ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Σχεδιασμός, κατασκευή, τεκμηρίωση και πειραματική εφαρμογή ενός IDE (Integrated Development Environment) σε JaVa, για JaVa.

Του φοιτητή
Μιχαήλ Κεμανετζή
Αρ.Μητρώου: **950636**

Επιβλέπων καθηγητής
Θεοδόσιος Χειμωνίδης

Θεσσαλονίκη 2008

Ευχαριστίες

Κατά την διάρκεια της παρούσας πτυχιακής εργασίας και κάποιες στιγμές κατά την διάρκεια των σπουδών μου με βοήθησαν πολύ οι υποδείξεις, παρατηρήσεις και συμβουλές του καθηγητή Θεοδόσιο Χειμωνίδη επιβλέποντα καθηγητή στην παρούσα πτυχιακή εργασία. Τον ευχαριστώ θερμά για αυτό και εκτιμώ επίσης πολύ τον προσωπικό χρόνο και χώρο που διέθεσε για να ολοκληρωθεί το παρόν.

Κατά την διάρκεια της φοίτησης μου στο τμήμα Πληροφορικής εκπαιδεύτηκα σε πολλά και διαφορετικά θέματα σχετικά και μη με την συγκεκριμένη πτυχιακή εργασία. Η συμβολή των καθηγητών και του προσωπικού του τμήματος ήταν πολύτιμη και καθοριστική για το σύνολο των γνώσεων και εμπειριών που απέκτησα.

Θα ήθελα να ευχαριστήσω και την οικογένεια μου για την υποστήριξη και υπομονή που πρόσφεραν ώστε να καταφέρω να φτάσω στο σημείο που είμαι τώρα. Έτοιμος να δείξω τις γνώσεις και εμπειρίες που απέκτησα σε ένα εκπαιδευτικό ίδρυμα που χαίρω άκρας εκτίμησης.

Περίληψη

Το αντικείμενο της εργασίας: Σχεδιασμός, κατασκευή, τεκμηρίωση και πειραματική εφαρμογή ενός Ενσωματωμένου Περιβάλλοντος Ανάπτυξης Προγραμμάτων, ***IDE*** *(****I****ntegrated* ***D****evelopment* ***E****nvironment), σε JAVA, για ανάπτυξη προγραμμάτων σε γλώσσα προγραμματισμού JAVA.*

Χαρακτήρας της προτεινόμενης εργασίας είναι κυρίως ο σχεδιασμός μοντέλου και η ανάπτυξη λογισμικού, που όμως συνοδεύεται από πλατιά διερεύνηση, κυρίως μέσα από το διαδίκτυο, υπαρχόντων ομοειδών ή ομόλογων μοντέλων.

Η διαπραγμάτευση του θέματος απαιτεί μια καλή γνώση αντικειμενοστραφούς προγραμματισμού και προγραμματιστική εμπειρία.

Από το εργαστήριο είχα όλα τα απαραίτητα βοηθήματα και εργαλεία, καθώς και μια ογκώδη βιβλιογραφία για να διαπραγματευτώ το θέμα στο σπίτι, και το μόνο που χρειαζόμουν ήταν να διαθέτω μια μοντέρνα παραθυρική πλατφόρμα (MS WIN, LINUX, MAC).

Το θέμα ολοκληρώθηκε με χρήση της JAVA και των σχετικών βιβλιοθηκών και εργαλείων ανάπτυξης προγραμμάτων μ' αυτήν. Στο πλαίσιο της εκπόνησης της πτυχιακής έγινε και παρουσίαση σε μορφή 1-2 σεμιναρίων με εκείνα τα μέρη των γνώσεων και εμπειριών που αποκτήθηκαν και που είχαν ίσως ένα γενικότερο ενδιαφέρον για τους καθηγητές και φοιτητές του τμήματος. Σημαντικό σύνολο υλικών και εργαλείων καθώς και μιας on-line βιβλιογραφίας, κυρίως εκπαιδευτικών πονημάτων (tutorials), εγχειριδίων οδηγιών χρήσης (user guides) και εγχειριδίων αναφοράς (user references), διατέθηκαν από το εργαστήριο και τον επιβλέποντα καθηγητή.

Το τμήμα μας δεν περιλάμβανε την JAVA και την δημιουργία εφαρμογών μ' αυτήν στο εκπαιδευτικό του πρόγραμμα κατά την διάρκεια φοίτησης μου οπότε χρειάστηκε να διαπραγματευτώ ένα "μη διδαγμένο" αντικείμενο, διαδικασία πολύ συνηθισμένη για ένα πληροφορικό, με την αλματώδη εξέλιξη των τεχνολογιών και των μέσων της πληροφορικής. Η ποιο πάνω ιδιομορφία θα ταλανίζει εφόρου ζωής τους πληροφορικούς και ιδιαίτερα τον δάσκαλο, που κάθε τόσο θα πρέπει να είναι έτοιμος να εισάγει αυτοδιδασκόμενος, άρτια και αποτελεσματικά, νέα αντικείμενα διδασκαλίας στο εκπαιδευτικό του πρόγραμμα. Με την παρούσα πτυχιακή και ιδιαίτερα τα απαιτούμενα σεμινάρια εμπλούτισα τις γνώσεις μου και γυμνάστηκα στην προετοιμασία και εκτέλεση εκπαιδευτικών σεμιναρίων.

Abstract

The object of work: Designing, development, documentation and experimentation of an IDE (Integrated Development Environment), written in the coding language JAVA, for developing programs in JAVA.

The purpose of the proposed work is mainly the designing of a software model and the development of this software, that however is accompanied by wide investigation, mainly through the internet, of existing, similar or homologous models.

The negotiation of the current subject requires a good knowledge of object oriented development and programming experience.

From the laboratory I had all the essential aids and tools, as well as a bulky bibliography in order to work with the subject at home, and the only I needed to have was a modern visual system platform (MS WIN, LINUX, MAC).

The subject was completed with use of JAVA and relative libraries and tools. While in development I did also presentations in the form of 1-2 seminars with those parts of knowledge and experiences that were acquired and that had perhaps a more general interest for the professors and students of the department. Important materials and tools as well as an on-line bibliography, mainly tutorials, user guides and user references, were been disposed by the laboratory and the supervising professor.

At the duration of my studies our department did not include the language of JAVA and the creation of applications in its educational program, therefore it needed to work using a non taught programming language, process very usual for a computer specialist, with the swift development of technologies and means of information technology. That is not very rare for those working in rapid developing technologies like Informatics and is something that will try the patience of the computer specialists and particularly the schoolteachers, that each so much will be supposed to self teach themselves, effective on new technologies so they incorporate them in their educational program. With the present final work and particularly the required seminars I enriched my knowledge and experience in the preparation and implementation of educational seminars.

ΠΕΡΙΕΧΟΜΕΝΑ

1 Εισαγωγή

Ο JavaBegginersBuilder JBB αναπτύχθηκε σε Java χωρίς την χρήση κάποιου IDE γιατί κατά την έναρξη του δεν υπήρχαν και πολλά εργαλεία για την συγγραφή παραθυρικών προγραμμάτων σε γλώσσα Java και χρησιμοποιείται για τον οπτικό προγραμματισμό νέων προγραμμάτων σε Java.

Η εφαρμογή προσφέρει ένα χώρο εργασίας όπου μπορεί κάποιος με οπτικό τρόπο να δημιουργεί και να επεξεργάζεται παράθυρα μιας εφαρμογής Java. Οι πληροφορίες της απεικόνισης αποθηκεύονται μέσα στον ίδιο τον κώδικα ώστε να μπορούν να μεταγλωττίζονται από οποιαδήποτε έκδοση Java χρησιμοποιείται ακόμα και εκτός του εργαλείου αυτού. Ταυτόχρονα έχει την δυνατότητα να βλέπει και να επεξεργάζεται τον κώδικα που δημιουργείται για κάθε ένα παράθυρο μέσα από ένα εύχρηστο «επεξεργαστή κειμένου κώδικα».

Όταν το νεοδημιουργηθέντα πρόγραμμα είναι έτοιμο για δοκιμή, τότε μέσα από το IDE δίνεται η δυνατότητα για μεταγλώττιση, εκτέλεση και capture των εξόδων κειμένου σε παράθυρο του IDE ώστε να είναι ποιο εύκολο η αποσφαλμάτωση της εφαρμογής.

Δέχεται plugins εύκολα και γρήγορα λόγο ενσωματωμένου ανοικτού interface ειδικά για αυτή την λειτουργία. Και υπάρχει και 1 plugin που μπορεί να χρησιμοποιηθεί ως sample. Για την οπτική επεξεργασία υπάρχει και ένας προχωρημένος Property Editor όπου με τεχνικές reflection που προσφέρονται από την Java εμφανίζει όλες τις ιδιότητες ενός οπτικού αντικειμένου και επιτρέπει την αλλαγή τους. Οι μπάρες με τα component γεμίζουν εύκολα με όσα οπτικά συστατικά είναι και JavaBeans αντικείμενα. Έχει μερικώς ένα wizard generator όπου επιλέγεις ένα πίνακα ή δύο (master-detail) και κάνει generate κώδικα και παράθυρο για την προβολή των πληροφοριών αυτών με χρήση grid αλλά και δυναμικά δημιουργημένων συστατικών σαν edit φόρμα. (με χρήση JDBC driver)

Για τις διεπαφές χρήστη έγινε χρήση των βιβλιοθηκών AWT και SWING. Σε επόμενα κεφάλαια δίνεται μια περιγραφή της AWT και συμπληρωματικά δίνονται και 2 παρουσιάσεις powerpoint με την πρώτη να μας κάνει την Εισαγωγή σε JAVA και την δεύτερη να μας εξηγεί την βιβλιοθήκη AWT.

2 Εισαγωγή στη JAVA

2.1 "Γεια σου, κόσμε".

Η *JAVA* είναι μία πλήρης αντικειμενοστραφής γλώσσα προγραμματισμού. Οποιοδήποτε πρόγραμμα πρέπει να έχει τουλάχιστον μία κλάση, που θα υλοποιεί την παρακάτω μέθοδο `main` :

```
class HelloWorld{
  public static void main(String[] args){
    // . . .
    //κώδικας που εκτελείται πρώτος π.χ.
    System.out.println("Γεια σου, κόσμε");
    // . . .
  }
}
```

Για την εκτέλεση του παραπάνω πλήρους προγράμματος χρειάζεται ένας συντάκτης(editor) κειμένου (οποιοσδήποτε), ένας μεταγλωττιστής (compiler) και ένας διερμηνέας (interpreter) της Java. Το Java Development Kit (JDK) προσφέρεται δωρεάν από την Sun Microsystems και περιέχει όλα τα απαραίτητα εργαλεία εκτός του συντάκτη κειμένου. Τα ονόματα των φακέλων με τον πηγαίο κώδικα πρέπει να έχουν κατάληξη `.java` και πρόθεμα το όνομα της κλάσης που περιέχουν (αν είναι περισσότερες από μία τότε μιας από αυτές), π.χ. το παραπάνω πρέπει να γραφεί σε έναν φάκελο ονομαζόμενο `Helloworld.java`. Αν διαθέτουμε το JDK το πρόγραμμά θα μεταγλωττιστεί με :

```
javac HelloWorld.java
```

και θα εκτελεστεί με :

```
java HelloWorld.
```

Πώς εργάζεται όμως το πρόγραμμά μας;

Δηλώνει μία κλάση που ονομάζεται `HelloWorld` με μία και μοναδική μέθοδο την `main`, χωρίς πεδία, άλλες μεθόδους ή άλλες εσωτερικές κλάσεις και διασυνδέσεις (θα εξηγήσω παρακάτω τη σημαίνει αυτό). Όπως φαίνεται τα άγκιστρα "{" και "}" χρησιμοποιούνται για τον διαχωρισμό μεταξύ της κλάσης και των περιεχομένων της (σώμα), μεταξύ των μεθόδων και των προτάσεων (statements) του ορισμού τους, όπως και για ομαδοποίηση προτάσεων (block statements) του προγράμματος.

Η `main` δηλώνεται `public` που σημαίνει ότι μπορεί να κληθεί από οποιοδήποτε άλλη κλάση και `static` που σημαίνει ότι η μέθοδος ανήκει στην κλάση και όχι στα στιγμιότυπα της. Επιστρέφει τιμή `void` δηλαδή τίποτα για την Java. Σαν παραμέτρους δέχεται ένα πίνακα από αντικείμενα `String` τα οποία είναι οι παράμετροι που θα περαστούν κατά την κλήση του προγράμματος από το λειτουργικό σύστημα.

Μέσα στην `main` γράφονται οι προτάσεις που τρέχουν πρώτες, όπως δημιουργία αντικειμένων, υπολογισμός τιμής εκφράσεων και γενικά οτιδήποτε χρειάζεται για να ξεκινήσει μία εφαρμογή.

Στο παράδειγμά μας εκτελείται μία μόνο πρόταση. Καλείται η μέθοδος `println` του `static` πεδίου `out` της κλάσης `System` με παράμετρο ένα `String`. Το πεδίο - αντικείμενο

`out` αντιστοιχεί στην καθιερωμένη έξοδο[1] του προγράμματος. Οπότε αποτέλεσμα είναι να εμφανιστεί στην οθόνη το μήνυμα "Γεια σου, κόσμε".

2.2 Μεταβλητές – Τύποι δεδομένων.

Οι μεταβλητές μπορούν να ονομάζονται σε οποιαδήποτε γλώσσα γιατί η Java χρησιμοποιεί Unicode χαρακτήρες στον πηγαίο κώδικα της. Τα αρχεία (φάκελοι) `.java` πριν μεταγλωττιστούν μεταφράζονται σε Unicode κείμενα σύμφωνα με την γλώσσα του συστήματος που γίνεται η μεταγλώττιση, π.χ. θα μπορούσε να δηλωθεί μία μεταβλητή με όνομα π η οποία θα περιέχει το γνωστό από τα μαθηματικά άρρητο αριθμό 3.14159265….

```
final static double π  = 3.14159265;
```

Η Java υποστηρίζει κάποιους στοιχειώδεις τύπους δεδομένων οι οποίοι αναφέρονται παρακάτω (Τα μεγέθη των τύπων αυτών είναι σταθερά για όλα τα συστήματα, σε αντίθεση με τις άλλες γλώσσες προγραμματισμού). Πέρα όμως από τους τύπους αυτούς η Java διαθέτει και σχετικές κλάσεις με αντικείμενα-τιμές όλων των βασικών τύπων.
Η δήλωση `final` δείχνει πώς η π δεν μπορεί να αλλάξει, είναι δηλαδή μία σταθερά[2].

Τύπος	**Μέγεθος / Μορφή**	**Περιγραφή**
Ακέραιοι (integers)		
`byte`	8-bit δυαδική μορφή	Προσημασμένος ακέραιος μήκους Byte
`short`	16-bit δυαδική μορφή	Προσημασμένος ακέραιος μήκους Short
`int`	32-bit δυαδική μορφή	Προσημασμένος ακέραιος μήκους Integer
`long`	64-bit δυαδική μορφή	Προσημασμένος ακέραιος μήκους Long
Πραγματικοί (real numbers)		
`float`	32-bit IEEE 754	Μονής-ακρίβειας πραγματικός αριθμός
`double`	64-bit IEEE 754	Διπλής-ακρίβειας πραγματικός αριθμός
Άλλοι (other types)		
`char`	16-bit Unicode χαρακτήρας	Ένας χαρακτήρας
`boolean`	Αληθές ή ψευδές	Μία Boolean μεταβλητή

2.3 Έλεγχος Ροής προγράμματος.

Η Java υποστηρίζει τις περισσότερες γνωστές μορφές ελέγχου ροής ενός προγράμματος όπως :

1. Επιλογή(Choice) `if-else, switch-case`
2. Βρόχους(loop) `for, while, do-while`
3. Εξαιρέσεις(Exception) `try-catch-finally, throw`
4. Διάφορα(miscellaneous) `break, continue, label: , return`

[1] Δεν είναι καθαρή Java η εντολή αυτή, γιατί δεν διαθέτουν όλα τα λειτουργικά συστήματα καθιερωμένη έξοδο στις εφαρμογές.
[2] Ισοδυναμεί με `const` της C++

2.4 Κλάσεις, αντικείμενα και διασυνδέσεις (interfaces).

Όλες οι νέες κλάσεις που δημιουργούνται εάν δεν είναι υπερκλάσεις κάποιας άλλης, τότε αυτόματα είναι υπερκλάσεις της βασικής Object. Η κλάση αυτή δίνει κάποιες κοινές λειτουργίες σε όλα τα στιγμιότυπα των υπερκλάσεών της. Στιγμιότυπα κάποιας κλάσης δημιουργούνται με την δεσμευμένη λέξη new π.χ.

```
String τοΟνομαΤηςΜεταβλητής = new String("Γεια σου, κόσμε");
```

Η καταστροφή των αντικειμένων γίνεται αυτόματα από την Java χωρίς να χρειάζεται κάποια ενέργεια του προγραμματιστή. Εξαλείφονται έτσι πολλά προβλήματα απώλειας μνήμης (memory leaks) κατά την εκτέλεση της εφαρμογής. Την εργασία καταστροφής των αντικειμένων που δεν προσπελαύνονται μέσο κάποιας μεταβλητής την αναλαμβάνει ο Συλλέκτης Τμημάτων[3] απελευθερωμένης μνήμης (Garbage Collector GC). Όταν χρειαστεί μνήμη το πρόγραμμα και δεν υπάρχει άμεσα διαθέσιμη, εκτελείται η λειτουργία συλλογής τμημάτων μνήμης από τον GC και ανακτάται η μη χρησιμοποιημένη μνήμη.
Κάθε κλάση μπορεί να είναι υπερκλάση μόνο μίας κλάσης. Στην περίπτωση που υπάρχει η ανάγκη για πολλαπλή κληρονομικότητα χρησιμοποιούνται οι διασυνδέσεις. Οι διασυνδέσεις ορίζονται όπως και οι κλάσεις, αλλά όπου "class" χρησιμοποιείται το "interface" και περιέχουν μόνο δηλώσεις μεθόδων χωρίς τις υλοποιήσεις τους, π.χ. :

```
interface Ψάξιμο{
  public int getPosition(String όνομα);
}
```

Έτσι μία κλάση πρέπει να υλοποιήσει αυτή την μέθοδο εάν θέλει να υποστηρίζει αυτήν την διασύνδεση, π.χ. :

```
class ΠίνακαςΟνομάτων extends Πίνακας implements Ψάξιμο{
  //. . .
  public int getPosition(String όνομα){
    int θέση = 0;
    //. . .
    //κώδικας μεθόδου
         //. . .
    return θέση
  }
}
```

2.5 Πακέτα (Packages).

Η Sun για την αποφυγή συγκρούσεων μεταξύ των ονομάτων των κλάσεων, που δημιουργούνται από διάφορους προγραμματιστές / εταιρίες, σκέφτηκε να δίνει μεγάλα ονόματα σ' αυτές, με μορφή που χρησιμοποιείται και στο Internet, όπου οι κλάσεις ομαδοποιούνται σε πακέτα (packages). Έτσι οι κλάσεις πρέπει να γράφονται με το πλήρες όνομά τους ή για ευκολία να χρησιμοποιείτε μόνο το τελικό κομμάτι του ονόματός τους (Πρέπει όμως τότε να

[3] κυριολεκτικά : Συλλέκτης Σκουπιδιών. Υπονοείται εδώ ο αχρείαστος πια χαρακτήρας των συλλεγμένων τμημάτων μνήμης. Μια συνεχής ανακύκλωση μνήμης.

οριστεί η βραχυγραφία με χρήση του όρου `import` στην αρχή του πηγαίου κώδικα) εάν δεν υπάρχει άλλη κλάση με το ίδιο τελικό κομμάτι. π.χ.

Η κλάση `System` που έχουμε χρησιμοποιήσει παραπάνω είναι στην πραγματικότητα η κλάση `java.lang.System`, αλλά το πακέτο `java.lang` γίνεται πάντα `import` στον κώδικά μας (γράφοντας `import java.lang.*;`). Με το * δηλώνουμε ότι θέλουμε να χρησιμοποιήσουμε με βραχυγραφία όλες τις κλάσεις του συγκεκριμένου πακέτου (Εάν δεν θέλαμε όλες, παρά μόνο την `System`, θα γράφαμε `import java.lang.System;`).

2.6 Τέλος Εισαγωγής στη Java.

Εδώ τελειώνει η εισαγωγή στον προγραμματισμό με Java. Για μεγαλύτερη εμβάθυνση στο προγραμματισμό αυτό, ανατρέξτε στη σχετική Βιβλιογραφία του τέλους των σημειώσεων.

3 Abstract Windowing Toolkit (AWT).

Στα ελληνικά το πακέτο αυτό θα μπορούσε να αποδοθεί ως Εργαλεία δημιουργίας συστήματος Αφηρημένων Παραθύρων.

Τα παραδείγματα των σημειώσεων όπως και της παρουσίασης είναι πλήρη και εκτελέσιμα. Εάν κάποιος θέλει να πειραματιστεί με τον προτεινόμενο σ' αυτά κώδικα, ώστε να κατανοήσει καλύτερα το AWT, δεν χρειάζεται παρά μία πλήρη αντιγραφή του.[4]

Θεωρείται ότι ο ακροατής της παρουσίασης και αναγνώστης του παρόντος, γνωρίζει στοιχειώδεις έννοιες αντικειμενοστραφούς προγραμματισμού.

Το AWT είναι μία βιβλιοθήκη ή πακέτο (package), κατά την ορολογία της SUN, αποτελούμενο από κλάσεις (classes) και διασυνδέσεις (interfaces). Η λειτουργία των κλάσεων αυτών είναι η παροχή στον προγραμματιστή ενός εργαλείου δημιουργίας του γραφικού περιβάλλοντος επικοινωνίας (GUI, Graphical User Interface), μεταξύ της εφαρμογής (Applet ή Application) και του χρήστη της.

Η JAVA έχει σχεδιαστεί έτσι ώστε οι εφαρμογές της να έχουν την δυνατότητα να εκτελούνται σε ποικίλα γραφικά λειτουργικά συστήματα (ή και μη γραφικά), όπου το κάθ' ένα έχει τα δικά του χαρακτηριστικά στον τρόπο επικοινωνίας με τον χρήστη. Συνεπώς το AWT πακέτο, προκειμένω να ικανοποιήσει το στόχο της μεταφερσιμότητας σε διαφορετικά περιβάλλοντα, υλοποιεί τα κοινά χαρακτηριστικά τους, περιορίζοντας έτσι τις φυσικές δυνατότητες κάθε συστήματος. Αυτό είναι σαφώς μειονέκτημα, που όμως αντισταθμίζεται, με το παραπάνω μάλιστα, από το γεγονός ότι η εφαρμογή μας είναι πια εκτελέσιμη παντού, όπου υπάρχει μία JavaVM[5], χωρίς επιπρόσθετο κώδικα ή μετατροπές στον ήδη υπάρχοντα.

Ένας λογικός διαχωρισμός των κλάσεων του AWT για την ευκολία της καλύτερης κατανόησης του είναι :

5. Γραφικές συνιστώσες (Components)
6. Πλάνα (Layouts)
7. Βοηθητικές Κλάσεις (Utilities)
8. Κλάσεις χειρισμού γεγονότων (Event Handling)

Στη συνέχεια θα αναλυθούν οι κατηγορίες αυτές, περιγράφοντας τις βασικές κλάσεις τους και παράλληλα θα δίνονται αντιπροσωπευτικά παραδείγματα κώδικα χρήσης των κλάσεων αυτών των κατηγοριών.

3.1 Γραφικές συνιστώσες (Components).

Οι κλάσεις της κατηγορίας αυτής χρησιμοποιούνται για την δημιουργία αντικειμένων, τα οποία προσφέρουν μία γραφική αναπαράσταση της πληροφορίας που περιέχουν, με διαφορετικό τρόπο το κάθ' ένα τους. Τα αντικείμενα αυτά αντιλαμβάνονται τις ενέργειες του χρήστη πάνω τους (πληκτρολόγιο, ποντίκι, κ.τ.λ.), στέλνοντας μηνύματα στους ενδιαφερόμενους αποδέκτες - Ακροατές (που είναι άλλα αντικείμενα και ονομάζονται *listeners* κατά SUN).

Η βασική κλάση από την οποία προέρχονται οι περισσότερες AWT συνιστώσες είναι η `java.awt.Component` (Δίνεται το ιεραρχικό δέντρο των γραφικών συνιστωσών στο παράρτημα). Η κλάση αυτή δίνει τις κοινές ιδιότητες όλων των συνιστωσών, όπως είναι το

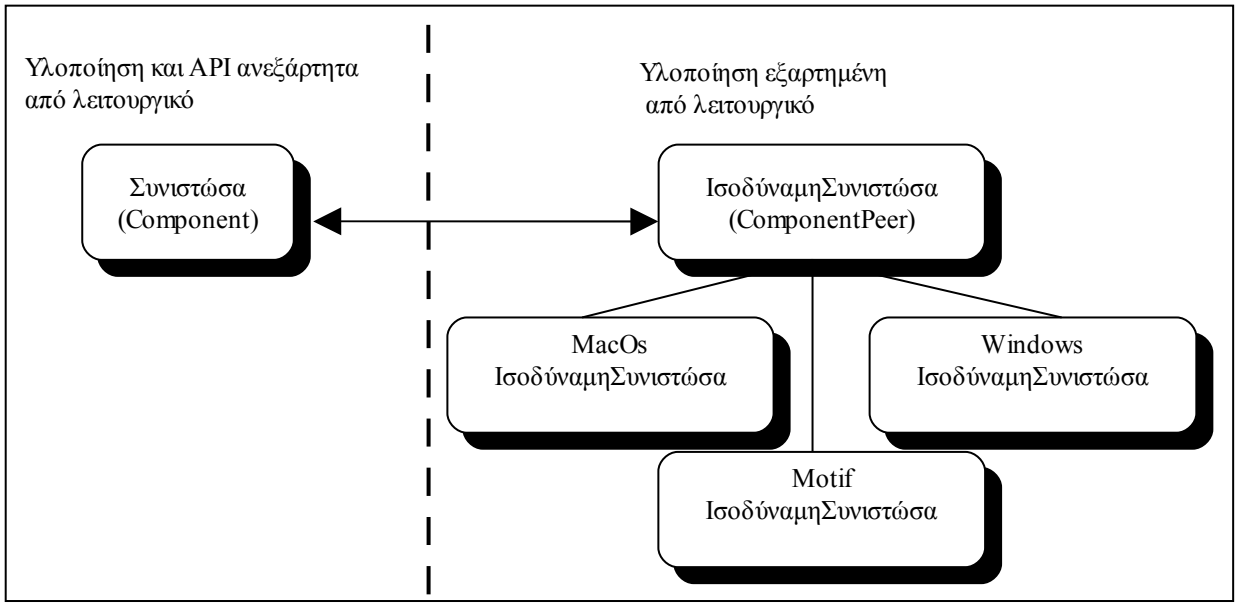
[4] Διατίθενται έτοιμα σε μηχαναγνώσιμη μορφή από τον συγγραφέα.

[5] JavaVM : Java Virtual Machine, είναι μία εφαρμογή η οποία εξομοιώνει μία υποθετική - εικονική μηχανή ικανή να εκτελεί μεταγλωττισμένο κώδικα Java.

μέγεθος, η γραμματοσειρά, η θέση(location) τους ,τα χρώματα, κτλ, καθώς επίσης και τις κοινές μεθόδους που έχουν όπως :

9. Εμφάνισε / κρύψε `setVisible(true/false)`
10. Ενεργοποίησε ή όχι `setEnabled(true/false)`
11. Σχεδίασε το `paint()`
12. Ξανασχεδίασε το `repaint()`
13. Πρόσθεσε / αφαίρεσε Ακροατές π.χ.`addMouseListener(myMouseListener)`
    οι Ακροατές από την 1.1 έκδοση της JAVA χρησιμοποιούνται για τον έλεγχο των γεγονότων που εισάγει ο χρήστης (και όχι μόνο). Θα αναφερθούμε παρακάτω εκτενέστερα στον χειρισμό γεγονότων.
14. Εγκυροποίησε `validate()`
    που σημαίνει, ότι μετά την εκτέλεση της μεθόδου, έχει δημιουργηθεί και τοποθετηθεί το σχετικό με το λειτουργικό σύστημα ισοδύναμο(peer) της γραφικής συνιστώσας του AWT.

Γραφική απεικόνιση των ισοδυνάμων (peers)



Οι AWT γραφικές συνιστώσες μπορούν να διαχωριστούν σε τρεις λογικές υποκατηγορίες

15. Περιέκτες (Containers)
16. Απλές συνιστώσες (Simple components – widgets)
17. Η κλάση `java.awt.canvas`

3.1.1 Περιέκτες (Containers).

Περιέκτες είναι οι γραφικές συνιστώσες οι οποίες έχουν την δυνατότητα να περιέχουν άλλες συνιστώσες, και είναι υπερκλάσεις της κλάσης `java.awt.Container`, η οποία είναι υπερκλάση της βασικής. Μέσο αυτής της κλάσης όλοι οι υποδοχείς έχουν κοινές ιδιότητες και μεθόδους όπως :

18. Πρόσθεσε / αφαίρεσε συνιστώσες π.χ.`add(Component comp)`
    ή `add(Component comp, Constraints περιορισμοί)`

19. Επέλεξε Διαχειριστή Πλάνου `setLayout(LayoutManager lm)`
20. Κάνε πλάνο συνιστωσών `doLayout()`

Οποιεσδήποτε συνιστώσες, περιέκτες ή μη μπορούν να περιέχονται μέσα σε περιέκτες, δημιουργώντας ένα δέντρο από συνιστώσες. Οι οποιοιδήποτε όμως περιέκτες δεν μπορούν να είναι ρίζες του δέντρου αυτού, να είναι δηλαδή αρχικοί περιέκτες. Οι περιέκτες που έχουν αυτή την δυνατότητα είναι του τύπου της κλάσης `java.awt.Window` ή κάποιας από τις υπερκλάσεις της, π.χ. `java.awt.Frame, java.awt.Dialog, java.awt. FileDialog`. Από αυτά, μόνο τα αντικείμενα τύπου `java.awt.Frame` έχουν πλήρη αυτονομία αφού δεν χρειάζεται να έχουν κάποιον γονέα που να τα δημιουργεί.

Υπάρχει και η κλάση `java.awt.Panel`, στιγμιότυπα της οποίας είναι οι ποιο απλοί περιέκτες, οι οποίοι χρησιμοποιούνται μόνο για σχετικές τοποθετήσεις γραφικών συνιστωσών στο πλάνο.

Από την παραπάνω κλάση απορρέει η `java.applet.Applet` (Παλιότερα ονομαζότανε `java.awt.Applet` γι' αυτό και την αναφέρω ως κλάση του AWT πακέτου). Η κλάση αυτή είναι ένας ειδικός περιέκτης τα αντικείμενα (στιγμιότυπα) του οποίου χρησιμοποιούνται ως αρχικοί περιέκτες, με τον περιορισμό, αυτά να τοποθετούνται μέσα σε κάποιά άλλη ειδική εφαρμογή, όπως είναι οι φυλλομετρητές (browsers) ή ο applet_viewer της SUN.

Συνήθως πριν την εμφάνιση των γραφικών συνιστωσών, δίνεται κάποιο μέγεθος σ' αυτές είτε ρητά `setSize(200,100)` είτε έμμεσα `pack()` όπου το μέγεθος γίνεται τόσο όσο χρειάζονται οι περιεχόμενες συνιστώσες για την εμφάνισή τους(με το επιθυμητό τους μέγεθος) στο πλάνο.

3.1.2 Απλές συνιστώσες (Simple Components).

Απλές συνιστώσες είναι αυτές οι οποίες προσφέρουν με τον δικό τους τρόπο κάποια χρηστικότητα στο γραφικό περιβάλλον επικοινωνίας. Αναλυτικά στο πακέτο AWT έχουμε :

Ετικέτα(Label)

Είναι η ποιο απλή συνιστώσα και το μόνο που κάνει είναι να εμφανίζει μία γραμμή από κείμενο, στοιχισμένο αριστερά, στο κέντρο ή δεξιά.

Πλήκτρο(Button)

Το πλήκτρο είναι όμοιο (όχι στην εμφάνιση) με την ετικέτα, συν του ότι αντιλαμβάνεται το πάτημα και στέλνει μήνυμα για την ενέργεια αυτή στους σχετικούς Ακροατές.

Επιλογή(Choice)

Η συνιστώσα αυτή επιτρέπει στον χρήστη να επιλέγει μέσα από μία λίστα αντικείμένων(κειμένου) ένα και μόνο ένα.

Λίστα(List)

Όμοιο με την Επιλογή αλλά επιτρέπει την επιλογή περισσοτέρων του ενός αντικειμένων.

ΚουτάκιΕλέγχου(Checkbox)

Το Checkbox μπορεί να έχει σαν τίτλο μία ετικέτα. Η λειτουργία του είναι να εμφανίζει με κάποιο τρόπο τιμές αληθείας (αληθές ή ψευδές).

ΠεδίοΚειμένου(TextField)

Η συνιστώσα αυτή περιέχει μία γραμμή από κείμενο, εμφανίζοντας την και επιτρέποντας(αν επιθυμούμε) την επεξεργασία της.

ΠεριοχήΚειμένου(TextArea)

Όμοια με το ΠεδίοΚειμένου, αλλά μπορεί να περιέχει περισσότερες από μία γραμμές.

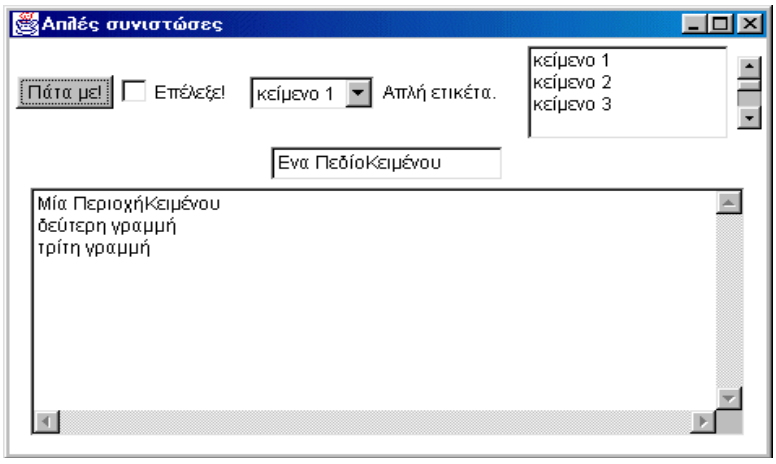
ΜπάραΚύλισης(ScrollBar)

Περιέχει κάποια τιμή από μία περιοχή ακεραίων τιμών [<ελάχιστη τιμή>, <μέγιστη τιμή>]. Την τιμή αυτή την εμφανίζει μέσο μίας μπάρας κύλισης, όπου μετακινώντας την, μπορεί να αλλάξει την περιεχόμενη τιμή. Συνήθως χρησιμοποιείται για την εμφάνιση επιφανειών μεγαλυτέρων από τον διαθέσιμο φυσικό χώρο. π.χ.

Έχουμε μία εικόνα μεγέθους 1000χ1000 pixel και ένα Παράθυρο μεγέθους 400χ400. Σίγουρα δεν μπορούμε να δούμε ολόκληρη την εικόνα στο παραπάνω Παράθυρο. Χρησιμοποιούμε λοιπόν, δύο μπάρες κύλισης (X, Y άξονες) με περιοχή τιμών από 0 ως 600 (1000-400), έτσι ώστε ανάλογα με την μετακίνηση αυτών να εμφανίζουμε το ανάλογο κομμάτι. Αν X μπάρα έχει τιμή 200 και Y μπάρα έχει τιμή 300 το κομμάτι που θα εμφανιστεί θα είναι μεταξύ (200,400) (πάνω αριστερή γωνία) και (700,900) (κάτω δεξιά γωνία)

Παράδειγμα κώδικα για απλές συνιστώσες.

```
1.  import java.awt.*;
2.
3.      public class SimpleWidgetFrame extends java.awt.Frame{
4.
5.         Button button = new Button("Πάτα με!");
6.         Checkbox checkbox = new Checkbox("Επέλεξε!");
7.         Choice choice = new Choice();
8.         Label label = new Label("Απλή ετικέτα.");
9.         List list = new List();
10.     Scrollbar scrollbar = new Scrollbar();
11.     TextField textfield = new TextField("Ενα ΠεδίοΚειμένου");
12.     TextArea textarea = new TextArea("Μία ΠεριοχήΚειμένου\nδεύτερη γραμμή\nτρίτη γραμμή");
13.     pubic SimpleWidgetFrame(){
14.        super("Ενα απλό παράθυρο με τις απλές συνιστώσες");
15.        setLayout(new FlowLayout());
16.
17.        choice.add("κείμενο
   1");
18.        choice.add("κείμενο
   2");
19.        choice.add("κείμενο
   3");
20.        list.add("κείμενο
   1");
21.        list.add("κείμενο
   2");
22.        list.add("κείμενο
   3");
23.        add(button);
24.        add(checkbox);
25.        add(choice);
26.        add(label);
27.        add(list);
28.        add(scrollbar);
29.        add(textfield);
```



```
30.         add(textarea);
31.         pack();
32.         setVisible(true);
33.      }
34.      public static void main(String[] args){
35.         SimpleWidgetFrame swf = new SimpleWidgetFrame();
36.      }
37.   }
```

3.1.3 `java.awt.canvas.`

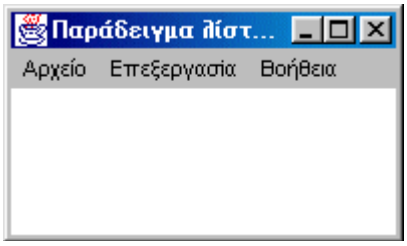
Την κλάση αυτή την αναφέρομε ξεχωριστά, γιατί δεν έχει κάποια άμεση λειτουργικότητα, αλλά συνήθως χρησιμοποιείται ως γονέας κλάσεων δημιουργούμενων από τον χρήστη. Συνήθως υπερκαλύπτεται η μέθοδος paint(Graphics g) ώστε η νέα κλάση να σχεδιάζει τα δικά της στοιχεία(εικόνες, γραφήματα, κτλ.).

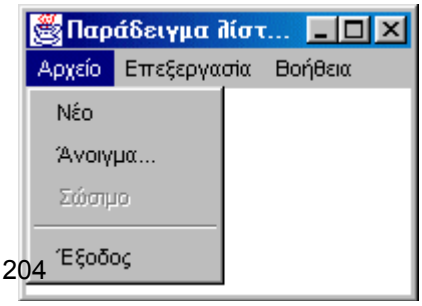
3.1.4 Λίστα επιλογών (menu).

Οι συνιστώσες που χρησιμοποιούνται για την δημιουργία μπαρών λίστας επιλογών (Menubars) δεν ανήκουν σε υπερκλάσεις της `java.awt.Component`, αλλά της `java.awt.MenuComponent`. Μία λίστα επιλογών αντιστοιχεί σε στιγμιότυπο της κλάσης `java.awt.Menu`. Μπορεί να εμφανιστεί μόνο μέσο μίας μπάρας λίστας επιλογών (`java.awt.MenuBar`) η οποία τοποθετείται σε κάποιο παράθυρο (`java.awt.Window`). Μία λίστα επιλογών περιέχει συνιστώσες λίστας επιλογών (`java.awt.MenuItem`) ή άλλες λίστες επιλογών, αφού η κλάση `java.awt.Menu` είναι υπερκλάση της κλάσης `java.awt.MenuItem`. Μία συνιστώσα λίστας επιλογών είναι της μορφής ετικέτας ή κουτάκι ελέγχου, συνιστώσες, που έχουν περιγραφεί παραπάνω.

Παράδειγμα κώδικα χρήσης λίστας επιλογών.

```
1)  import java.awt.*;
2)
3)  public class MenuFrame extends Frame{
4)
5)     MenuItem fileNew = new MenuItem("Νέο");
6)     MenuItem fileOpen = new MenuItem("Άνοιγμα...");
7)     MenuItem fileSave = new MenuItem("Σώσιμο");
8)     MenuItem fileExit = new MenuItem("Εξοδος");
9)     MenuItem editUndo = new MenuItem("Αναίρεση");
10) MenuItem editCut = new MenuItem("Αποκοπή");
11) MenuItem editCopy = new MenuItem("Αντιγραφή");
12) MenuItem editPaste = new MenuItem("Επικόλληση");
13) MenuItem helpContents = new MenuItem("Περιεχόμενα");
14) MenuItem helpAbout = new MenuItem("Σχετικά με το πρόγραμμα...");
15)
16) public MenuFrame(){
17)    super("Παράδειγμα λίστας επιλογών");
18)    MenuBar menubar = new MenuBar();
19)
20)    Menu fileMenu = new Menu("Αρχείο");
21)    Menu editMenu = new Menu("Επεξεργασία");
22)    Menu helpMenu = new Menu("Βοήθεια");
23)
24)    fileMenu.add(fileNew);
```





```
25)         fileMenu.add(fileOpen);
26)         fileSave.setEnabled(false);
27)         fileMenu.add(fileSave);
28)         fileMenu.addSeparator();
29)         fileMenu.add(fileExit);
30)         editUndo.setEnabled(false);
31)         editMenu.add(editUndo);
32)         editMenu.addSeparator();
33)         editCut.setEnabled(false);
34)         editMenu.add(editCut);
35)         editCopy.setEnabled(false);
36)         editMenu.add(editCopy);
37)         editPaste.setEnabled(false);
38)         editMenu.add(editPaste);
39)         helpMenu.add(helpContents);
40)         helpMenu.addSeparator();
41)         helpMenu.add(helpAbout);
42)         menubar.add(fileMenu);
43)         menubar.add(editMenu);
44)         menubar.add(helpMenu);
45)         menubar.setHelpMenu(helpMenu);
46)
47)         setMenuBar(menubar);
48)         setSize(new Dimension(200, 100));
49)         setVisible(true);
50)     }
51)     public static void main(String[] args){
52)         MenuFrame me = new MenuFrame();
53)     }
54) }
```

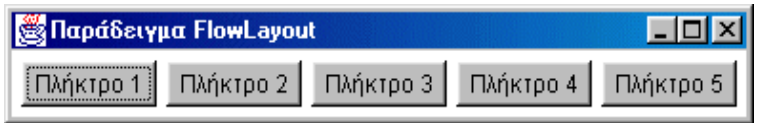
3.2 Πλάνα (layouts)

Η τοποθέτηση των συνιστωσών μέσα σε περιέκτες μπορεί να γίνει είτε δίνοντας ρητά την θέση `setLocation(x, y)` και το μέγεθος του αντικειμένου ή λέγοντας στον περιέκτη ποιον Διαχειριστή Πλάνου (`LayoutManager`) να χρησιμοποιήσει. Οι διαχειριστές πλάνων υπολογίζουν την θέση και το μέγεθος των συνιστωσών βάσει κάποιων κανόνων. π.χ.

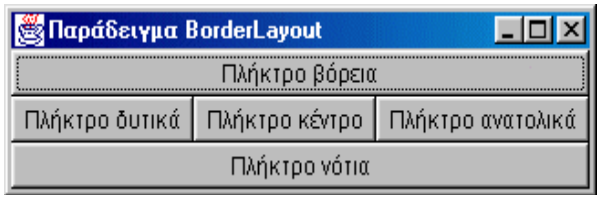
21. ο `FlowLayout` τοποθετεί τις συνιστώσες σε μία ροή από συνιστώσες. Όσες χωράνε στη γραμμή(ή ανά στήλη) σχεδιάζονται εκεί, τα υπόλοιπα παρακάτω με όμοιο τρόπο. Υπάρχει η δυνατότητα επίσης να οριστούν τα κενά μεταξύ των συνιστωσών κατά τον x άξονα (`setHgap()`) και κατά τον y άξονα (`setVgap()`).
22. ο `BorderLayout` τοποθετεί τις συνιστώσες σε μία από τις πέντε περιοχές που ορίζει (βορράς, νότος, δύση, ανατολή, κέντρο).
23. ο `GridLayout` τοποθετεί τις συνιστώσες σε ένα πλέγμα από v γραμμές και μ στήλες, ξεκινώντας από αριστερά προς τα δεξιά και προς τα κάτω.
24. Ανάλογα με άλλους κανόνες υπάρχουν και οι `GridBagLayout`, `BoxLayout`, `OverlayLayout`, κτλ.

Παράδειγμα FlowLayout.

```
1.  import java.awt.*;
2.
3.  public class FlowLayoutFrame extends Frame{
4.
5.     public FlowLayoutFrame(){
6.        super("Παράδειγμα FlowLayout");
7.        setLayout(new FlowLayout(FlowLayout.RIGHT));
8.
9.        add(new Button("Πλήκτρο 1"));
10.    add(new Button("Πλήκτρο 2"));
11.    add(new Button("Πλήκτρο
       3"));
12.    add(new Button("Πλήκτρο
       4"));
13.    add(new Button("Πλήκτρο 5"));
14.
15.    pack();
16.    setVisible(true);
17.  }
18.  public static void main(String[] args){
19.     FlowLayoutFrame me = new FlowLayoutFrame();
20.  }
21. }
```



Παράδειγμα BorderLayout.




```
1.  import java.awt.*;
2.
3.  public class BorderLayoutFrame extends
    Frame{
4.
5.     public BorderLayoutFrame(){
6.        super("Παράδειγμα BorderLayout");
7.        setLayout(new BorderLayout());
8.
9.        add(new Button("Πλήκτρο βόρεια"), BorderLayout.NORTH);
10.    add(new Button("Πλήκτρο νότια"), BorderLayout.SOUTH);
11.    add(new Button("Πλήκτρο δυτικά"), BorderLayout.WEST);
12.    add(new Button("Πλήκτρο ανατολικά"), BorderLayout.EAST);
13.    add(new Button("Πλήκτρο κέντρο"), BorderLayout.CENTER);
14.
15.    pack();
16.    setVisible(true);
17.  }
18.  public static void main(String[] args){
19.     BorderLayoutFrame me = new BorderLayoutFrame();
20.  }
21. }
```

Παράδειγμα GridLayout.

```
1   import java.awt.*;
2
3   public class GridLayoutFrame extends Frame{
4
5     public GridLayoutFrame(){
6       super("Παράδειγμα GridLayout");
7       setLayout(new GridLayout(3, 3));
8
9       add(new Button("κελί 1"));
10    add(new Button("κελί 2"));
11    add(new Button("κελί 3"));
12    add(new Button("κελί 4"));
13    add(new Button("κελί 5"));
14    add(new Button("κελί 6"));
15    add(new Button("κελί 7"));
16    add(new Button("κελί 8"));
17    add(new Button("κελί 9"));
18
19    pack();
20    setVisible(true);
21  }
22  public static void main(String[] args){
23    GridLayoutFrame me = new GridLayoutFrame();
24  }
25 }
```

Φυσικά μπορούν να δημιουργηθούν νέοι Διαχειριστές Πλάνων έχοντας νέους κανόνες. Η χρήση των Διαχειριστών Πλάνων δίνει το πλεονέκτημα στο γραφικό περιβάλλον επικοινωνίας να τοποθετεί τις συνιστώσες του σε σχέση μεταξύ τους χωρίς να περιορίζονται από τα εκάστοτε μεγέθη των ισοδύναμων συνιστωσών στο αντίστοιχο γραφικό λειτουργικό σύστημα.

3.3 Βοηθητικές κλάσεις (Utilities).

Εδώ ανήκουν όλες οι κλάσεις που χρησιμοποιούνται για τον καλύτερο ορισμό του γραφικού περιβάλλοντος. Π.χ.

25. `Dimension` για διαστάσεις
26. `Insets` για ένθετα στις συνιστώσες (περιθώρια)
27. `Point` για τοποθετήσεις στη θέση x, y
28. `Polygon` για αναπαράσταση πολυγωνικών σχημάτων
29. `Rectangle` για αναπαράσταση παραλληλόγραμμου

30. `Color` για αναπαράσταση χρωμάτων
31. `Font` για αναπαράσταση γραμματοσειρών
32. `Cursor` για αναπαράσταση μορφών του ποντικιού
33. `Image` για αναπαράσταση εικόνων
34. `MediaTracker` για παρακολούθηση κατάστασης πολυμέσων
35. Και πολλές άλλες

3.4 Κλάσεις χειρισμού γεγονότων.

Σ' αυτή την κατηγορία ανήκουν οι κλάσεις που χρησιμοποιούνται για των χειρισμό των ενεργειών του χρήστη και όχι μόνο. Κάθε συνιστώσα μπορεί να δεχτεί κάποιους ενδιαφερόμενους Ακροατές (listeners) για τα γεγονότα που αντιλαμβάνεται, ώστε κάθε φορά που συμβαίνει κάποια ενέργεια, να ειδοποιούνται οι ενδιαφερόμενοι Ακροατές. Ένα αντικείμενο δηλώνει ότι μπορεί να δέχεται κάποιο είδος γεγονότων με το να υλοποιεί την αντίστοιχη διασύνδεση (interface) του γεγονότος. π.χ. έστω ότι ένα αντικείμενο myObject θέλει να δέχεται τις ενέργειες του ποντικιού πάνω σε μία ετικέτα. Η διασύνδεση που πρέπει να υλοποιήσει είναι η `java.awt.event. MouseListener`. Για την υλοποίηση της διασύνδεση αυτής πρέπει να υλοποιηθούν οι εξής μέθοδοι στον Ακροατή :

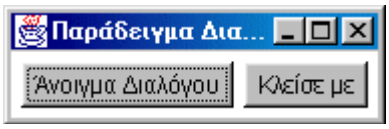
```
36.MouseClicked(MouseEvent e)
37.MouseEntered(MouseEvent e)
38.MouseExited(MouseEvent e)
39.MousePressed(MouseEvent e)
40.MouseReleased(MouseEvent e)
```

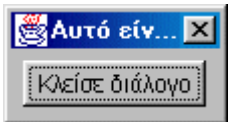
Στη συνέχεια στην ετικέτα(ή άλλη συνιστώσα) την οποία θέλουμε να παρακολουθήσουμε εκτελούμε την μέθοδο προσθήκης `MouseListener` Ακροατών, την `addMouseListener(myObject)` με παράμετρο το ενδιαφερόμενο αντικείμενο. Μετά από αυτό κάθε ενέργεια του ποντικιού που αντιστοιχεί στον `MouseListener` θα στέλνεται στο `myObject`.

Ομοίως προσθέτουμε και χειριζόμαστε οποιοδήποτε τύπο γεγονότων (ενεργειών του χρήστη) όπως :

41. WindowListener για ακρόαση παραθυρικών γεγονότων όπως κλείσιμο, άνοιγμα παραθύρου
42. ActionListener για ακρόαση ενεργειών πάνω σε συνιστώσες όπως πάτημα πλήκτρου, επιλογή από μία λίστα επιλογών (menu)
43. MouseMotionListener για ακρόαση της κίνησης του ποντικιού
44. κ.ο.κ.

Παράδειγμα χρήσης γεγονότων με Διάλογο (Dialog) και Παράθυρο (Frame).






```
1)  import java.awt.*;
2)  import java.awt.event.*;
3)
4)  public class DialogFrame extends Frame
5)     implements ActionListener, WindowListener{
6)
7)     Dialog dialog;
8)     Button openDialogButton;
9)     Button closeDialogButton;
10) Button closeFrameButton;
11)
12) public DialogFrame(){
13)    super("Παράδειγμα Διαλόγου(Dialog) και (Παραθύρου)Frame");
14)    setLayout(new FlowLayout());
15)    addWindowListener(this);
16)
17)    openDialogButton = new Button("Άνοιγμα Διαλόγου");
18)    openDialogButton.addActionListener(this);
19)    closeFrameButton = new Button("Κλείσε με");
20)    closeFrameButton.addActionListener(this);
21)
22)    add(openDialogButton);
23)    add(closeFrameButton);
24)
25)    pack();
26)    setVisible(true);
27) }
28) public void showDialog(){
29)    dialog = new Dialog(this, "Αυτό είναι ένας Διάλογος", true);
30)    dialog.setLayout(new FlowLayout());
31)    dialog.addWindowListener(this);
32)
33)    closeDialogButton = new Button("Κλείσε διάλογο");
34)    closeDialogButton.addActionListener(this);
35)
36)    dialog.add(closeDialogButton);
37)
38)    dialog.pack();
39)    dialog.setVisible(true);
40) }
41) //υλοποίηση interface ActionListener
42) public void actionPerformed(ActionEvent e){
43)    String buttonCommand = e.getActionCommand();
44)    if(buttonCommand.equals("Άνοιγμα Διαλόγου"))
45)       showDialog();
46)    else if(buttonCommand.equals("Κλείσε διάλογο"))
47)       dialog.dispose();
48)    else if(buttonCommand.equals("Κλείσε με"))
49)       processEvent(new WindowEvent(this, WindowEvent.WINDOW_CLOSING));
50) }
51) //υλοποίηση interface WindowListener
52) public void windowClosing(WindowEvent e){
53)    Window originator = e.getWindow();
54)    if(originator.equals(this)){
55)       this.dispose();
56)       System.exit(0);
57)    } else if(originator.equals(dialog))
58)       dialog.dispose();
```

```
59)     }
60)     public void windowActivated(WindowEvent e){ }
61)     public void windowDeactivated(WindowEvent e){ }
62)     public void windowDeiconified(WindowEvent e){ }
63)     public void windowClosed(WindowEvent e){ }
64)     public void windowIconified(WindowEvent e){ }
65)     public void windowOpened(WindowEvent e){ }
66)
67)     public static void main(String[] args){
68)        DialogFrame me = new DialogFrame();
69)     }
70)  }
```

3.5 Περίληψη AWT.

Το AWT πακέτο είναι μία πολύ ισχυρή βιβλιοθήκη για την δημιουργία GUIs, με δυνατότητα εκτέλεσης σε πολλαπλά γραφικά λειτουργικά συστήματα, χρησιμοποιώντας την εμφάνιση και λειτουργικότητα του τοπικού (native) περιβάλλοντος. Υπάρχουν όμως φορές, που θέλουμε μία εφαρμογή να έχει το ίδιο γραφικό περιβάλλον επικοινωνίας σε όλα τα συστήματα, δηλαδή να έχει τη δικιά του μορφή & αίσθηση (*look & feel* κατά SUN). Αυτό επιτυγχάνεται μέσο ενός άλλου πακέτου του SWING (`javax.swing`). Το πακέτο αυτό προσφέρει λειτουργίες που δεν είναι άμεσα συνδεδεμένες με το τοπικό σύστημα γι' αυτό και ονομάζεται και *lightweighted* ενώ το AWT ονομάζεται *heavyweighted*. Το μέλλον στρέφεται προς το SWING πακέτο γιατί προσφέρει πολλά περισσότερα από το AWT (είναι χτισμένο πάνω σ' αυτό), ελαχιστοποιώντας το προγραμματιστικό έργο για την δημιουργία ενός γραφικού περιβάλλοντος επικοινωνίας.

4 Ανάλυση και Σχεδίαση με UML

Η πρώτη φορά που έπρεπε να κάνω κάποια ολοκληρωμένη εφαρμογή οπότε και έπρεπε να ψάξω , να μάθω πως ξεκινάνε την ανάλυση και σχεδίαση αυτής. Βρήκα αρκετές μεθοδολογίες άλλες περιγράφανε το πως θα γίνει, άλλες περιγράφανε το πως γίνει αλλά και πως αυτό θα περιγραφεί, και άλλες μόνο το πως θα περιγραφεί. Για κάποιον που ξεκινάει θα ήταν ίσως ποιο εύκολο να ξεκινήσω με κάποια μεθοδολογία που έχει και τον τρόπο και τα σύμβολα. Δεν διάλεξα αυτό τον τρόπο όμως γιατί αυτό που σίγουρα ξέρω είναι το να ακολουθήσω χωρίς προηγούμενη εμπειρία κάποια μεθοδολογία θα σήμαινε πως δεν θα έφτανα ποτέ στην σχεδίαση – υλοποίηση μέχρι να μάθω τον συγκεκριμένο τρόπο και φυσικά να τον προσαρμόσω στα μέτρα μου. Η καλύτερη λύση που επέλεξα ήταν να διαλέξω μία μεθοδολογία που μας δίνει τα σύμβολα – εργαλεία να αποτυπώσουμε τις ιδέες μας χωρίς να μας περιορίζει στον τρόπο που θα προχωρήσουμε με την ανάπτυξη της εφαρμογής. Διάλεξα την UML (Unified Modeling Language).

Μην επιλέγοντας μεθοδολογία φυσικά είναι σαν να επέλεξα αυτόματα την
- Ανάλυση
- Σχεδίαση
-Υλοποίηση

σαν τα βήματα της ανάπτυξης. Και έτσι «θεωρητικά» προχώρησα.

4.1 Πρώτο βήμα στην ανάλυση – Λίστα απαιτήσεων

Το πρώτο βήμα είναι να περιγράψουμε τι ακριβώς θα δώσουμε σαν τελική εφαρμογή. Δεν χρειάζεται ειδικά σύμβολα και είναι απλά μία λίστα από τίτλους με όλες τις λειτουργίες και απαιτήσεις που πρέπει να πληρεί η εφαρμογή για να θεωρηθεί ολοκληρωμένη.

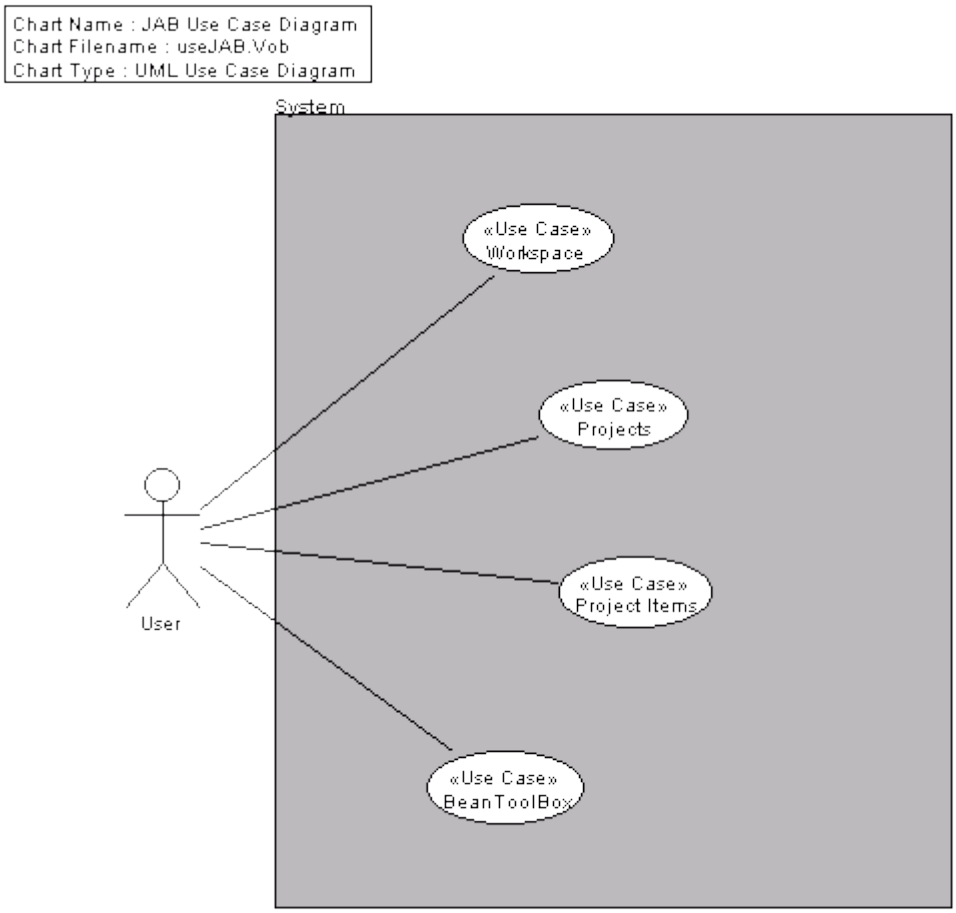
Λίστα απαιτήσεων

1. Χρήση γλώσσας προγραμματισμού JaVa
2. Παραθυρικό περιβάλλον
3. Διαχείριση αρχείων – έργων. (* έργα είναι οι εφαρμογές που θα αναπτύσσονται μέσω του εργαλείου που αναπτύσσουμε)
    a. Δημιουργία έργου
    b. Αποθήκευση έργου
    c. Άνοιγμα έργου
    d. Προσθήκη αντικειμένων στο έργο
4. Επεξεργασία κώδικα έργου.
5. Οπτική σχεδίαση οθονών έργου.
6. Ύπαρξη παλέτας οπτικών αντικειμένων που να μπορούν να προστεθούν στα οπτικά αντικείμενα που βρίσκονται υπό σχεδίαση.

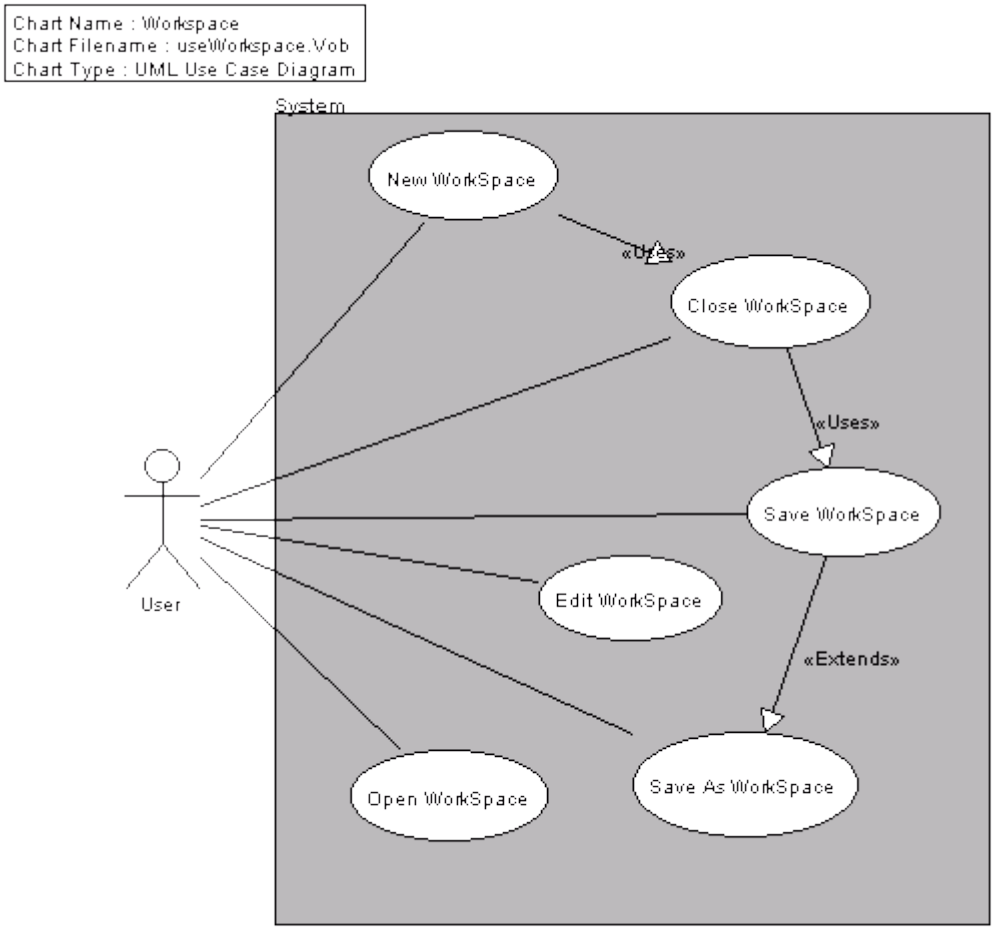
7. Προβολή ιδιοτήτων των οπτικών αντικειμένων που βρίσκονται υπό σχεδίαση.
8. Μεταγλώττιση έργου μέσα από το εργαλείο που αναπτύσσουμε.
9. Εκτέλεση έργου μέσα από το εργαλείο που αναπτύσσουμε.
10. Βοηθητικά εργαλεία (Wizards) δημιουργίας οπτικών αντικειμένων.
11. Ανοικτή αρχιτεκτονική ώστε να δέχεται πρόσθετες εφαρμογές.
12. Δημιουργία μίας πρόσθετης εφαρμογής ως παράδειγμα.
13. Δυναμική αλλαγή – παραμετροποίηση της οπτικής εμφάνισης της εφαρμογής.

4.2 Διαγράμματα χρήσης εφαρμογής – Use Cases

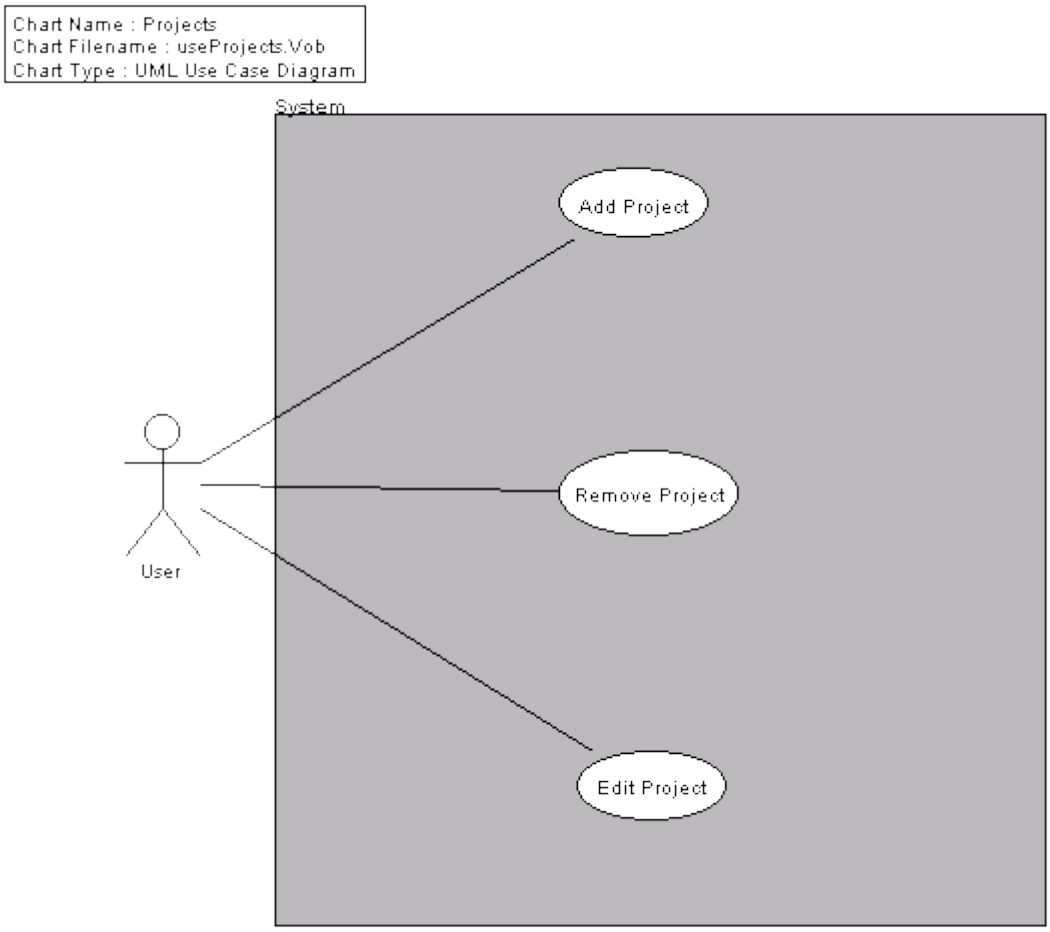
Γενικό διάγραμμα εφαρμογής



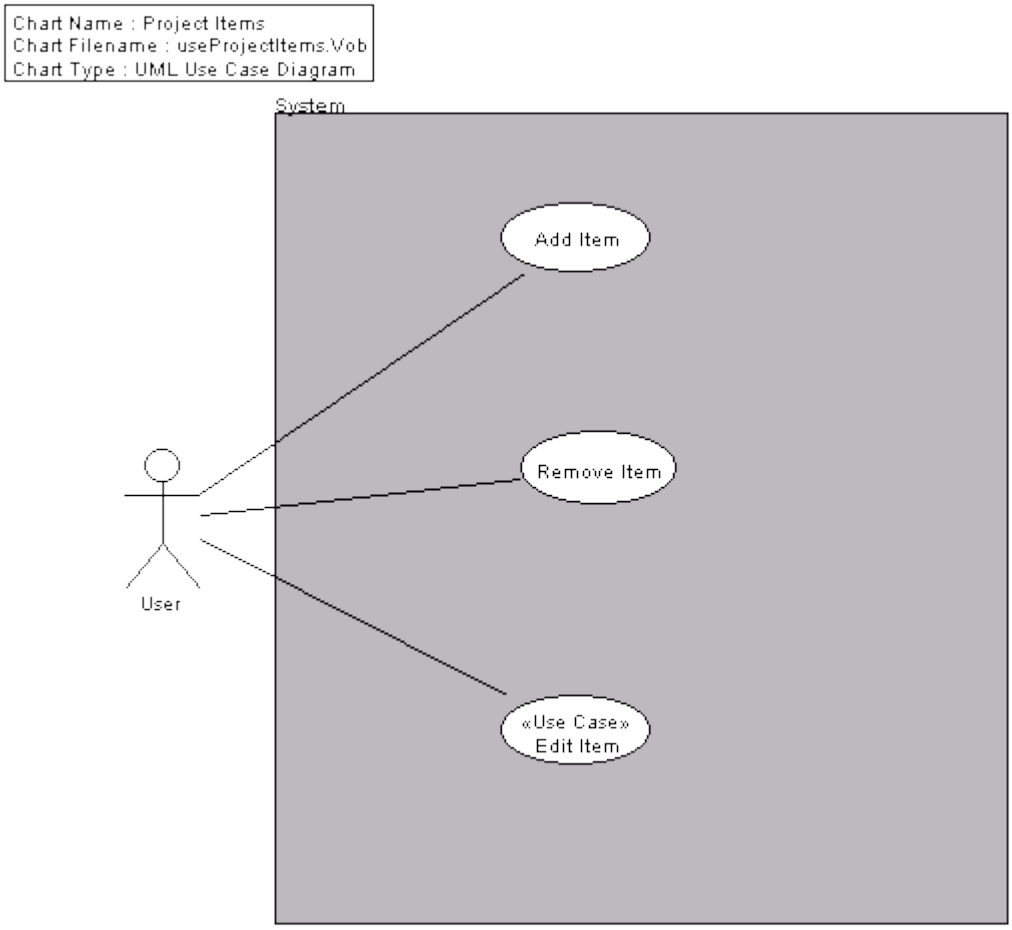
Διάγραμμα Χώρου Εργασίας - WorkSpace



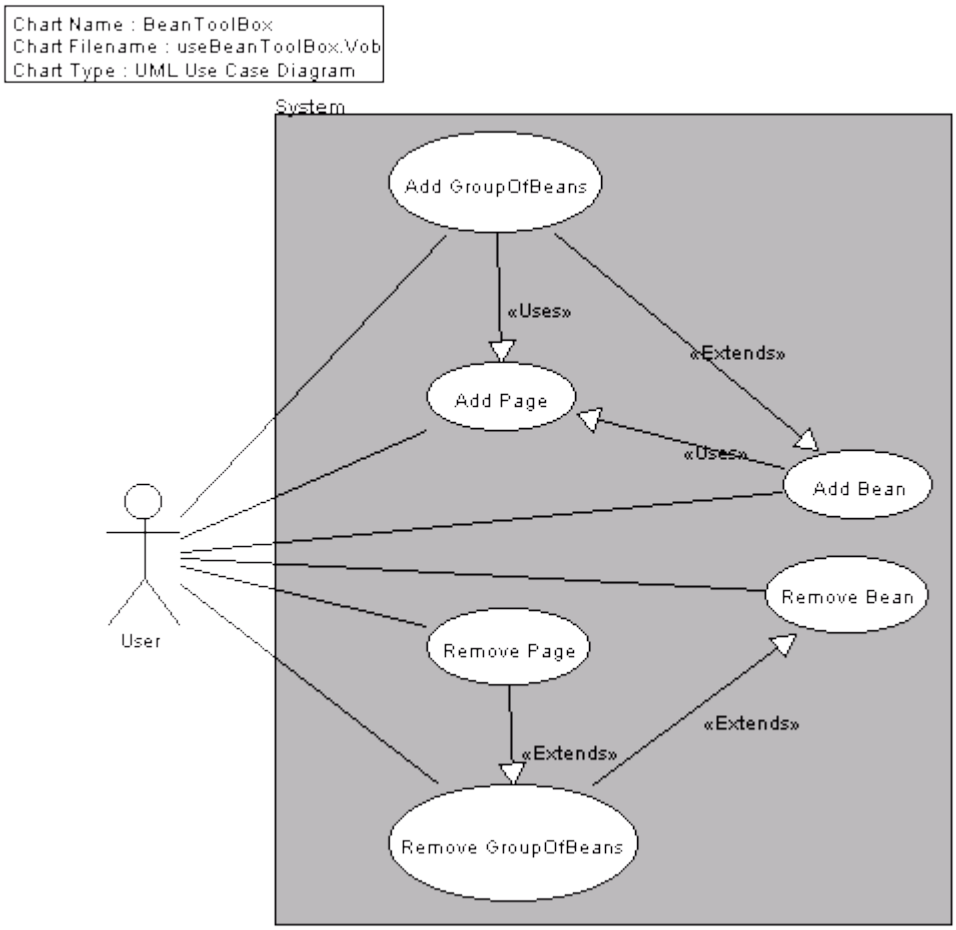
Διάγραμμα χρήσης Έργου



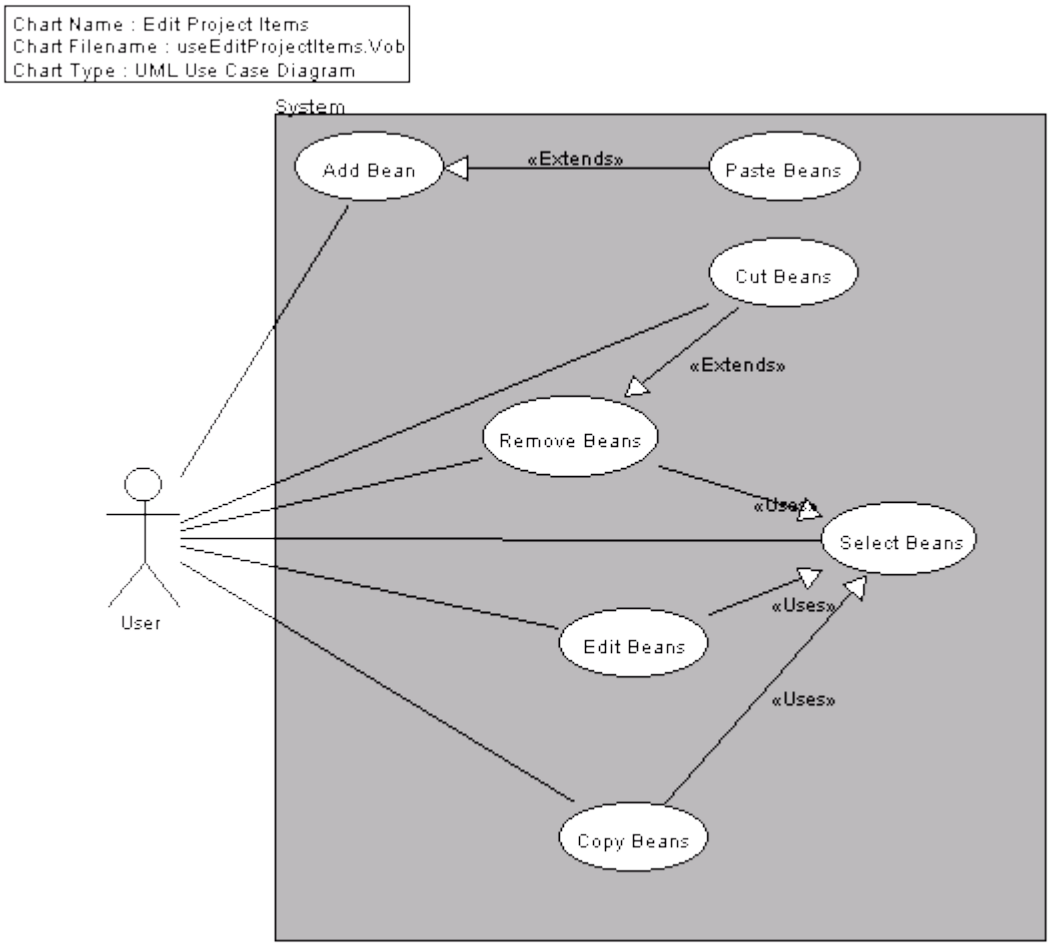
Διάγραμμα χρήσης αντικειμένων έργου



Διάγραμμα χρήσης μπάρας οπτικών συστατικών (JavaBeans)

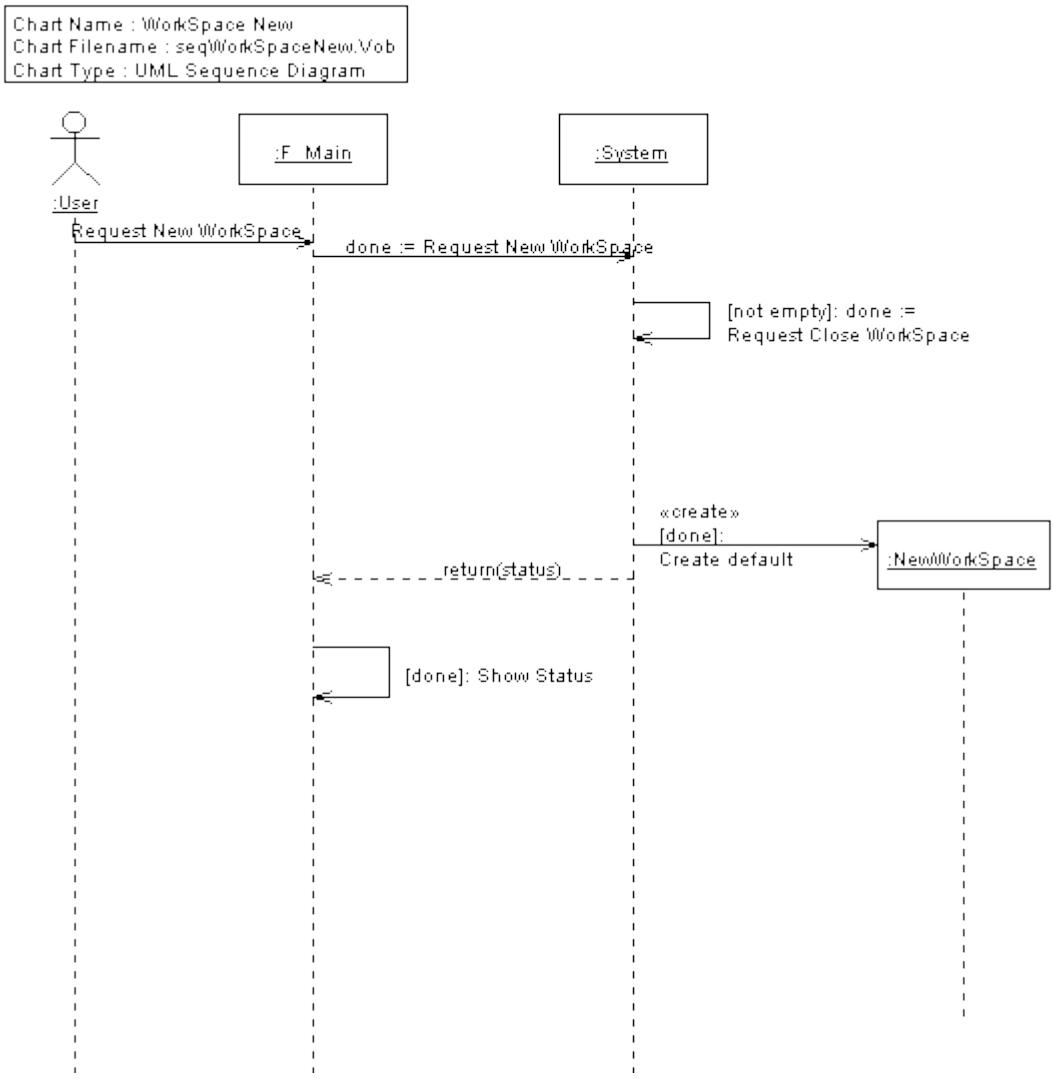


Διάγραμμα οπτικής επεξεργασίας οπτικών συστατικών

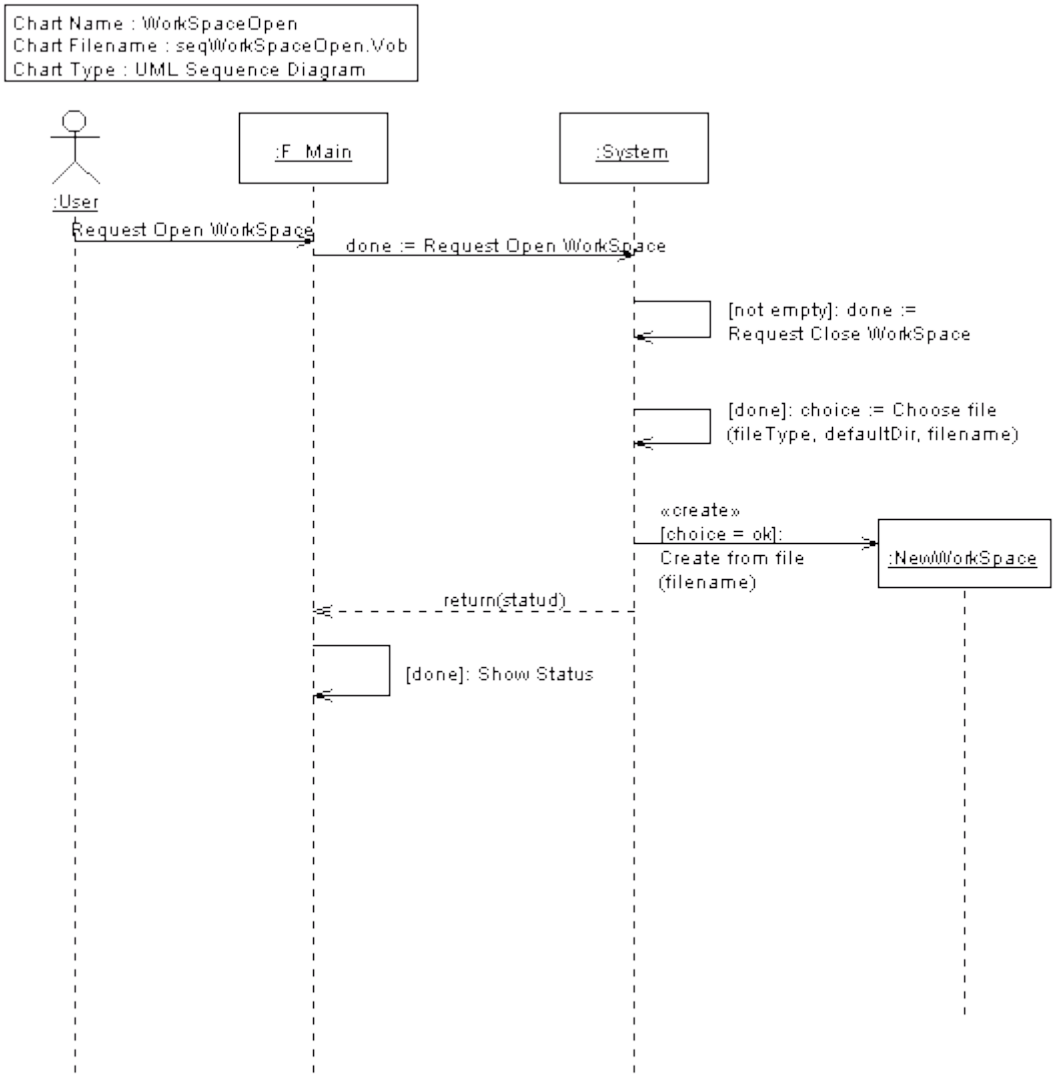


4.3 Διαγράμματα αλληλουχίας ενεργειών – Sequence Diagrams

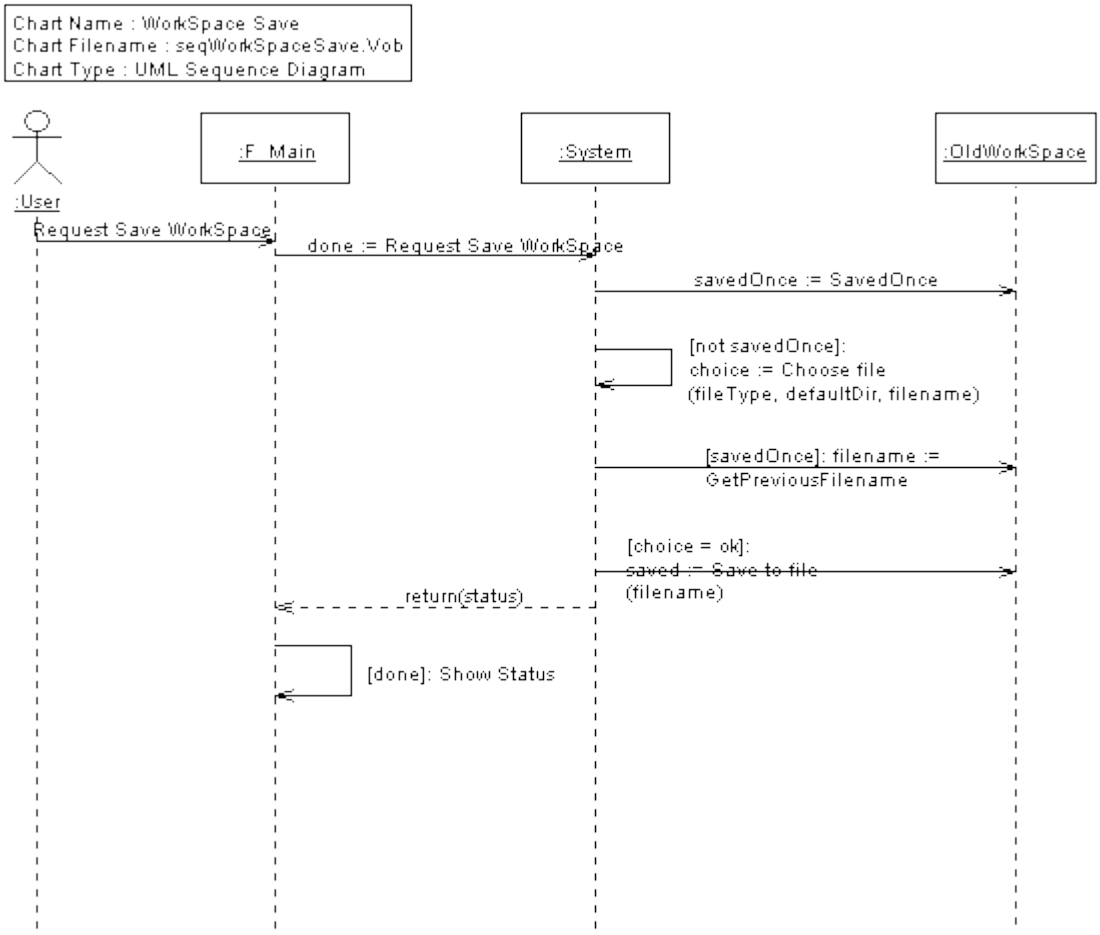
Διάγραμμα ενεργειών λειτουργίας Δημιουργία νέου χώρου εργασίας



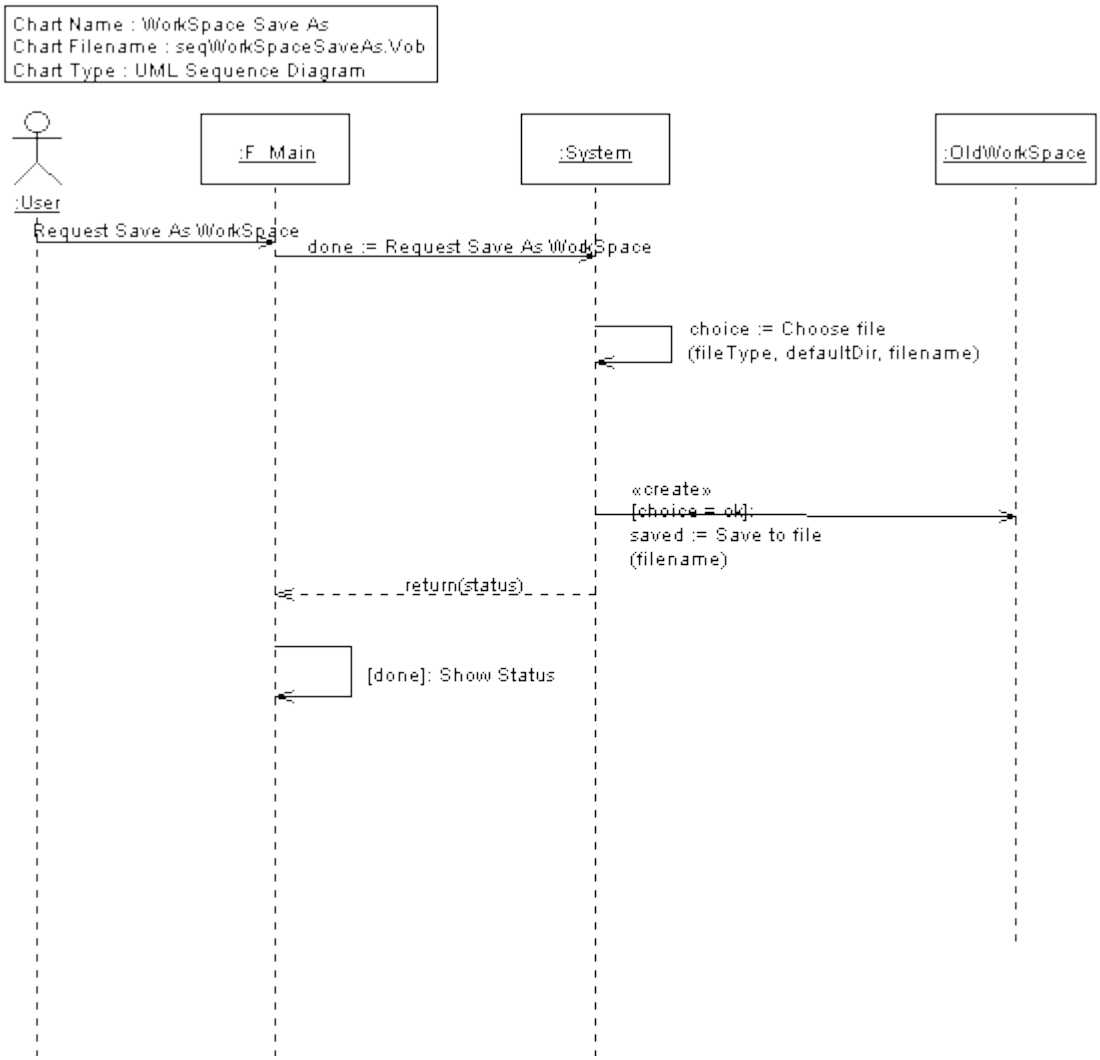
ενεργειών λειτουργίας Άνοιγμα υπάρχοντος χώρου εργασίας



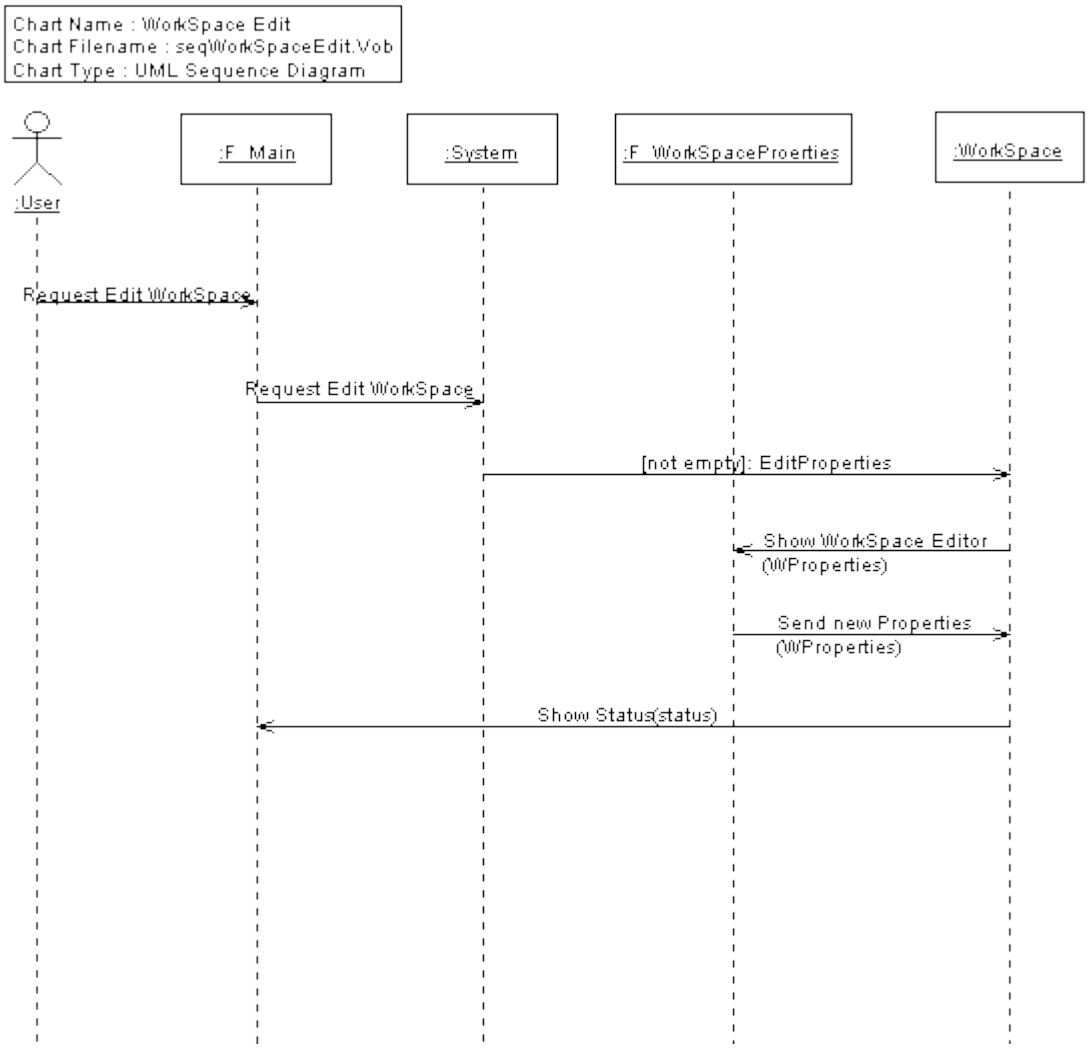
Διάγραμμα ενεργειών λειτουργίας Αποθήκευση χώρου εργασίας



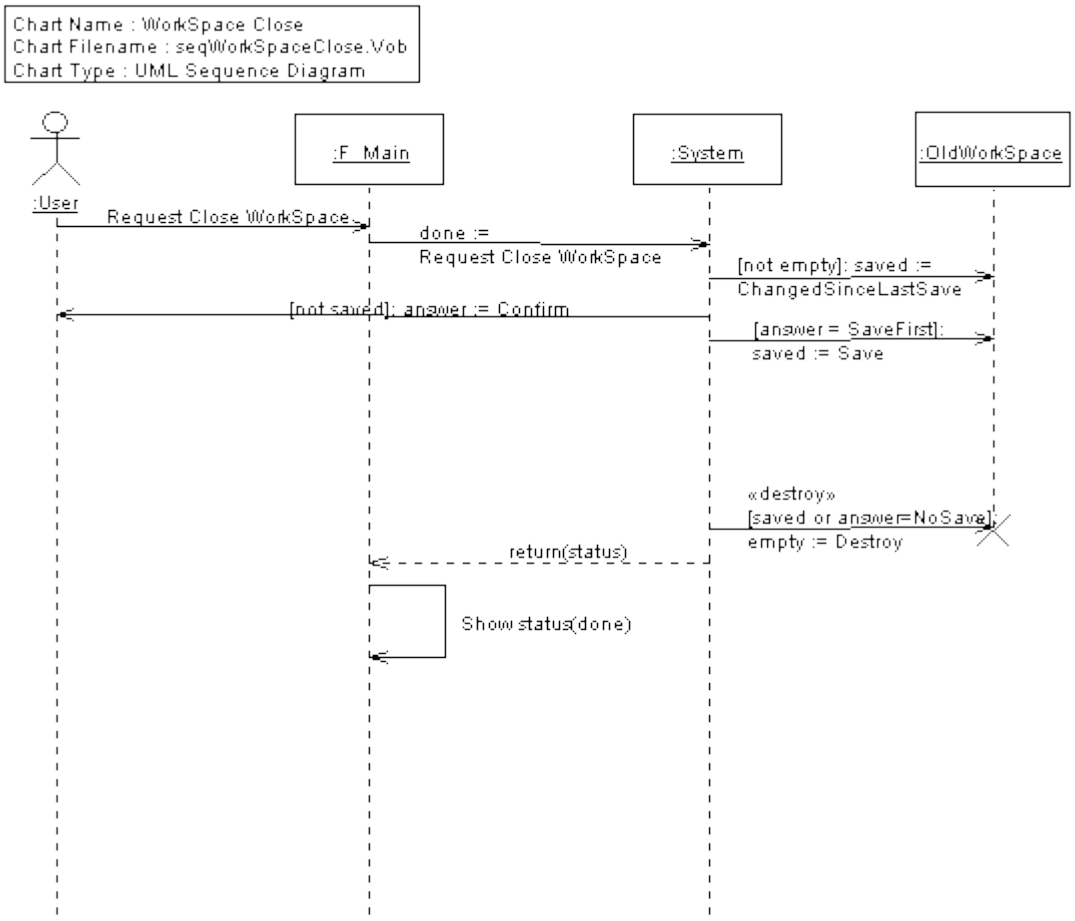
Διάγραμμα ενεργειών λειτουργίας Χώρου εργασίας με άλλο όνομα αρχείου



Διάγραμμα ενεργειών λειτουργίας Ιδιότητες χώρου εργασίας

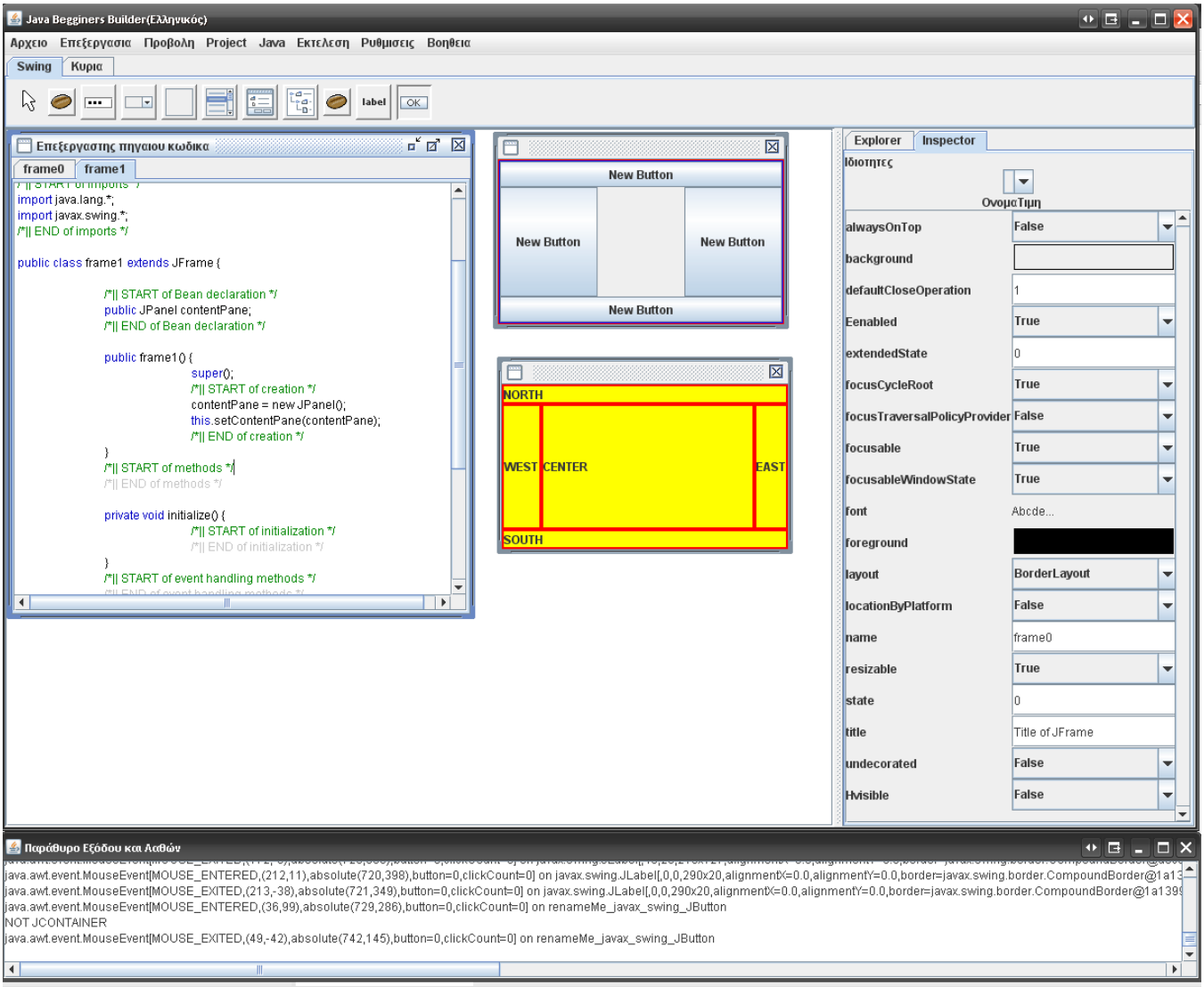


Διάγραμμα ενεργειών λειτουργίας Κλείσιμο χώρου εργασίας



5 Εγχειρίδιο

Κεντρική οθόνη εφαρμογής



Βλέπουμε πως η κεντρική διεπαφή της εφαρμογής είναι της μορφής Πολλαπλών Οθονών σε ένα παράθυρο (MDI: Multi Document Interface). Αποτελείται από:

1. Κεντρικό μενού λειτουργιών
2. Μπάρα αντικειμένων που μπορούν να προστεθούν σε νέα παράθυρα
3. Επεξεργαστή πηγαίου κώδικα
4. Οπτική επεξεργασία διεπαφής
5. Επεξεργάσιμη λίστα ιδιοτήτων επιλεγμένου αντικειμένου.
6. Λίστα αντικειμένων χώρου εργασίας
7. Διεπαφή εμφάνισης σχολίων εκτέλεσης εφαρμογής – για αποσφαλμάτωση.

5.1 Κεντρικό μενού λειτουργιών

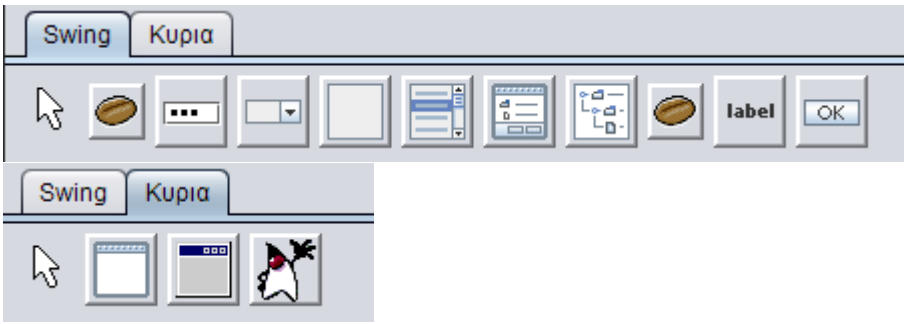
Το κεντρικό μενού αποτελείται από τις εξής λειτουργίες:

- **Αρχείο**
  - Νέο
    - New File
    - New DB Form
  - Άνοιγμα
  - Αποθήκευση
  - Αποθήκευση ως
  - Κλείσιμο
  - ----------
  - Νέος χώρος εργασίας
  - Άνοιγμα χώρου εργασίας
  - Αποθήκευση Χώρου εργασίας
  - Αποθήκευση ως Χώρου εργασίας
  - Κλείσιμο χώρου εργασίας
  - ----------
  - Εκτύπωση
  - ----------
  - Έξοδος
- Επεξεργασία
  - Αναίρεση
- Προβολή
  - Πληροφορίες κλάσης
- Project
  - Νέο
  - Άνοιγμα
  - Αποθήκευση
  - Αποθήκευση ως
  - Κλείσιμο

    - Αφαίρεση
- Java
    - Επιλογές
- Εκτέλεση
    - Μεταγλώττιση
    - Χτίσιμο όλων
    - Χτίσιμο αλλαγμένων
    - Εκτέλεση
    - ----------
    - Αποσφαλμάτωση
    - ----------
    - Επιλογές
- Ρυθμίσεις
    - Γενικές επιλογές
    - Πρόσθετα
        - Εμφάνιση μνήμης
    - Look & Feel
        - Metal
        - Nimbus
        - CDE/Motif
        - Windows
        - Windows Classic
- Βοήθεια
    - Βοήθεια
    - ----------
    - Σχετικά

Αρκετές από αυτές δεν έχουν ολοκληρωθεί και οπότε παραμένουν ανενεργές.
Κάποιες ενεργοποιούνται όταν πληρούνται κάποιες προϋποθέσεις.

5.2 Μπάρα αντικειμένων που μπορούν να προστεθούν σε νέα παράθυρα

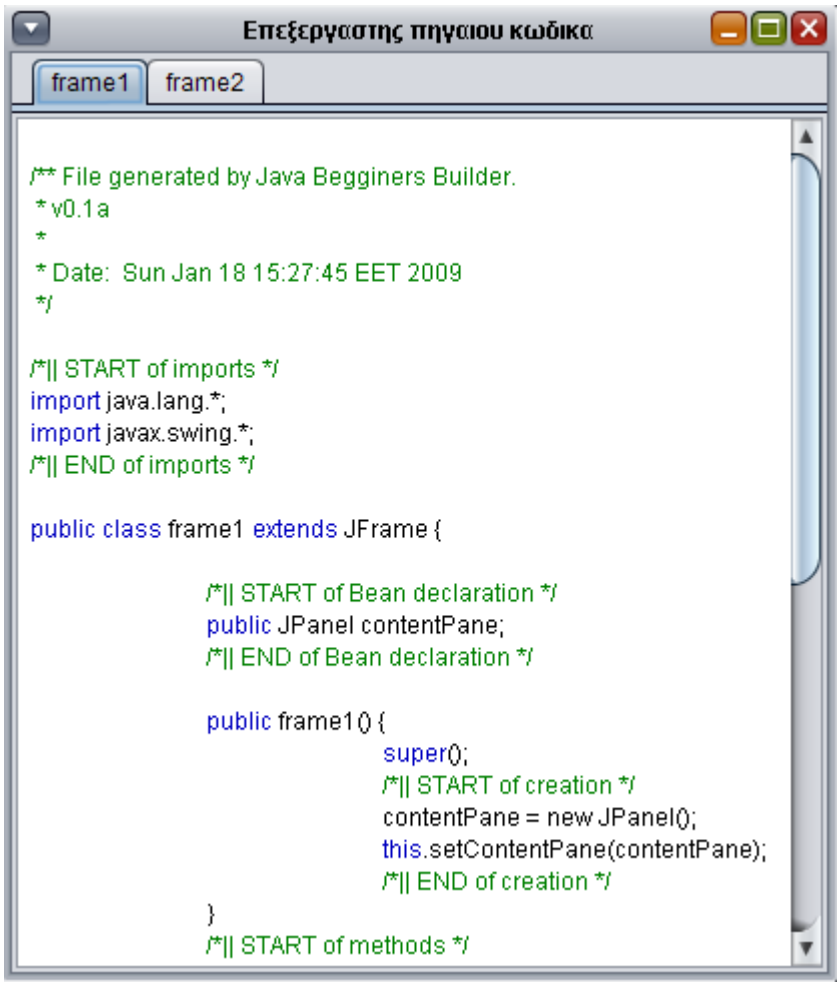


Στις μπάρες αυτές προστίθενται οι κλάσεις οι οποίες υποστηρίζονται από το πρόγραμμα και έχουν υλοποιήσει τις διεπαφές κώδικα (Interfaces) που χρειάζεται για να χαρακτηριστούν ως Beans. Μία διεπαφή είναι και η BeanInfo.

Ένα Bean μπορεί να χαρακτηριστεί ως ένα αυτόνομο αντικείμενο που περιγράφει τον τρόπο που μπορεί να χρησιμοποιηθεί από άλλο κώδικα ή από κάποιο σχεδιαστικό οπτικών διεπαφών χρήστη.

Τα έχω χωρίσει σε δύο κατηγορίες λόγο του ότι δεν έχω βάλει και μεγάλο αριθμό από το σύνολο που υπάρχουν. Σε αυτά που είναι αρχικοί αποδέκτες άλλων συστατικών: Jframe, JinternalFrame, Japplet.

5.3 Επεξεργασία πηγαίου κώδικα



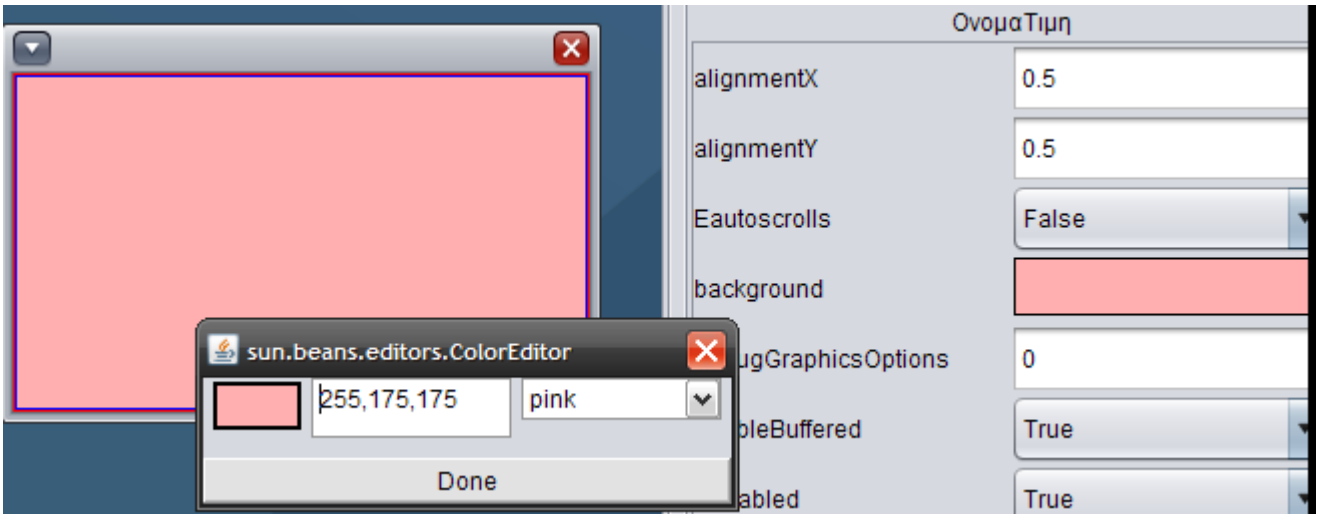
Όταν δημιουργούμε ένα νέο χώρο εργασίας αυτόματα δημιουργούνται και δύο διεπαφές χρήστη για σχεδίαση. Αν τις ανοίξουμε από την διεπαφή Λίστα αντικειμένων χώρου εργασίας αυτόματα θα ανοίξει και ο κώδικας που έχει γίνει generate για την συγκεκριμένη διεπαφή.

Ο επεξεργαστής κειμένου καταλαβαίνει κάποιες ειδικές λέξεις κώδικα και τις χρωματίζει ανάλογα. Επίσης καταλαβαίνει τα σχόλια τις java όπως και τα ειδικά σχόλια αρχή και τέλους των σημείων που περιέχουν τον αυτόματα παραγόμενο κώδικα. (/*||)

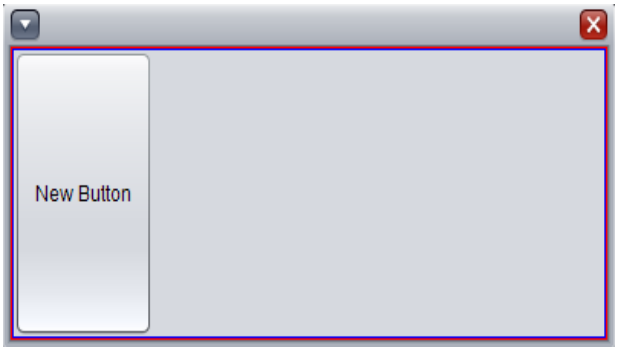
5.4 Οπτική επεξεργασία διεπαφής

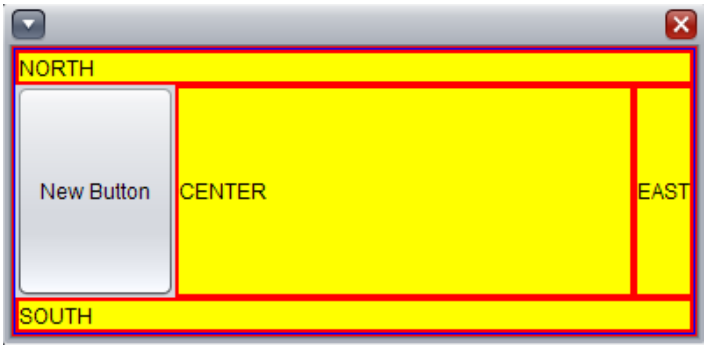
Η οπτική επεξεργασία διεπαφών έχει γίνει έτσι ώστε να είναι εύκολο να τοποθετούνται νέα συστατικά γνωρίζοντας και βλέποντας το σημείο τοποθέτησης με κίτρινο και κόκκινο περίγραμμα. Επίσης με την χρήση της λίστας επεξεργαζόμενων ιδιοτήτων του επιλεγμένου οπτικού συστατικού είναι εύκολες οι όποιες αλλαγές.

Επιλέγοντας το εσωτερικό συστατικό από ένα παράθυρο μπορούμε να αλλάξουμε κάποιο από τις ιδιότητες του. Στην παρακάτω εικόνα δείχνουμε πως αλλάζει το “background”



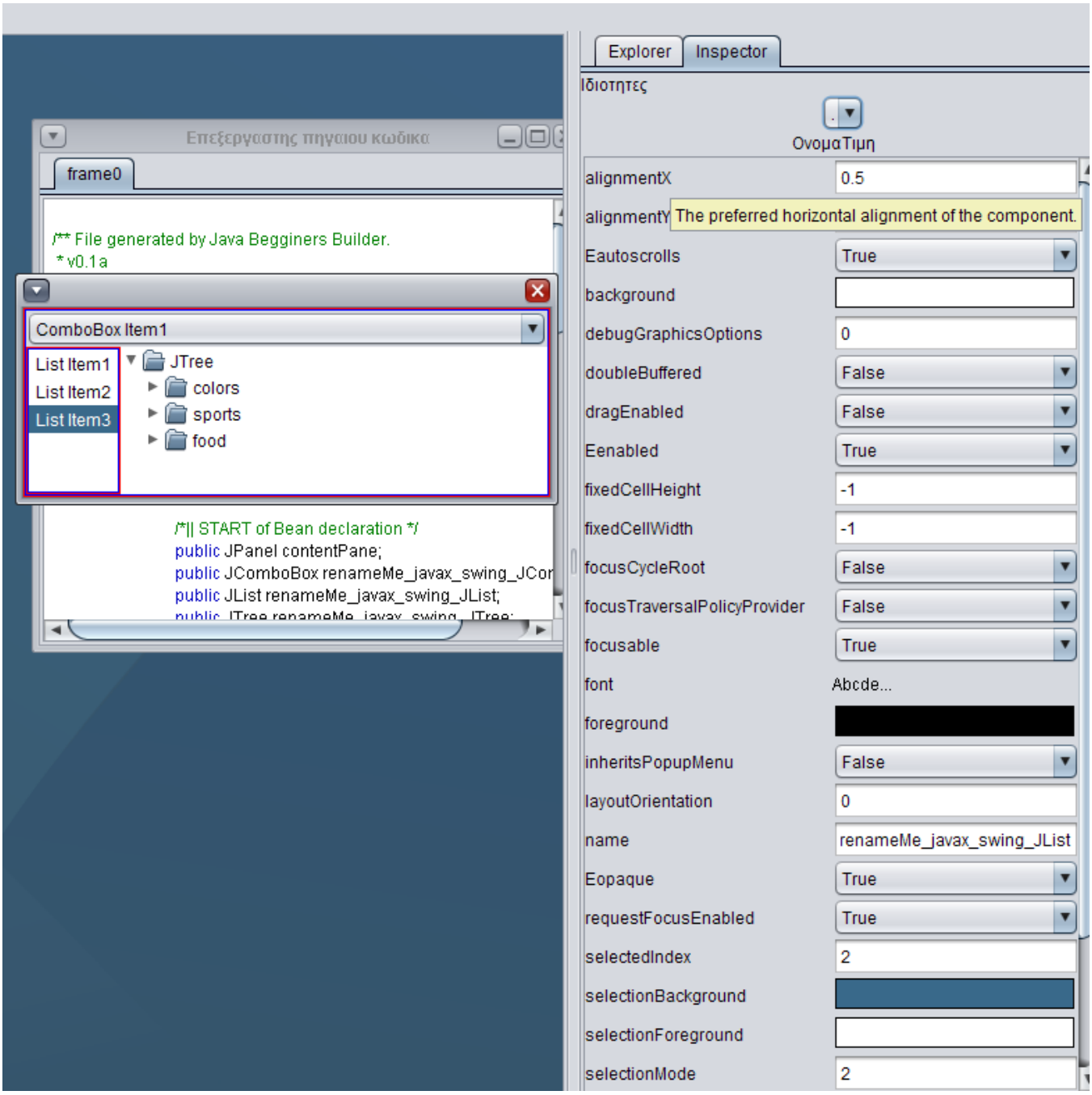
Στις επόμενες εικόνες βλέπουμε πως προσθέτουμε ένα νέο αντικείμενο – συστατικό. Όταν επιλέξουμε από την μπάρα συστατικών και πάμε τον δείκτη του ποντικιού πάνω στην υπό σχεδίαση διεπαφή ανάλογα με το τη “layout” έχει ως ιδιότητα τότε το πρόγραμμα μας δείχνει τις πιθανές θέσεις που μπορεί να τοποθετηθεί το νέο συστατικό.





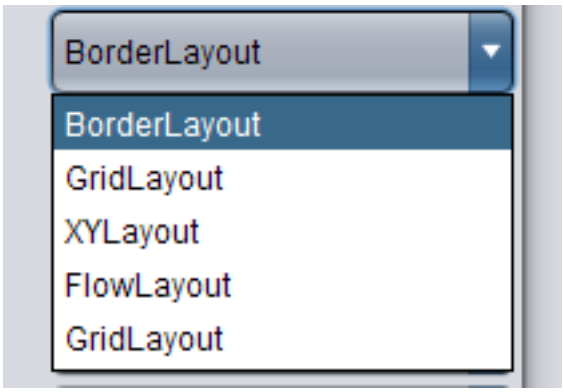
Λόγο του ότι κάποιες ιδιότητες δημιουργούν πρόβλημα αν δοθούν πραγματικά στα συστατικά έχει φτιαχτεί ένα ενδιάμεσο συστατικό “βιτρίνα” (proxy) όπου αποθηκεύει αυτές τις τιμές χωρίς να τις εκχωρεί πραγματικά στα ίδια τα συστατικά αλλά να δημιουργεί τις τιμές αυτές ώστε να ανατεθούν την ώρα της εκτέλεσης.
(π.χ. Η ιδιότητα “Visible” είναι μια τέτοια περίπτωση. Κατά την σχεδίαση θέλουμε το συστατικό πάντα να είναι εμφανίσιμο αλλά αν έχει “Visible”= false να μην εμφανίζεται κατά την εκτέλεση)

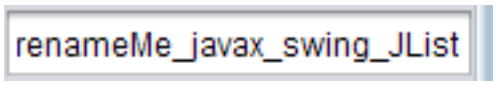
5.5 Επεξεργάσιμη λίστα ιδιοτήτων επιλεγμένου αντικειμένου

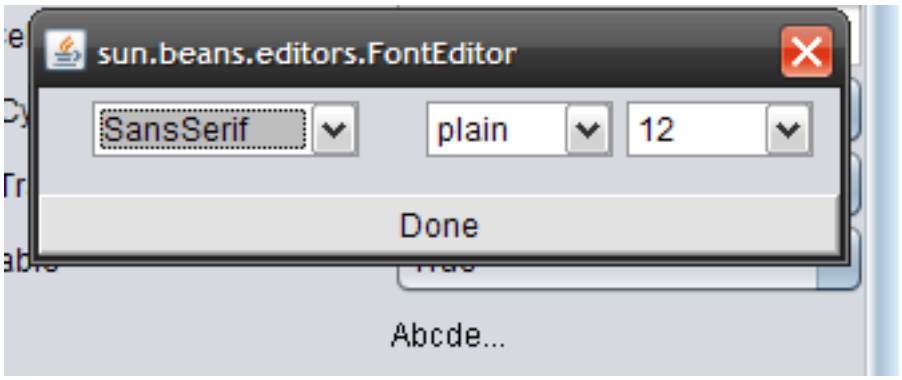


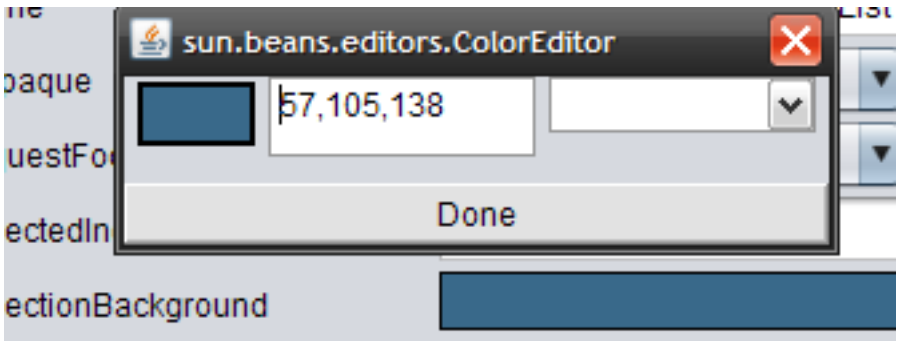
Σε κάθε οπτικό προγραμματισμό δεν λείπει φυσικά και η λίστα ιδιοτήτων ενός επιλεγμένου οπτικού αντικειμένου. Η λίστα αυτή ανάλογα με τις “public” ιδιότητες του επιλεγμένου αντικειμένου εμφανίζει για κάθε μία ιδιότητα τον αντίστοιχο επεξεργαστή.

Παραδείγματα επεξεργαστών ιδιοτήτων για τους οποίους γράφτηκε ειδικός κώδικας ώστε να τα υποστηρίζει η λίστα ιδιοτήτων της εφαρμογής.

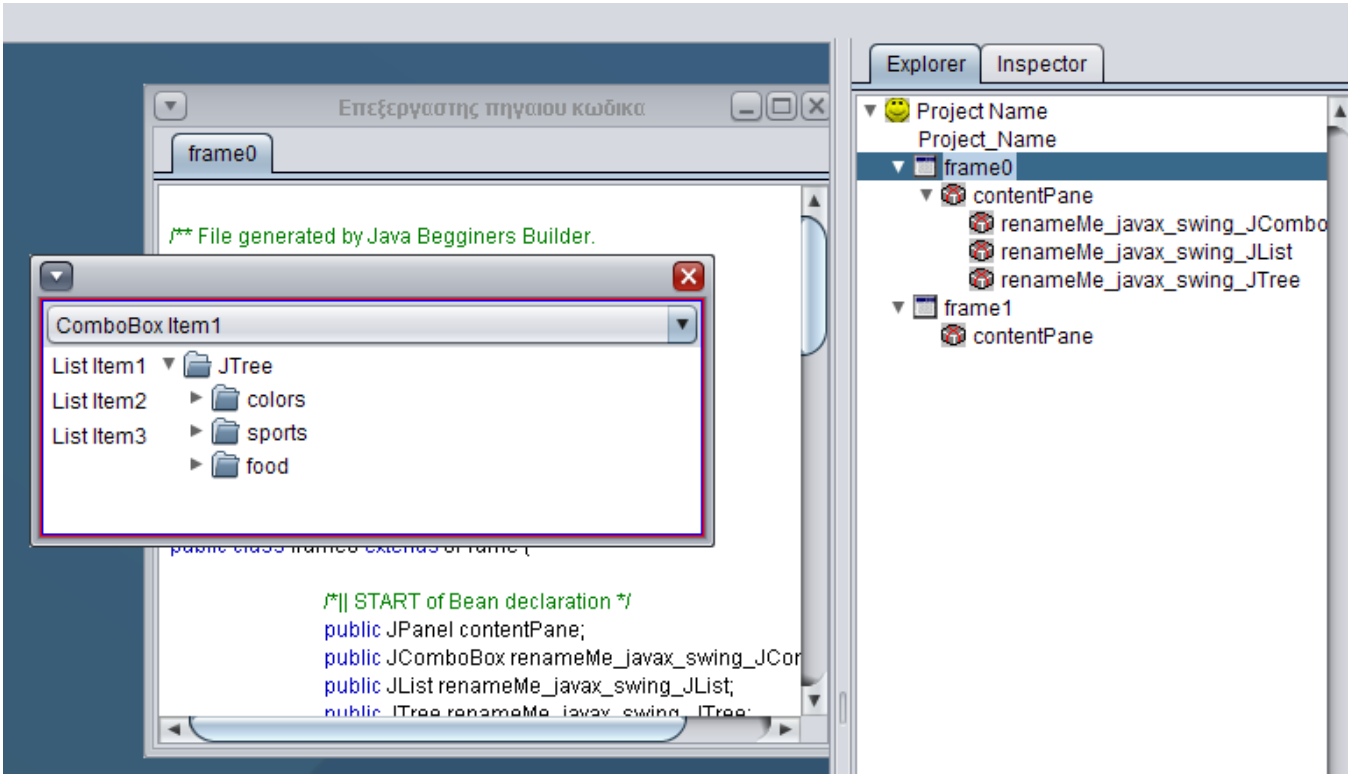








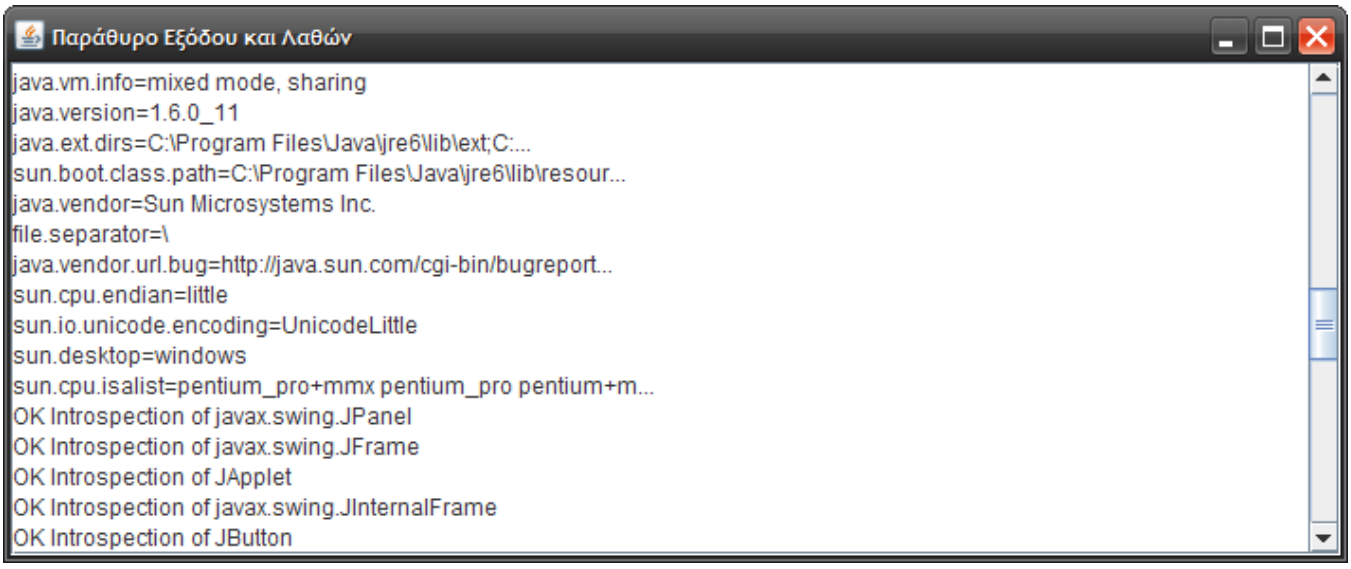
5.6 Λίστα αντικειμένων χώρου εργασίας



Στα δεξιά της κεντρικής διεπαφής υπάρχει μία λίστα που έχε σε δενδροειδή μορφή τα αντικείμενα τα οποία απαρτίζουν την υπό δημιουργία εφαρμογή. π.χ. μετά από δημιουργία ενός νέου έργου αυτόματα προστίθενται και 2 κενές φόρμες frame0 και frame1. Στην μία από αυτές πρόσθεσα τρία συστατικά ένα JCombobox, ένα JList και ένα Jtree. Το αποτέλεσμα είναι αυτό που βλέπετε και στην εικόνα.

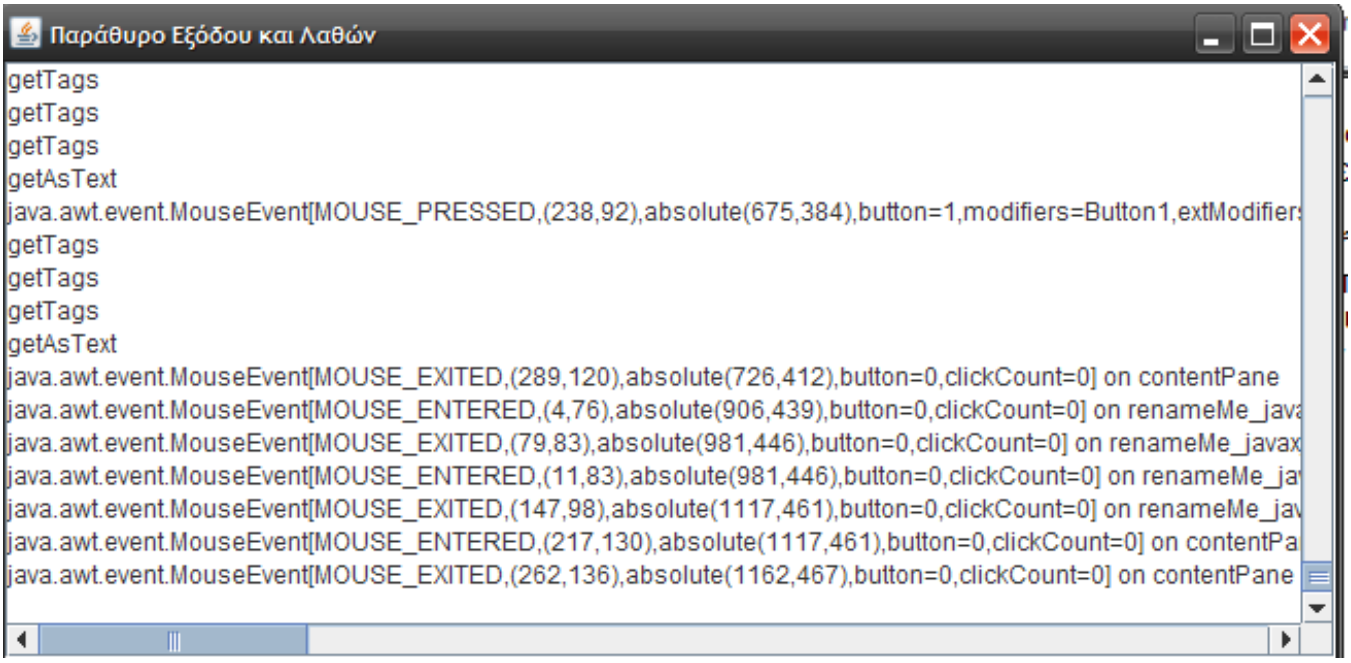
Κάθε κλαδί του δέντρου ανάλογα με το τι συστατικό είναι και αν αυτό μπορεί να περιέχει και άλλα συστατικά μπορεί να έχει αριστερά του ένα σύμβολο ώστε να εμφανίζονται ή να κρύβονται τα συστατικά αυτά.

5.7 Διεπαφή εμφάνισης σχολίων εκτέλεσης εφαρμογής – για αποσφαλμάτωση



Η καθορισμένη έξοδος της κεντρικής εφαρμογής είναι συνδεδεμένη σε αυτό το οπτικό αντικείμενο και βρίσκεται κατά το ξεκίνημα της εφαρμογής στο κάτω μέρος της οθόνης.

Λόγο της εκπαιδευτικής φύσης του όλου προγράμματος υπάρχουν διάφορα γεγονότα που παράγουν μηνύματα που περιγράφουν με λεπτομέρεια το γεγονός. Χρησιμοποιήθηκε αρκετά κατά την ανάπτυξη του ίδιου του προγράμματος γιατί κατά την συγγραφή του δεν υπήρχαν IDE που να βοηθούν στην αποσφαλμάτωση των τότε Java εφαρμογών.



6 Συμπεράσματα

Αρχικώς θα διατυπώσω το προφανές: Η εφαρμογή για να θεωρηθεί ολοκληρωμένη και χρήσιμη ως και ανταγωνιστική αντίστοιχων πλέον εργαλείων χρειάζεται πολύ δουλειά ακόμη από μία ομάδα έμπειρων μηχανικών λογισμικού.

Αυτό δεν σημαίνει όμως ότι δεν είχε την χρησιμότητα της η μέχρι το στάδιο που έφτασε ανάπτυξη της. Η πτυχιακή ξεκίνησε μία εποχή όπου η Java ήταν ακόμα νέα και πολλά υποσχόμενη χωρίς καν ακόμα να βρίσκεται στο πρόγραμμα μαθημάτων του τμήματος. Όπως έδειξε και ο χρόνος η Java καθιερώθηκε σαν η καλύτερη γλώσσα σε πολλούς τύπους εφαρμογών. Πολλές μεγάλες εταιρίες με χρόνια εμπειρίας και ήδη έμπειρο προσωπικό ανέπτυξαν IDE με δυνατότητες ανταγωνιστικές ως προς τις αντίστοιχες διανομές άλλων εταιριών. Φυσικά και το τμήμα Πληροφορικής πολύ γρήγορα υιοθέτησε την Java και πρόσθεσε αντίστοιχα πολλά μαθήματα σχετικά με την Java.

Για ένα προγραμματιστή όμως το να εργαστεί στην δημιουργία του εργαλείου με το οποίο θα εργάζεται στο υπόλοιπο της ενασχόλησης του με την Πληροφορική του δίνει γνώσεις και απαιτήσεις που δύσκολα θα μπορούσε να σκεφτεί ότι χρειάζεται ή ότι μπορούν να υπάρχουν ώστε να κάνουν την εργασία του ποιο εύκολη και αποδοτική.

Θεωρώ ότι ως θέμα πτυχιακής για εκπαιδευτικούς σκοπούς υπήρξε πετυχημένο γιατί μου δίδαξε σημαντικές πλευρές της ανάπτυξης ενός λογισμικού, και λόγο της χρήσης της Java και τον αντικειμενοστραφή τρόπο υλοποίησης εφαρμογών, απολύτως απαραίτητο σήμερα σε οποιαδήποτε σχεδόν ανάπτυξη νέων εφαρμογών.

- Πριν την υλοποίηση χρειάζεται σωστή ανάλυση του θέματος και στη συνέχεια ο σχεδιασμός. Η υλοποίηση είναι ένα μικρό κομμάτι μιας εφαρμογής
- Από την αρχή μιας εφαρμογής πρέπει να γίνεται εκτίμηση του χρόνου ολοκλήρωσης ώστε να μην υπερεκτιμούμε της δυνατότητες μας και βγαίνουμε εκτός χρόνου και χωρίς πλήρες εφαρμογή χωρίς λάθη και ελλείψεις σε σημαντικές λειτουργίες.

7 Βιβλιογραφία

JAVA Tutorials

Getting Started (Digital Cat)
The Java Tutorial (Sun Microsystems)
Inside the Java Tutorial (Sun Microsystems)
Java Distributed Computing | Tutorials (Sun Microsystems)

JAVA Βιβλία

JAVA Expert Solutions *by Mark Wutka, et. al.*
JAVA By Example *by Clayton Walnum*
Developing Professional JAVA ™ Applets *by K.C. Hopson, Stephen, E. Ingram*
Creating Web Applets with JAVA ™ *by David Gulbransen, Kenrick Rawlings*
JAVA Developer's Guide *by Jamie Jaworski*
JAVA Reference Guide *by SAMS Publishing*
JAVA Unleashed *by SAMS Publishing*
Special Edition Using JAVA 1.1, Third Edition *by Joe Weber*
Tricks of the JAVA Programming Gurus *by Glenn L. Vanderburg. et al.*

JAVA References

Documentation (Sun Microsystems)
Java Distributed Computing Specs (Sun Microsystems)
Java HotSpot (Sun Microsystems)
Jini Specs (Sun Microsystems)
The Java Language Specification (Sun Microsystems)
The Java Virtual Machine Specification (Sun Microsystems)
The JIT Compiler Interface Specification (Sun Microsystems)
JDK 1.2 Documentation (Sun Microsystems)
Java Platform 1.2 API Specification (Sun Microsystems)
Java Management Documents (Sun Microsystems)
Java Code Conventions (Sun Microsystems)

ΠΑΡΑΡΤΗΜΑ Α:
Πηγαίος Κώδικας Εφαρμογής

Δέντρο καταλόγου αρχείων πηγαίου κώδικα

```
└──jbb
       JBBMain.java
   ├──beans
   │   └──DB
   │          JDBCon.java
   │          JDBConBeanInfo.java
   ├──gui
   │   │   AboutDialog.java
   │   │   FBeansBar.java
   │   │   FMain.java
   │   │   FMenuBar.java
   │   │   FProjectExplorer.java
   │   │   FPropertiesPanel.java
   │   ├──editors
   │   │       FSourceEditor.java
   │   ├──wiz
   │   │   └──db
   │   │          DWizDBForm.form
   │   │          DWizDBForm.java
   │   │          DWizDBForm.xml
   │   └──workspace
   │           CProjectExplorerCellRenderer.java
   │           FItem.java
   │           FProjectItem.java
   │           FWorkspace.java
   ├──international
   │       GlobalRes.java
   │       GlobalRes_el.java
   │       World.java
   ├──plugins
   │       PMemoryView.java
   ├──system
   │   │   CKnownBeans.java
   │   │   CSystem.java
   │   ├──editors
   │   │   │   ISourceDocument.java
   │   │   ├──html
   │   │   │       CHTMLDocument.java
   │   │   └──java
   │   │           CJavaContext.java
   │   │           CJavaDocument.java
   │   │           CJavaEditorKit.java
   │   │           CToken.java
   │   ├──io
   │   │       CFileFilter.java
   │   │       COutputFrame.java
   │   ├──layouts
   │   │       XYLayout.java
   │   ├──propertyEditors
   │   │       CCustomEditor.java
   │   │       LayoutManagerEditor.java
   │   │       PropertyButton.java
   │   │       PropertyCanvas.java
   │   │       PropertyDialog.java
   │   │       PropertySelector.java
   │   │       PropertyText.java
   │   └──workspace
   │           CItem.java
   │           CProject.java
   │           CProjectItem.java
   │           CWorkspace.java
   └──tools
       │   Runner.java
       │   RunningAppWatcher.java
       │   Tools.java
       └──compilers
               CJavaCompiler.java
```

jbb\JBBMain.java

```
package jbb;

import jbb.gui.*;
import jbb.international.*;
import jbb.system.io.*;
import jbb.tools.*;
import jbb.system.*;
import jbb.gui.wiz.db.*;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.Icon;
import java.util.Properties;
import java.beans.Beans;

import java.sql.*;
import java.util.Enumeration;

public class JBBMain {

	private static final World world = new World();
	private static COutputFrame output;

	public static void main(String[] args) {
		JFrame intro = new JFrame(World.getS("JBBTitle") + ".....");
		Icon icn = World.getIcn("JBB");
		JLabel lbl = new JLabel(icn);
		intro.getContentPane().add(lbl);
		intro.pack();
		intro.validate();
		Tools.initialize();
		intro.setLocation(Tools.getScreenCenter(intro.getSize()));
		output = new COutputFrame(false);
		intro.setVisible(true);
/* //Force JBB to come up in the System L&F
		try {

	UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
		} catch (Exception e) {
			System.err.println("Error loading System L&F: " + e);
			System.err.println("Using Cross Platform L&F instead");
		}
*/
		Properties spr = System.getProperties();
		spr.list(System.out);

		Beans.setDesignTime(true);
		Beans.setGuiAvailable(true);

		new CSystem();
//
    try     {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //loads the driver
    } catch (ClassNotFoundException e) {
System.out.println(e);
    }
System.out.println("JDBC db drivers list - start");
		for(Enumeration e = DriverManager.getDrivers(); e.hasMoreElements(); )
{
```

```
System.out.println(e.nextElement().getClass().getName());
            }
System.out.println("JDBC db drivers list - end");

//
//////////////////////////////
//////////////////////////////
            DWizDBForm wizDBForm = new DWizDBForm();
            wizDBForm.setVisible(true);// show();
/*
            intro.setVisible(true);
            try {
                  Thread.currentThread().sleep(3600000);
            } catch (InterruptedException e) {
System.out.println(e);
            }
*/
//////////////////////////////
//////////////////////////////
            intro.dispose();

      }
}
```

jbb\beans\DB\JBBCon.java

```
/*
 * Bean.java
 *
 * Created on 22 Éïýíéïò 2003, 7:40 ìì
 */

package jbb.beans.DB;

import java.beans.*;
import java.sql.*;
import java.util.*;
/**
 *
 * @author  OurHome
 */
public class JDBCon extends Object implements java.io.Serializable {

  private PropertyChangeSupport propertySupport;

  /** Holds value of property driverName. */
  private String driverName;

  /** Holds value of property DBURL. */
  private String DBURL;

  /** Holds value of property loginName. */
  private String loginName;

  /** Holds value of property loginPassword. */
  private String loginPassword;

  /** Holds value of property DBTypes. */
  private Vector DBTypes;

  /** Holds value of property DBConnection. */
  private Connection DBConnection;

  /** Holds value of property DBMetaData. */
  private DatabaseMetaData DBMetaData;

  /** Holds value of property DBDriver. */
  private Driver DBDriver;

  /** Creates new Bean */
  public JDBCon() {
    propertySupport = new PropertyChangeSupport( this );
    DBDriver = null;
    DBConnection = null;
    DBMetaData = null;
    DBTypes = new Vector();
    driverName = "sun.jdbc.odbc.JdbcOdbcDriver";
    //DBURL = "jdbc:odbc:xxxxx_odbc_name_xxxx";
    loginName = "";
    loginPassword = "";

    DBURL = "jdbc:odbc:JDBC_TEST_DS";
  }
```

```
  public void addPropertyChangeListener(PropertyChangeListener listener) {
    propertySupport.addPropertyChangeListener(listener);
  }

  public void removePropertyChangeListener(PropertyChangeListener listener) {
    propertySupport.removePropertyChangeListener(listener);
  }

  /** Getter for property connected.
   * @return Value of property connected.
   *
   */
  public boolean isConnected() throws SQLException {
    boolean b = ((DBConnection != null) && !(DBConnection.isClosed()));
    return b;
    //return this.connected;
  }

  /** Setter for property connected.
   * @param connected New value of property connected.
   *
   */
  public void setConnected(boolean connected) throws ClassNotFoundException,
SQLException {
    if (connected) {
      if (!isConnected()) {
        DoConnect();
      }
    } else {
      if (isConnected()) {
        DoDisconnect();
      }
    }
  }

  /** Getter for property driverName.
   * @return Value of property driverName.
   *
   */
  public String getDriverName() {
    return this.driverName;
  }

  /** Setter for property driverName.
   * @param driverName New value of property driverName.
   *
   */
  public void setDriverName(String driverName) {
    SilentClose();
    this.driverName = driverName;
  }

  /** Getter for property DB_URL.
   * @return Value of property DB_URL.
   *
   */
  public String getDBURL() {
    return this.DBURL;
  }

  /** Setter for property DB_URL.
```

```
     * @param DB_URL New value of property DB_URL.
     *
     */
    public void setDBURL(String DBURL) {
      SilentClose();
      this.DBURL = DBURL;
    }

    /** Getter for property loginName.
     * @return Value of property loginName.
     *
     */
    public String getLoginName() {
      return this.loginName;
    }

    /** Setter for property loginName.
     * @param loginName New value of property loginName.
     *
     */
    public void setLoginName(String loginName) {
      SilentClose();
      this.loginName = loginName;
    }

    /** Getter for property loginPassword.
     * @return Value of property loginPassword.
     *
     */
    public String getLoginPassword() {
      return this.loginPassword;
    }

    /** Setter for property loginPassword.
     * @param loginPassword New value of property loginPassword.
     *
     */
    public void setLoginPassword(String loginPassword) {
      SilentClose();
      this.loginPassword = loginPassword;
    }

    /** Indexed getter for property DBTypes.
     * @param index Index of the property.
     * @return Value of the property at <CODE>index</CODE>.
     *
     */
    public Vector getDBTypes() {
      return this.DBTypes;
    }

    public void Open() throws ClassNotFoundException, SQLException {
      setConnected(true);
    }

    public void Close() throws ClassNotFoundException, SQLException {
      setConnected(false);
    }

    protected void DoConnect() throws ClassNotFoundException, SQLException {
      //try {
```

```
    if ((driverName != null) && (driverName != "")) {
      //try     {
        //loads and registers the driver
        Class.forName(driverName);
      //} catch (ClassNotFoundException e) {
      //  System.err.println(e);
      //}
    }
    DBDriver = DriverManager.getDriver(DBURL);
    //todo: set properties on DBDriver and then
    //connect using the DBDriver
    DBConnection = DriverManager.getConnection(DBURL, loginName, loginPassword);
    Refresh();
    propertySupport.firePropertyChange("connected", false, true);
  //} catch (SQLException e) {
  //  System.err.println(e);
  //}
}

/** Getter for property connection.
 * @return Value of property connection.
 *
 */
public Connection getDBConnection() {
  return this.DBConnection;
}

/** Getter for property DBMetaData.
 * @return Value of property DBMetaData.
 *
 */
public DatabaseMetaData getDBMetaData() {
  return this.DBMetaData;
}

/** Getter for property DBDriver.
 * @return Value of property DBDriver.
 *
 */
public Driver getDBDriver() {
  return this.DBDriver;
}

protected void DoDisconnect() throws SQLException {
  //try {
    DBConnection.close();
    DBConnection = null;
    //a refresh could clear the metadata but its better not to!
    //Refresh();
    propertySupport.firePropertyChange("connected", true, false);
  //} catch (SQLException e) {
  //  System.err.println(e);
  //}
}

public void Refresh() throws SQLException {
  DBMetaData = null;
  DBTypes.clear();
  if (isConnected()) {
    DBMetaData = DBConnection.getMetaData();
    ResultSet rs = DBMetaData.getTableTypes();
```

```
      if (rs!=null) {
        String s;
        while(rs.next()){
          s = rs.getString("TABLE_TYPE");
          if (!DBTypes.contains(s))
            DBTypes.add(s);
        }
        //rs.close();
        //rs = DBMetaData.getTables(null, null, null, null);
        //displayResults(rs);

        rs.close();
      }
    }
  }

 static void displayResults(ResultSet r) throws SQLException {
  //r.beforeFirst();
  ResultSetMetaData rmeta = r.getMetaData();
  // Use meta data to obtain the number of columns
  int numColumns=rmeta.getColumnCount();
  String text="";
  for(int i=1;i<=numColumns;++i) {
   if(i<numColumns)
    text+=rmeta.getColumnName(i)+" | ";
   else
    text+=rmeta.getColumnName(i);
  }
  text+="\n";
  String s;
  while(r.next()){
   for(int i=1;i<=numColumns;++i) {
    if(i<numColumns)
     text+=r.getString(i)+" | ";
    else {
                  s = r.getString(i);
                  if (!r.wasNull()) s.trim();
      text+=s;
    }
   }
   text+="\n";
  }
System.out.println(text);
 }

  private void SilentClose() {
    try {
      setConnected(false);
    } catch (Exception e) {
      e.getLocalizedMessage();
      e.getMessage();
      e.printStackTrace();
    }
  }

}
```

jbb\beans\DB\JBBConBeanInfo.java

```
package jbb.beans.DB;

import java.beans.*;

public class JDBConBeanInfo extends SimpleBeanInfo {

    // Bean descriptor information will be obtained from introspection.//GEN-
FIRST:BeanDescriptor
    private static BeanDescriptor beanDescriptor = null;
    private static BeanDescriptor getBdescriptor(){
//GEN-HEADEREND:BeanDescriptor
    // Here you can add code for customizing the BeanDescriptor.

         return beanDescriptor;     } //GEN-LAST:BeanDescriptor

     // Properties information will be obtained from introspection.//GEN-
FIRST:Properties
    private static PropertyDescriptor[] properties = null;
    private static PropertyDescriptor[] getPdescriptor(){//GEN-HEADEREND:Properties

    // Here you can add code for customizing the properties array.

        return properties;     } //GEN-LAST:Properties

    // Event set information will be obtained from introspection.//GEN-FIRST:Events
    private static EventSetDescriptor[] eventSets = null;
    private static EventSetDescriptor[] getEdescriptor(){//GEN-HEADEREND:Events

    // Here you can add code for customizing the event sets array.

        return eventSets;     } //GEN-LAST:Events

    // Method information will be obtained from introspection.//GEN-FIRST:Methods
    private static MethodDescriptor[] methods = null;
    private static MethodDescriptor[] getMdescriptor(){//GEN-HEADEREND:Methods

    // Here you can add code for customizing the methods array.

        return methods;     } //GEN-LAST:Methods


    private static final int defaultPropertyIndex = -1;//GEN-BEGIN:Idx
    private static final int defaultEventIndex = -1;//GEN-END:Idx


 //GEN-FIRST:Superclass

  // Here you can add code for customizing the Superclass BeanInfo.

 //GEN-LAST:Superclass

  /**
   * Gets the bean's <code>BeanDescriptor</code>s.
   *
   * @return BeanDescriptor describing the editable
   * properties of this bean.  May return null if the
   * information should be obtained by automatic analysis.
   */
  public BeanDescriptor getBeanDescriptor() {
```

```
    return getBdescriptor();
  }
  /**
   * Gets the bean's <code>PropertyDescriptor</code>s.
   *
   * @return An array of PropertyDescriptors describing the editable
   * properties supported by this bean.  May return null if the
   * information should be obtained by automatic analysis.
   * <p>
   * If a property is indexed, then its entry in the result array will
   * belong to the IndexedPropertyDescriptor subclass of PropertyDescriptor.
   * A client of getPropertyDescriptors can use "instanceof" to check
   * if a given PropertyDescriptor is an IndexedPropertyDescriptor.
   */
  public PropertyDescriptor[] getPropertyDescriptors() {
    return getPdescriptor();
  }
  /**
   * Gets the bean's <code>EventSetDescriptor</code>s.
   * @return  An array of EventSetDescriptors describing the kinds of
   * events fired by this bean.  May return null if the information
   * should be obtained by automatic analysis.
   */
  public EventSetDescriptor[] getEventSetDescriptors() {
    return getEdescriptor();
  }
  /**
   * Gets the bean's <code>MethodDescriptor</code>s.
   * @return  An array of MethodDescriptors describing the methods
   * implemented by this bean.  May return null if the information
   * should be obtained by automatic analysis.
   */
  public MethodDescriptor[] getMethodDescriptors() {
    return getMdescriptor();
  }
  /**
   * A bean may have a "default" property that is the property that will
   * mostly commonly be initially chosen for update by human's who are
   * customizing the bean.
   * @return  Index of default property in the PropertyDescriptor array
   *          returned by getPropertyDescriptors.
   * <P>    Returns -1 if there is no default property.
   */
  public int getDefaultPropertyIndex() {
    return defaultPropertyIndex;
  }
  /**
   * A bean may have a "default" event that is the event that will
   * mostly commonly be used by human's when using the bean.
   * @return Index of default event in the EventSetDescriptor array
   *          returned by getEventSetDescriptors.
   * <P>    Returns -1 if there is no default event.
   */
  public int getDefaultEventIndex() {
    return defaultEventIndex;
  }
}
```

jbb\gui\AboutDialog.java

```
package jbb.gui;

import jbb.international.*;
import jbb.tools.*;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.event.ActionListener;
import java.awt.event.WindowListener;
import java.awt.event.ActionEvent;
import java.awt.event.WindowEvent;
import java.awt.Container;
import java.awt.BorderLayout;
import java.awt.Point;

//import java.util.*;

public class AboutDialog extends JDialog implements ActionListener, WindowListener
{

	private JButton btnOK;
	private JPanel btnsPanel;
	private JLabel mainIcon;

	public AboutDialog(JFrame mainFrame) {
		//will be getText
		super(mainFrame, World.getS("About") + " " + World.getS("JBBTitle"),
true);

		btnOK = new JButton("            " + World.getS("OK") + "
");
		btnsPanel = new JPanel();
		mainIcon = new JLabel(World.getIcn("JBBInfo"));

		//btnsPanel.setBorder(new BevelBorder(BevelBorder.LOWERED));
		//btnsPanel.add(btnOK);
		Container cp = getContentPane();
		cp.setLayout(new BorderLayout());
		cp.add(btnOK, BorderLayout.SOUTH);
		cp.add(mainIcon, BorderLayout.CENTER);

		setResizable(false);
		pack();

		Point screenCenter = Tools.getScreenCenter(this.getSize());
		setLocation(screenCenter.x, screenCenter.y);

		btnOK.addActionListener(this);
		addWindowListener(this);

		setVisible(true);
	}
	private void closeAbout() {
		setVisible(false);
		dispose();
	}
```

```
	public void actionPerformed(ActionEvent e) {
		closeAbout();
	}
	public void windowClosing(WindowEvent e) {
		closeAbout();
	}
	public void windowClosed(WindowEvent e) {
	}
	public void windowActivated(WindowEvent e) {
	}
	public void windowDeactivated(WindowEvent e) {
	}
	public void windowIconified(WindowEvent e) {
	}
	public void windowDeiconified(WindowEvent e) {
	}
	public void windowOpened(WindowEvent e) {
	}
}
```

jbb\gui\FBeansBar.java

```
package jbb.gui;

import jbb.international.*;
import jbb.gui.*;
import jbb.system.workspace.*;
import jbb.system.*;

import javax.swing.border.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.beans.BeanInfo;

public class FBeansBar extends JPanel {

	public  static final String SelectingArrow = World.getS("Selection");

	private static final ImageIcon DefaultBeanImageIcon32 =
World.getImgIcn("DefaultBeanIcon32");
	private static final Icon SelectArrow32 = World.getIcn("SelectArrow32");

	private FMain fMain;
	private Hashtable hash = new Hashtable();
	private JTabbedPane tp = new JTabbedPane();
	private JComponent lblSelectArrow;
	private Vector selectArrows = new Vector();

	public String selectedID = SelectingArrow;
	public JComponent selectedComponent;

	public FBeansBar(FMain newFMain) {
		super();
		fMain = newFMain;
		setLayout(new BoxLayout(this, BoxLayout.X_AXIS));

		String tmps = SelectingArrow;
		JLabel lbl = new JLabel(SelectArrow32);
		lbl.setToolTipText(tmps);
	lbl.addMouseListener(new CBeansBarListener(tmps, fMain));
		selectedComponent = lbl;//(JComponent)
		lblSelectArrow = selectedComponent;
		selectArrows.add(lbl);
		selectedComponent.setBorder(new BevelBorder(BevelBorder.LOWERED));
//		add(lbl);

//		hash.put(getTabName(tmps), new JPanel());
//		( (JPanel)hash.get(getTabName(tmps)) ).add(lbl);
		String s;
		BeanInfo BI;
		Image img;
		ImageIcon imgIcn;
		for(Enumeration e = CKnownBeans.hash.keys(); e.hasMoreElements(); ) {
			tmps = (String)e.nextElement();
			BI = (BeanInfo)CKnownBeans.hash.get(tmps);
			img = BI.getIcon(BeanInfo.ICON_COLOR_32x32);
			if (img == null) {
			imgIcn = DefaultBeanImageIcon32;
```

```
                  } else {
                        imgIcn = new ImageIcon(img);
                  }
            lbl = new JLabel((Icon)imgIcn);
            lbl.setToolTipText(tmps);
            lbl.addMouseListener(new CBeansBarListener(tmps, fMain));
            lbl.setBorder(new BevelBorder(BevelBorder.RAISED));
        s = getTabName(tmps);
        if (hash.containsKey(s))
                  ( (JPanel)hash.get(s) ).add(lbl);
              else {
                  hash.put(s, new JPanel());
                        JLabel lblSelectArrowExtra = new
JLabel(((JLabel)lblSelectArrow).getIcon());

      lblSelectArrowExtra.setToolTipText(lblSelectArrow.getToolTipText());
                  lblSelectArrowExtra.addMouseListener(new
CBeansBarListener(lblSelectArrow.getToolTipText(), fMain));
                  ( (JPanel)hash.get(s) ).add(lblSelectArrowExtra);
                  selectArrows.add(lblSelectArrowExtra);
                  ( (JPanel)hash.get(s) ).add(lbl);
              }
            }
            //JScrollPane sp;
            for(Enumeration e = hash.keys(); e.hasMoreElements(); ) {
                  s = (String)e.nextElement();
                  //sp = new JScrollPane((JPanel)hash.get(s));
                  //sp.revalidate();
                  //sp.setVerticalScrollBarPolicy(sp.VERTICAL_SCROLLBAR_NEVER);
                  //sp.setHorizontalScrollBarPolicy(sp.HORIZONTAL_SCROLLBAR_ALWAYS)
;
            //tp.add(s, sp);
            ((FlowLayout)
((JPanel)hash.get(s)).getLayout()).setAlignment(FlowLayout.LEFT);
            tp.add(s, (JPanel)hash.get(s));
            }
            add(tp);
      setButtonSelecting();
      }
      private String getTabName(String s) {
            if (     s.equals("javax.swing.JFrame")
                        || s.equals("javax.swing.JApplet")
                        || s.equals("javax.swing.JInternalFrame")
                        || s.equals("javax.swing.JDialog")
                        || s.equals(SelectingArrow))
                  return World.getS("Main");
            else
                  return World.getS("Swing");
      }
      public void setButtonSelecting() {
            selectedID = SelectingArrow;
            selectedComponent.setBorder(new BevelBorder(BevelBorder.RAISED));
            selectedComponent = lblSelectArrow;
            selectedComponent.setBorder(new BevelBorder(BevelBorder.LOWERED));
            //and also update the button group
      }
}

class CBeansBarListener implements MouseListener {

      private FMain fMain;
```

```
	private String ID;

	public CBeansBarListener(String newID, FMain newFMain) {
		fMain = newFMain;
		ID = newID;
System.out.println("FBeansBar added(icon) ID:< " + ID + " >");
	}
	public void mouseClicked(MouseEvent me) {
		CProject p;
System.out.println("Selected bean ID : " + ID);
		if (ID.equals("javax.swing.JFrame")) {  //get this from somewhere else
as constant (final) or file
			p = CSystem.workspace.getSelectedProject();
			if (p != null) {
				( (JComponent)me.getComponent() ).setBorder(new
BevelBorder(BevelBorder.LOWERED));
				fMain.beansBar.repaint();
				p.addProjectItem("javax.swing.JFrame");
				( (JComponent)me.getComponent() ).setBorder(new
BevelBorder(BevelBorder.RAISED));
			}
		} else if (ID.equals("javax.swing.JApplet")) {
			p = CSystem.workspace.getSelectedProject();
			if (p != null) {
				( (JComponent)me.getComponent() ).setBorder(new
BevelBorder(BevelBorder.LOWERED));
				fMain.beansBar.repaint();
				p.addProjectItem("javax.swing.JApplet");
				( (JComponent)me.getComponent() ).setBorder(new
BevelBorder(BevelBorder.RAISED));
			}
		} else if (ID.equals(fMain.beansBar.SelectingArrow)) {
			fMain.beansBar.selectedComponent.setBorder(new
BevelBorder(BevelBorder.RAISED));
			( (JComponent)me.getComponent() ).setBorder(new
BevelBorder(BevelBorder.LOWERED));
			fMain.beansBar.selectedID = ID;
			fMain.beansBar.selectedComponent = (JComponent)me.getComponent();
			//add any other special actions i need
		} else {
			//store for drop to desktopFrames
			fMain.beansBar.selectedComponent.setBorder(new
BevelBorder(BevelBorder.RAISED));
			( (JComponent)me.getComponent() ).setBorder(new
BevelBorder(BevelBorder.LOWERED));
			fMain.beansBar.selectedID = ID;
			fMain.beansBar.selectedComponent = (JComponent)me.getComponent();
		}
	}
	public void mousePressed(MouseEvent me) {
	}
	public void mouseEntered(MouseEvent me) {
	}
	public void mouseExited(MouseEvent me) {
	}
	public void mouseReleased(MouseEvent me) {
	}
}
```

jbb\gui\FMain.java

```
package jbb.gui;

import jbb.international.*;
import jbb.system.*;
import jbb.system.workspace.*;
import jbb.gui.workspace.*;
import jbb.system.propertyEditors.*;
import jbb.tools.*;
//import jbb.gui.wiz.db.*;

import javax.swing.JTabbedPane;
import javax.swing.JFrame;
import java.awt.BorderLayout;
import javax.swing.JPanel;
import javax.swing.JSplitPane;
//import javax.swing.border.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.LayoutManager;
import java.beans.PropertyEditorManager;
import java.awt.Dimension;
import java.awt.Container;
import java.util.Vector;

public class FMain extends JFrame {

	private FMenuBar menuBar;

	private CKnownBeans knownBeans;
	public FBeansBar beansBar;

//private static JSplitPane eastPane; //used by projectExplorer & propertiesPanel
	private static JSplitPane eastCenterPane; //used by propertiesPanel &
fWorkspace

	private	JTabbedPane eastPane;
	public		FProjectExplorer projectExplorer;

	public		FWorkspace fWorkspace;

	public		FPropertiesPanel propertiesPanel;

/**
start JAB() constructor
*/
	public FMain() {
		super();

		this.setTitle(World.getS("JBBTitle"));
		Dimension screenSize = Tools.getScreenSize();
		this.setBounds(0, 0, screenSize.width, screenSize.height-150);
		this.addWindowListener(new FMainListener());

		knownBeans			= new CKnownBeans();
		fWorkspace			= new FWorkspace(this);
		beansBar				= new FBeansBar(this);
		projectExplorer	= new FProjectExplorer(this);
		propertiesPanel	= new FPropertiesPanel(this);
```

```
            propertiesPanel.fMain = this;
//          eventsPanel       = new FEventsPanel(this);
            eastPane = new JTabbedPane();
            eastPane.add(World.getText("Explorer"), projectExplorer);
            eastPane.add(World.getText("Inspector"), propertiesPanel);
            eastPane.setPreferredSize(getPreferredSize());
//          eastPane                            = new
JSplitPane(JSplitPane.VERTICAL_SPLIT, true
//                                                                      ,
projectExplorer, propertiesPanel);
            eastCenterPane          = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
true
                                                                        ,
fWorkspace, eastPane);
//

            Container cp = this.getContentPane();

            menuBar = new FMenuBar();
            this.setJMenuBar(menuBar);
            cp.add(beansBar, BorderLayout.NORTH);
//          cp.add(eastPane, BorderLayout.EAST);
//          cp.add(fWorkspace, BorderLayout.CENTER);
            cp.add(eastCenterPane, BorderLayout.CENTER);
            PropertyEditorManager.registerEditor( LayoutManager.class,
LayoutManagerEditor.class);

      }
}

/**
   class that listens for FMain events
*/
class FMainListener extends WindowAdapter {
      public void windowClosing(WindowEvent evt) {
            CSystem.exit(0);
      }
}
```

jbb\gui\FMenuBar.java

```
package jbb.gui;

import jbb.international.*;
import jbb.system.*;
import jbb.plugins.*;
import jbb.system.workspace.*;
import jbb.tools.compilers.*;
import jbb.gui.wiz.db.*;
import javax.swing.UIManager;
import javax.swing.JMenuBar;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JSeparator;
import javax.swing.JMenuItem;
import javax.swing.SwingUtilities;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.Enumeration;
import java.awt.Component;
import java.awt.Frame;

public class FMenuBar extends JMenuBar {
	private CSystem system;
	private JMenu fileMenu;
	private JMenu fileNewMenu;
	private JMenu editMenu;
	private JMenu viewMenu;
	private JMenu projectMenu;
	private JMenu javaMenu;
	private JMenu runMenu;
	private JMenu configurationMenu;
	private JMenu configurationPluginsMenu;
	private JMenu configurationLFMenu;
	private JMenu helpMenu;
	public FMenuBar() {
		super();
		setBorderPainted(true);

		fileMenu = new JMenu(World.getS("File"));

		fileNewMenu = new JMenu(World.getS("New"));
		fileNewMenu.add(new CMenuItemAction(World.getS("New File"),
"NewFile"));
		fileNewMenu.getItem(fileNewMenu.getItemCount()-1).setEnabled(false);
		fileNewMenu.add(new CMenuItemAction(World.getS("New DB Form"),
"NewDBForm"));
		fileMenu.add(fileNewMenu);
		fileMenu.add(new CMenuItemAction(World.getS("Open"), "OpenFile"));
		fileMenu.getItem(fileMenu.getItemCount()-1).setEnabled(false);
		fileMenu.add(new CMenuItemAction(World.getS("Save"), "SaveFile"));
		fileMenu.getItem(fileMenu.getItemCount()-1).setEnabled(false);
		fileMenu.add(new CMenuItemAction(World.getS("Save As"), "SaveAsFile"));
		fileMenu.getItem(fileMenu.getItemCount()-1).setEnabled(false);
		fileMenu.add(new CMenuItemAction(World.getS("Close"), "CloseFile"));
		fileMenu.getItem(fileMenu.getItemCount()-1).setEnabled(false);
		fileMenu.add(new JSeparator());
		fileMenu.add(new CMenuItemAction(World.getText("New Workspace"),
"NewWorkspaceFile"));
```

```
            fileMenu.add(new CMenuItemAction(World.getText("Open Workspace"),
"OpenWorkspaceFile"));
            fileMenu.getItem(fileMenu.getItemCount()-1).setEnabled(false);
            fileMenu.add(new CMenuItemAction(World.getText("Save Workspace"),
"SaveWorkspaceFile"));
            fileMenu.add(new CMenuItemAction(World.getText("Save As Workspace"),
"SaveAsWorkspaceFile"));
            fileMenu.add(new CMenuItemAction(World.getText("Close Workspace"),
"CloseWorkspaceFile"));
            fileMenu.getItem(fileMenu.getItemCount()-1).setEnabled(false);
            fileMenu.add(new JSeparator());
            fileMenu.add(new CMenuItemAction(World.getS("Print"), "Print"));
            fileMenu.getItem(fileMenu.getItemCount()-1).setEnabled(false);
            fileMenu.add(new JSeparator());
            fileMenu.add(new CMenuItemAction(World.getS("Exit"), "Exit"));

            editMenu = new JMenu(World.getS("Edit"));
            editMenu.add(new CMenuItemAction(World.getS("Undo"), "UndoEdit"));
            editMenu.getItem(editMenu.getItemCount()-1).setEnabled(false);

            viewMenu = new JMenu(World.getS("View"));
            viewMenu.add(new CMenuItemAction(World.getText("Class Info"),
"ClassInfoView"));
            viewMenu.getItem(viewMenu.getItemCount()-1).setEnabled(false);

            projectMenu = new JMenu(World.getS("Project"));
            projectMenu.add(new CMenuItemAction(World.getS("New"), "NewProject"));
            projectMenu.add(new CMenuItemAction(World.getS("Open"),
"OpenProject"));
            projectMenu.getItem(projectMenu.getItemCount()-1).setEnabled(false);
            projectMenu.add(new CMenuItemAction(World.getS("Save"),
"SaveProject"));
            projectMenu.add(new CMenuItemAction(World.getS("Save As"),
"SaveAsProject"));
            projectMenu.add(new CMenuItemAction(World.getS("Close"),
"CloseProject"));
            projectMenu.getItem(projectMenu.getItemCount()-1).setEnabled(false);
            projectMenu.add(new CMenuItemAction(World.getS("Remove"),
"RemoveProject"));
            projectMenu.getItem(projectMenu.getItemCount()-1).setEnabled(false);
            projectMenu.add(new JSeparator());
            projectMenu.add(new CMenuItemAction(World.getS("Options"),
"OptionsProject"));
            projectMenu.getItem(projectMenu.getItemCount()-1).setEnabled(false);

            javaMenu = new JMenu(World.getS("Java"));
            javaMenu.add(new CMenuItemAction(World.getText("Options"),
"OptionsJava"));

            runMenu = new JMenu(World.getS("Run"));
            runMenu.add(new CMenuItemAction(World.getS("Compile"), "CompileRun"));
            runMenu.getItem(runMenu.getItemCount()-1).setEnabled(false);
            runMenu.add(new CMenuItemAction(World.getText("Build All"),
"BuildAllRun"));
            runMenu.add(new CMenuItemAction(World.getText("Build Dirty"),
"BuildDirtyRun"));
            runMenu.getItem(runMenu.getItemCount()-1).setEnabled(false);
            runMenu.add(new JSeparator());
            runMenu.add(new CMenuItemAction(World.getS("Run"), "RunRun"));
            runMenu.add(new JSeparator());
//this should be a subMenu
```

```
            runMenu.add(new CMenuItemAction(World.getS("Debug"), "DebugRun"));
            runMenu.getItem(runMenu.getItemCount()-1).setEnabled(false);
            runMenu.add(new JSeparator() );
            runMenu.add(new CMenuItemAction(World.getS("Options"), "OptionsRun"));
            runMenu.getItem(runMenu.getItemCount()-1).setEnabled(false);

            configurationMenu = new JMenu(World.getS("Configuration"));
            configurationMenu.add(new CMenuItemAction(World.getText("General
options"), "GeneralConfiguration"));
            configurationMenu.getItem(configurationMenu.getItemCount()-
1).setEnabled(false);

            configurationPluginsMenu = new JMenu(World.getS("Plugins"));
            configurationPluginsMenu.add(new
CMenuItemAction(World.getText("MemoryView"), "Plugin_PMemoryView"));

            configurationLFMenu = new JMenu(World.getS("Look & Feel"));

      configurationMenu.add(configurationPluginsMenu);
            configurationMenu.add(configurationLFMenu);

            helpMenu = new JMenu(World.getS("Help"));
            helpMenu.add(new CMenuItemAction(World.getS("Help"), "Help"));
            helpMenu.getItem(helpMenu.getItemCount()-1).setEnabled(false);
            helpMenu.add(new JSeparator() );
            helpMenu.add(new CMenuItemAction(World.getS("About"), "About"));

            /** Adding all available look & feels
              */
            String name, className;
            UIManager.LookAndFeelInfo[] LFIs;
            LFIs = UIManager.getInstalledLookAndFeels();
            for(int i=0 ; i<LFIs.length ; i++) {
                  name = LFIs[i].getName();
                  className = LFIs[i].getClassName();
                  configurationLFMenu.add(new CMenuItemAction(name, className));
            }
            this.add(fileMenu);
            this.add(editMenu);
            this.add(viewMenu);
            this.add(projectMenu);
            this.add(javaMenu);
            this.add(runMenu);
            this.add(configurationMenu);
            this.add(helpMenu);
      }
}
/** Instances of this class listen menu events
  */
class CMenuItemAction extends JMenuItem implements ActionListener {
      private String itemID = null;
      public CMenuItemAction(String newItemTitle, String newItemID) {
            super(newItemTitle);
            itemID = newItemID;
            this.addActionListener(this);
      }
      public void actionPerformed(ActionEvent ae) {
            if (itemID.equals("NewWorkspaceFile")) {
                  CSystem.system.workspace.addProject();
            } else if (itemID.equals("NewProject")) {
System.out.println("$$$$$$$$$$$$$$$$$$$$$$$$4");
```

```
                        CSystem.system.workspace.addAppletProject();
System.out.println("@@@@@@@@@@@@@@@@@@@@@@@@2");
            } else if (itemID.equals("NewDBForm")) {
            DWizDBForm wizDBForm = new DWizDBForm();
                        wizDBForm.setVisible(true);//show();
//                CSystem.system.workspace.getSelectedProject().addProjectItem();
            } else if (itemID.equals("SaveWorkspaceFile")) {
                  CSystem.system.workspace.save();
            } else if (itemID.equals("SaveAsWorkspaceFile")) {
                  CSystem.system.workspace.saveAs();
            } else if (itemID.equals("Plugin_PMemoryView")) {
                  PMemoryView mv = new PMemoryView();
            } else if (itemID.equals("OptionsJava")) {
            } else if (itemID.equals("BuildAllRun")) {
                  CSystem.system.workspace.save();
                  if (CSystem.system.workspace.saved) {
                        CJavaCompiler javaCompiler = new CJavaCompiler();
                        CProject selProject =
CSystem.system.workspace.getSelectedProject();

      javaCompiler.javaCompile(selProject.getJavaFilenamesForBuild());
                  } else {
                        //BuildAll Canceled
                  }
            } else if (itemID.equals("RunRun")) {
                  CProject selProject =
CSystem.system.workspace.getSelectedProject();
                  selProject.run();
            } else if (itemID.equals("About")) {
                        AboutDialog dlg = new AboutDialog( (JFrame)
(CSystem.system.fMain) );
            } else if (itemID.equals("Exit")) {
                  CSystem.exit(0);
            } else {
                  checkLFs();
System.out.println(itemID);
            }
      }
      /** Depending on the itemID the Look & Feel changes.
        *
        * @see itemID
        */
      public void checkLFs() {
            UIManager.LookAndFeelInfo[] LFIs = null;
            String name, className;
            Component tmp;
            LFIs = UIManager.getInstalledLookAndFeels();
            for(int i=0 ; i<LFIs.length ; i++) {
                  className = LFIs[i].getClassName();
                  if (itemID.equals(className)) {
                        try {
                              UIManager.setLookAndFeel(className);
                        } catch (javax.swing.UnsupportedLookAndFeelException e) {
System.out.println(e);
                        } catch (java.lang.IllegalAccessException e) {
System.out.println(e);
                        } catch (java.lang.InstantiationException e) {
System.out.println(e);
                        } catch (java.lang.ClassNotFoundException e) {
System.out.println(e);
                        }
```

```
                        Frame frames[] = CSystem.system.fMain.getFrames();
                        for(int j=0; i<frames.length;i++)
                              SwingUtilities.updateComponentTreeUI(frames[i]);
                  }
            }
      }
}
```

jbb\gui\FProjectExplorer.java

```
package jbb.gui;

import jbb.international.*;
import jbb.gui.*;
import jbb.system.workspace.*;
import jbb.gui.workspace.*;

import javax.swing.JScrollPane;
//import javax.swing.border.*;
import java.awt.event.MouseListener;
import java.awt.Dimension;
import java.awt.event.MouseEvent;
import javax.swing.JTree;
import javax.swing.tree.DefaultTreeModel;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.TreeModel;
import javax.swing.tree.TreePath;
import java.util.Enumeration;


public class FProjectExplorer extends JScrollPane implements MouseListener {

	private static final DefaultTreeModel emptyModel = new DefaultTreeModel(
			new DefaultMutableTreeNode( World.getText("No project
selected")));

	private JTree theTree;
	private FMain fMain;
	private CProject projectView;

	public FProjectExplorer(FMain newMain) {
		super();
		fMain = newMain;

		theTree = new JTree(emptyModel);
		this.setViewportView(theTree);

		theTree.addMouseListener(this);
		theTree.setCellRenderer(new CProjectExplorerCellRenderer(fMain));

		this.setVerticalScrollBarPolicy(VERTICAL_SCROLLBAR_ALWAYS);
	}

	public DefaultMutableTreeNode getBeanForName(String searchName) {
//should use a hash search(faster perfomance) / how? keep in tree and in a hash
		DefaultTreeModel dtm = (DefaultTreeModel)theTree.getModel();
		DefaultMutableTreeNode dmtnRoot =
(DefaultMutableTreeNode)dtm.getRoot();
		DefaultMutableTreeNode dmtnChild;
		for(Enumeration e = dmtnRoot.children(); e.hasMoreElements(); ) {
			dmtnChild = (DefaultMutableTreeNode)e.nextElement();
			if (searchName.equals(dmtnChild.toString()))
				return	dmtnChild;
		}
		return null;
	}
	public void setProject(TreeModel newTM) {
		theTree.setModel(newTM);
```

```
			updateProjectTree();
		}

		public void updateProjectTree() {
			theTree.revalidate();
			this.revalidate();
		}

		public void setProject(CProject newP) {
			this.setProject((TreeModel)newP);
}
		public void mouseClicked(MouseEvent e) {
			int selRow = theTree.getRowForLocation(e.getX(), e.getY());
			TreePath selPath = theTree.getPathForLocation(e.getX(), e.getY());
			if(selRow != -1) {
				if(e.getClickCount() == 1) {
System.out.println("single click" + selRow + selPath);
				} else if (e.getClickCount() == 2) {
System.out.println("double click" + selRow + selPath);
					Object obj = selPath.getLastPathComponent();
					if (obj instanceof CProjectItem) {
						fMain.fWorkspace.addBoundGUI((CProjectItem)obj);
					}
				}
			}
		}
		public void mouseEntered(MouseEvent e) {
		}
		public void mouseExited(MouseEvent e) {
		}
		public void mousePressed(MouseEvent e) {
		}
		public void mouseReleased(MouseEvent e) {
		}
		public Dimension getPreferredSize() {
			Dimension screenSize = fMain.getSize();
			return new Dimension(screenSize.width/5, screenSize.height/5);
		}
		//public Dimension getMaximumSize() {
		//	Dimension screenSize = fMain.getSize();
		//	return new Dimension(screenSize.width/3, screenSize.height);
		//}
}
```

jbb\gui\FPropertiesPaneljava

```
package jbb.gui;

import jbb.international.*;
import jbb.gui.workspace.*;
import jbb.system.*;
import jbb.system.propertyEditors.*;

import javax.swing.*;
//import javax.swing.border.*;
//import javax.swing.event.*;
import java.awt.*;
import java.lang.reflect.*;
//import java.util.*;
import java.beans.*;
import jbb.system.workspace.CItem;

public class FPropertiesPanel extends JPanel implements PropertyChangeListener {

//should change status
	private final static int USE_NONE = 0;
	private final static int USE_NORMAL = 1;
	private final static int USE_HIDDEN = 2;
	private final static int USE_EXPERT = 3;
	private final static int USE_EXPERT_HIDDEN = 4;
	private final static int USE_NOT_READ_WRITE = 5;
	private final static int USE_NO_EDITOR = 6;
	private final static int USE_NULL_VALUE = 7;
	private final static int USE_NO_DISPLAYABLE_EDITOR = 8;

	public FMain fMain;

	private Object selectedBean = null;
	private PropertyDescriptor[] prds;
	private PropertyEditor[] editors;
	private JLabel[] labels;
	private Object[] values;
	private Component[] views;
	private Method[] getters;
	private Method[] setters;
	private int[] use;
	private int normProperties;
	private boolean topContainer;

	private JComboBox chooser = new JComboBox();
	private JPanel bodyTitle = new JPanel(new GridLayout(1,2));
	private Box body = new Box(BoxLayout.Y_AXIS);
	private JPanel propertiesSheet = new JPanel();
	private JScrollPane sp;

	public FPropertiesPanel(JFrame newFrame) {
		super(new BorderLayout());
		//fMain = newMain;
		chooser.setMaximumSize(chooser.getPreferredSize());
		body.add(chooser);
		JLabel lblName = new JLabel(World.getS("Name"));
		lblName.setMaximumSize(lblName.getPreferredSize());
		JLabel lblValue = new JLabel(World.getS("Value"));
		lblValue.setMaximumSize(lblValue.getPreferredSize());
```

```
/*
            lblName.setHorizontalAlignment(lblName.LEFT);
            lblName.setVerticalAlignment(lblName.TOP);
            lblValue.setHorizontalAlignment(lblName.RIGHT);
            lblValue.setVerticalAlignment(lblName.TOP);
*/
            bodyTitle.add(lblName);
            bodyTitle.add(lblValue);
            bodyTitle.setMaximumSize(bodyTitle.getPreferredSize());

            body.add(bodyTitle);
            sp = new JScrollPane(propertiesSheet);
            sp.setVerticalScrollBarPolicy(sp.VERTICAL_SCROLLBAR_ALWAYS);
            sp.setHorizontalScrollBarPolicy(sp.HORIZONTAL_SCROLLBAR_NEVER);
            body.add(sp);
            JLabel title = new JLabel(World.getS("Properties"));//will be tabbed
pane properties/events
            this.add(title, BorderLayout.NORTH);
            this.add(body, BorderLayout.CENTER);
      this.setMinimumSize(new Dimension(170, 100));
      this.setMaximumSize(new Dimension(170, 2000));
      }
      public void setSelectedBean(Object obj) {
            if ( obj == null )
                  propertiesSheet.removeAll();
            else
                  if (selectedBean != obj) {
                        selectedBean = obj;
                        updateSheet();
                  }
      }
      private void updateSheet() {
            propertiesSheet.removeAll();
            prds = null;
            editors = null;
            labels = null;
            values = null;
            views = null;
            getters = null;
            setters = null;
            use = null;
            normProperties = 0;
            if (selectedBean != null) {
//System.err.println("1" + selectedBean);
//System.err.println("2" + selectedBean.getClass());
//System.err.println("3" + selectedBean.getClass().getName());
BeanInfo BI=null;
//should get all usable properties in a different way, faster way. Probably storing
them in a beans JBB package
BI = (BeanInfo)CKnownBeans.hash.get(selectedBean.getClass().getName());
/*
if ( BI == null ) {
            try {
                  BI =
Introspector.getBeanInfo(Class.forName( selectedBean.getClass().getName() ),
Introspector.USE_ALL_BEANINFO);
            } catch (IntrospectionException ie) {
            } catch (ClassNotFoundException cnfe) {
            }
}//end of BI = introspector....
*/
```

```
//System.err.println("4" + BI);
                  Object[] args = { };
                  //initialize arrays
//System.gc();
//System.runFinalization();
                  prds = BI.getPropertyDescriptors();
                  editors = new PropertyEditor[prds.length];
                  labels = new JLabel[prds.length];
                  values = new Object[prds.length];
                  views = new JComponent[prds.length];
                  getters = new Method[prds.length];
                  setters = new Method[prds.length];
                  use = new int[prds.length];
                  Class prdType = null;
                  normProperties = 0;
                  for (int i = 0 ; i < prds.length ; i++ ) {
//System.gc();
//System.runFinalization();
                        use[i] = USE_NORMAL;
                        labels[i] = new JLabel(prds[i].getDisplayName());
                        if (prds[i].isHidden()) {
                              use[i] = USE_HIDDEN;
                              labels[i].setText("H" + labels[i].getText() );
                        }
                        if (prds[i].isExpert()) {
                              labels[i].setText("E" + labels[i].getText() );
                              if (use[i] == USE_HIDDEN) {
                                    use[i] = USE_EXPERT_HIDDEN;
                              } else {
                                    use[i] = USE_EXPERT;
                              }
                        }
                        labels[i].setToolTipText(prds[i].getShortDescription());
                        getters[i] = prds[i].getReadMethod();
                        setters[i] = prds[i].getWriteMethod();
//System.err.println(i + "/" + prds.length);
//System.err.println("1.");
                        values[i] = null;
                        if (getters[i] != null && setters[i] != null) {
                              try {
                                    values[i] = getters[i].invoke(selectedBean,
args);
                              } catch (Exception e) {
System.err.println("2.");
System.out.println(e);
                              }
//System.err.println("3.");
                              Class editorClass = prds[i].getPropertyEditorClass();
                              editors[i] = null;
//System.err.println("4.");
                              if (editorClass != null)
                                    try {
                                          editors[i] =
(PropertyEditor)editorClass.newInstance();
                                    } catch (Exception e) {
                                          //do nothing
                                    }
//System.err.println("5.");
                              if (editors[i] == null) {
                                    prdType = prds[i].getPropertyType();
```

```
                                    editors[i] =
PropertyEditorManager.findEditor(prdType);
                                    //if editor does not exist use custom editor
(for now)
                                    if (editors[i] == null) {
//                                      editors[i] = new CCustomEditor(frame);
                                    }
                              }
                              // If i can't edit this component, skip it.
//System.err.println("6.");
                              if (editors[i] == null) {
//System.err.println("Warning: Can't find public property editor for property \"" +
labels[i].getText() + "\".  Skipping.");
//System.err.println("7.");
                                    if (prdType == null)
                                          prdType = prds[i].getPropertyType();

//System.err.println("needs public property editor for class \"" +
prdType.getName() + "\".  Skipping.");
                                    use[i] = USE_NO_EDITOR;
                              } else {
                                    // Don't try to set null values:
//System.err.println("8.");
                                    if (values[i] == null) {
//System.err.println("Warning: Property \"" + labels[i].getText() + "\" has null
initial value.  Skipping.");
                                          use[i] = USE_NULL_VALUE;
                                    } else {
                                          editors[i].setValue(values[i]);

      editors[i].addPropertyChangeListener(this);
                                          // Now figure out how to display it...
//System.err.println("9.");
                                          if (editors[i].isPaintable() &&
editors[i].supportsCustomEditor()) {
                                                views[i] = new
PropertyCanvas(fMain, editors[i]);
                                                normProperties += 1;
                                          } else if
(editors[i].supportsCustomEditor()) {
                                                views[i] = new
PropertyButton(fMain, editors[i]);
                                                normProperties += 1;
                                          } else if (editors[i].getTags() != null)
{
                                                views[i] = new
PropertySelector(editors[i]);
                                                normProperties += 1;
                                          } else if (editors[i].getAsText() !=
null) {
                                                views[i] = new
PropertyText(editors[i]);
                                                normProperties += 1;
                                          } else {
//System.err.println("Warning: Property \"" + labels[i].getText() + "\" has non-
displayabale editor.  Skipping.");
                                                use[i] = USE_NO_DISPLAYABLE_EDITOR;
                                          }
                                    }
                              }
                        } else {
```

```
                                    use[i] = USE_NOT_READ_WRITE;
                              }
                        }
                  }
                  propertiesSheet.setVisible(false);
                  if (normProperties > 0) {
                        propertiesSheet.setLayout(new GridLayout(normProperties, 2));
                        for (int i = 0 ; i < prds.length ; i++ ) {
                              switch (use[i]) {
                                    case USE_NO_DISPLAYABLE_EDITOR: {
//                                        propertiesSheet.add(labels[i]);
//                                        propertiesSheet.add(new JLabel("No
displayableEditor"));
                                        break;
                                    }
                                    case USE_NULL_VALUE: {
//                                        propertiesSheet.add(labels[i]);
//                                        propertiesSheet.add(new JLabel("Null value"));
                                        break;
                                    }
                                    case USE_NO_EDITOR: {
//                                        propertiesSheet.add(labels[i]);
//                                        propertiesSheet.add(new JLabel("No editor"));
                                        break;
                                    }
                                    case USE_NOT_READ_WRITE: {
//                                        propertiesSheet.add(labels[i]);
//                                        propertiesSheet.add(new JLabel("Read only"));
                                        break;
                                    }
                                    case USE_EXPERT_HIDDEN: {
                                        propertiesSheet.add(labels[i]);
                                        propertiesSheet.add(views[i]);
                                        break;
                                    }
                                    case USE_HIDDEN: {
                                        propertiesSheet.add(labels[i]);
                                        propertiesSheet.add(views[i]);
                                        break;
                                    }
                                    case USE_EXPERT: {
                                        propertiesSheet.add(labels[i]);
                                        propertiesSheet.add(views[i]);
                                        break;
                                    }
                                    case USE_NORMAL: {
                                        propertiesSheet.add(labels[i]);
                                        propertiesSheet.add(views[i]);
                                        break;
                                    }
                                    case USE_NONE: {
//                                        propertiesSheet.add(labels[i]);
//                                        propertiesSheet.add(new JLabel("NONE"));
                                        break;
                                    }
                              }
                        }
                  }
                  propertiesSheet.revalidate();
                  //the size it should be????
```

```
            Dimension dm = ((GridLayout)
(propertiesSheet.getLayout())).preferredLayoutSize(propertiesSheet);
            dm.width = dm.width + 40;
            int w = dm.width/4;
            if (w<170)
              w = 170;
            this.setMinimumSize(new Dimension(w, dm.height));
            w = dm.width/2;
            if (w<170)
              w = 170;
            this.setPreferredSize(new Dimension(w, dm.height));
            this.setMaximumSize(dm);
            propertiesSheet.setVisible(true);
      }
      public void propertyChange(PropertyChangeEvent evt) {
System.out.println("PropertyChanged: "+evt.getPropertyName()+"\n"+evt);
            propertiesSheet.setVisible(false);
            if (evt.getSource() instanceof PropertyEditor) {
                  PropertyEditor editor = (PropertyEditor)evt.getSource();
                  for (int i = 0; i < editors.length; i++) {
                        if (editors[i] == editor) {
                                    Object value = editor.getValue();
                                 if (values[i] == value)
                                    return;
                                 if (selectedBean instanceof FProjectItem)
                                 {
                                          if (evt.getPropertyName() ==
"Name")
                                          {

                                          }

                                    }
                               values[i] = value;
                               try {
                                     Object args[] = { value };
                                     args[0] = value;
                                     setters[i].invoke(selectedBean, args);
                                     // We add the changed property to the targets
wrapper
                                     // so that we know precisely what bean
properties have
                                     // changed for the target bean and we're able
to
                                     // generate initialization statements for only
those
                                     // modified properties at code generation time.
//
      targetWrapper.getChangedProperties().addElement(properties[i]);
//XXXXXXX
//editor.getJavaInitializationString()
                                     } catch (InvocationTargetException e) {
                                     if (e.getTargetException() instanceof
PropertyVetoException) {
System.err.println("WARNING: Vetoed; reason is: " +
e.getTargetException().getMessage());
                                     } else {
System.err.println("Exception while updating " + prds[i].getName());
System.err.println(e.getTargetException());
                                     }
                               } catch (Exception e) {
```

```
System.err.println("Unexpected exception while updating " + prds[i].getName());
System.err.println(e);
                                }
                                if (views[i] != null && views[i] instanceof
PropertyCanvas) {
                                      views[i].repaint();
                                }
                                break;
                          }
                    }
              }
              // Now re-read all the properties and update the editors
              // for any other properties that have changed.
              for (int i = 0; i < prds.length; i++) {
                    Object obj;
                    try {
                          Object args[] = { };
                          obj = getters[i].invoke(selectedBean, args);
                    } catch (Exception e) {
                          obj = null;
                    }
                    if (obj == values[i] || (obj != null && obj.equals(values[i]))) {
                          // The property is equal to its old value.
                          continue;
                    }
                    values[i] = obj;
                    // Make sure we have an editor for this property...
                    if (editors[i] == null) {
                          continue;
                    }
                    // The property has changed!  Update the editor.
                    editors[i].setValue(obj);
                    if (views[i] != null) {
                          views[i].repaint();
                    }
              }
System.out.println("End of PropertyChanged:\n"+evt);
              // Make sure the target bean gets repainted.
              if (Beans.isInstanceOf(selectedBean, Component.class)) {
                    ((Component)(Beans.getInstanceOf(selectedBean,
Component.class))).repaint();
              }
              propertiesSheet.setVisible(true);
        }
}
```

jbb\gui\editors\FSourceEditor.java

```
package jbb.gui.editors;

import jbb.gui.*;
import jbb.system.editors.java.*;
import jbb.system.editors.html.*;
import jbb.system.workspace.*;

import javax.swing.*;
import javax.swing.text.*;
import java.awt.*;

public class FSourceEditor extends JTabbedPane {

      private FMain fMain;

      public FSourceEditor(FMain newFMain) {
            super();
            fMain = newFMain;
      }

      public void addBoundEditor(CProjectItem pi) {
            //for now make it simple. Will add another class that will use
            JEditorPane ep = new JEditorPane();
            //if java source
    if (pi.getType()==pi.TYPE_APPLET_HTML){
            ep.setContentType("text/html");
    } else {
      CJavaEditorKit kit = new CJavaEditorKit();
      ep.setEditorKitForContentType("text/java", kit);
      CJavaContext styles = kit.getStylePreferences();
      Style s;
            s = styles.getStyleForScanValue(CToken.COMMENT.getScanValue());
      StyleConstants.setForeground(s, new Color(0, 128, 0));
      s = styles.getStyleForScanValue(CToken.STRINGVAL.getScanValue());
            StyleConstants.setForeground(s, new Color(192, 0, 0));
      Color keyword = new Color(0, 0, 192);
      for (int code = 70; code <= 130; code++) {
        s = styles.getStyleForScanValue(code);
        if (s != null) {
                  StyleConstants.setForeground(s, keyword);
            }
            }
            ep.setContentType("text/java");
    }

            ep.setBackground(Color.white);
            ep.setFont(new Font("Courier", 0, 12));
            ep.setEditable(true);

            ep.setDocument(pi.getDoc());
            JScrollPane sp = new JScrollPane(ep);
            addTab(pi.getNam(), sp);
      }
}
```

jbb\gui\wiz\db\DWizDBForm.form

```
<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.0" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
  <NonVisualComponents>
    <Component class="jbb.beans.DB.JDBCon" name="jDBCon1">
    </Component>
  </NonVisualComponents>
  <Properties>
    <Property name="title" type="java.lang.String" value="DB Form Wizard"/>
    <Property name="defaultCloseOperation" type="int" value="2"/>
    <Property name="name" type="java.lang.String" value="fDWizDBForm"/>
  </Properties>
  <SyntheticProperties>
    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
  </SyntheticProperties>
  <Events>
    <EventHandler event="windowClosing" listener="java.awt.event.WindowListener"
parameters="java.awt.event.WindowEvent" handler="exitForm"/>
  </Events>
  <AuxValues>
    <AuxValue name="designerSize" type="java.awt.Dimension" value="-84,-
19,0,5,115,114,0,18,106,97,118,97,46,97,119,116,46,68,105,109,101,110,115,105,111,1
10,65,-114,-39,-41,-
84,95,68,20,2,0,2,73,0,6,104,101,105,103,104,116,73,0,5,119,105,100,116,104,120,112
,0,0,0,-12,0,0,1,60"/>
  </AuxValues>

  <Layout class="org.netbeans.modules.form.compat2.layouts.DesignBoxLayout"/>
  <SubComponents>
    <Container class="javax.swing.JTabbedPane" name="tbpDBWiz">
      <Properties>
        <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[200, 200]"/>
        </Property>
        <Property name="doubleBuffered" type="boolean" value="true"/>
      </Properties>

      <Layout
class="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLayout"/
>
      <SubComponents>
        <Container class="javax.swing.JPanel" name="pnlConnection">
          <Constraints>
            <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLa
yout"
value="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLayout$J
TabbedPaneConstraintsDescription">
              <JTabbedPaneConstraints tabName="Connection">
                <Property name="tabTitle" type="java.lang.String"
value="Connection"/>
              </JTabbedPaneConstraints>
            </Constraint>
          </Constraints>

          <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"/>
          <SubComponents>
```

```
            <Container class="javax.swing.JPanel" name="jPanel5">
              <Constraints>
                <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
                  <BorderConstraints direction="North"/>
                </Constraint>
              </Constraints>

              <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"/>
              <SubComponents>
                <Component class="javax.swing.JTextField" name="edtDriverName">
                  <Properties>
                    <Property name="text" type="java.lang.String"
value="sun.jdbc.odbc.JdbcOdbcDriver"/>
                    <Property name="name" type="java.lang.String"
value="edtDriverName"/>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtDB_URL"/>
                    </Property>
                  </Properties>
                  <Events>
                    <EventHandler event="keyTyped"
listener="java.awt.event.KeyListener" parameters="java.awt.event.KeyEvent"
handler="edtDriverNameKeyTyped"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="1" gridWidth="2"
gridHeight="1" fill="1" ipadX="209" ipadY="9" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="6" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JLabel" name="lblDriverName">
                  <Properties>
                    <Property name="labelFor" type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtDriverName"/>
                    </Property>
                    <Property name="text" type="java.lang.String" value="Driver
Name"/>
                    <Property name="name" type="java.lang.String"
value="lblDriverName"/>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtDriverName"/>
                    </Property>
                  </Properties>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
```

```
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="0" gridWidth="2"
gridHeight="1" fill="1" ipadX="310" ipadY="3" insetsTop="6" insetsLeft="6"
insetsBottom="0" insetsRight="6" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JTextField" name="edtDB_URL">
                  <Properties>
                    <Property name="text" type="java.lang.String"
value="jdbc:odbc:JDBC_TEST_DS"/>
                    <Property name="name" type="java.lang.String"
value="edtDB_URL"/>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtUsername"/>
                    </Property>
                  </Properties>
                  <Events>
                    <EventHandler event="keyTyped"
listener="java.awt.event.KeyListener" parameters="java.awt.event.KeyEvent"
handler="edtDB_URLKeyTyped"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="3" gridWidth="2"
gridHeight="1" fill="1" ipadX="227" ipadY="9" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="6" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JLabel" name="lblDB_Url">
                  <Properties>
                    <Property name="labelFor" type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtDB_URL"/>
                    </Property>
                    <Property name="text" type="java.lang.String" value="DB_URL"/>
                    <Property name="name" type="java.lang.String"
value="lblDB_Url"/>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtDB_URL"/>
                    </Property>
                  </Properties>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="2" gridWidth="2"
gridHeight="1" fill="1" ipadX="334" ipadY="3" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="6" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
```

```
                </Component>
                <Component class="javax.swing.JTextField" name="edtUsername">
                  <Properties>
                    <Property name="name" type="java.lang.String"
value="edtUsername"/>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtPassword"/>
                    </Property>
                  </Properties>
                  <Events>
                    <EventHandler event="keyTyped"
listener="java.awt.event.KeyListener" parameters="java.awt.event.KeyEvent"
handler="edtUsernameKeyTyped"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="5" gridWidth="1"
gridHeight="1" fill="1" ipadX="186" ipadY="9" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="3" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JLabel" name="lblUsername">
                  <Properties>
                    <Property name="labelFor" type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtUsername"/>
                    </Property>
                    <Property name="text" type="java.lang.String"
value="Username"/>
                    <Property name="name" type="java.lang.String"
value="lblUsername"/>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtUsername"/>
                    </Property>
                  </Properties>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="4" gridWidth="1"
gridHeight="1" fill="1" ipadX="131" ipadY="3" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="3" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JTextField" name="edtPassword">
                  <Properties>
                    <Property name="name" type="java.lang.String"
value="edtPassword"/>
                  </Properties>
                  <Events>
```

```
                    <EventHandler event="keyTyped"
listener="java.awt.event.KeyListener" parameters="java.awt.event.KeyEvent"
handler="edtPasswordKeyTyped"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="1" gridY="5" gridWidth="1"
gridHeight="1" fill="1" ipadX="186" ipadY="9" insetsTop="0" insetsLeft="3"
insetsBottom="0" insetsRight="6" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JLabel" name="lblPassword">
                  <Properties>
                    <Property name="labelFor" type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtPassword"/>
                    </Property>
                    <Property name="text" type="java.lang.String"
value="Password"/>
                    <Property name="name" type="java.lang.String"
value="lblPassword"/>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="edtPassword"/>
                    </Property>
                  </Properties>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="1" gridY="4" gridWidth="1"
gridHeight="1" fill="1" ipadX="132" ipadY="3" insetsTop="0" insetsLeft="3"
insetsBottom="0" insetsRight="6" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
              </SubComponents>
            </Container>
            <Container class="javax.swing.JPanel" name="jPanel2">
              <Constraints>
                <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
                  <BorderConstraints direction="South"/>
                </Constraint>
              </Constraints>

              <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"/>
              <SubComponents>
                <Component class="javax.swing.JButton" name="btnTestConnection">
                  <Properties>
                    <Property name="text" type="java.lang.String" value="Test
Connection"/>
```

```
                </Properties>
                <Events>
                  <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="btnTestConnectionActionPerformed"/>
                </Events>
                <Constraints>
                  <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
                    <BorderConstraints direction="North"/>
                  </Constraint>
                </Constraints>
              </Component>
              <Component class="javax.swing.JButton" name="btnMoveToTables">
                <Properties>
                  <Property name="text" type="java.lang.String" value="Next
=&gt;"/>
                </Properties>
                <Events>
                  <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="btnMoveToTablesActionPerformed"/>
                </Events>
                <Constraints>
                  <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
                    <BorderConstraints direction="East"/>
                  </Constraint>
                </Constraints>
              </Component>
            </SubComponents>
          </Container>
        </SubComponents>
      </Container>
      <Container class="javax.swing.JPanel" name="pnlTables">
        <Properties>
          <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
            <Dimension value="[2147483647, 2147483647]"/>
          </Property>
        </Properties>
        <Events>
          <EventHandler event="componentShown"
listener="java.awt.event.ComponentListener"
parameters="java.awt.event.ComponentEvent" handler="pnlTablesComponentShown"/>
        </Events>
        <Constraints>
          <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLa
yout"
value="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLayout$J
TabbedPaneConstraintsDescription">
            <JTabbedPaneConstraints tabName="Tables">
              <Property name="tabTitle" type="java.lang.String" value="Tables"/>
            </JTabbedPaneConstraints>
          </Constraint>
        </Constraints>
```

```
          <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"/>
          <SubComponents>
            <Container class="javax.swing.JPanel" name="jPanel4">
              <Properties>
                <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                  <Dimension value="[386, 219]"/>
                </Property>
                <Property name="name" type="java.lang.String" value="null"/>
                <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                  <Dimension value="[120, 219]"/>
                </Property>
              </Properties>
              <Constraints>
                <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
                  <BorderConstraints direction="Center"/>
                </Constraint>
              </Constraints>

              <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"/>
              <SubComponents>
                <Component class="javax.swing.JComboBox" name="cbTableType">
                  <Properties>
                    <Property name="editable" type="boolean" value="true"/>
                    <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[124, 25]"/>
                    </Property>
                    <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 25]"/>
                    </Property>
                    <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[32767, 25]"/>
                    </Property>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="cbMainTable"/>
                    </Property>
                  </Properties>
                  <Events>
                    <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="cbTableTypeActionPerformed"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
```

```
                    <GridBagConstraints gridX="0" gridY="1" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="6" anchor="10" weightX="0.0" weightY="0.0"/>
                  </Constraint>
                </Constraints>
              </Component>
              <Component class="javax.swing.JLabel" name="lblTableType">
                <Properties>
                  <Property name="labelFor" type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                    <ComponentRef name="cbTableType"/>
                  </Property>
                  <Property name="text" type="java.lang.String" value="Table
Type"/>
                  <Property name="toolTipText" type="java.lang.String"
value="null"/>
                  <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                    <Dimension value="[39, 16]"/>
                  </Property>
                  <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                    <Dimension value="[39, 16]"/>
                  </Property>
                  <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                    <Dimension value="[39, 16]"/>
                  </Property>
                </Properties>
                <Constraints>
                  <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                    <GridBagConstraints gridX="0" gridY="0" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="6" insetsLeft="6"
insetsBottom="0" insetsRight="6" anchor="11" weightX="1.0" weightY="0.0"/>
                  </Constraint>
                </Constraints>
              </Component>
              <Component class="javax.swing.JLabel" name="lblMainTable">
                <Properties>
                  <Property name="labelFor" type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                    <ComponentRef name="cbMainTable"/>
                  </Property>
                  <Property name="text" type="java.lang.String" value="Main
Table"/>
                  <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                    <Dimension value="[51, 16]"/>
                  </Property>
                  <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                    <Dimension value="[51, 16]"/>
                  </Property>
                  <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                    <Dimension value="[51, 16]"/>
                  </Property>
                </Properties>
```

```
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="2" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="3" anchor="10" weightX="0.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JComboBox" name="cbMainTable">
                  <Properties>
                    <Property name="editable" type="boolean" value="true"/>
                    <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[124, 25]"/>
                    </Property>
                    <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 25]"/>
                    </Property>
                    <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[32767, 25]"/>
                    </Property>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="cbDetailTable"/>
                    </Property>
                  </Properties>
                  <Events>
                    <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="cbMainTableActionPerformed"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="3" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="3" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JLabel" name="lblDetailTable">
                  <Properties>
                    <Property name="labelFor" type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="cbDetailTable"/>
                    </Property>
                    <Property name="text" type="java.lang.String" value="Detail
Table"/>
                    <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 16]"/>
                    </Property>
```

```
                    <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 16]"/>
                    </Property>
                    <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 16]"/>
                    </Property>
                  </Properties>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="1" gridY="2" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="3"
insetsBottom="0" insetsRight="6" anchor="10" weightX="0.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JComboBox" name="cbDetailTable">
                  <Properties>
                    <Property name="editable" type="boolean" value="true"/>
                    <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[124, 25]"/>
                    </Property>
                    <Property name="name" type="java.lang.String" value="null"/>
                    <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 25]"/>
                    </Property>
                    <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[32767, 25]"/>
                    </Property>
                  </Properties>
                  <Events>
                    <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="cbDetailTableActionPerformed"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="1" gridY="3" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="3"
insetsBottom="0" insetsRight="6" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Container class="javax.swing.JPanel" name="jPanel1">
                  <Properties>
                    <Property name="name" type="java.lang.String" value="null"/>
                    <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[120, 87]"/>
                    </Property>
                  </Properties>
```

```
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="5" gridWidth="2"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="3" insetsLeft="6"
insetsBottom="0" insetsRight="6" anchor="10" weightX="0.0" weightY="2.0"/>
                    </Constraint>
                  </Constraints>

                  <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignBoxLayout"/>
                  <SubComponents>
                    <Container class="javax.swing.JScrollPane" name="jScrollPane1">
                      <Properties>
                        <Property name="name" type="java.lang.String"
value="null"/>
                        <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                          <Dimension value="[55, 87]"/>
                        </Property>
                        <Property name="autoscrolls" type="boolean" value="true"/>
                      </Properties>

                      <Layout
class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayout"/
>
                      <SubComponents>
                        <Container class="javax.swing.JPanel"
name="pnlFields_MainTable">

                          <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignBoxLayout">
                            <Property name="axis" type="int" value="1"/>
                          </Layout>
                        </Container>
                      </SubComponents>
                    </Container>
                    <Container class="javax.swing.JScrollPane" name="jScrollPane2">
                      <Properties>
                        <Property name="name" type="java.lang.String"
value="null"/>
                        <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                          <Dimension value="[55, 87]"/>
                        </Property>
                        <Property name="autoscrolls" type="boolean" value="true"/>
                      </Properties>

                      <Layout
class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayout"/
>
                      <SubComponents>
                        <Container class="javax.swing.JPanel"
name="pnlFields_DetailTable">

                          <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignBoxLayout">
                            <Property name="axis" type="int" value="1"/>
                          </Layout>
```

```
                        </Container>
                      </SubComponents>
                    </Container>
                  </SubComponents>
                </Container>
                <Component class="javax.swing.JLabel" name="lblMasterKey">
                  <Properties>
                    <Property name="labelFor" type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="cbMasterKey"/>
                    </Property>
                    <Property name="text" type="java.lang.String" value="Master
Key"/>
                    <Property name="toolTipText" type="java.lang.String"
value="null"/>
                    <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[51, 16]"/>
                    </Property>
                    <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[51, 16]"/>
                    </Property>
                    <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[51, 16]"/>
                    </Property>
                  </Properties>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="6" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="3" anchor="10" weightX="0.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JComboBox" name="cbMasterKey">
                  <Properties>
                    <Property name="editable" type="boolean" value="true"/>
                    <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[124, 25]"/>
                    </Property>
                    <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 25]"/>
                    </Property>
                    <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[32767, 25]"/>
                    </Property>
                    <Property name="nextFocusableComponent"
type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="cbDetailTable"/>
                    </Property>
                  </Properties>
                  <Constraints>
```

```
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="7" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="6"
insetsBottom="0" insetsRight="3" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JLabel" name="lblDetailKey">
                  <Properties>
                    <Property name="labelFor" type="java.awt.Component"
editor="org.netbeans.modules.form.ComponentChooserEditor">
                      <ComponentRef name="cbDetailKey"/>
                    </Property>
                    <Property name="text" type="java.lang.String" value="Detail
Key"/>
                    <Property name="toolTipText" type="java.lang.String"
value="null"/>
                    <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 16]"/>
                    </Property>
                    <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 16]"/>
                    </Property>
                    <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 16]"/>
                    </Property>
                  </Properties>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="1" gridY="6" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="3"
insetsBottom="0" insetsRight="6" anchor="10" weightX="0.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JComboBox" name="cbDetailKey">
                  <Properties>
                    <Property name="editable" type="boolean" value="true"/>
                    <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[124, 25]"/>
                    </Property>
                    <Property name="name" type="java.lang.String" value="null"/>
                    <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[55, 25]"/>
                    </Property>
                    <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
                      <Dimension value="[32767, 25]"/>
                    </Property>
                  </Properties>
```

```
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="1" gridY="7" gridWidth="1"
gridHeight="1" fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="3"
insetsBottom="0" insetsRight="6" anchor="10" weightX="1.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JButton" name="btnMoveToOptions">
                  <Properties>
                    <Property name="text" type="java.lang.String" value="Next
=&amp;gt;"/>
                  </Properties>
                  <Events>
                    <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="btnMoveToOptionsActionPerformed"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="1" gridY="8" gridWidth="1"
gridHeight="1" fill="0" ipadX="0" ipadY="0" insetsTop="3" insetsLeft="0"
insetsBottom="0" insetsRight="0" anchor="13" weightX="0.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JButton"
name="btnMoveBackToConnection">
                  <Properties>
                    <Property name="text" type="java.lang.String" value="&amp;lt;=
Back"/>
                  </Properties>
                  <Events>
                    <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="btnMoveBackToConnectionActionPerformed"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagConstra
intsDescription">
                      <GridBagConstraints gridX="0" gridY="8" gridWidth="1"
gridHeight="1" fill="0" ipadX="0" ipadY="0" insetsTop="3" insetsLeft="0"
insetsBottom="0" insetsRight="0" anchor="17" weightX="0.0" weightY="0.0"/>
                    </Constraint>
                  </Constraints>
                </Component>
              </SubComponents>
            </Container>
          </SubComponents>
        </Container>
        <Container class="javax.swing.JPanel" name="pnlOptions">
          <Constraints>
```

```
            <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLa
yout"
value="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLayout$J
TabbedPaneConstraintsDescription">
              <JTabbedPaneConstraints tabName="Options">
                <Property name="tabTitle" type="java.lang.String" value="Options"/>
              </JTabbedPaneConstraints>
            </Constraint>
          </Constraints>

          <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"/>
          <SubComponents>
            <Container class="javax.swing.JPanel" name="jPanel3">
              <Constraints>
                <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
                  <BorderConstraints direction="South"/>
                </Constraint>
              </Constraints>

              <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"/>
              <SubComponents>
                <Component class="javax.swing.JButton" name="btnGenerate">
                  <Properties>
                    <Property name="text" type="java.lang.String"
value="Generate"/>
                  </Properties>
                  <Events>
                    <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="btnGenerateActionPerformed"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
                      <BorderConstraints direction="North"/>
                    </Constraint>
                  </Constraints>
                </Component>
                <Component class="javax.swing.JButton" name="btnMoveBackToTables">
                  <Properties>
                    <Property name="text" type="java.lang.String" value="&lt;=
Back"/>
                  </Properties>
                  <Events>
                    <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="btnMoveBackToTablesActionPerformed"/>
                  </Events>
                  <Constraints>
                    <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
```

```
                    <BorderConstraints direction="West"/>
                  </Constraint>
                </Constraints>
              </Component>
            </SubComponents>
          </Container>
        </SubComponents>
      </Container>
    </SubComponents>
  </Container>
</SubComponents>
</Form>
```

jbb\gui\wiz\db\DWizDBForm.java

```
/*
 * DWizDBForm.java
 *
 * Created on 10 Áðñßëéïò 2003, 8:36 ìì
 */
package jbb.gui.wiz.db;

import jbb.system.workspace.*;
import jbb.beans.DB.*;
import java.sql.*;
import java.util.*;
import javax.swing.*;

/**
 *
 * @author  OurHome
 */
public class DWizDBForm extends javax.swing.JFrame {

  private boolean conHasChanged = true;
  /** Creates new form DWizDBForm */
  public DWizDBForm() {
    initComponents();
  }

  /** This method is called from within the constructor to
   * initialize the form.
   * WARNING: Do NOT modify this code. The content of this method is
   * always regenerated by the Form Editor.
   */
  private void initComponents() {//GEN-BEGIN:initComponents
    java.awt.GridBagConstraints gridBagConstraints;

    jDBCon1 = new jbb.beans.DB.JDBCon();
    tbpDBWiz = new javax.swing.JTabbedPane();
    pnlConnection = new javax.swing.JPanel();
    jPanel5 = new javax.swing.JPanel();
    edtDriverName = new javax.swing.JTextField();
    lblDriverName = new javax.swing.JLabel();
    edtDB_URL = new javax.swing.JTextField();
    lblDB_Url = new javax.swing.JLabel();
    edtUsername = new javax.swing.JTextField();
    lblUsername = new javax.swing.JLabel();
    edtPassword = new javax.swing.JTextField();
    lblPassword = new javax.swing.JLabel();
    jPanel2 = new javax.swing.JPanel();
    btnTestConnection = new javax.swing.JButton();
    btnMoveToTables = new javax.swing.JButton();
    pnlTables = new javax.swing.JPanel();
    jPanel4 = new javax.swing.JPanel();
    cbTableType = new javax.swing.JComboBox();
    lblTableType = new javax.swing.JLabel();
    lblMainTable = new javax.swing.JLabel();
    cbMainTable = new javax.swing.JComboBox();
    lblDetailTable = new javax.swing.JLabel();
    cbDetailTable = new javax.swing.JComboBox();
    jPanel1 = new javax.swing.JPanel();
    jScrollPane1 = new javax.swing.JScrollPane();
    pnlFields_MainTable = new javax.swing.JPanel();
```

```
    jScrollPane2 = new javax.swing.JScrollPane();
    pnlFields_DetailTable = new javax.swing.JPanel();
    lblMasterKey = new javax.swing.JLabel();
    cbMasterKey = new javax.swing.JComboBox();
    lblDetailKey = new javax.swing.JLabel();
    cbDetailKey = new javax.swing.JComboBox();
    btnMoveToOptions = new javax.swing.JButton();
    btnMoveBackToConnection = new javax.swing.JButton();
    pnlOptions = new javax.swing.JPanel();
    jPanel3 = new javax.swing.JPanel();
    btnGenerate = new javax.swing.JButton();
    btnMoveBackToTables = new javax.swing.JButton();

    getContentPane().setLayout(new javax.swing.BoxLayout(getContentPane(),
javax.swing.BoxLayout.X_AXIS));

    setTitle("DB Form Wizard");
    setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    setName("fDWizDBForm");
    addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
      }
    });

    tbpDBWiz.setMinimumSize(new java.awt.Dimension(200, 200));
    tbpDBWiz.setDoubleBuffered(true);
    pnlConnection.setLayout(new java.awt.BorderLayout());

    jPanel5.setLayout(new java.awt.GridBagLayout());

    edtDriverName.setText("sun.jdbc.odbc.JdbcOdbcDriver");
    edtDriverName.setName("edtDriverName");
    edtDriverName.setNextFocusableComponent(edtDB_URL);
    edtDriverName.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyTyped(java.awt.event.KeyEvent evt) {
        edtDriverNameKeyTyped(evt);
      }
    });

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 1;
    gridBagConstraints.gridwidth = 2;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.ipadx = 209;
    gridBagConstraints.ipady = 9;
    gridBagConstraints.weightx = 1.0;
    gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 6);
    jPanel5.add(edtDriverName, gridBagConstraints);

    lblDriverName.setLabelFor(edtDriverName);
    lblDriverName.setText("Driver Name");
    lblDriverName.setName("lblDriverName");
    lblDriverName.setNextFocusableComponent(edtDriverName);
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 0;
    gridBagConstraints.gridwidth = 2;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.ipadx = 310;
```

```
gridBagConstraints.ipady = 3;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.insets = new java.awt.Insets(6, 6, 0, 6);
jPanel5.add(lblDriverName, gridBagConstraints);

edtDB_URL.setText("jdbc:odbc:JDBC_TEST_DS");
edtDB_URL.setName("edtDB_URL");
edtDB_URL.setNextFocusableComponent(edtUsername);
edtDB_URL.addKeyListener(new java.awt.event.KeyAdapter() {
  public void keyTyped(java.awt.event.KeyEvent evt) {
    edtDB_URLKeyTyped(evt);
  }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.gridwidth = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.ipadx = 227;
gridBagConstraints.ipady = 9;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 6);
jPanel5.add(edtDB_URL, gridBagConstraints);

lblDB_Url.setLabelFor(edtDB_URL);
lblDB_Url.setText("DB_URL");
lblDB_Url.setName("lblDB_Url");
lblDB_Url.setNextFocusableComponent(edtDB_URL);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.gridwidth = 2;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.ipadx = 334;
gridBagConstraints.ipady = 3;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 6);
jPanel5.add(lblDB_Url, gridBagConstraints);

edtUsername.setName("edtUsername");
edtUsername.setNextFocusableComponent(edtPassword);
edtUsername.addKeyListener(new java.awt.event.KeyAdapter() {
  public void keyTyped(java.awt.event.KeyEvent evt) {
    edtUsernameKeyTyped(evt);
  }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.ipadx = 186;
gridBagConstraints.ipady = 9;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 3);
jPanel5.add(edtUsername, gridBagConstraints);

lblUsername.setLabelFor(edtUsername);
lblUsername.setText("Username");
lblUsername.setName("lblUsername");
```

```
lblUsername.setNextFocusableComponent(edtUsername);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.ipadx = 131;
gridBagConstraints.ipady = 3;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 3);
jPanel5.add(lblUsername, gridBagConstraints);

edtPassword.setName("edtPassword");
edtPassword.addKeyListener(new java.awt.event.KeyAdapter() {
  public void keyTyped(java.awt.event.KeyEvent evt) {
    edtPasswordKeyTyped(evt);
  }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.ipadx = 186;
gridBagConstraints.ipady = 9;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.insets = new java.awt.Insets(0, 3, 0, 6);
jPanel5.add(edtPassword, gridBagConstraints);

lblPassword.setLabelFor(edtPassword);
lblPassword.setText("Password");
lblPassword.setName("lblPassword");
lblPassword.setNextFocusableComponent(edtPassword);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.ipadx = 132;
gridBagConstraints.ipady = 3;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.insets = new java.awt.Insets(0, 3, 0, 6);
jPanel5.add(lblPassword, gridBagConstraints);

pnlConnection.add(jPanel5, java.awt.BorderLayout.NORTH);

jPanel2.setLayout(new java.awt.BorderLayout());

btnTestConnection.setText("Test Connection");
btnTestConnection.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
    btnTestConnectionActionPerformed(evt);
  }
});

jPanel2.add(btnTestConnection, java.awt.BorderLayout.NORTH);

btnMoveToTables.setText("Next =>");
btnMoveToTables.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
    btnMoveToTablesActionPerformed(evt);
  }
});
```

```
jPanel2.add(btnMoveToTables, java.awt.BorderLayout.EAST);

pnlConnection.add(jPanel2, java.awt.BorderLayout.SOUTH);

tbpDBWiz.addTab("Connection", pnlConnection);

pnlTables.setLayout(new java.awt.BorderLayout());

pnlTables.setMaximumSize(new java.awt.Dimension(2147483647, 2147483647));
pnlTables.addComponentListener(new java.awt.event.ComponentAdapter() {
  public void componentShown(java.awt.event.ComponentEvent evt) {
    pnlTablesComponentShown(evt);
  }
});

jPanel4.setLayout(new java.awt.GridBagLayout());

jPanel4.setPreferredSize(new java.awt.Dimension(386, 219));
jPanel4.setName("null");
jPanel4.setMinimumSize(new java.awt.Dimension(120, 219));
cbTableType.setEditable(true);
cbTableType.setPreferredSize(new java.awt.Dimension(124, 25));
cbTableType.setMinimumSize(new java.awt.Dimension(55, 25));
cbTableType.setMaximumSize(new java.awt.Dimension(32767, 25));
cbTableType.setNextFocusableComponent(cbMainTable);
cbTableType.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
    cbTableTypeActionPerformed(evt);
  }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 6);
jPanel4.add(cbTableType, gridBagConstraints);

lblTableType.setLabelFor(cbTableType);
lblTableType.setText("Table Type");
lblTableType.setToolTipText("null");
lblTableType.setPreferredSize(new java.awt.Dimension(39, 16));
lblTableType.setMinimumSize(new java.awt.Dimension(39, 16));
lblTableType.setMaximumSize(new java.awt.Dimension(39, 16));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.insets = new java.awt.Insets(6, 6, 0, 6);
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;
gridBagConstraints.weightx = 1.0;
jPanel4.add(lblTableType, gridBagConstraints);

lblMainTable.setLabelFor(cbMainTable);
lblMainTable.setText("Main Table");
lblMainTable.setPreferredSize(new java.awt.Dimension(51, 16));
lblMainTable.setMinimumSize(new java.awt.Dimension(51, 16));
lblMainTable.setMaximumSize(new java.awt.Dimension(51, 16));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
```

```
    gridBagConstraints.gridy = 2;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 3);
    jPanel4.add(lblMainTable, gridBagConstraints);

    cbMainTable.setEditable(true);
    cbMainTable.setPreferredSize(new java.awt.Dimension(124, 25));
    cbMainTable.setMinimumSize(new java.awt.Dimension(55, 25));
    cbMainTable.setMaximumSize(new java.awt.Dimension(32767, 25));
    cbMainTable.setNextFocusableComponent(cbDetailTable);
    cbMainTable.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        cbMainTableActionPerformed(evt);
      }
    });

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 3;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 3);
    gridBagConstraints.weightx = 1.0;
    jPanel4.add(cbMainTable, gridBagConstraints);

    lblDetailTable.setLabelFor(cbDetailTable);
    lblDetailTable.setText("Detail Table");
    lblDetailTable.setPreferredSize(new java.awt.Dimension(55, 16));
    lblDetailTable.setMinimumSize(new java.awt.Dimension(55, 16));
    lblDetailTable.setMaximumSize(new java.awt.Dimension(55, 16));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 2;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.insets = new java.awt.Insets(0, 3, 0, 6);
    jPanel4.add(lblDetailTable, gridBagConstraints);

    cbDetailTable.setEditable(true);
    cbDetailTable.setPreferredSize(new java.awt.Dimension(124, 25));
    cbDetailTable.setName("null");
    cbDetailTable.setMinimumSize(new java.awt.Dimension(55, 25));
    cbDetailTable.setMaximumSize(new java.awt.Dimension(32767, 25));
    cbDetailTable.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        cbDetailTableActionPerformed(evt);
      }
    });

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 3;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.insets = new java.awt.Insets(0, 3, 0, 6);
    gridBagConstraints.weightx = 1.0;
    jPanel4.add(cbDetailTable, gridBagConstraints);

    jPanel1.setLayout(new javax.swing.BoxLayout(jPanel1,
javax.swing.BoxLayout.X_AXIS));

    jPanel1.setName("null");
    jPanel1.setMinimumSize(new java.awt.Dimension(120, 87));
    jScrollPane1.setName("null");
```

```
    jScrollPane1.setMinimumSize(new java.awt.Dimension(55, 87));
    jScrollPane1.setAutoscrolls(true);
    pnlFields_MainTable.setLayout(new javax.swing.BoxLayout(pnlFields_MainTable,
javax.swing.BoxLayout.Y_AXIS));

    jScrollPane1.setViewportView(pnlFields_MainTable);

    jPanel1.add(jScrollPane1);

    jScrollPane2.setName("null");
    jScrollPane2.setMinimumSize(new java.awt.Dimension(55, 87));
    jScrollPane2.setAutoscrolls(true);
    pnlFields_DetailTable.setLayout(new
javax.swing.BoxLayout(pnlFields_DetailTable, javax.swing.BoxLayout.Y_AXIS));

    jScrollPane2.setViewportView(pnlFields_DetailTable);

    jPanel1.add(jScrollPane2);

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 5;
    gridBagConstraints.gridwidth = 2;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.insets = new java.awt.Insets(3, 6, 0, 6);
    gridBagConstraints.weighty = 2.0;
    jPanel4.add(jPanel1, gridBagConstraints);

    lblMasterKey.setLabelFor(cbMasterKey);
    lblMasterKey.setText("Master Key");
    lblMasterKey.setToolTipText("null");
    lblMasterKey.setPreferredSize(new java.awt.Dimension(51, 16));
    lblMasterKey.setMinimumSize(new java.awt.Dimension(51, 16));
    lblMasterKey.setMaximumSize(new java.awt.Dimension(51, 16));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 6;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 3);
    jPanel4.add(lblMasterKey, gridBagConstraints);

    cbMasterKey.setEditable(true);
    cbMasterKey.setPreferredSize(new java.awt.Dimension(124, 25));
    cbMasterKey.setMinimumSize(new java.awt.Dimension(55, 25));
    cbMasterKey.setMaximumSize(new java.awt.Dimension(32767, 25));
    cbMasterKey.setNextFocusableComponent(cbDetailTable);
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 7;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 3);
    gridBagConstraints.weightx = 1.0;
    jPanel4.add(cbMasterKey, gridBagConstraints);

    lblDetailKey.setLabelFor(cbDetailKey);
    lblDetailKey.setText("Detail Key");
    lblDetailKey.setToolTipText("null");
    lblDetailKey.setPreferredSize(new java.awt.Dimension(55, 16));
    lblDetailKey.setMinimumSize(new java.awt.Dimension(55, 16));
    lblDetailKey.setMaximumSize(new java.awt.Dimension(55, 16));
    gridBagConstraints = new java.awt.GridBagConstraints();
```

```
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.insets = new java.awt.Insets(0, 3, 0, 6);
jPanel4.add(lblDetailKey, gridBagConstraints);

cbDetailKey.setEditable(true);
cbDetailKey.setPreferredSize(new java.awt.Dimension(124, 25));
cbDetailKey.setName("null");
cbDetailKey.setMinimumSize(new java.awt.Dimension(55, 25));
cbDetailKey.setMaximumSize(new java.awt.Dimension(32767, 25));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 7;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.insets = new java.awt.Insets(0, 3, 0, 6);
gridBagConstraints.weightx = 1.0;
jPanel4.add(cbDetailKey, gridBagConstraints);

btnMoveToOptions.setText("Next =>");
btnMoveToOptions.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
    btnMoveToOptionsActionPerformed(evt);
  }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 8;
gridBagConstraints.insets = new java.awt.Insets(3, 0, 0, 0);
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
jPanel4.add(btnMoveToOptions, gridBagConstraints);

btnMoveBackToConnection.setText("<= Back");
btnMoveBackToConnection.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
    btnMoveBackToConnectionActionPerformed(evt);
  }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.insets = new java.awt.Insets(3, 0, 0, 0);
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
jPanel4.add(btnMoveBackToConnection, gridBagConstraints);

pnlTables.add(jPanel4, java.awt.BorderLayout.CENTER);

tbpDBWiz.addTab("Tables", pnlTables);

pnlOptions.setLayout(new java.awt.BorderLayout());

jPanel3.setLayout(new java.awt.BorderLayout());

btnGenerate.setText("Generate");
btnGenerate.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
    btnGenerateActionPerformed(evt);
  }
});
```

```
    jPanel3.add(btnGenerate, java.awt.BorderLayout.NORTH);

    btnMoveBackToTables.setText("<= Back");
    btnMoveBackToTables.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnMoveBackToTablesActionPerformed(evt);
      }
    });

    jPanel3.add(btnMoveBackToTables, java.awt.BorderLayout.WEST);

    pnlOptions.add(jPanel3, java.awt.BorderLayout.SOUTH);

    tbpDBWiz.addTab("Options", pnlOptions);

    getContentPane().add(tbpDBWiz);

    pack();
  }//GEN-END:initComponents

  private void edtPasswordKeyTyped(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_edtPasswordKeyTyped
    conHasChanged = true;
  }//GEN-LAST:event_edtPasswordKeyTyped

  private void edtUsernameKeyTyped(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_edtUsernameKeyTyped
    conHasChanged = true;
  }//GEN-LAST:event_edtUsernameKeyTyped

  private void edtDB_URLKeyTyped(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_edtDB_URLKeyTyped
    conHasChanged = true;
  }//GEN-LAST:event_edtDB_URLKeyTyped

  private void edtDriverNameKeyTyped(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_edtDriverNameKeyTyped
    conHasChanged = true;
  }//GEN-LAST:event_edtDriverNameKeyTyped

  private void btnGenerateActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnGenerateActionPerformed
    // Add your handling code here:
  }//GEN-LAST:event_btnGenerateActionPerformed

  private void btnMoveBackToConnectionActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_btnMoveBackToConnectionActionPerformed
    tbpDBWiz.setSelectedComponent(pnlConnection);
  }//GEN-LAST:event_btnMoveBackToConnectionActionPerformed

  private void btnMoveToOptionsActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnMoveToOptionsActionPerformed
    tbpDBWiz.setSelectedComponent(pnlOptions);
  }//GEN-LAST:event_btnMoveToOptionsActionPerformed

  private void btnMoveBackToTablesActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnMoveBackToTablesActionPerformed
    tbpDBWiz.setSelectedComponent(pnlTables);
  }//GEN-LAST:event_btnMoveBackToTablesActionPerformed
```

```
  private void btnMoveToTablesActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnMoveToTablesActionPerformed
    tbpDBWiz.setSelectedComponent(pnlTables);
  }//GEN-LAST:event_btnMoveToTablesActionPerformed

  private void cbDetailTableActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_cbDetailTableActionPerformed
    cbTableActionPerformed(cbDetailTable, pnlFields_DetailTable);
  }//GEN-LAST:event_cbDetailTableActionPerformed
  private void cbTableActionPerformed(JComboBox cbTable, JPanel pnlFields) {
    pnlFields.removeAll();
    if (cbTable==cbDetailTable) {
        lblMasterKey.setVisible(false);
        cbMasterKey.setVisible(false);
        lblDetailKey.setVisible(false);
        cbDetailKey.setVisible(false);
    }
    cbDetailKey.removeAllItems();
    cbMasterKey.removeAllItems();
    DatabaseMetaData dbMD = jDBCon1.getDBMetaData();
    if ((dbMD!=null) && (cbTable.getSelectedIndex()!=-1)) {
      String sTBL = cbTable.getSelectedItem().toString();
      if (!sTBL.equals("<NO DETAIL>")) {
        if (cbTable==cbDetailTable) {
          lblMasterKey.setVisible(true);
          cbMasterKey.setVisible(true);
          lblDetailKey.setVisible(true);
          cbDetailKey.setVisible(true);
        }
        try {
          ResultSet rs = dbMD.getColumns(null, null, sTBL, null);
          if (rs!=null) {
            String sCN;
            String sTN;
            JCheckBox chk;
            while (rs.next()) {
              sCN = rs.getString("COLUMN_NAME");
              if (!rs.wasNull()) {
                sTN = rs.getString("TYPE_NAME");//DATA_TYPE
                if (!rs.wasNull()) {
                  chk = new JCheckBox(sCN+": "+sTN);
                  chk.setToolTipText(sCN);
                  pnlFields.add(chk);
                  cbMasterKey.addItem(sCN);
                  cbDetailKey.addItem(sCN);
                }
              }
            }
            rs.close();
          }
        } catch (SQLException e) {
System.err.println(e);
        }
      }
    }
    pnlFields.validate();
    pnlFields.repaint(10);
  }
  private void cbMainTableActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_cbMainTableActionPerformed
    cbTableActionPerformed(cbMainTable, pnlFields_MainTable);
```

```
  }//GEN-LAST:event_cbMainTableActionPerformed

  private void cbTableTypeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_cbTableTypeActionPerformed
    cbMainTable.removeAllItems();
    cbDetailTable.removeAllItems();
    cbDetailTable.addItem("<NO DETAIL>");
    DatabaseMetaData dbMD = jDBCon1.getDBMetaData();
    if ((dbMD!=null) && (cbTableType.getSelectedIndex()!=-1)) {
      String[] s = {cbTableType.getSelectedItem().toString()};
      try {
                ResultSet rs = dbMD.getTables(null, null, null, s);
        if (rs!=null) {
          String sTBL;
          while (rs.next()) {
            sTBL = rs.getString("TABLE_NAME");
            if (!rs.wasNull()) {
              cbMainTable.addItem(sTBL);
              cbDetailTable.addItem(sTBL);
            }
          }
          rs.close();
        }
      } catch (SQLException e) {
System.err.println(e);
      }
    }
  }//GEN-LAST:event_cbTableTypeActionPerformed

  private void DoConnect(boolean isTest) {
    try{
      jDBCon1.Close();
    } catch (ClassNotFoundException e) {
System.err.println(e);
            } catch (SQLException e) {
System.err.println(e);
    }
            jDBCon1.setDriverName(edtDriverName.getText());
            jDBCon1.setDBURL(edtDB_URL.getText());
            jDBCon1.setLoginName(edtUsername.getText());
            jDBCon1.setLoginPassword(edtPassword.getText());
    try {
      jDBCon1.Open();
      if (isTest) {
        System.out.println("Connection OK!!");
        jDBCon1.Close();
      }
    } catch (ClassNotFoundException e) {
System.err.println(e);
            } catch (SQLException e) {
System.err.println(e);
    }
  }

  /**
   * @param evt
   */
  private void pnlTablesComponentShown(java.awt.event.ComponentEvent evt) {//GEN-
FIRST:event_pnlTablesComponentShown
    if (conHasChanged) {
      DoConnect(false);
```

```
      cbTableType.removeAllItems();
      for (Enumeration e = jDBCon1.getDBTypes().elements(); e.hasMoreElements();) {
        cbTableType.addItem(e.nextElement().toString());
      }
      cbTableType.setSelectedItem("TABLE");
      conHasChanged = false;
    }
  }//GEN-LAST:event_pnlTablesComponentShown

  private void btnTestConnectionActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_btnTestConnectionActionPerformed
    DoConnect(true);
    conHasChanged = true;
  }//GEN-LAST:event_btnTestConnectionActionPerformed

  /** Exit the Application */
  private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
    try {
      jDBCon1.Close();
    } catch (ClassNotFoundException e) {
System.err.println(e);
            } catch (SQLException e) {
System.err.println(e);
    }
  }//GEN-LAST:event_exitForm

  // Variables declaration - do not modify//GEN-BEGIN:variables
  private javax.swing.JLabel lblDriverName;
  private javax.swing.JButton btnMoveToTables;
  private javax.swing.JPanel pnlFields_MainTable;
  private javax.swing.JLabel lblDB_Url;
  private javax.swing.JComboBox cbDetailKey;
  private javax.swing.JLabel lblDetailKey;
  private javax.swing.JScrollPane jScrollPane2;
  private javax.swing.JScrollPane jScrollPane1;
  private javax.swing.JComboBox cbTableType;
  private javax.swing.JComboBox cbMasterKey;
  private javax.swing.JLabel lblTableType;
  private javax.swing.JLabel lblMasterKey;
  private jbb.beans.DB.JDBCon jDBCon1;
  private javax.swing.JPanel pnlConnection;
  private javax.swing.JPanel jPanel5;
  private javax.swing.JPanel jPanel4;
  private javax.swing.JPanel jPanel3;
  private javax.swing.JButton btnTestConnection;
  private javax.swing.JPanel jPanel2;
  private javax.swing.JPanel jPanel1;
  private javax.swing.JPanel pnlOptions;
  private javax.swing.JTextField edtUsername;
  private javax.swing.JButton btnMoveBackToTables;
  private javax.swing.JLabel lblDetailTable;
  private javax.swing.JTabbedPane tbpDBWiz;
  private javax.swing.JPanel pnlFields_DetailTable;
  private javax.swing.JLabel lblUsername;
  private javax.swing.JTextField edtDB_URL;
  private javax.swing.JComboBox cbMainTable;
  private javax.swing.JTextField edtPassword;
  private javax.swing.JButton btnMoveBackToConnection;
  private javax.swing.JLabel lblMainTable;
  private javax.swing.JButton btnGenerate;
  private javax.swing.JPanel pnlTables;
```

```
  private javax.swing.JLabel lblPassword;
  private javax.swing.JButton btnMoveToOptions;
  private javax.swing.JComboBox cbDetailTable;
  private javax.swing.JTextField edtDriverName;
  // End of variables declaration//GEN-END:variables

}
```

jbb\gui\wiz\db\DWizDBForm.xml

```
<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.0" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
  <SyntheticProperties>
    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
  </SyntheticProperties>
  <Events>
    <EventHandler event="windowClosing" listener="java.awt.event.WindowListener"
parameters="java.awt.event.WindowEvent" handler="exitForm"/>
  </Events>

  <Layout class="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"/>
  <SubComponents>
    <Container class="javax.swing.JPanel" name="pnlLeft">
      <Properties>
        <Property name="name" type="java.lang.String" value="pnlLeft"/>
        <Property name="alignmentY" type="float" value="0.47058824"/>
      </Properties>
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
          <BorderConstraints direction="West"/>
        </Constraint>
      </Constraints>

      <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"/>
      <SubComponents>
        <Container class="javax.swing.JPanel" name="pnlMaster">
          <Properties>
            <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
              <Dimension value="[32000, 50]"/>
            </Property>
          </Properties>
          <Constraints>
            <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
              <BorderConstraints direction="Center"/>
            </Constraint>
          </Constraints>

          <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignGridLayout">
            <Property name="columns" type="int" value="1"/>
            <Property name="rows" type="int" value="2"/>
          </Layout>
          <SubComponents>
            <Component class="javax.swing.JLabel" name="lblMaster">
              <Properties>
                <Property name="text" type="java.lang.String" value="jLabel1"/>
              </Properties>
            </Component>
            <Component class="javax.swing.JTextField" name="jTextField1">
              <Properties>
```

```
                <Property name="text" type="java.lang.String" value="jTextField1"/>
              </Properties>
            </Component>
          </SubComponents>
        </Container>
        <Container class="javax.swing.JPanel" name="pnlDetail">
          <Constraints>
            <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstrain
tsDescription">
              <BorderConstraints direction="South"/>
            </Constraint>
          </Constraints>

          <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignGridLayout">
            <Property name="columns" type="int" value="1"/>
            <Property name="rows" type="int" value="2"/>
          </Layout>
        </Container>
      </SubComponents>
    </Container>
  </SubComponents>
</Form>
```

jbb\gui\workspace\CProjectExplorerCellRenderer.java

```
package jbb.gui.workspace;

import jbb.system.workspace.*;
import jbb.gui.*;
import jbb.international.*;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Component;
import javax.swing.JTree;
import javax.swing.Icon;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.JLabel;
import javax.swing.tree.TreeCellRenderer;
import javax.swing.UIManager;

public class CProjectExplorerCellRenderer extends JLabel implements
TreeCellRenderer {

	private FMain fMain;

	private boolean valueSelected;
	private boolean valueHasFocus;

	private static final Icon ProjectIcon = World.getIcn("ProjectIcon");
	private static final Icon FrameIcon = World.getIcn("FrameIcon");
	private static final Icon AppletIcon = World.getIcn("AppletIcon");
	private static final Icon UserClassIcon = World.getIcn("UserClassIcon");
	private static final Icon AppClassIcon = World.getIcn("AppClassIcon");
	private static final Icon ComponentIcon = World.getIcn("ComponentIcon");
	private static final Icon NonComponentIcon =
World.getIcn("NonComponentIcon");

	//should add others
	private Color textSelectionColor;
	private Color textNonSelectionColor;
	private Color backgroundSelectionColor;
	private Color backgroundNonSelectionColor;
	private Color borderSelectionColor;

	public CProjectExplorerCellRenderer(FMain newMain) {
		super();
		fMain = newMain;

		setHorizontalAlignment(JLabel.LEFT);
		textSelectionColor = UIManager.getColor("Tree.selectionForeground");
		textNonSelectionColor = UIManager.getColor("Tree.textForeground");
		backgroundSelectionColor =
UIManager.getColor("Tree.selectionBackground");
		backgroundNonSelectionColor =
UIManager.getColor("Tree.textBackground");
		borderSelectionColor = UIManager.getColor("Tree.selectionBorderColor");
	}
	public Component getTreeCellRendererComponent(JTree tree, Object value,
			boolean sel, boolean expanded, boolean leaf, int row, boolean
hasFocus) {
	int type;
		if (value instanceof CProjectItem) {
```

```
                  CProjectItem prItem = (CProjectItem)(value);
                  type = prItem.getType();
            } else if (value instanceof CItem) {
                        CItem item = (CItem)(value);
                        type = CProjectItem.TYPE_ITEM;
            } else {
//System.out.println(value);
type = CProjectItem.TYPE_ROOT;
            }
            switch (type) {
                  case CProjectItem.TYPE_ROOT: {
                        setIcon(ProjectIcon);
                        break;
                  }
                  case CProjectItem.TYPE_JFRAME: {
                        setIcon(FrameIcon);
                        break;
                  }
                  case CProjectItem.TYPE_JAPPLET: {
                        setIcon(AppletIcon);
                        break;
                  }
                  case CProjectItem.TYPE_USERCLASS: {
                        setIcon(UserClassIcon);
                        break;
                  }
                  case CProjectItem.TYPE_COMPONENT: {
                        setIcon(ComponentIcon);
                        break;
                  }
                  case CProjectItem.TYPE_APPCLASS: {
                        setIcon(AppClassIcon);
                        break;
                  }
                  case CProjectItem.TYPE_OTHER: {
                        setIcon(NonComponentIcon);
                        break;
                  }
                  case CProjectItem.TYPE_ITEM: {
                        //depends on class. Maybe small icon(16x16) of bean
                        setIcon(NonComponentIcon);
                        break;
                  }
            }
            setEnabled(tree.isEnabled());
            setText(value.toString());
            if(sel)
                  setForeground(textSelectionColor);
            else
                setForeground(textNonSelectionColor);
              valueSelected = sel;
      valueHasFocus = hasFocus;
            return this;
      }
      public void paint(Graphics g) {
            int textStart;
            Color clr;
            Icon ci = getIcon();
            if(ci != null && getText() != null)
                  textStart = ci.getIconWidth() + getIconTextGap() - 1;
            else
```

```
                textStart = 0;
            if(valueSelected) {
                clr = backgroundSelectionColor;
            } else {
                clr = backgroundNonSelectionColor;
            }
            if(clr == null)
                clr = getBackground();
            g.setColor(clr);
            g.fillRect(textStart, 0, getWidth() - textStart + 2, getHeight());
            if (valueHasFocus) {
                g.setColor(borderSelectionColor);
                g.drawRect(textStart, 0, getWidth() - textStart + 2,
getHeight());
            }
            super.paint(g);
      }
}
```

jbb\gui\workspace\FItem.java

```
package jbb.gui.workspace;

import jbb.system.workspace.*;

import java.awt.*;
import javax.swing.*;
import java.util.Vector;
//import javax.swing.tree.*;
//import java.awt.event.*;
//import java.io.*;
//import java.beans.*;
//import java.lang.reflect.Method;
//import java.lang.reflect.InvocationTargetException;
//import java.applet.Applet;
//import sunw.beanbox.PropertyHookup;

public class FItem extends JPanel {
      //implements MouseListener, MouseMotionListener {

      private FProjectItem boundProjectItem;
      private CItem cItem;
//    private NodeObject boundNodeObject;
      private Component theBeanComponent;
      private int theBeanType;
      private Object theBean;

//          private transient boolean active;
//          private transient Cursor cursor = defaultCursor;

//          private final static int borderWidth = 5;
//          private final static int resizeDelta = 8;

      // Shorthands for the cursors.
      private static final Cursor nwResizeCursor =
Cursor.getPredefinedCursor(Cursor.NW_RESIZE_CURSOR);
      private static final Cursor neResizeCursor =
Cursor.getPredefinedCursor(Cursor.NE_RESIZE_CURSOR);
      private static final Cursor swResizeCursor =
Cursor.getPredefinedCursor(Cursor.SW_RESIZE_CURSOR);
      private static final Cursor seResizeCursor =
Cursor.getPredefinedCursor(Cursor.SE_RESIZE_CURSOR);
      private static final Cursor moveCursor     =
Cursor.getPredefinedCursor(Cursor.MOVE_CURSOR);
      private static final Cursor defaultCursor  =
Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR);

/*          private static Vector invisibleWrappers = new Vector();
            private transient boolean sawMouseDown;

            // Table that maps events set names to EventDescriptors.
            private transient Hashtable esdMap;
            // List of WrapperEventTargets that we fire events at.
            private Vector eventTargets;
            // List of properties that we bound into
            transient private Vector propertyTargets;

            private static int hashBarWidth = 4;
            private static int hashBarLength = 104;
```

```
            private static Image xHashBar;
            private static Image yHashBar;

            // List of this wrappers' beans' changed properties
            private transient Vector changedProperties;
            private transient boolean isFromPrototype;
*/
//    Object bean, String beanLabel, String beanName
      public FItem(FProjectItem fPI, CItem ci) {
            super();
            boundProjectItem = fPI;
            setName(ci.getNam());
            cItem = ci;
            theBean = cItem.getObj();
            setLayout(new BorderLayout(0,0));
            if (theBean instanceof Component) {
                  theBeanComponent = (Component)theBean;
                  if (ci.getPos() == "ContentPane") {
                        //theBeanComponent.setBackground(new Color(0,0,255));
                        //I should disable this and use my custom CFullLayout
instead.
                        ((JComponent)theBeanComponent).setPreferredSize(new
Dimension(5000, 5000));
                  }
            } else {
                  theBeanComponent = new JLabel(cItem.getNam());
            }
//System.out.println(this);
//System.out.println(theBeanComponent);

//          theBeanComponent.setLocation(0, 0); //borderWidth, borderWidth);

//          theBeanComponent.setSize(50, 50);
            theBeanComponent.addMouseListener(boundProjectItem);
            add(theBeanComponent);
//          setSize(theBeanComponent.getSize());
//System.out.println(this);
//System.out.println(theBeanComponent);
//          theBeanComponent.validate();
            revalidate();
      }
      public Container getContainer() {
//theBean should always be a container since this method is called
            return      (Container)theBean;
      }

      public Object getPosition() {
            return      cItem.getPos();
      }

      public CItem getCItem() {
            return      cItem;
      }
}
```

jbb\gui\workspace\FProjectItem.java

```
package jbb.gui.workspace;

import jbb.international.*;
import jbb.gui.*;
import jbb.gui.workspace.*;
import jbb.system.*;
import jbb.system.workspace.*;
import jbb.system.layouts.*;
import jbb.tools.*;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.event.InternalFrameListener;
import javax.swing.event.InternalFrameEvent;
import javax.swing.border.*;


public class FProjectItem extends JInternalFrame implements InternalFrameListener,
ComponentListener, MouseListener, KeyListener {

	private static final int SELECTING_STATUS	= 0; //when user selects GUIs
	private static final int LOCK_STATUS		= 1; //when user
selects place (of selected container)for new bean
	private static final int CHOOSING_STATUS	= 2; //when user selects
container for new bean

	private static final int UNKNOWN_TYPE		= 0;
	private static final int NULL_TYPE		= 1;
	private static final int BORDER_TYPE		= 2;
	private static final int GRID_TYPE		= 3;
	private static final int FLOW_TYPE		= 4;
	private static final int BOX_TYPE		= 5;
	private static final int CARD_TYPE		= 6;
	private static final int XY_TYPE		= 7;


	private static FProjectItem selectedProjectItem = null;
	private FItem selectedItem = null;

	private int selectedType = CProjectItem.TYPE_ROOT;
	private Object selectedObject = null;

	private Component selectedComponent = null;
//	private DefaultMutableTreeNode selectedItemComponent = null;

	private DefaultMutableTreeNode focusContainerDmtn = null;
	private Container focusContainer = null;
	private Component focusComponent = null;
	private Object focusObject = null;
	private Border selectionBorder = new CompoundBorder(new LineBorder(Color.red,
1), new LineBorder(Color.blue, 1));
	private Border emptyBorder = new EmptyBorder(0,0,0,0);
	private Border oldSelectedItemBorder = null;
	private int backContainers = 0;
```

```
	private Vector componentsToRemove = new Vector();
	private Vector componentsToRemovePosition = new Vector();
	private int guiStatus = SELECTING_STATUS;

	public CProjectItem boundPI;
	private FBeansBar beansBar = null;
	private FPropertiesPanel propertiesPanel;

	public FProjectItem(CProjectItem pi) {
		super();
		boundPI = pi;
		beansBar = CSystem.fMain.beansBar;
//should be size of cprojectitem
		int posX = 100;//valueOf(pi.getProperty("SizeX")).intValue();
		int posY = 50;//valueOf(pi.getProperty("SizeY")).intValue();
		int sizeX = 300;//valueOf(pi.getProperty("SizeX")).intValue();
		int sizeY = 200;//valueOf(pi.getProperty("SizeY")).intValue();
		setBounds(posX, posY, sizeX, sizeY);
		setClosable(true);
		setResizable(true);
		setDefaultCloseOperation(DISPOSE_ON_CLOSE);//I should hide it
		propertiesPanel = CSystem.fMain.propertiesPanel;
//		copyPropertiesFromTopObject();
		FItem fItem = new FItem(this, pi.getContentPane());
		Container cntr = fItem.getContainer();
		this.setContentPane(fItem);
		addChildren(fItem, pi.getContentPane());
		//I should use the GlassPane
		addInternalFrameListener(this);
		addComponentListener(this);
		setVisible(true);
	}

	private void copyPropertiesFromJComponent() {
System.err.println("updatedPropsFromJComponent");
/*		switch (selectedGUIType) {
			case NodeObject.TYPE_JFRAME: {
				JFrame frm = (JFrame)selectedObject;
				selectedProjectItem.setTitle(frm.getTitle());
				selectedProjectItem.setNam(frm.getNam());
				selectedProjectItem.setResizable(frm.isResizable());
				selectedProjectItem.setFont(frm.getFont());
				selectedProjectItem.setBackground(frm.getBackground());
				selectedProjectItem.setForeground(frm.getForeground());
				selectedProjectItem.setSize(frm.getSize());
		selectedProjectItem.setContentPane(frm.getContentPane());
				break;
			}
			case NodeObject.TYPE_JCOMPONENT: {
			//all read-write properties
				break;
			}
		}
*/	}
	private void addChildren(Container cntr, CProjectItem pi) {
		CItem child;
		int layoutType = getLayoutType(cntr.getLayout());
		FItem fItem = null;
		for(Enumeration e = pi.children(); e.hasMoreElements(); ) {
			child = (CItem)e.nextElement();
			fItem = new FItem(this, child);
```

```
                addToContainer(cntr, fItem, layoutType);
        }
    }
    private void addChildren(Container cntr, CItem pi) {
        CItem child;
        int layoutType = getLayoutType(cntr.getLayout());
        FItem fItem = null;
        for(Enumeration e = pi.children(); e.hasMoreElements(); ) {
                child = (CItem)e.nextElement();
                fItem = new FItem(this, child);
                addToContainer(cntr, fItem, layoutType);
        }
    }
    private int getLayoutType(LayoutManager lm) {
        int layoutType = UNKNOWN_TYPE;
        if (lm == null)
                layoutType = NULL_TYPE;
        else if (lm instanceof BorderLayout)
                layoutType = BORDER_TYPE;
        else if (lm instanceof XYLayout)
                layoutType = XY_TYPE;
        else if (lm instanceof GridLayout)
                layoutType = GRID_TYPE;
        else if (lm instanceof FlowLayout)
                layoutType = FLOW_TYPE;
        else if (lm instanceof BoxLayout)
                layoutType = BOX_TYPE;
        else if (lm instanceof CardLayout)
                layoutType = CARD_TYPE;
        return  layoutType;
    }
    private void addToContainer(Container cntr, FItem fItem, int layoutType) {
        switch (layoutType) {
                case BORDER_TYPE: {
                        cntr.add(fItem, (String)fItem.getPosition());
                        if (!fItem.getCItem().isLeaf())
                                addChildren(fItem.getContainer(), fItem.getCItem());
                        break;
                }
                case XY_TYPE: {
                        cntr.add(fItem);
                        fItem.setBounds( (Rectangle)fItem.getPosition() );
                        if (!fItem.getCItem().isLeaf())
                                addChildren(fItem.getContainer(), fItem.getCItem());
                        break;
                }
                case CARD_TYPE: {
                        break;
                }
                case GRID_TYPE: {
                        break;
                }
                case FLOW_TYPE: {}
                case BOX_TYPE: {}
                case UNKNOWN_TYPE: {
                        cntr.add(fItem);
                        if (!fItem.getCItem().isLeaf())
                                addChildren(fItem.getContainer(), fItem.getCItem());
                        break;
                }
                case NULL_TYPE: {
```

```
System.err.println("null type layout");
                        //add a default layout or just add components with no
layout
                        //if possible (with exact positioning)
                        break;
                  }
            }
      }

/*

      //i should add which property has changed
      //so that i do not have to copy all
      public static void propertyHasChanged() {
            if (selectedGUIComponent == null)
                  selectedProjectItem.copyPropertiesFromTopObject();
            else
                  selectedProjectItem.copyPropertiesFromJComponent();
      }
      private void copyPropertiesFromTopObject() {
            switch (selectedGUIType) {
                  case NodeObject.TYPE_JFRAME: {
                        JFrame frm = (JFrame)selectedObject;
                        selectedProjectItem.setTitle(frm.getTitle());
                        selectedProjectItem.setNam(frm.getNam());
                        selectedProjectItem.setResizable(frm.isResizable());
                        selectedProjectItem.setFont(frm.getFont());
                        selectedProjectItem.setBackground(frm.getBackground());
                        selectedProjectItem.setForeground(frm.getForeground());
                        selectedProjectItem.setSize(frm.getSize());
                        selectedProjectItem.setContentPane(frm.getContentPane());
System.err.println("updatedPropsFromTopObj JFrame");
                        break;
                  }
                  case NodeObject.TYPE_JAPPLET: {
                  //all read-write properties
                        break;
                  }
            }
      }
      private void copyPropertiesToTopObject() {
            switch (selectedGUIType) {
                  case NodeObject.TYPE_JFRAME: {
                        JFrame frm = (JFrame)selectedObject;
                        frm.setSize(selectedProjectItem.getSize());
System.err.println("updatedPropsToTopObj JFrame");
                        break;
                  }
                  case NodeObject.TYPE_JAPPLET: {
                        // add some properties
                        break;
                  }
            }
      }
*/
      private void addNewBean(int newType) {
            if (newType == 0) {
                  if (focusObject == this.getContentPane()) {
                        backContainers = 0;
                        focusContainer = getContentPane();
System.out.println("CONTENT PANE");
```

```
                  } else {
                        //should check | if not a JContainer then go back
                        if (focusObject instanceof JPanel) {//also JDesktopPane,
etc
                              backContainers = 0;
                              focusContainer = (Container)focusObject;
System.out.println("JCONTAINER");
                        } else {
                              backContainers = -1;
                              //twice, because an FItem object is actually a
component inside a JPanel
System.out.println("NOT JCONTAINER");
                              focusContainer =
((Component)focusObject).getParent().getParent();
                        }
                  }
            } else if (newType == -1) {
                  if (focusContainer != this.getContentPane()) {
                        focusContainer =
((Component)focusContainer).getParent().getParent();
                        backContainers--;
                  }
            }
            addPossiblePositions();
      }
      private void addPossiblePositions() {
//int debugPos = 0;
            guiStatus = CHOOSING_STATUS;
            CItem ci;
            Container cntr = null;
            int objType;
            Object objItem;
            boolean childFound = false;
            Component c = focusContainer.getParent();
//System.err.println(c);
//System.err.println(debugPos); debugPos++;
            //can we have errors here????
            DefaultMutableTreeNode child = null;
            if ( c instanceof FItem ) {
                  objType = CProjectItem.TYPE_ITEM;
                  child = ((FItem)c).getCItem();
                  objItem = (((FItem)c).getCItem()).getObj();
                  cntr = focusContainer;
            } else { //this should never occur
                        child = boundPI;
                        objType = boundPI.getType();  //--------something
else---------;
                        objItem = boundPI.getObj();
                        cntr = getContentPane();
            }
//System.err.println(debugPos); debugPos++;
            childFound = true;
            if (childFound) {
                  focusContainerDmtn = child;
                  int layoutType = getLayoutType(focusContainer.getLayout());
                  DefaultMutableTreeNode dmtn = child;
                  Hashtable borderHash = new Hashtable();
                  Object position;
                  Object obj;
//System.err.println(debugPos); debugPos++;
                  for(Enumeration e = dmtn.children(); e.hasMoreElements(); ) {
```

```
//System.err.println("for "+debugPos); debugPos++;
                        child = (DefaultMutableTreeNode)e.nextElement();
//System.err.println(debugPos); debugPos++;
                        ci = (CItem)child;
//System.err.println(debugPos); debugPos++;
                        position = ci.getPos();
//System.err.println(debugPos); debugPos++;
                        obj = ci.getObj();
//System.err.println(jc + "nl \n nl" + position);
//System.err.println(debugPos); debugPos++;
                        switch (layoutType) {
                              case BORDER_TYPE: {
//System.err.println(debugPos); debugPos++;
                                    borderHash.put((String)position, obj);
//System.err.println(debugPos); debugPos++;
                                    break;
                              }
                              case CARD_TYPE: {
                                    break;
                              }
                              case XY_TYPE: {}
                              case GRID_TYPE: {}
                              case FLOW_TYPE: {}
                              case BOX_TYPE: {}
                              case UNKNOWN_TYPE: {
                                    //do_nothing
                                    break;
                              }
                              case NULL_TYPE: {
System.err.println("null type layout");
                                    //add a default layout or just add components
with no layout
                                    //if possible
                                    break;
                              }
                        }
                  }
                  JComponent jc;
//System.err.println("forin "+debugPos); debugPos++;
                  try {
                        switch (layoutType) {
                              case BORDER_TYPE: {
                                    obj = borderHash.get(BorderLayout.NORTH);
                                    if (obj == null) {
                                          jc = new JLabel(World.getS("NORTH"));
                                          jc.setBackground(Color.yellow);
                                          jc.setOpaque(true);
jc.setBorder(new CompoundBorder(new LineBorder(Color.red, 2), jc.getBorder()));
                                          jc.addMouseListener(this);
                                          focusContainer.add(jc,
BorderLayout.NORTH);
                                          componentsToRemove.add(jc);

      componentsToRemovePosition.add(BorderLayout.NORTH);
                                    }
                                    obj = borderHash.get(BorderLayout.SOUTH);
                                    if (obj == null) {
                                          jc = (JComponent)new
JLabel(World.getS("SOUTH"));
                                          jc.setBackground(Color.yellow);
                                          jc.setOpaque(true);
```

```
jc.setBorder(new CompoundBorder(new LineBorder(Color.red, 2), jc.getBorder()));
                                          jc.addMouseListener(this);
                                          focusContainer.add(jc,
BorderLayout.SOUTH);
                                          componentsToRemove.add(jc);

      componentsToRemovePosition.add(BorderLayout.SOUTH);
                                    }
                                    obj = borderHash.get(BorderLayout.WEST);
                                    if (obj == null) {
                                          jc = new JLabel(World.getS("WEST"));
                                          jc.setBackground(Color.yellow);
                                          jc.setOpaque(true);
jc.setBorder(new CompoundBorder(new LineBorder(Color.red, 2), jc.getBorder()));
                                          jc.addMouseListener(this);
                                          focusContainer.add(jc,
BorderLayout.WEST);
                                          componentsToRemove.add(jc);

      componentsToRemovePosition.add(BorderLayout.WEST);
                                    }
                                    obj = borderHash.get(BorderLayout.EAST);
                                    if (obj == null) {
                                          jc = new JLabel(World.getS("EAST"));
                                          jc.setBackground(Color.yellow);
                                          jc.setOpaque(true);
jc.setBorder(new CompoundBorder(new LineBorder(Color.red, 2), jc.getBorder()));
                                          jc.addMouseListener(this);
                                          focusContainer.add(jc,
BorderLayout.EAST);
                                          componentsToRemove.add(jc);

      componentsToRemovePosition.add(BorderLayout.EAST);
                                    }
                                    obj = borderHash.get(BorderLayout.CENTER);
                                    if (obj == null) {
                                          jc = new JLabel(World.getS("CENTER"));
                                          jc.setBackground(Color.yellow);
                                          jc.setOpaque(true);
jc.setBorder(new CompoundBorder(new LineBorder(Color.red, 2), jc.getBorder()));
                                          jc.addMouseListener(this);
                                          focusContainer.add(jc,
BorderLayout.CENTER);
                                          componentsToRemove.add(jc);

      componentsToRemovePosition.add(BorderLayout.CENTER);
                                    }
                                    break;
                              }
                              case CARD_TYPE: {
                                    break;
                              }
                              case XY_TYPE: {
                                    //do nothing
                                    break;
                              }
                              case GRID_TYPE: {}
                              case FLOW_TYPE: {}
                              case BOX_TYPE: {}
                              case UNKNOWN_TYPE: {
                                    jc = new JLabel(World.getS("HERE"));
```

```
                                    jc.setBackground(Color.yellow);
                                    jc.setOpaque(true);
jc.setBorder(new CompoundBorder(new LineBorder(Color.red, 2), jc.getBorder()));
                                    jc.addMouseListener(this);
                                    focusContainer.add(jc);
                                    componentsToRemove.add(jc);
                                    componentsToRemovePosition.add(null);
                                    break;
                              }
                              case NULL_TYPE: {
System.err.println("null type layout");
                                    //add a default layout or just add components
with no layout
                                    //if possible
                                    break;
                              }
                        }
                  } catch (Exception ex) {
System.err.println(ex);
                  }
//System.err.println("end for "+debugPos); debugPos++;
                  focusContainer.validate();
            }
      }
      //listen for key events
      public void keyTyped(KeyEvent e) {
System.out.println(e.getKeyCode());
            if (e.getKeyCode() == KeyEvent.VK_DELETE) {
System.out.println("Delete");
            }
      }
      public void keyPressed(KeyEvent e) {
      }
      public void keyReleased(KeyEvent e) {
      }
      //listen for internalFrame events
      public void internalFrameActivated(InternalFrameEvent e) {
            selectedProjectItem = this;
//maybe keep old selected item for each FProjectItem
            selectedItem = null;
            selectedType = selectedProjectItem.boundPI.getType();
            selectedObject = selectedProjectItem.boundPI.getObj();
            propertiesPanel.setSelectedBean(selectedObject);
            selectedComponent = null;
      }
      public void internalFrameClosing(InternalFrameEvent e) {
            if (this == selectedProjectItem) {
                  selectedProjectItem = null;
//maybe keep old selected item for each FProjectItem
                  selectedType = CProjectItem.TYPE_ROOT;
                  selectedItem = null;
                  selectedObject = null;
                  propertiesPanel.setSelectedBean(selectedObject);
                  selectedComponent = null;
            }
      }
      public void internalFrameClosed(InternalFrameEvent e) {
//System.out.println(e);
      }
      public void internalFrameDeactivated(InternalFrameEvent e) {
//System.err.println(e);
```

```
		}
		public void internalFrameDeiconified(InternalFrameEvent e) {
//System.out.println(e);
		}
		public void internalFrameIconified(InternalFrameEvent e) {
//System.out.println(e);
		}
		public void internalFrameOpened(InternalFrameEvent e) {
//System.out.println(e);
		}
		//listen for component events
		public void componentResized(ComponentEvent e) {
//System.err.println("Resized");
//			copyPropertiesToTopObject();
		}
		public void componentMoved(ComponentEvent e) {
//System.out.println(e);
		}
		public void componentShown(ComponentEvent e) {
//System.out.println(e);
		}
		public void componentHidden(ComponentEvent e) {
//System.out.println(e);
		}
		//listen for mouse events for the internalFrame
		//and for all the sub components that contains
		public void mouseClicked(MouseEvent e) {
//System.out.println(e);
		}
		public void mouseEntered(MouseEvent e) {
System.out.println(e);
			if (selectedProjectItem == this
			&& guiStatus == SELECTING_STATUS
			&& beansBar.selectedID != beansBar.SelectingArrow) {
				focusObject = e.getSource();
				if (focusObject instanceof Container)//check for XY
					selectedProjectItem.addNewBean(0);
			}
		}
		public void mouseExited(MouseEvent e) {
System.out.println(e);
			if (selectedProjectItem == this
			&& guiStatus == CHOOSING_STATUS) {
				Component c;
				for(Enumeration en = componentsToRemove.elements();
en.hasMoreElements(); ) {
					c = (Component)en.nextElement();
					focusContainer.remove(c);
				}
				guiStatus = SELECTING_STATUS;
				componentsToRemove.clear();
				componentsToRemovePosition.clear();
				focusContainer.validate();
				focusContainer.repaint();
			}
		}
		public void mousePressed(MouseEvent e) {
System.out.println(e);
			if (selectedProjectItem == this
				&& guiStatus == CHOOSING_STATUS
				&& SwingUtilities.isLeftMouseButton(e)) {
```

```
                        doCheckForXY(e);
                }       else if (selectedProjectItem == this
                        && guiStatus == LOCK_STATUS
                        && SwingUtilities.isLeftMouseButton(e)) {
                                doRemoveAllAddNew(e);
                }       else if (selectedProjectItem == this
                        && guiStatus == SELECTING_STATUS
                        && SwingUtilities.isLeftMouseButton(e)) {
                                doSelectNewGUIComponent(e);
                }       else if (selectedProjectItem == this
                        && guiStatus == CHOOSING_STATUS
                        && SwingUtilities.isRightMouseButton(e)) {
                                selectedProjectItem.addNewBean(-1);
                }
        }
        public void mouseReleased(MouseEvent e) {
//System.out.println(e);
        }

        //actions for above events
        private void doCheckForXY(MouseEvent e) {
                if (focusContainer.getLayout() instanceof XYLayout) {
                        Object obj = Tools.instantiate(null, beansBar.selectedID, true);
                        if (obj instanceof Component) {
                                Component c = (Component)obj;
                                c.setLocation(e.getX(), e.getY());
                                Rectangle r = new Rectangle(c.getBounds());
                                CItem ci = new CItem(boundPI, c.getName(), c, r);
                                FItem newComponent = new FItem(this, ci);
System.out.println(c.getBounds());
                                focusContainerDmtn.add(ci);
                                if (focusContainerDmtn instanceof CProjectItem) {

        ((CProjectItem)focusContainerDmtn).updateJavaSource(ci);
                                } else {
                                        ((CItem)focusContainerDmtn).updateJavaSource(ci);
                                }
                                addToContainer(focusContainer, newComponent,
getLayoutType(focusContainer.getLayout()));
//focusContainer.list(System.out, 0);
                                //newComponent.setPos(c.getBounds());
CSystem.fMain.projectExplorer.revalidate();
CSystem.fMain.projectExplorer.repaint();
                                guiStatus = SELECTING_STATUS;
                                beansBar.setButtonSelecting();
                                focusContainer.validate();
                                focusContainer.repaint();
                                //set new focus
System.out.println("added Component");
                        } else {
                                //make a Label component for design time
System.out.println("must add Non Component");
//like a TTable in Delphi.
                        }
                } else
System.out.println("status = LOCK");
                        guiStatus = LOCK_STATUS;
        }
        private void doRemoveAllAddNew(MouseEvent e) {
                JComponent jcSearch = (JComponent)e.getSource();
//              FItem db;
```

```
            JComponent jc;
            Object pos;
            Enumeration enPos = componentsToRemovePosition.elements();
            for(Enumeration en = componentsToRemove.elements();
en.hasMoreElements(); ) {
                  jc = (JComponent)en.nextElement();
                  pos = enPos.nextElement();
                  focusContainer.remove(jc);
                  if (jc == jcSearch) {
                        focusContainer.remove(jc);
                        Object obj = Tools.instantiate(null, beansBar.selectedID,
false);
                        CItem ci;
                        if (obj instanceof Component) {
                              Component c = (Component)obj;
                              ci = new CItem(boundPI, c.getName(), c, pos);
                        } else {
                              ci = new CItem(boundPI,
Tools.getNewName(beansBar.selectedID), obj, pos);
                        }
                              FItem newComponent = new FItem(this, ci);
                              focusContainerDmtn.add(ci);
                              if (focusContainerDmtn instanceof CProjectItem) {

      ((CProjectItem)focusContainerDmtn).updateJavaSource(ci);
                              } else {

      ((CItem)focusContainerDmtn).updateJavaSource(ci);
                              }
                              addToContainer(focusContainer, newComponent,
getLayoutType(focusContainer.getLayout()));
//
//focusContainer.list(System.out, 0);
//
CSystem.fMain.projectExplorer.revalidate();
CSystem.fMain.projectExplorer.repaint();
                  }
            }
            guiStatus = SELECTING_STATUS;
            beansBar.setButtonSelecting();
            focusContainer.validate();
            focusContainer.repaint();
            componentsToRemove.clear();
            componentsToRemovePosition.clear();
      }
      private void doSelectNewGUIComponent(MouseEvent e) {
            //also show a border and catch resize events for the component if in
XYLayout
                  selectedComponent = (Component)e.getSource();
            if (selectedComponent != getContentPane()) {
                  if (selectedItem != null) {
                        if (oldSelectedItemBorder != null) {
                              selectedItem.setBorder(oldSelectedItemBorder);
                        } else {
                              selectedItem.setBorder(emptyBorder);
                        }
                  }
                  selectedItem = (FItem)(selectedComponent.getParent());
                  oldSelectedItemBorder = selectedItem.getBorder();
                  selectedItem.setBorder(selectionBorder);
```

```
		propertiesPanel.setSelectedBean(selectedItem.getCItem().getObj());
			} else {
				selectedItem = null;
				propertiesPanel.setSelectedBean(selectedComponent);
			}
		}
}
```

jbb\gui\workspace\FWorkspace.java

```
package jbb.gui.workspace;

import jbb.international.*;
import jbb.system.workspace.*;
import jbb.gui.*;
import jbb.gui.editors.*;

import javax.swing.*;
import java.awt.Dimension;
import java.awt.Rectangle;

public class FWorkspace extends JScrollPane {

	private Dimension preferredDimension = new Dimension(5000,5000);
	private JDesktopPane dp = new JDesktopPane();
	private FMain fMain;

	// Currently only one editor will be visible.
	private JInternalFrame editorFrame =
			new JInternalFrame(World.getText("Source
Editor"),true,true,true,true);
	private FSourceEditor editor;

	public FWorkspace(FMain newFMain) {
		super();
		fMain = newFMain;
		setViewportView(dp);
		editor = new FSourceEditor(fMain);
		editorFrame.getContentPane().add(editor);
		dp.add(editorFrame);
		editorFrame.setBounds(0,0,300,150);//or use the last bounds;
		editorFrame.show();
	}
	public void addBoundGUI(CProjectItem pi) {
		int objType = pi.getType();
		Object obj = pi.getObj();
		switch (objType) {
			case CProjectItem.TYPE_JFRAME: {
				editor.addBoundEditor(pi);
				addBoundGUI((JFrame)(obj), pi);
				break;
			}
			case CProjectItem.TYPE_JAPPLET: {
				editor.addBoundEditor(pi);
				addBoundGUI((JApplet)(obj), pi);
				break;
			}
			case CProjectItem.TYPE_APPCLASS:
			case CProjectItem.TYPE_APPLET_HTML: {
				editor.addBoundEditor(pi);
				break;
			}
		}
	}
	private void addBoundGUI(JFrame newJF, CProjectItem pi) {
		FProjectItem fpi = new FProjectItem(pi);
		dp.add(fpi);
		fpi.setBounds((Rectangle)pi.getPos());
```

```
			fpi.show();
			fpi.requestFocus();
			dp.revalidate();
		}
		private void addBoundGUI(JApplet newApplet, CProjectItem pi) {
			FProjectItem fpi = new FProjectItem(pi);
			dp.add(fpi);
			fpi.setBounds((Rectangle)pi.getPos());
			fpi.show();
			fpi.requestFocus();
			dp.revalidate();
		}

//temporary
		public Dimension getMinimumSize() {
			Dimension dm = fMain.propertiesPanel.getMaximumSize();
			Dimension dmMain = fMain.getSize();
			Dimension dmWork = new Dimension(dmMain.width-dm.width, 100);
System.out.println("************************");
System.out.println("dm:"+dm);
System.out.println("dmWork:"+dmWork);
			return dmWork;
		}
		public Dimension getPreferredSize() {
			return preferredDimension;
		}

}
```

jbb\international\GlobalRes.java

```
package jbb.international;

import javax.swing.*;
import java.util.*;
import java.io.File;

public class GlobalRes extends ListResourceBundle {
      private static final Icon JBB = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"JBB.gif");
      private static final Icon JBBInfo = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"JBBInfo.gif");
      private static final Icon SelectArrow32 = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"SimpleArrow32.gif");
      private static final Icon DefaultBeanIcon32 = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"DefaultBeanIcon24.gif");

      private static final Icon ProjectIcon = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"ProjectIcon.gif");
      private static final Icon FrameIcon = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"FrameIcon.gif");
      private static final Icon AppletIcon = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"AppletIcon.gif");
      private static final Icon UserClassIcon = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"UserClassIcon.gif");
      private static final Icon AppClassIcon = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"AppClassIcon.gif");
      private static final Icon JComponentIcon = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"JComponentIcon.gif");
      private static final Icon ComponentIcon = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"JComponentIcon.gif");
      private static final Icon NonComponentIcon = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"JComponentIcon.gif");

      public Object[][] getContents() {
            return contents;
      }

       private static final Object[][] contents = {
            { "JBBDisplayName",                          "JBB" },
            { "JBBTitle",                                      "Java Begginers
Builder" },
            { "JBBVersion",                                "v0.1a" },
            { "Exit",                                            "Exit" },
            { "OK",
      "OK" },
            { "Cancel",                                        "Cancel" },
            { "Hello",                                         "Hello" },
            { "New",                                                 "New" },
            { "Open",                                                "Open" },
            { "Save",                                                "Save" },
            { "Save As",                                       "Save As" },
            { "Close",                                         "Close" },
            { "Help",                                                "Help" },
            { "About",                                         "About" },
            { "File",                                                "File" },
            { "Look & Feel",                         "Look & Feel" },
            { "Help",                                                "Help" },
            { "Swing",                                         "Swing" },
```

```
            { "Source Editor",                          "Source Editor" },
            { "Selection",                                    "Selection" },
            { "No project selected",        "No project selected" },
            { "Name",                                                 "Name" },
            { "Value",                                              "Value" },
            { "Properties",                                   "Properties" },
            { "Explorer",                                           "Explorer" },
            { "Inspector",                                    "Inspector" },
            { "New Workspace",                          "New Workspace" },
            { "Open Workspace",                         "Open Workspace" },
            { "Save Workspace",                         "Save Workspace" },
            { "Save As Workspace",          "Save As Workspace" },
            { "Close Workspace",                  "Close Workspace" },
            { "Print",                                              "Print" },
            { "Edit",                                                 "Edit" },
            { "Undo",                                                 "Undo" },
            { "View",                                                 "View" },
            { "Class Info",                                   "Class Info" },
            { "Project",                                            "Project" },
            { "Remove",                                             "Remove" },
            { "Options",                                            "Options" },
            { "Java",                                                 "Java" },
            { "Options",                                            "Options" },
            { "Run",                                                  "Run" },
            { "Compile",                                            "Compile" },
            { "Build All",                                    "Build All" },
            { "Build Dirty",                            "Build Dirty" },
            { "Debug",                                              "Debug" },
            { "Options",                                            "Options" },
            { "Configuration",                          "Configuration" },
            { "General options",                  "General options" },
            { "Plugins",                                            "Plugins" },
            { "MemoryView",                                   "Memory Viewer" },
            { "Output & Error window", "Output & Error window"},
            { "Starting",                                           "Starting"},
            { "started",                                            "started"},

            { "I/O Error during exec(...).", "I/O Error during exec(...)."},

            { "JBB",                                                      JBB },
            { "JBBInfo",                                            JBBInfo },
            { "SelectArrow32",                          SelectArrow32 },
            { "ProjectIcon",                            ProjectIcon },
            { "DefaultBeanIcon32",          DefaultBeanIcon32 },
            { "FrameIcon",                                    FrameIcon },
            { "AppletIcon",                             AppletIcon },
            { "UserClassIcon",                          UserClassIcon },
            { "JComponentIcon",                   JComponentIcon },
            { "ComponentIcon",                          ComponentIcon },
            { "NonComponentIcon",                 NonComponentIcon },
        };
}
```

jbb\international\GlobalRes_el.java

```
package jbb.international;

import javax.swing.*;
import java.util.*;
import java.io.File;

public class GlobalRes_el extends ListResourceBundle {

	private static final Icon JBB = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"JBB_el.gif");
	private static final Icon JBBInfo = (Icon)new
ImageIcon("Resources\\Images"+File.separator+"JBBInfo_el.gif");

	public Object[][] getContents() {
		return contents;
	}

	static private final Object[][] contents = {
		{ "JBBTitle",                                    "Java Begginers
Builder(Åëëçíéêüò)" },
		{ "Exit",                                              "Åîïäïò" },
		{ "OK",
	"ÏÊ" },
		{ "New",                                               "Íåï" },
		{ "Open",
	"¢íïéãìá" },
		{ "Save",
	"ÁðïèÞêåõóç" },
		{ "Cancel",                                      "Áêõñï" },
		{ "Hello",                                       "Ãåéá óáò" },
		{ "Save As",                                     "Áðïèçêåõóç
ùò" },
		{ "Close",                                       "Êëåéóéìï" },
		{ "Help",
	"Âïçèåéá" },
		{ "About",                                       "Ó÷åôéêá" },
		{ "File",                                              "Áñ÷åßï" },
		{ "Look & Feel",                  "Look & Feel" },
		{ "Main",                                              "Êõñéá" },
		{ "Swing",                                       "Swing" },
		{ "Source Editor",                "Åðåîåñãáóôçò ðçãáéïõ êùäéêá"
},
		{ "Selection",                                 "Åðéëïãç" },
		{ "No project selected",      "Äåí åðéëå÷ôçêå project" },
		{ "Name",                                              "Ïíïìá" },
		{ "Value",                                       "Ôéìç" },
		{ "Properties",                                "Éäéïôçôåò" },
		{ "Explorer",                                        "Explorer" },
		{ "Inspector",                                 "Inspector" },
		{ "New Workspace",                "Íåïò ×ùñïò åñãáóéáò" },
		{ "Open Workspace",               "Áíïéãìá ×ùñïõ åñãáóéáò" },
		{ "Save Workspace",               "Áðïèçêåõóç ×ùñïõ
åñãáóéáò" },
		{ "Save As Workspace",        "Áðïèçêåõóç ùò ×ùñïõ åñãáóéáò" },
		{ "Close Workspace",                "Êëåéóéìï ×ùñïõ åñãáóéáò" },
		{ "Print",                                       "Åêôõðùóç" },
		{ "Edit",
	"Åðåîåñãáóéá" },
```

```
            { "Undo",
        "Áíáéñåóç" },
            { "View",
        "Ðñïâïëç" },
            { "Class Info",                                    "Ðëçñïöïñéåò êëáóçò" },
            { "Project",                                             "Project" },
            { "Remove",                                              "Áöáéñåóç" },
            { "Options",                                             "Åðéëïãåò" },
            { "Java",                                                      "Java" },
            { "Run",
        "Åêôåëåóç" },
            { "Compile",                                             "Ìåôáãëùôôéóç" },
            { "Build All",                                    "×ôéóéìï ïëùí" },
            { "Build Dirty",                             "×ôéóéìï áëëáãìåíùí" },
            { "Debug",
        "Áðïóöáëìáôùóç" },
            { "Configuration",                           "Ñõèìéóåéò" },
            { "General options",                    "Ãåíéêåò åðéëïãåò" },
            { "Plugins",                                             "Ðñïóèåôá" },
            { "MemoryView",                                   "Åìöáíéóç Ìíçìçò" },
            { "Output & Error window","ÐáñÜèõñï Åîüäïõ êáé Ëáèþí"},
            { "Starting",                                            "Åêôýëáóç"},
            { "started",                                             "óå åêôýëáóç"},

            { "I/O Error during exec(...).", "I/O Error during exec(...)."},

            { "JBB",                                                       JBB},
            { "JBBInfo",                                             JBBInfo},
        };
}
```

jbb\international\World.java

```
package jbb.international;

import javax.swing.ImageIcon;
import javax.swing.Icon;
import java.util.ResourceBundle;
import java.util.Locale;
import java.util.ListResourceBundle;
import java.util.MissingResourceException;


public class World {

	private static Locale defaultLocale;
	private static ResourceBundle res;


	public World() {
		defaultLocale = Locale.getDefault();
		getIt();
	}

	public World(Locale dl) {
		defaultLocale = dl;
		getIt();
	}

	private static void getIt() {
			res = ListResourceBundle.getBundle("jbb.international.GlobalRes",
defaultLocale);
	}

	public static String getS(String skey) {
		try {
			return res.getString(skey);
		} catch (MissingResourceException e) {
//System.err.println("| " + skey + " | w1 " + e);
			return skey;
		}
	}
//++
//the skey will be splitted to tokens and
//getS for every one
//if one fails all Fail or only that ????
	public static String getText(String skey) {
		try {
			return res.getString(skey);
		} catch (MissingResourceException e) {
//System.err.println("| " + skey + " | w2 " + e);
			return skey;
		}
	}
	public static Icon getIcn(String skey) {
		try {
			return (Icon)(getImgIcn(skey));
		} catch (Exception e) {
//System.err.println("| " + skey + " | w3 " + e);
			return null;
		}
```

```
	}
	public static ImageIcon getImgIcn(String skey) {
		try {
			return (ImageIcon)res.getObject(skey);
		} catch (MissingResourceException e) {
//System.err.println("| " + skey + " | w4 " + e);
//System.err.println("got it from current folder");
			return (new ImageIcon(skey));
		} catch (Exception e) {
//System.err.println("| " + skey + " | w5 " + e);
			return null;
		}
	}
}
```

jbb\plugins\PMemoryView.java

```
package jbb.plugins;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.text.MessageFormat;

import javax.swing.Timer;

/** Frame to display amount of free memory in the running application.
* <P>
* Handy for use with NetBeans Developer's internal execution. Then the statistic
* of free memory in the whole environment is displayed.
*
* @version 1.0
*/
public class PMemoryView extends javax.swing.JFrame {
  private Runtime r = Runtime.getRuntime ();
  /** message of free memory */
  private static MessageFormat msgMemory = new MessageFormat("Used {2} bytes of {0}
allocated memory");
  /** default update time */
  private static final int UPDATE_TIME = 1000;
  /** timer to invoke updating */
  private Timer timer;

  /** Initializes the Form */
  public PMemoryView() {
    initComponents ();

    setTitle ("Memory View");
    doGarbage.setText ("Garbage Collect");
    doRefresh.setText ("Refresh Now");
    doClose.setText ("Close");

    txtTime.setText ("Refresh time:");
    doTime.setText ("Apply");
    time.setText (String.valueOf (UPDATE_TIME));
    time.selectAll ();
    time.requestFocus ();

    updateStatus ();

    timer = new Timer (UPDATE_TIME, new ActionListener () {
      public void actionPerformed (ActionEvent ev) {
        updateStatus ();
      }
    });
    timer.setRepeats (true);

    pack ();
    Dimension d = Toolkit.getDefaultToolkit ().getScreenSize ();
    Dimension m = this.getSize ();
    d.width -= m.width;
    d.height -= m.height;
    d.width /= 2;
```

```
    d.height /= 2;
    this.setLocation (d.width, d.height);
    this.setVisible (true);
  }
  /** Starts the timer.
  */
  public void addNotify () {
    super.addNotify ();
    timer.start ();
  }
  /** Stops the timer.
  */
  public void removeNotify () {
    super.removeNotify ();
    timer.stop ();
  }
  /** Updates the status of all components */
  private void updateStatus () {
    long free = r.freeMemory ();
    long total = r.totalMemory ();
    // when bigger than integer then divide by two
    while (total > Integer.MAX_VALUE) {
      total = total >> 1;
      free = free >> 1;
    }
    int taken = (int) (total - free);

    status.setMaximum ((int)total);
    status.setValue (taken);

    text.setText (msgMemory.format (new Object[] {
      new Long (total),
      new Long (free),
      new Integer (taken)
    }));
    text.invalidate ();
    validate ();
  }
  /** This method is called from within the constructor to
   * initialize the form.
   * WARNING: Do NOT modify this code. The content of this method is
   * always regenerated by the FormEditor.
   */
  private void initComponents () {//GEN-BEGIN:initComponents
    addWindowListener (new java.awt.event.WindowAdapter () {
      public void windowClosing (java.awt.event.WindowEvent evt) {
        exitForm (evt);
      }
    }
    );

    jPanel1 = new javax.swing.JPanel ();
    jPanel1.setLayout (new java.awt.BorderLayout ());

    text = new javax.swing.JLabel ();

    jPanel1.add (text, java.awt.BorderLayout.SOUTH);

    status = new javax.swing.JProgressBar ();

    jPanel1.add (status, java.awt.BorderLayout.CENTER);
```

```
getContentPane ().add (jPanel1, java.awt.BorderLayout.CENTER);

jPanel2 = new javax.swing.JPanel ();

doGarbage = new javax.swing.JButton ();
doGarbage.addActionListener (new java.awt.event.ActionListener () {
  public void actionPerformed (java.awt.event.ActionEvent evt) {
    doGarbageActionPerformed (evt);
  }
}
);

jPanel2.add (doGarbage);

doRefresh = new javax.swing.JButton ();
doRefresh.addActionListener (new java.awt.event.ActionListener () {
  public void actionPerformed (java.awt.event.ActionEvent evt) {
    doRefreshActionPerformed (evt);
  }
}
);

jPanel2.add (doRefresh);

doClose = new javax.swing.JButton ();
doClose.addActionListener (new java.awt.event.ActionListener () {
  public void actionPerformed (java.awt.event.ActionEvent evt) {
    doCloseActionPerformed (evt);
  }
}
);

jPanel2.add (doClose);


getContentPane ().add (jPanel2, java.awt.BorderLayout.SOUTH);

jPanel3 = new javax.swing.JPanel ();
jPanel3.setLayout (new java.awt.BorderLayout (0, 20));

txtTime = new javax.swing.JLabel ();

jPanel3.add (txtTime, java.awt.BorderLayout.WEST);

time = new javax.swing.JTextField ();

jPanel3.add (time, java.awt.BorderLayout.CENTER);

doTime = new javax.swing.JButton ();
doTime.addActionListener (new java.awt.event.ActionListener () {
  public void actionPerformed (java.awt.event.ActionEvent evt) {
    setRefreshTime (evt);
  }
}
);

jPanel3.add (doTime, java.awt.BorderLayout.EAST);
```

```
    getContentPane ().add (jPanel3, java.awt.BorderLayout.NORTH);

  }//GEN-END:initComponents

  /** Exit the plugin */
  private void exitForm (java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
    this.dispose();
  }//GEN-LAST:event_exitForm

  private void setRefreshTime (java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_setRefreshTime
    try {
      int rate = Integer.valueOf (time.getText ()).intValue ();
      timer.setDelay (rate);
    } catch (NumberFormatException ex) {
      time.setText (String.valueOf (timer.getDelay ()));
    }
    time.selectAll ();
    time.requestFocus ();
  }//GEN-LAST:event_setRefreshTime


  private void doCloseActionPerformed (java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_doCloseActionPerformed
    exitForm (null);
  }//GEN-LAST:event_doCloseActionPerformed


  private void doRefreshActionPerformed (java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_doRefreshActionPerformed
    updateStatus ();
  }//GEN-LAST:event_doRefreshActionPerformed

  private void doGarbageActionPerformed (java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_doGarbageActionPerformed
    System.gc ();
    updateStatus ();
  }//GEN-LAST:event_doGarbageActionPerformed


  // Variables declaration - do not modify//GEN-BEGIN:variables
  private javax.swing.JPanel jPanel1;
  private javax.swing.JLabel text;
  private javax.swing.JProgressBar status;
  private javax.swing.JPanel jPanel2;
  private javax.swing.JButton doGarbage;
  private javax.swing.JButton doRefresh;
  private javax.swing.JButton doClose;
  private javax.swing.JPanel jPanel3;
  private javax.swing.JLabel txtTime;
  private javax.swing.JTextField time;
  private javax.swing.JButton doTime;
  // End of variables declaration//GEN-END:variables
}
```

jbb\system\CKnownBeans.java

```
package jbb.system;

import java.util.Hashtable;
import java.beans.Introspector;
import java.beans.BeanInfo;
import java.beans.IntrospectionException;

public class CKnownBeans {
//The Hashtable will be used to hold BeanInfo Objects(that implement BeanInfo
interface)
      public static Hashtable hash = new Hashtable();

      public CKnownBeans() {
//all default knownbeans must be added and also those added by the user
//use packages of beans like delphi

            BeanInfo BI;
//          ClassBrowser cb = new ClassBrowser();
            try {
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JPanel"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JPanel", BI);
System.out.println("OK Introspection of javax.swing.JPanel");
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JFrame"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JFrame", BI);
System.out.println("OK Introspection of javax.swing.JFrame");
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JApplet"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JApplet", BI);
System.out.println("OK Introspection of JApplet");
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JInternalFrame"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JInternalFrame", BI);
System.out.println("OK Introspection of javax.swing.JInternalFrame");
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JButton"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JButton", BI);
System.out.println("OK Introspection of JButton");
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JLabel"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JLabel", BI);
System.out.println("OK Introspection of JLabel");
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JComboBox"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JComboBox", BI);
System.out.println("OK Introspection of JComboBox");
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JTree"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JTree", BI);
System.out.println("OK Introspection of JTree");
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JList"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JList", BI);
System.out.println("OK Introspection of JList");
                  BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JPasswordField"), Introspector.USE_ALL_BEANINFO);
                  hash.put("javax.swing.JPasswordField", BI);
```

```
System.out.println("OK Introspection of JPasswordField");
                    BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JFileChooser"), Introspector.USE_ALL_BEANINFO);
                    hash.put("javax.swing.JFileChooser", BI);
System.out.println("OK Introspection of JFileChooser");
                    BI = Introspector.getBeanInfo(
Class.forName("javax.swing.JComponent"), Introspector.USE_ALL_BEANINFO);
                    hash.put("javax.swing.JComponent", BI);
System.out.println("OK Introspection of JComponent");
                    BI = Introspector.getBeanInfo( Class.forName("java.util.Vector"),
Class.forName("java.lang.Object"));
                    hash.put("java.util.Vector", BI);
System.out.println("OK Introspection of java.util.Vector");

/*
      System.out.println(cb.showClass(Class.forName("javax.swing.JFrame")));
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");
                    System.out.println("*******************************");


System.out.println(cb.showClass(Class.forName("javax.swing.JInternalFrame")));
*/
System.out.println("OK Introspection");
            } catch (IntrospectionException ie) {
System.out.println("IntrospectionException");
            } catch (ClassNotFoundException cnfe) {
System.out.println("ClassNotFoundException");
            }
      }//end constructor

}//end class
```

jbb\system\CSystem.java

```
package jbb.system;

import jbb.system.workspace.*;
import jbb.gui.*;

//import java.awt.*;
//import javax.swing.*;
//import javax.swing.text.*;
//import java.util.*;
//import java.beans.*;

public class CSystem {

	public static CWorkspace workspace;
	public static FMain fMain;
	public static CSystem system;

	public CSystem() {
		system = this;
		workspace = new CWorkspace();
		fMain = new FMain();
		fMain.setVisible(true);
	}

	public static void exit(int exitCode) {
System.out.println("Exiting normally");
		System.exit(exitCode);
	}
}
```

jbb\system\editors\ISourceDocument.java

```
package jbb.system.editors;

import javax.swing.text.*;

/**
 * This interface will be used by subclasses of AbstractDocument.
 * The methods defined here will be used to handle all types of
 * documents using named positions of the document for insert and
 * remove actions.
 */
public interface ISourceDocument {

      /**
       * A hashtable could be used to keep the associations between the
       * Positions and their names.
       *
       * <code>public Hashtable positions</code>;
       */

      /**
       * Inserts content after the specified position from
       * the positions Hashtable.
       *
       * @param pos  the specified position.
       * @return     returns <code>true<code> if <code>pos<code> was found
       *             and the insertion happend successfully.
       */
      public boolean insertAfterPos(String pos, String text);
      public boolean insertAfterPos(String pos, String text,
                  String newPosName, boolean overwrite);
      public void addPosition(String pos, Position posValue);
      public void addPosition(String pos, int offsetValue);

}
```

jbb\system\editors\html\CHTMLDocument.java

```
package jbb.system.editors.html;

import jbb.system.editors.*;

import java.awt.Color;
import java.io.*;
import javax.swing.event.*;
import javax.swing.text.*;
import java.util.*;
import javax.swing.text.html.*;

/**
 * A document to represent html source code.
 */
public class CHTMLDocument extends HTMLDocument implements ISourceDocument {

	/**
	 * This will be used for implementing the interface ISourceDocument
	 */
	private Hashtable positions = new Hashtable();
	/**
	 * Key to be used in AttributeSet's holding a value of CToken.
	 */
	static final Object CommentAttribute = new AttributeKey("comment");
	static final Object JBBMarkAttribute = new AttributeKey("JBBmark");

	public CHTMLDocument() {
		super(/*new GapContent()*/);
		positions.put("Start", this.getStartPosition());
		positions.put("End", this.getEndPosition());
	}

	/**
	 * Fetch a reasonable location to start scanning
	 * given the desired start location.  This allows
	 * for adjustments needed to accomodate multiline
	 * comments.
	 */
	public int getScannerStart(int p) {
		Element elem = getDefaultRootElement();
		int lineNum = elem.getElementIndex(p);
		Element line = elem.getElement(lineNum);
		AttributeSet a = line.getAttributes();
		while (a.isDefined(CommentAttribute) && lineNum > 0) {
			lineNum -= 1;
			line = elem.getElement(lineNum);
			a = line.getAttributes();
		}
		return line.getStartOffset();
	}
	/**
	 * Updates document structure as a result of text insertion.  This
	 * will happen within a write lock.  The superclass behavior of
	 * updating the line map is executed followed by marking any comment
	 * areas that should backtracked before scanning.
	 *
	 * @param chng the change event
	 * @param attr the set of attributes
```

```
	 */
	protected void insertUpdate(DefaultDocumentEvent chng, AttributeSet attr) {
		super.insertUpdate(chng, attr);
		// update comment marks
		Element root = getDefaultRootElement();
		DocumentEvent.ElementChange ec = chng.getChange(root);
		if (ec != null) {
			Element[] added = ec.getChildrenAdded();
			boolean inComment = false;
			for (int i = 0; i < added.length; i++) {
				Element elem = added[i];
				int p0 = elem.getStartOffset();
				int p1 = elem.getEndOffset();
				String s;
				try {
					s = getText(p0, p1 - p0);
				} catch (BadLocationException e) {
					s = null;
				}
				if (inComment) {
					MutableAttributeSet a = (MutableAttributeSet)
elem.getAttributes();
					a.addAttribute(CommentAttribute, CommentAttribute);
					int index = s.indexOf("*/");
					if (index >= 0) { // found an end of comment
						inComment = false;
					}
				} else { // scan for multiline comment
					int index = s.indexOf("/*||");
					if (index >= 0) { // found a JBBMark
						MutableAttributeSet a =
(MutableAttributeSet)elem.getAttributes();
						a.addAttribute(JBBMarkAttribute,
JBBMarkAttribute);
					}
					index = s.indexOf("/*");
					if (index >= 0) { // found a start of comment, see if
it spans multiple lines
						MutableAttributeSet a = (MutableAttributeSet)
elem.getAttributes();
						a.addAttribute(CommentAttribute,
CommentAttribute);
						index = s.indexOf("*/", index);
						if (index < 0) { // it spans multiple lines
							inComment = true;
						}
					}
				}
			}
		}

	}
	/**
	 * Updates any document structure as a result of text removal.
	 * This will happen within a write lock.  The superclass behavior of
	 * updating the line map is executed followed by placing a lexical
	 * update command on the analyzer queue.
	 *
	 * @param chng the change event
	 */
	protected void removeUpdate(DefaultDocumentEvent chng) {
```

```
			super.removeUpdate(chng);
			// update comment marks(not the same as insertUpdate)
		}

		// Implementation of ISourceDocument.

		public boolean insertAfterPos(String pos, String text) {
			Object obj = positions.get(pos);
			if (obj==null) {
				return false;
			}
			Position p = (Position)obj;
			int offset =  p.getOffset();
			if ((pos=="End") && (offset>0)) {
				offset--;
			}
			SimpleAttributeSet sas = new SimpleAttributeSet();
			//sas.addAttribute(CommentAttribute, CommentAttribute);
			try {
				this.insertString(offset, text, sas);
				return true;
			} catch (BadLocationException e) {
				return false;
			}
		}
		public boolean insertAfterPos(String pos, String text,
				String newPosName, boolean overwrite) {
			Object obj = positions.get(pos);
			if (obj==null) {
				return false;
			}
			Position p = (Position)obj;
			obj = positions.get(newPosName+"Start");
			int offsetStart = -1;
			int offsetEnd = -1;
			if (overwrite) {
				if  (obj!=null) {
					offsetStart = ((Position)obj).getOffset();
					obj = positions.get(newPosName+"End");
					offsetEnd = ((Position)obj).getOffset();
					try {
						this.remove(offsetStart, Math.abs(offsetEnd -
offsetStart));
					} catch (BadLocationException e) {
						return false;
					}
				}
			} else if (obj!=null)
				return true;
			int offset;
			if (offsetStart > -1) {
				offset = offsetStart;
			} else {
				offset = p.getOffset();
			}
			SimpleAttributeSet sas = new SimpleAttributeSet();
			try {
				this.insertString(offset, text, sas);
				addPosition(newPosName + "Start", offset);
				addPosition(newPosName + "End", offset + text.length());//check
to see if it is correct
```

```
                return true;
            } catch (BadLocationException e) {
                return false;
            }
        }
        public void addPosition(String pos, Position posValue) {
            positions.put(pos, posValue);
        }
        public void addPosition(String pos, int offsetValue) {
            try {
                addPosition(pos, createPosition(offsetValue));
            } catch (BadLocationException e) {
                //something usefull
            }
        }

        // Inner classes.

        static class AttributeKey {
            String attribute;
            private AttributeKey(String attr) {
                attribute = attr;
            }
            public String toString() {
                return attribute;
            }
        }

}
```

jbb\system\editors\java\CJavaContext.java

```
package jbb.system.editors.java;

import java.awt.*;
import java.util.Vector;
import javax.swing.text.*;

/**
 * A collection of styles used to render java text.
 * This class also acts as a factory for the views used
 * to represent the java documents.  Since the rendering
 * styles are based upon view preferences, the views need
 * a way to gain access to the style settings which is
 * facilitated by implementing the factory in the style
 * storage.  Both functionalities can be widely shared across
 * java document views.
 */
public class CJavaContext extends StyleContext implements ViewFactory {
	/**
	 * The styles representing the actual token types.
	 */
	Style[] tokenStyles;
	/**
	 * Cache of foreground colors to represent the
	 * various tokens.
	 */
	transient Color[] tokenColors;
	/**
	 * Cache of fonts to represent the various tokens.
	 */
	transient Font[] tokenFonts;
	/**
	 * Constructs a set of styles to represent java lexical
	 * tokens.  By default there are no colors or fonts specified.
	 */
	public CJavaContext() {
		super();
		Style root = getStyle(DEFAULT_STYLE);
		tokenStyles = new Style[CToken.MaximumScanValue + 1];
		CToken[] tokens = CToken.all;
		int n = tokens.length;
		for (int i = 0; i < n; i++) {
			CToken t = tokens[i];
			Style parent = getStyle(t.getCategory());
			if (parent == null) {
				parent = addStyle(t.getCategory(), root);
			}
			Style s = addStyle(null, parent);
			s.addAttribute(CToken.TokenAttribute, t);
			tokenStyles[t.getScanValue()] = s;
		}
	}
	/**
	 * Fetch the foreground color to use for a lexical
	 * token with the given value.
	 *
	 * @param attr attribute set from a token element
	 *  that has a Token in the set.
	 */
```

```
	public Color getForeground(int code) {
		if (tokenColors == null) {
			tokenColors = new Color[CToken.MaximumScanValue + 1];
		}
		if ((code >= 0) && (code < tokenColors.length)) {
			Color c = tokenColors[code];
			if (c == null) {
				Style s = tokenStyles[code];
				c = StyleConstants.getForeground(s);
			}
			return c;
		}
		return Color.black;
	}
	/**
	 * Fetch the font to use for a lexical
	 * token with the given scan value.
	 */
	public Font getFont(int code) {
		if (tokenFonts == null) {
			tokenFonts = new Font[CToken.MaximumScanValue + 1];
		}
		if (code < tokenFonts.length) {
			Font f = tokenFonts[code];
			if (f == null) {
				Style s = tokenStyles[code];
				f = getFont(s);
			}
			return f;
		}
		return null;
	}
	/**
	 * Fetches the attribute set to use for the given
	 * scan code.  The set is stored in a table to
	 * facilitate relatively fast access to use in
	 * conjunction with the scanner.
	 */
	public Style getStyleForScanValue(int code) {
		if (code < tokenStyles.length) {
			return tokenStyles[code];
		}
		return null;
	}

	// Implementation of ViewFactory methods
	public View create(Element elem) {
		return new CJavaView(elem);
	}

	/**
	 * View that uses the lexical information to determine the
	 * style characteristics of the text that it renders.  This
	 * simply colorizes the various tokens and assumes a constant
	 * font family and size.
	 */
	class CJavaView extends WrappedPlainView {

		CJavaDocument.CScanner lexer;
		boolean lexerValid;
```

```
		/**
		 * Construct a simple colorized view of java
		 * text.
		 */
		CJavaView(Element elem) {
			super(elem);
			CJavaDocument doc = (CJavaDocument)getDocument();
			lexer = doc.createScanner();
//			lexerValid = false;
		}
		/**
		 * Renders using the given rendering surface and area
		 * on that surface.  This is implemented to invalidate
		 * the lexical scanner after rendering so that the next
		 * request to drawUnselectedText will set a new range
		 * for the scanner.
		 *
		 * @param g the rendering surface to use
		 * @param a the allocated region to render into
		 *
		 * @see View#paint
		 */
		public void paint(Graphics g, Shape a) {
			super.paint(g, a);
			lexerValid = false;
		}
		/**
		* Renders the given range in the model as normal unselected
		* text.  This is implemented to paint colors based upon the
		* token-to-color translations.  To reduce the number of calls
		* to the Graphics object, text is batched up until a color
		* change is detected or the entire requested range has been
		* reached.
		*
		* @param g the graphics context
		* @param x the starting X coordinate
		* @param y the starting Y coordinate
		* @param p0 the beginning position in the model
		* @param p1 the ending position in the model
		* @returns the location of the end of the range
		* @exception BadLocationException if the range is invalid
		*/
		protected int drawUnselectedText(Graphics g, int x, int y,
				int p0, int p1) throws BadLocationException {
			//i should take the segment from p0 and p1 and continue the
proccessing
			Document doc = getDocument();
//System.out.println(doc);
//System.out.println(p0 + " -- " + p1);
//System.out.println(getElement().getAttributes().isDefined(CJavaDocument.JBBMarkAt
tribute));
//System.out.println(getElement().getAttributes().getAttributeNames());
			Color last = null;
			int mark = p0;
			for (; p0 < p1; ) {
				updateScanner(p0);
				int p = Math.min(lexer.getEndOffset(), p1);
				p = (p <= p0) ? p1 : p;
				Color fg = getForeground(lexer.token);
				if (fg != last && last != null) {
					// color change, flush what we have
```

```
                                    g.setColor(last);
                                    Segment text = getLineBuffer();
                                    if (text.toString().indexOf("/*||") > 0) {
                                          g.setColor(Color.lightGray);
//g.setColor(JBBMarkColor);
                                    }
//System.out.println("1-"+text);
                                    doc.getText(mark, p0 - mark, text);
//System.out.println("2-"+text);
                                    x = Utilities.drawTabbedText(text, x, y, g, this,
mark);
                                    mark = p0;
                              }
                              last = fg;
                              p0 = p;
                        }
                        // flush remaining
                        g.setColor(last);
                        Segment text = getLineBuffer();
                        if (text.toString().indexOf("/*||") > 0) {
                              g.setColor(Color.lightGray); //g.setColor(JBBMarkColor);
                        }
                        doc.getText(mark, p1 - mark, text);
                        x = Utilities.drawTabbedText(text, x, y, g, this, mark);
                        return x;
                  }
                  /**
                   * Update the scanner (if necessary) to point to the appropriate
                   * token for the given start position needed for rendering.
                   */
                  void updateScanner(int p) {
                        try {
                              if (!lexerValid) {
                                    CJavaDocument doc = (CJavaDocument)getDocument();
                                    lexer.setRange(doc.getScannerStart(p),
doc.getLength());
                                    lexerValid = true;
                              }
                              while (lexer.getEndOffset() <= p) {
                                    lexer.scan();
                              }
                        } catch (Throwable e) {
                              // can't adjust scanner... calling logic
                              // will simply render the remaining text.
//e.printStackTrace();
                        }
                  }
            }
}
```

jbb\system\editors\java\CJavaDocument.java

```
package jbb.system.editors.java;

import jbb.system.editors.*;

import java.awt.Color;
import java.io.*;
import javax.swing.event.*;
import javax.swing.text.*;
import java.util.*;

/**
 * A document to represent java source code.
 */
public class CJavaDocument extends PlainDocument implements ISourceDocument {

	/**
	 * This will be used for implementing the interface ISourceDocument
	 */
	private Hashtable positions = new Hashtable();
	/**
	 * Key to be used in AttributeSet's holding a value of CToken.
	 */
	static final Object CommentAttribute = new AttributeKey("comment");
	static final Object JBBMarkAttribute = new AttributeKey("JBBmark");

	public CJavaDocument() {
		super(new GapContent());
		positions.put("Start", this.getStartPosition());
		positions.put("End", this.getEndPosition());
	}
	/**
	 * Create a lexical analyzer for this document.
	 */
	public CScanner createScanner() {
		CScanner s;
		try {
			s = new CScanner();
		} catch (IOException e) {
			s = null;
		}
		return s;
	}
	/**
	 * Fetch a reasonable location to start scanning
	 * given the desired start location.  This allows
	 * for adjustments needed to accomodate multiline
	 * comments.
	 */
	public int getScannerStart(int p) {
		Element elem = getDefaultRootElement();
		int lineNum = elem.getElementIndex(p);
		Element line = elem.getElement(lineNum);
		AttributeSet a = line.getAttributes();
		while (a.isDefined(CommentAttribute) && lineNum > 0) {
			lineNum -= 1;
			line = elem.getElement(lineNum);
			a = line.getAttributes();
		}
		return line.getStartOffset();
```

```
	}
	/**
	 * Updates document structure as a result of text insertion.  This
	 * will happen within a write lock.  The superclass behavior of
	 * updating the line map is executed followed by marking any comment
	 * areas that should backtracked before scanning.
	 *
	 * @param chng the change event
	 * @param attr the set of attributes
	 */
	protected void insertUpdate(DefaultDocumentEvent chng, AttributeSet attr) {
		super.insertUpdate(chng, attr);
		// update comment marks
		Element root = getDefaultRootElement();
		DocumentEvent.ElementChange ec = chng.getChange(root);
		if (ec != null) {
			Element[] added = ec.getChildrenAdded();
			boolean inComment = false;
			for (int i = 0; i < added.length; i++) {
				Element elem = added[i];
				int p0 = elem.getStartOffset();
				int p1 = elem.getEndOffset();
				String s;
				try {
					s = getText(p0, p1 - p0);
				} catch (BadLocationException e) {
					s = null;
				}
				if (inComment) {
					MutableAttributeSet a = (MutableAttributeSet)
elem.getAttributes();
					a.addAttribute(CommentAttribute, CommentAttribute);
					int index = s.indexOf("*/");
					if (index >= 0) { // found an end of comment
						inComment = false;
					}
				} else { // scan for multiline comment
					int index = s.indexOf("/*||");
					if (index >= 0) { // found a JBBMark
						MutableAttributeSet a =
(MutableAttributeSet)elem.getAttributes();
						a.addAttribute(JBBMarkAttribute,
JBBMarkAttribute);
					}
					index = s.indexOf("/*");
					if (index >= 0) { // found a start of comment, see if
it spans multiple lines
						MutableAttributeSet a = (MutableAttributeSet)
elem.getAttributes();
						a.addAttribute(CommentAttribute,
CommentAttribute);
						index = s.indexOf("*/", index);
						if (index < 0) { // it spans multiple lines
							inComment = true;
						}
					}
				}
			}
		}

	}
```

```
	/**
	 * Updates any document structure as a result of text removal.
	 * This will happen within a write lock.  The superclass behavior of
	 * updating the line map is executed followed by placing a lexical
	 * update command on the analyzer queue.
	 *
	 * @param chng the change event
	 */
	protected void removeUpdate(DefaultDocumentEvent chng) {
		super.removeUpdate(chng);
		// update comment marks(not the same as insertUpdate)
	}

	// Implementation of ISourceDocument.

	public boolean insertAfterPos(String pos, String text) {
		Object obj = positions.get(pos);
		if (obj==null) {
			return false;
		}
		Position p = (Position)obj;
		int offset =  p.getOffset();
		if ((pos=="End") && (offset>0)) {
			offset--;
		}
		SimpleAttributeSet sas = new SimpleAttributeSet();
		//sas.addAttribute(CommentAttribute, CommentAttribute);
		try {
			this.insertString(offset, text, sas);
			return true;
		} catch (BadLocationException e) {
			return false;
		}
	}
	public boolean insertAfterPos(String pos, String text,
			String newPosName, boolean overwrite) {
		Object obj = positions.get(pos);
		if (obj==null) {
			return false;
		}
		Position p = (Position)obj;
		obj = positions.get(newPosName+"Start");
		int offsetStart = -1;
		int offsetEnd = -1;
		if (overwrite) {
			if  (obj!=null) {
				offsetStart = ((Position)obj).getOffset();
				obj = positions.get(newPosName+"End");
				offsetEnd = ((Position)obj).getOffset();
				try {
					this.remove(offsetStart, Math.abs(offsetEnd -
offsetStart));
				} catch (BadLocationException e) {
					return false;
				}
			}
		} else if (obj!=null)
			return true;
		int offset;
		if (offsetStart > -1) {
			offset = offsetStart;
```

```
			} else {
				offset = p.getOffset();
			}
			SimpleAttributeSet sas = new SimpleAttributeSet();
			try {
				this.insertString(offset, text, sas);
				addPosition(newPosName + "Start", offset);
				addPosition(newPosName + "End", offset + text.length());//check
to see if it is correct
				return true;
			} catch (BadLocationException e) {
				return false;
			}
		}
		public void addPosition(String pos, Position posValue) {
			positions.put(pos, posValue);
		}
		public void addPosition(String pos, int offsetValue) {
			try {
				addPosition(pos, createPosition(offsetValue));
			} catch (BadLocationException e) {
				//something usefull
			}
		}

		// Inner classes.

		static class AttributeKey {
			String attribute;
			private AttributeKey(String attr) {
				attribute = attr;
			}
			public String toString() {
				return attribute;
			}
		}

		public class CScanner extends sun.tools.java.Scanner {

			int p0 = 0;

			private CScanner() throws IOException {
				super(new LocalEnvironment());
				scanComments = true;
//				CScanner.
			}
			/**
			 * Sets the range of the scanner.  This should be called
			 * to reinitialize the scanner to the desired range.
			 */
			public void setRange(int newP0, int newP1) throws IOException {
				useInputStream(new DocumentInputStream(newP0, newP1));
				p0 = newP0;
			}
			/**
			 * This fetches the starting location of the current
			 * CToken in the document.
			 */
			public final int getStartOffset() {
				int begOffs = (int)(pos & MAXFILESIZE);
				return p0 + begOffs;
```

```
	}
	/**
	 * This fetches the ending location of the current
	 * token in the document.
	 */
	public final int getEndOffset() {
		int endOffs = (int)(getEndPos() & MAXFILESIZE);
		return p0 + endOffs;
	}
}

/**
 * Class to provide InputStream functionality from a portion of a
 * Document. (Would it be better if it was a Reader?)
 */
class DocumentInputStream extends InputStream {

	Segment segment;
	int p0;    // start position
	int p1;    // end position
	int pos;   // pos in document
	int index; // index into array of the segment

	public DocumentInputStream(int newP0, int newP1) {
		this.segment = new Segment();
		p0 = newP0;
		p1 = Math.min(getLength(), newP1);
		pos = p0;
		try {
			loadSegment();
		} catch (IOException ioe) {
			// throw new Error("unexpected: " + ioe);
		}
	}
	/**
	 * Reads the next byte of data from this input stream. The value
	 * byte is returned as an <code>int</code> in the range
	 * <code>0</code> to <code>255</code>. If no byte is available
	 * because the end of the stream has been reached, the value
	 * <code>-1</code> is returned. This method blocks until input data
	 * is available, the end of the stream is detected, or an exception
	 * is thrown.
	 *
	 * @return     the next byte of data, or <code>-1</code> if the
	 *             end of the stream is reached.
	 * @exception  IOException  if an I/O error occurs.
	 */
	public int read() throws IOException {
		if (index >= segment.offset + segment.count) {
			if (pos >= p1) {
				// no more data
				return -1;
			}
			loadSegment();
		}
		return segment.array[index++];
	}
	private void loadSegment() throws IOException {
		try {
			int n = Math.min(1024, p1 - pos);
			getText(pos, n, segment);
```

```
                        pos += n;
                        index = segment.offset;
                    } catch (BadLocationException e) {
                        throw new IOException("Bad location");
                    }
                }
            }

            static class LocalEnvironment extends sun.tools.java.Environment {
                public void error(Object source, int where, String err,
                        Object arg1, Object arg2, Object arg3) {
                    // should do something useful...
                    System.err.println(err);
                    System.err.println("location: " + where);
                }
            }

}
```

jbb\system\editors\java\CJavaEditorKit.java

```
package jbb.system.editors.java;

import javax.swing.text.*;

/**
 * This kit supports a fairly minimal handling of
 * editing java text content.  It supports syntax
 * highlighting and produces the lexical structure
 * of the document as best it can.
 */
public class CJavaEditorKit extends DefaultEditorKit {

	CJavaContext preferences;

	public CJavaEditorKit() {
		super();
	}
	public CJavaContext getStylePreferences() {
		if (preferences == null) {
			preferences = new CJavaContext();
		}
		return preferences;
	}
	public void setStylePreferences(CJavaContext prefs) {
		preferences = prefs;
	}
	/**
	 * Get the MIME type of the data that this
	 * kit represents support for.  This kit supports
	 * the type <code>text/java</code>.
	 */
	public String getContentType() {
		return "text/java";
	}
	/**
	 * Create a copy of the editor kit.  This
	 * allows an implementation to serve as a prototype
	 * for others, so that they can be quickly created.
	 */
	public Object clone() {
		CJavaEditorKit kit = new CJavaEditorKit();
		kit.preferences = preferences;
		return kit;
	}
	/**
	 * Creates an uninitialized text storage model
	 * that is appropriate for this type of editor.
	 *
	 * @return the model
	 */
	public Document createDefaultDocument() {
		return new CJavaDocument();
	}
	/**
	 * Fetches a factory that is suitable for producing
	 * views of any models that are produced by this
	 * kit.  The default is to have the UI produce the
	 * factory, so this method has no implementation.
```

```
         *
         * @return the view factory
         */
        public final ViewFactory getViewFactory() {
              return getStylePreferences();
        }
}
```

jbb\system\editors\java\CToken.java

```
package jbb.system.editors.java;

import java.io.Serializable;
import sun.tools.java.Constants;

public class CToken implements Serializable {

	public static final int MaximumScanValue = Constants.INLINENEWINSTANCE + 1;
	/**
	 * Key to be used in AttributeSet's holding a value of CToken.
	 */
	public static final Object TokenAttribute = new AttributeKey();

	private String representation;
	private int scanValue;
	private String tokenType = "";

	CToken(String newRepresentation, int newScanValue, String newTokenType) {
		representation = newRepresentation;
		scanValue = newScanValue;
		tokenType = newTokenType;
	}
	/**
	 * Numeric value of this CToken.  This is the value
	 * returned by the scanner and is the tie between
	 * the lexical scanner and the tokens.
	 */
	 public int getScanValue() {
		return scanValue;
	 }
	/**
	 * Specifies the category of the CToken as a
	 * string that can be used as a label.
	 */
	public String getCategory() {
		return tokenType;
	}
	/**
	 * Returns a hashcode for this set of attributes.
	 * @return	a hashcode value for this set of attributes.
	 */
	public final int hashCode() {
		return scanValue;
	}
	/**
	 * Compares this object to the specifed object.
	 * The result is <code>true</code> if and only if the argument is not
	 * <code>null</code> and is a <code>CToken</code> object with the same
	 * scanValue.
	 * @param	obj	the object to compare this CToken with.
	 * @return	<code>true</code> if the objects are equal;
	 *		<code>false</code> otherwise.
	 */
	public final boolean equals(Object obj) {
		if (obj instanceof CToken)
			return (scanValue == ((CToken)obj).scanValue);
		else
			return false;
```

```
	}
	public String toString() {
		return representation;
	}


	static class AttributeKey {
		private AttributeKey() {
		}
		public String toString() {
			return "ctoken";
		}
	}

	//All CTokens

	/*
	 * Operators
	 */
	public static final CToken COMMA =
			new CToken(Constants.opNames[Constants.COMMA], Constants.COMMA,
"Operator");
	public static final CToken ASSIGN =
			new CToken(Constants.opNames[Constants.ASSIGN], Constants.ASSIGN,
"Operator");
	public static final CToken ASGMUL =
			new CToken(Constants.opNames[Constants.ASGMUL], Constants.ASGMUL,
"Operator");
	public static final CToken ASGDIV =
			new CToken(Constants.opNames[Constants.ASGDIV], Constants.ASGDIV,
"Operator");
	public static final CToken ASGREM =
			new CToken(Constants.opNames[Constants.ASGREM], Constants.ASGREM,
"Operator");
	public static final CToken ASGADD =
			new CToken(Constants.opNames[Constants.ASGADD], Constants.ASGADD,
"Operator");
	public static final CToken ASGSUB =
			new CToken(Constants.opNames[Constants.ASGSUB], Constants.ASGSUB,
"Operator");
	public static final CToken ASGLSHIFT =
			new CToken(Constants.opNames[Constants.ASGLSHIFT],
Constants.ASGLSHIFT, "Operator");
	public static final CToken ASGRSHIFT =
			new CToken(Constants.opNames[Constants.ASGRSHIFT],
Constants.ASGRSHIFT, "Operator");
	public static final CToken ASGURSHIFT =
			new CToken(Constants.opNames[Constants.ASGURSHIFT],
Constants.ASGURSHIFT, "Operator");
	public static final CToken ASGBITAND =
			new CToken(Constants.opNames[Constants.ASGBITAND],
Constants.ASGBITAND, "Operator");
	public static final CToken ASGBITOR =
			new CToken(Constants.opNames[Constants.ASGBITOR],
Constants.ASGBITOR, "Operator");
	public static final CToken ASGBITXOR =
			new CToken(Constants.opNames[Constants.ASGBITOR],
Constants.ASGBITOR, "Operator");
	public static final CToken COND =
			new CToken(Constants.opNames[Constants.COND], Constants.COND,
"Operator");
```

```
	public static final CToken OR =
			new CToken(Constants.opNames[Constants.OR], Constants.OR,
"Operator");
	public static final CToken AND =
			new CToken(Constants.opNames[Constants.AND], Constants.AND,
"Operator");
	public static final CToken BITOR =
			new CToken(Constants.opNames[Constants.BITOR], Constants.BITOR,
"Operator");
	public static final CToken BITXOR =
			new CToken(Constants.opNames[Constants.BITXOR], Constants.BITXOR,
"Operator");
	public static final CToken BITAND =
			new CToken(Constants.opNames[Constants.BITAND], Constants.BITAND,
"Operator");
	public static final CToken NE =
			new CToken(Constants.opNames[Constants.NE], Constants.NE,
"Operator");
	public static final CToken EQ =
			new CToken(Constants.opNames[Constants.EQ], Constants.EQ,
"Operator");
	public static final CToken GE =
			new CToken(Constants.opNames[Constants.GE], Constants.GE,
"Operator");
	public static final CToken GT =
			new CToken(Constants.opNames[Constants.GT], Constants.GT,
"Operator");
	public static final CToken LE =
			new CToken(Constants.opNames[Constants.LE], Constants.LE,
"Operator");
	public static final CToken LT =
			new CToken(Constants.opNames[Constants.LT], Constants.LT,
"Operator");
	public static final CToken INSTANCEOF =
			new CToken(Constants.opNames[Constants.INSTANCEOF],
Constants.INSTANCEOF, "Operator");
	public static final CToken LSHIFT =
			new CToken(Constants.opNames[Constants.LSHIFT], Constants.LSHIFT,
"Operator");
	public static final CToken RSHIFT =
			new CToken(Constants.opNames[Constants.RSHIFT], Constants.RSHIFT,
"Operator");
	public static final CToken URSHIFT =
			new CToken(Constants.opNames[Constants.URSHIFT],
Constants.URSHIFT, "Operator");
	public static final CToken ADD =
			new CToken(Constants.opNames[Constants.ADD], Constants.ADD,
"Operator");
	public static final CToken SUB =
			new CToken(Constants.opNames[Constants.SUB], Constants.SUB,
"Operator");
	public static final CToken DIV =
			new CToken(Constants.opNames[Constants.DIV], Constants.DIV,
"Operator");
	public static final CToken REM =
			new CToken(Constants.opNames[Constants.REM], Constants.REM,
"Operator");
	public static final CToken MUL =
			new CToken(Constants.opNames[Constants.MUL], Constants.MUL,
"Operator");
	public static final CToken CAST =
```

```
                  new CToken(Constants.opNames[Constants.CAST], Constants.CAST,
"Operator");
      public static final CToken POS =
                  new CToken(Constants.opNames[Constants.POS], Constants.POS,
"Operator");
      public static final CToken NEG =
                  new CToken(Constants.opNames[Constants.NEG], Constants.NEG,
"Operator");
      public static final CToken NOT =
                  new CToken(Constants.opNames[Constants.NOT], Constants.NOT,
"Operator");
      public static final CToken BITNOT =
                  new CToken(Constants.opNames[Constants.BITNOT], Constants.BITNOT,
"Operator");
      public static final CToken PREINC =
                  new CToken(Constants.opNames[Constants.PREINC], Constants.PREINC,
"Operator");
      public static final CToken PREDEC =
                  new CToken(Constants.opNames[Constants.PREDEC], Constants.PREDEC,
"Operator");
      public static final CToken NEWARRAY =
                  new CToken(Constants.opNames[Constants.NEWARRAY],
Constants.NEWARRAY, "Operator");
      public static final CToken NEWINSTANCE =
                  new CToken(Constants.opNames[Constants.NEWINSTANCE],
Constants.NEWINSTANCE, "Operator");
      public static final CToken NEWFROMNAME =
                  new CToken(Constants.opNames[Constants.NEWFROMNAME],
Constants.NEWFROMNAME, "Operator");
      public static final CToken POSTINC =
                  new CToken(Constants.opNames[Constants.POSTINC],
Constants.POSTINC, "Operator");
      public static final CToken POSTDEC =
                  new CToken(Constants.opNames[Constants.POSTDEC],
Constants.POSTDEC, "Operator");
      public static final CToken FIELD =
                  new CToken(Constants.opNames[Constants.FIELD], Constants.FIELD,
"Operator");
      public static final CToken METHOD =
                  new CToken(Constants.opNames[Constants.METHOD], Constants.METHOD,
"Operator");
      public static final CToken ARRAYACCESS =
                  new CToken(Constants.opNames[Constants.ARRAYACCESS],
Constants.ARRAYACCESS, "Operator");
      public static final CToken NEW =
                  new CToken(Constants.opNames[Constants.NEW], Constants.NEW,
"Operator");
      public static final CToken INC =
                  new CToken(Constants.opNames[Constants.INC], Constants.INC,
"Operator");
      public static final CToken DEC =
                  new CToken(Constants.opNames[Constants.DEC], Constants.DEC,
"Operator");
      public static final CToken CONVERT =
                  new CToken(Constants.opNames[Constants.CONVERT],
Constants.CONVERT, "Operator");
      public static final CToken EXPR =
                  new CToken(Constants.opNames[Constants.EXPR], Constants.EXPR,
"Operator");
      public static final CToken ARRAY =
```

```
                  new CToken(Constants.opNames[Constants.ARRAY], Constants.ARRAY,
"Operator");
      public static final CToken GOTO =
                  new CToken(Constants.opNames[Constants.GOTO], Constants.GOTO,
"Operator");
      /*
       * Value tokens
       */
      public static final CToken IDENT =
                  new CToken(Constants.opNames[Constants.IDENT], Constants.IDENT,
"Value");
      public static final CToken BOOLEANVAL =
                  new CToken(Constants.opNames[Constants.BOOLEANVAL],
Constants.BOOLEANVAL, "Value");
      public static final CToken BYTEVAL =
                  new CToken(Constants.opNames[Constants.BYTEVAL],
Constants.BYTEVAL, "Value");
      public static final CToken CHARVAL =
                  new CToken(Constants.opNames[Constants.CHARVAL],
Constants.CHARVAL, "Value");
      public static final CToken SHORTVAL =
                  new CToken(Constants.opNames[Constants.SHORTVAL],
Constants.SHORTVAL, "Value");
      public static final CToken INTVAL =
                  new CToken(Constants.opNames[Constants.INTVAL], Constants.INTVAL,
"Value");
      public static final CToken LONGVAL =
                  new CToken(Constants.opNames[Constants.LONGVAL],
Constants.LONGVAL, "Value");
      public static final CToken FLOATVAL =
                  new CToken(Constants.opNames[Constants.FLOATVAL],
Constants.FLOATVAL, "Value");
      public static final CToken DOUBLEVAL =
                  new CToken(Constants.opNames[Constants.DOUBLEVAL],
Constants.DOUBLEVAL, "Value");
      public static final CToken STRINGVAL =
                  new CToken(Constants.opNames[Constants.STRINGVAL],
Constants.STRINGVAL, "Value");
      /*
       * Type keywords
       */
      public static final CToken BYTE =
                  new CToken(Constants.opNames[Constants.BYTE], Constants.BYTE,
"Type");
      public static final CToken CHAR =
                  new CToken(Constants.opNames[Constants.CHAR], Constants.CHAR,
"Type");
      public static final CToken SHORT =
                  new CToken(Constants.opNames[Constants.SHORT], Constants.SHORT,
"Type");
      public static final CToken INT =
                  new CToken(Constants.opNames[Constants.INT], Constants.INT,
"Type");
      public static final CToken LONG =
                  new CToken(Constants.opNames[Constants.LONG], Constants.LONG,
"Type");
      public static final CToken FLOAT =
                  new CToken(Constants.opNames[Constants.FLOAT], Constants.FLOAT,
"Type");
      public static final CToken DOUBLE =
```

```
                  new CToken(Constants.opNames[Constants.DOUBLE], Constants.DOUBLE,
"Type");
      public static final CToken VOID =
                  new CToken(Constants.opNames[Constants.VOID], Constants.VOID,
"Type");
      public static final CToken BOOLEAN =
                  new CToken(Constants.opNames[Constants.BOOLEAN],
Constants.BOOLEAN, "Type");
      /*
       * Expression keywords
       */
      public static final CToken TRUE =
                  new CToken(Constants.opNames[Constants.TRUE], Constants.TRUE,
"Expression");
      public static final CToken FALSE =
                  new CToken(Constants.opNames[Constants.FALSE], Constants.FALSE,
"Expression");
      public static final CToken THIS =
                  new CToken(Constants.opNames[Constants.THIS], Constants.THIS,
"Expression");
      public static final CToken SUPER =
                  new CToken(Constants.opNames[Constants.SUPER], Constants.SUPER,
"Expression");
      public static final CToken NULL =
                  new CToken(Constants.opNames[Constants.NULL], Constants.NULL,
"Expression");
      /*
       * Statement keywords
       */
      public static final CToken IF =
                  new CToken(Constants.opNames[Constants.IF], Constants.IF,
"Statement");
      public static final CToken ELSE =
                  new CToken(Constants.opNames[Constants.ELSE], Constants.ELSE,
"Statement");
      public static final CToken FOR =
                  new CToken(Constants.opNames[Constants.FOR], Constants.FOR,
"Statement");
      public static final CToken WHILE =
                  new CToken(Constants.opNames[Constants.WHILE], Constants.WHILE,
"Statement");
      public static final CToken DO =
                  new CToken(Constants.opNames[Constants.DO], Constants.DO,
"Statement");
      public static final CToken SWITCH =
                  new CToken(Constants.opNames[Constants.SWITCH], Constants.SWITCH,
"Statement");
      public static final CToken CASE =
                  new CToken(Constants.opNames[Constants.CASE], Constants.CASE,
"Statement");
      public static final CToken DEFAULT =
                  new CToken(Constants.opNames[Constants.DEFAULT],
Constants.DEFAULT, "Statement");
      public static final CToken BREAK =
                  new CToken(Constants.opNames[Constants.BREAK], Constants.BREAK,
"Statement");
      public static final CToken CONTINUE =
                  new CToken(Constants.opNames[Constants.CONTINUE],
Constants.CONTINUE, "Statement");
      public static final CToken RETURN =
```

```
                  new CToken(Constants.opNames[Constants.RETURN], Constants.RETURN,
"Statement");
      public static final CToken TRY =
                  new CToken(Constants.opNames[Constants.TRY], Constants.TRY,
"Statement");
      public static final CToken CATCH =
                  new CToken(Constants.opNames[Constants.CATCH], Constants.CATCH,
"Statement");
      public static final CToken FINALLY =
                  new CToken(Constants.opNames[Constants.FINALLY],
Constants.FINALLY, "Statement");
      public static final CToken THROW =
                  new CToken(Constants.opNames[Constants.THROW], Constants.THROW,
"Statement");
      public static final CToken STAT =
                  new CToken(Constants.opNames[Constants.STAT], Constants.STAT,
"Statement");
      public static final CToken EXPRESSION =
                  new CToken(Constants.opNames[Constants.EXPRESSION],
Constants.EXPRESSION, "Statement");
      public static final CToken DECLARATION =
                  new CToken(Constants.opNames[Constants.DECLARATION],
Constants.DECLARATION, "Statement");
      public static final CToken VARDECLARATION =
                  new CToken(Constants.opNames[Constants.VARDECLARATION],
Constants.VARDECLARATION, "Statement");
      /*
       * Declaration keywords
       */
      public static final CToken IMPORT =
                  new CToken(Constants.opNames[Constants.IMPORT], Constants.IMPORT,
"Declaration");
      public static final CToken CLASS =
                  new CToken(Constants.opNames[Constants.CLASS], Constants.CLASS,
"Declaration");
      public static final CToken EXTENDS =
                  new CToken(Constants.opNames[Constants.EXTENDS],
Constants.EXTENDS, "Declaration");
      public static final CToken IMPLEMENTS =
                  new CToken(Constants.opNames[Constants.IMPLEMENTS],
Constants.IMPLEMENTS, "Declaration");
      public static final CToken INTERFACE =
                  new CToken(Constants.opNames[Constants.INTERFACE],
Constants.INTERFACE, "Declaration");
      public static final CToken PACKAGE =
                  new CToken(Constants.opNames[Constants.PACKAGE],
Constants.PACKAGE, "Declaration");
      /*
       * Modifier keywords
       */
      public static final CToken PRIVATE =
                  new CToken(Constants.opNames[Constants.PRIVATE],
Constants.PRIVATE, "Modifier");
      public static final CToken PUBLIC =
                  new CToken(Constants.opNames[Constants.PUBLIC], Constants.PUBLIC,
"Modifier");
      public static final CToken PROTECTED =
                  new CToken(Constants.opNames[Constants.PROTECTED],
Constants.PROTECTED, "Modifier");
      public static final CToken CONST =
```

```
                  new CToken(Constants.opNames[Constants.CONST], Constants.CONST,
"Modifier");
      public static final CToken STATIC =
                  new CToken(Constants.opNames[Constants.STATIC], Constants.STATIC,
"Modifier");
      public static final CToken TRANSIENT =
                  new CToken(Constants.opNames[Constants.TRANSIENT],
Constants.TRANSIENT, "Modifier");
      public static final CToken SYNCHRONIZED =
                  new CToken(Constants.opNames[Constants.SYNCHRONIZED],
Constants.SYNCHRONIZED, "Modifier");
      public static final CToken NATIVE =
                  new CToken(Constants.opNames[Constants.NATIVE], Constants.NATIVE,
"Modifier");
      public static final CToken FINAL =
                  new CToken(Constants.opNames[Constants.FINAL], Constants.FINAL,
"Modifier");
      public static final CToken VOLATILE =
                  new CToken(Constants.opNames[Constants.VOLATILE],
Constants.VOLATILE, "Modifier");
      public static final CToken ABSTRACT =
                  new
CToken(Constants.opNames[Constants.ABSTRACT],Constants.ABSTRACT, "Modifier");
      /*
       * Punctuation
       */
      public static final CToken SEMICOLON =
                  new CToken(Constants.opNames[Constants.SEMICOLON],
Constants.SEMICOLON, "Punctuation");
      public static final CToken COLON =
                  new CToken(Constants.opNames[Constants.COLON], Constants.COLON,
"Punctuation");
      public static final CToken QUESTIONMARK =
                  new CToken(Constants.opNames[Constants.QUESTIONMARK],
Constants.QUESTIONMARK, "Punctuation");
      public static final CToken LBRACE =
                  new CToken(Constants.opNames[Constants.LBRACE], Constants.LBRACE,
"Punctuation");
      public static final CToken RBRACE =
                  new CToken(Constants.opNames[Constants.RBRACE], Constants.RBRACE,
"Punctuation");
      public static final CToken LPAREN =
                  new CToken(Constants.opNames[Constants.LPAREN], Constants.LPAREN,
"Punctuation");
      public static final CToken RPAREN =
                  new CToken(Constants.opNames[Constants.RPAREN], Constants.RPAREN,
"Punctuation");
      public static final CToken LSQBRACKET =
                  new CToken(Constants.opNames[Constants.LSQBRACKET],
Constants.LSQBRACKET, "Punctuation");
      public static final CToken RSQBRACKET =
                  new CToken(Constants.opNames[Constants.RSQBRACKET],
Constants.RSQBRACKET, "Punctuation");
      public static final CToken THROWS =
                  new CToken(Constants.opNames[Constants.THROWS], Constants.THROWS,
"Punctuation");
      /*
       * Special tokens
       */
      public static final CToken ERROR =
```

```
                  new CToken(Constants.opNames[Constants.ERROR], Constants.ERROR,
"Special");
      public static final CToken COMMENT =
                  new CToken(Constants.opNames[Constants.COMMENT],
Constants.COMMENT, "Special");
      public static final CToken TYPE =
                  new CToken(Constants.opNames[Constants.TYPE], Constants.TYPE,
"Special");
      public static final CToken LENGTH =
                  new CToken(Constants.opNames[Constants.LENGTH], Constants.LENGTH,
"Special");
      public static final CToken INLINERETURN =
                  new CToken(Constants.opNames[Constants.INLINERETURN],
Constants.INLINERETURN, "Special");
      public static final CToken INLINEMETHOD =
                  new CToken(Constants.opNames[Constants.INLINEMETHOD],
Constants.INLINEMETHOD, "Special");
      public static final CToken INLINENEWINSTANCE =
                  new CToken(Constants.opNames[Constants.INLINENEWINSTANCE],
Constants.INLINENEWINSTANCE, "Special");
      public static final CToken UNSCANNED =
                  new CToken("unscanned", MaximumScanValue, "Special");

      static CToken[] operators = {
                  COMMA, ASSIGN, ASGMUL, ASGDIV, ASGREM, ASGADD, ASGSUB, ASGLSHIFT,
                  ASGRSHIFT, ASGURSHIFT, ASGBITAND, ASGBITOR, ASGBITXOR, COND, OR,
AND,
                  BITOR, BITXOR, BITAND, NE, EQ, GE, GT, LE, LT, INSTANCEOF,
LSHIFT,
                  RSHIFT, URSHIFT, ADD, SUB, DIV, REM, MUL, CAST, POS, NEG, NOT,
BITNOT,
                  PREINC, PREDEC, NEWARRAY, NEWINSTANCE, NEWFROMNAME, POSTINC,
POSTDEC,
                  FIELD, METHOD, ARRAYACCESS, NEW, INC, DEC, CONVERT, EXPR, ARRAY,
GOTO
      };

      static CToken[] values = {
                  IDENT, BOOLEANVAL, BYTEVAL, CHARVAL, SHORTVAL, INTVAL, LONGVAL,
                  FLOATVAL, DOUBLEVAL, STRINGVAL
      };
      static CToken[] types = {
                  BYTE, CHAR, SHORT, INT, LONG, FLOAT, DOUBLE, VOID, BOOLEAN
      };
      static CToken[] expressions = {
                  TRUE, FALSE, THIS, SUPER, NULL
      };
      static CToken[] statements = {
                  IF, ELSE, FOR, WHILE, DO, SWITCH, CASE, DEFAULT, BREAK,
                  CONTINUE, RETURN, TRY, CATCH, FINALLY, THROW, STAT, EXPRESSION,
                  DECLARATION, VARDECLARATION
      };
      static CToken[] declarations = {
                  IMPORT, CLASS, EXTENDS, IMPLEMENTS, INTERFACE, PACKAGE
      };
      static CToken[] modifiers = {
                  PRIVATE, PUBLIC, PROTECTED, CONST, STATIC, TRANSIENT,
SYNCHRONIZED,
                  NATIVE, FINAL, VOLATILE, ABSTRACT
      };
      static CToken[] punctuations = {
```

```
                    SEMICOLON, COLON, QUESTIONMARK, LBRACE, RBRACE, LPAREN, RPAREN,
                    LSQBRACKET, RSQBRACKET, THROWS
        };
        static CToken[] specials = {
                    ERROR, COMMENT, TYPE, LENGTH, INLINERETURN, INLINEMETHOD,
                    INLINENEWINSTANCE, UNSCANNED
        };
        static CToken[] all = {
                    COMMA, ASSIGN, ASGMUL, ASGDIV, ASGREM, ASGADD, ASGSUB, ASGLSHIFT,
                    ASGRSHIFT, ASGURSHIFT, ASGBITAND, ASGBITOR, ASGBITXOR, COND, OR,
AND,
                    BITOR, BITXOR, BITAND, NE, EQ, GE, GT, LE, LT, INSTANCEOF,
LSHIFT,
                    RSHIFT, URSHIFT, ADD, SUB, DIV, REM, MUL, CAST, POS, NEG, NOT,
BITNOT,
                    PREINC, PREDEC, NEWARRAY, NEWINSTANCE, NEWFROMNAME, POSTINC,
POSTDEC,
                    FIELD, METHOD, ARRAYACCESS, NEW, INC, DEC, CONVERT, EXPR, ARRAY,
GOTO,
                    IDENT, BOOLEANVAL, BYTEVAL, CHARVAL, SHORTVAL, INTVAL, LONGVAL,
                    FLOATVAL, DOUBLEVAL, STRINGVAL, BYTE, CHAR, SHORT, INT, LONG,
FLOAT,
                    DOUBLE, VOID, BOOLEAN,  TRUE, FALSE, THIS, SUPER, NULL, IF, ELSE,
FOR,
                    WHILE, DO, SWITCH, CASE, DEFAULT, BREAK, CONTINUE, RETURN, TRY,
                    CATCH, FINALLY, THROW, STAT, EXPRESSION,  DECLARATION,
VARDECLARATION,
                    IMPORT, CLASS, EXTENDS, IMPLEMENTS, INTERFACE, PACKAGE, PRIVATE,
                    PUBLIC, PROTECTED, CONST, STATIC, TRANSIENT, SYNCHRONIZED,
NATIVE,
                    FINAL, VOLATILE, ABSTRACT, SEMICOLON, COLON, QUESTIONMARK,
LBRACE,
                    RBRACE, LPAREN, RPAREN, LSQBRACKET, RSQBRACKET, THROWS, ERROR,
                    COMMENT, TYPE, LENGTH, INLINERETURN, INLINEMETHOD,
INLINENEWINSTANCE,
                    UNSCANNED
        };
}
```

jbb\system\io\CFileFilter.java

```
package jbb.system.io;

import java.io.File;
import java.util.Hashtable;
import java.util.Enumeration;
import javax.swing.*;
import javax.swing.filechooser.*;

public class CFileFilter extends FileFilter {

	private Hashtable extensions = null;
	private String description = null;

	public CFileFilter() {
		this.extensions = new Hashtable();
	}

	public CFileFilter(String extension, String description) {
		this();
		if (extension != null)
			addExtension(extension);
		if (description != null)
			setDescription(description);
	}

	public CFileFilter(String[] extensions, String description) {
		this();
		for (int i=0; i<extensions.length; i++)
			addExtension(extensions[i]);
		if(description!=null)
			setDescription(description);
	}

	public boolean accept(File fl) {
		if (fl != null) {
			if (fl.isDirectory())
				return true;
			if ((getExtension(fl) != null) &&
(extensions.get(getExtension(fl)) != null))
				return true;
		}
		return false;
	}

	public String getExtension(File fl) {
		if (fl != null) {
			String flName = fl.getName();
			int i = flName.lastIndexOf('.');
			if (( i>0 ) && ( i<(flName.length()-1) ))
				return flName.substring(i+1);
		}
		return null;
	}

	public void addExtension(String extension) {
		if (extensions == null)
			extensions = new Hashtable();
		extensions.put(extension, this);
```

```
	}

	public String getDescription() {
		if (description == null) {
			description = " (";
			Enumeration e = extensions.keys();
			if (e != null) {
				description += "*." + (String)e.nextElement();
				while (e.hasMoreElements())
					description += ", *." + (String)e.nextElement();
			}
			description += ") ";
		}
		return description;
	}

	public void setDescription(String newDescription) {
		description = newDescription;
	}
}
```

jbb\system\io\COutputFrame.java

```
package jbb.system.io;

import jbb.international.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;

public class COutputFrame extends JFrame {

	private boolean logFile;
	private JScrollPane sp;
	private JTextArea aTextArea = new JTextArea();
	private PrintStream aPrintStream = new PrintStream(
		new FilteredStream(new ByteArrayOutputStream()));

	public COutputFrame(boolean logFile) {
		super(World.getText("Output & Error window"));
		this.logFile = logFile;
		System.setOut(aPrintStream);
		System.setErr(aPrintStream);
		getContentPane().setLayout(new BorderLayout());
		sp = new JScrollPane(aTextArea);
		getContentPane().add(BorderLayout.CENTER, sp);
		Dimension dim = getToolkit().getScreenSize();
		setSize(dim.width, 150);
		setLocation(0, dim.height-150);
		setVisible(true);
	}
	class FilteredStream extends FilterOutputStream {
		public FilteredStream(OutputStream aStream) {
			super(aStream);
		}
		private void doWrite(String aString) throws IOException {
			aTextArea.append(aString);
			sp.validate();
		sp.getVerticalScrollBar().setValue(
				sp.getVerticalScrollBar().getMaximum());
		}
		public void write(byte b[]) throws IOException {
			doWrite(new String(b));
		}
		public void write(byte b[], int off, int len) throws IOException {
			String aString = new String(b , off , len);
	doWrite(aString);
			if (logFile) {
				FileWriter aWriter = new FileWriter("error.log", true);
				aWriter.write(aString);
				aWriter.close();
			}
		}

	}
}
```

jbb\system\layouts\XYLayout.java

```
package jbb.system.layouts;

import java.awt.*;
//import java.util.*;

//public class XYLayout implements LayoutManager2 {
public class XYLayout implements LayoutManager {

	/**
	* The hashtable that will keep the associations
	* between the components and their constraints
	*/
//	private Hashtable components;

	/** Constructs a new empty XYLayout */
	public XYLayout() {
//		components = new Hashtable();
	}

	public void addLayoutComponent(Component cmp, Object constraints) {
//		synchronized (cmp.getTreeLock()) {
//			if (constraints instanceof Rectangle) {
//				components.put(cmp, (Rectangle)constraints);
//			} else
//				throw new IllegalArgumentException("cannot add to layout:
constraint must be a Rectangle");
//		}
	}
	public void addLayoutComponent(String strRect, Component cmp) {
//		synchronized (cmp.getTreeLock()) {
			//should translate string to rectangle
			//and then call
			//addLayoutComponent(cmp, rect);
//System.err.println("not implemented yet");
//			addLayoutComponent(cmp, cmp.getBounds());
//				throw new IllegalArgumentException("cannot add to layout:
String must be intX intY intWidth intHeight");
//		}
	}
	public void removeLayoutComponent(Component cmp) {
//		synchronized (cmp.getTreeLock()) {
//			Object dummy = components.remove(cmp);
//		}
	}
	private Rectangle getMinLocationSize() {
//		Rectangle minRect = null;
//		Rectangle rect = null;
//		Component cmp;
//		for(Enumeration e = components.elements(); e.hasMoreElements(); ) {
//			cmp = (Component)e.nextElement();
//			rect = (Rectangle)components.get(cmp);
//			if (minRect == null)
//				minRect = rect;
//			else {
//				if ((rect.x + rect.width) < (minRect.x + minRect.width)) {
//					minRect.x = rect.x;
//					minRect.width = rect.width;
//				}
```

```
//                      if ((rect.y + rect.height) < (minRect.y + minRect.height))
{
//                            minRect.y = rect.y;
//                            minRect.height = rect.height;
//                      }
//                }
//          }
//          if (minRect == null)
                return new Rectangle(0, 0, 0, 0);
//          else
//                return minRect;
      }
      public Dimension minimumLayoutSize(Container cntr) {
//          synchronized (cntr.getTreeLock()) {
//                Rectangle rect = getMinLocationSize();
//                Insets insets = cntr.getInsets();
//                rect.width += rect.x + insets.left + insets.right;
//                rect.height += rect.y + insets.top + insets.bottom;
//                return rect.getSize();
//          }
            return cntr.getMinimumSize();
      }
      public Dimension preferredLayoutSize(Container cntr) {
            return cntr.getPreferredSize();
      }
      public Dimension maximumLayoutSize(Container cntr) {
            return cntr.getMaximumSize();
//          return new Dimension(Integer.MAX_VALUE, Integer.MAX_VALUE);
      }
      public float getLayoutAlignmentX(Container cntr) {
            return (float)0.5;
      }
      public float getLayoutAlignmentY(Container cntr) {
            return (float)0.5;
      }
      public void invalidateLayout(Container cntr) {
      }
      public void layoutContainer(Container cntr) {
/*          synchronized (cntr.getTreeLock()) {
                for(int i = 0; i < cntr.getComponentCount(); i++) {
                      cntr.getComponent(i).invalidate();
                }
            }
*/
/*                Component cmp = null;
                Rectangle rect = null;
                Object obj = null;
                for(Enumeration e = components.keys(); e.hasMoreElements(); ) {
                      obj = e.nextElement();
                      cmp = (Component)obj;
                      obj = components.get(cmp);
                      rect = (Rectangle)obj;
                      cmp.setBounds(rect);
                      System.err.println(cmp + "\n" + rect);
                }
*/
      }
}
```

jbb\system\propertyEditors\CCustomEditor.java

```
package jbb.system.propertyEditors;

import jbb.gui.*;

import java.beans.PropertyEditorSupport;
import java.awt.Component;
import javax.swing.JFrame;

public class CCustomEditor extends PropertyEditorSupport {

	JFrame frame;

	/**
	*/
	public CCustomEditor(JFrame newFrame) {
		super();
		frame = newFrame;
	}
	/**
	*/
	public String getJavaInitializationString() {
		//should return a default initialization string for all classes
		//or make a custom editor for all classes
			return "null";
	}
	/**
	*/
	public boolean supportsCustomEditor() {
		return true;
	}
	/**
	*/
	public Component getCustomEditor() {
		FPropertiesPanel fpp = new FPropertiesPanel(frame);
		fpp.setSelectedBean(this.getValue());
		return fpp;
	}
}
```

jbb\system\propertyEditors\LayoutManagerEditor.java

```
package jbb.system.propertyEditors;

import jbb.system.layouts.*;

import java.beans.*;
import java.awt.*;

public class LayoutManagerEditor extends PropertyEditorSupport {

//    LayoutManager layoutObject = null;

      public String getJavaInitializationString() {
System.err.println("getJavaInitializationString");
            if (((LayoutManager)getValue()) instanceof BorderLayout) {
                  return ("new java.awt.BorderLayout()");
            } else if (((LayoutManager)getValue()) instanceof GridLayout) {
                  return ("new java.awt.GridLayout()"); //should give parameters
            } else if (((LayoutManager)getValue()) instanceof XYLayout) {
                  return ("new XYLayout()");
            } else if (((LayoutManager)getValue()) instanceof FlowLayout) {
                  return ("new java.awt.FlowLayout()"); //should give parameters
            } else if (((LayoutManager)getValue()) instanceof CardLayout) {
                  return ("new java.awt.CardLayout()");  //should give parameters
            } else
                  return "null";
      }
/*
      public Object getValue() {
            return layoutObject;
      }
      public void setValue(Object newLayoutObject) {
                  layoutObject = (LayoutManager)newLayoutObject;
            if (newLayoutObject instanceof BorderLayout) {
                  layoutObject = new java.awt.BorderLayout();
            } else if (newLayoutObject instanceof GridLayout) {
                  layoutObject = new java.awt.GridLayout();
            } else if (newLayoutObject instanceof XYLayout) {
                  layoutObject = new XYLayout();
            } else if (newLayoutObject instanceof FlowLayout) {
                  layoutObject = new java.awt.FlowLayout();
            } else if (newLayoutObject instanceof CardLayout) {
                  layoutObject = new java.awt.CardLayout();
            } else if (newLayoutObject instanceof LayoutManager) {
                  layoutObject = (LayoutManager)newLayoutObject;
            } else
                  layoutObject = null;
      }
*/
      public String getAsText() {
System.err.println("getAsText");
            if (((LayoutManager)getValue()) instanceof BorderLayout) {
                  return ("BorderLayout");
            } else if (((LayoutManager)getValue()) instanceof GridLayout) {
                  return ("GridLayout"); //should give parameters
            } else if (((LayoutManager)getValue()) instanceof XYLayout) {
                  return ("XYLayout");
            } else if (((LayoutManager)getValue()) instanceof FlowLayout) {
                  return ("FlowLayout"); //should give parameters
```

```
            } else if (((LayoutManager)getValue()) instanceof CardLayout) {
                  return ("CardLayout");  //should give parameters
            } else
                  return "null";
      }
      public void setAsText(String text) throws java.lang.IllegalArgumentException
{
System.err.println("setAsText");
            if (text.toLowerCase().equals("borderlayout")) {
                  setValue(new BorderLayout());
            } else if (text.toLowerCase().equals("gridlayout")) {
                  setValue(new GridLayout());
            } else if (text.toLowerCase().equals("xylayout")) {
                  setValue(new XYLayout());
            } else if (text.toLowerCase().equals("flowlayout")) {
                  setValue(new FlowLayout());
            } else if (text.toLowerCase().equals("cardlayout")) {
                  setValue(new CardLayout());
            } else {
System.err.println(text);
System.err.println(text.toLowerCase());
                  throw new java.lang.IllegalArgumentException(text);
            }
      }
      public String[] getTags() {
System.err.println("getTags");
            String result[] = {     "BorderLayout"
                                                      ,     "GridLayout"
                                                      ,     "XYLayout"
                                                      ,     "FlowLayout"
                                                      ,     "GridLayout" };
            return result;
      }
}
```

jbb\system\propertyEditors\PropertyButton.java

```
package jbb.system.propertyEditors;

import jbb.international.*;

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.beans.*;

public class PropertyButton extends JLabel implements MouseListener {

	private JFrame frame;
	private PropertyEditor editor;
	private boolean ignoreClick = false;

	public PropertyButton(JFrame frame, PropertyEditor pe) {
		super(World.getS("Edit"));
		this.frame = frame;
		editor = pe;
		addMouseListener(this);
	}
	public void mouseClicked(MouseEvent evt) {
		if (!ignoreClick) {
			try {
				ignoreClick = true;
				int x = frame.getX() + frame.getWidth()/2 -
this.getWidth()/2;
				int y = frame.getY() + frame.getHeight()/2 -
this.getHeight()/2;
				new PropertyDialog(frame, editor, x, y);
			} finally {
				ignoreClick = false;
			}
		}
	}
	public void mousePressed(MouseEvent evt) {
	}
	public void mouseReleased(MouseEvent evt) {
	}
	public void mouseEntered(MouseEvent evt) {
	}
	public void mouseExited(MouseEvent evt) {
	}
}
```

jbb\system\propertyEditors\PropertyCanvas.java

```
package jbb.system.propertyEditors;

// Support for drawing a property value in a Canvas.
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.beans.*;

public class PropertyCanvas extends JComponent implements MouseListener {

	private JFrame frame;
	private PropertyEditor editor;
	private static boolean ignoreClick = false;

	public PropertyCanvas(JFrame frame, PropertyEditor pe) {
		super();
		this.frame = frame;
		editor = pe;
		addMouseListener(this);
	}
	public void paint(Graphics g) {
		Rectangle box = new Rectangle(2, 2, getWidth() - 4, getHeight() - 4);
		editor.paintValue(g, box);
	}
	public void mouseClicked(MouseEvent evt) {
		if (!ignoreClick) {
			try {
				ignoreClick = true;
				int x = frame.getX() + frame.getWidth()/2 - getWidth()/2;
				int y = frame.getY() + frame.getHeight()/2 - getHeight()/2;
				new PropertyDialog(frame, editor, x, y);
			} finally {
				ignoreClick = false;
			}
		}
	}
	public void mousePressed(MouseEvent evt) {
	}
	public void mouseReleased(MouseEvent evt) {
	}
	public void mouseEntered(MouseEvent evt) {
	}
	public void mouseExited(MouseEvent evt) {
	}
}
```

jbb\system\propertyEditors\PropertyDialog.java

```
package jbb.system.propertyEditors;

// Support for PropertyEditor with custom editors.
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.beans.*;

public class PropertyDialog extends JDialog implements ActionListener {

	private Button doneButton;
	private Component body;
	private final static int vPad = 5;
	private final static int hPad = 4;

	PropertyDialog(JFrame frame, PropertyEditor pe, int x, int y) {
		super(frame, pe.getClass().getName(), true);
		setResizable(true);
		setDefaultCloseOperation(DISPOSE_ON_CLOSE);
		Container cp = getContentPane();
		body = pe.getCustomEditor();
		cp.add(body, BorderLayout.CENTER);
		doneButton = new Button("Done");
		doneButton.addActionListener(this);
		cp.add(doneButton, BorderLayout.SOUTH);
//should calculate location by size after pack
		pack();
		setLocation(x-this.getWidth()/2, y-this.getWidth()/2);
		setVisible(true);
	}
	public void actionPerformed(ActionEvent evt) {
		// Button down.
		dispose();
	}
//	public void doLayout() {
//		Insets ins = getInsets();
//		Dimension bodySize = body.getPreferredSize();
//		Dimension buttonSize = doneButton.getPreferredSize();
//		int width = ins.left + 2*hPad + ins.right + bodySize.width;
//		int height = ins.top + 3*vPad + ins.bottom + bodySize.height +
buttonSize.height;
//		body.setBounds(ins.left+hPad, ins.top+vPad,bodySize.width,
bodySize.height);
//		doneButton.setBounds((width-buttonSize.width)/2, ins.top+(2*hPad) +
bodySize.height, buttonSize.width, buttonSize.height);
//		setSize(width, height);
//	}
}
```

jbb\system\propertyEditors\PropertySelector.java

```
package jbb.system.propertyEditors;

// Support for PropertyEditors that use tags.
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.beans.*;

public class PropertySelector extends JComboBox implements ItemListener {

	private PropertyEditor editor;

	public PropertySelector(PropertyEditor pe) {
		super(pe.getTags());
		editor = pe;
		String tags[] = editor.getTags();
		setSelectedIndex(0);
		// This is a noop if the getAsText is not a tag.
		setSelectedItem(editor.getAsText());
		setEditable(false);
		addItemListener(this);
	}
	public void itemStateChanged(ItemEvent evt) {
		String s = (String)getSelectedItem();
		editor.setAsText(s);
	}
//	public void repaint() {
//		setSelectedItem(editor.getAsText());
//	}
}
```

jbb\system\propertyEditors\PropertyText.java

```
package jbb.system.propertyEditors;

// Support for a PropertyEditor that uses text.
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.beans.*;

public class PropertyText extends JTextField implements KeyListener, FocusListener
{

	private PropertyEditor editor;

	public PropertyText(PropertyEditor pe) {
		super(pe.getAsText());
		editor = pe;
		addKeyListener(this);
		addFocusListener(this);
	}
//	public void repaint() {
//		setText(editor.getAsText());
//	}
	protected void updateEditor() {
		try {
			if (editor.getAsText()!=getText())
			editor.setAsText(getText());
		} catch (IllegalArgumentException ex) {
			// Quietly ignore.
		}
	}
	// Focus listener methods.
	public void focusGained(FocusEvent e) {
	}
	public void focusLost(FocusEvent e) {
		updateEditor();
	}
	// Keyboard listener methods.
	public void keyReleased(KeyEvent e) {
		if (e.getKeyCode() == KeyEvent.VK_ENTER) {
			updateEditor();
		}
	}
	public void keyPressed(KeyEvent e) {
	}
	public void keyTyped(KeyEvent e) {
	}
}
```

jbb\system\workspace\CItem.java

```
package jbb.system.workspace;

import jbb.tools.*;

import java.awt.Component;
import javax.swing.tree.DefaultMutableTreeNode;
//import javax.swing.tree.TreeNode;
//import java.util.*;
//import java.beans.*;

public class CItem extends DefaultMutableTreeNode {

	private CProjectItem projectItem;

	private String name = null;
	private Object object = null;
	private Object position = null;
	private boolean visible = true;

	public CItem(CProjectItem newProjectItem, String newName,
			Object newObj, Object newPosition) {
		super( new String(newName) );
		projectItem = newProjectItem;
		name = newName;
		object = newObj;
		position = newPosition;
		if (object instanceof Component)
			( (Component)object ).setName(newName);
	}
	public String getNam() {
		return name;
	}
	public void setNam(String newName) {
		//better check if setName(String s) exists
		if (object instanceof Component) {
			( (Component)object ).setName(newName);
			name = newName;
		}
		else
			name = newName;
	}
	public final void updateJavaSource(CItem ci) {
		//code can be improved
		String temp, temp2;
		temp2 = "import " + Tools.getPackage(ci.getObj()) + ".*;\n";
		projectItem.sd.insertAfterPos("ImportEnd", temp2, temp2,
projectItem.NO_OVERWRITE);
		temp = Tools.getClassRightName(ci.getObj());
		temp2 = "\tpublic " + temp + " " + ci.getNam() + ";\n";
		projectItem.sd.insertAfterPos("BeanDeclarationStart", temp2, temp2,
projectItem.OVERWRITE);
		temp2 = "\t\t" + ci.getNam() + " = new " + temp + "();\n";
		projectItem.sd.insertAfterPos("CreationStart", temp2, temp2,
projectItem.OVERWRITE);
		if (ci.getObj() instanceof Component) {
			//if layout = ....
			temp2 = "\t\t" + getNam() + ".add(" + ci.getNam() + ");\n";
```

```
                    projectItem.sd.insertAfterPos("CreationStart", temp2, temp2,
projectItem.OVERWRITE);
                    //else ...
                    //if XYLayout then ...
              }
       }
       public final void propertyChange(CItem ci) {

       }
       public Object getObj() {
              return object;
       }
       public void setObj(Object newObject) {
              object = newObject;
       }
       public Object getPos() {
              return position;
       }
       public void setPos(Object newPosition) {
              position = newPosition;
       }
       public @Override String toString() {
              return name;
       }
}
```

jbb\system\workspace\CProject.java

```
package jbb.system.workspace;

import jbb.system.*;
import jbb.international.*;
import jbb.tools.*;

import java.awt.Rectangle;
import javax.swing.tree.TreeNode;
import javax.swing.tree.DefaultTreeModel;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JApplet;
import java.util.Vector;
import java.util.Enumeration;
import java.util.Properties;
import java.io.File;
import java.io.IOException;
import java.lang.Runtime;
//import java.beans.*;

public class CProject extends DefaultTreeModel {

	private static final String defaultProjectName = World.getText("Project
Name");
	private static final String defaultProjectFileName =
World.getText("Project_Name");

	public static final int TYPE_APPLICATION = 1;
	public static final int TYPE_APPLET = 2;
	public static final int TYPE_USER_COMPONENT = 3;
	public static final int TYPE_CONSOLE_APPLICATION = 4;


	public CWorkspace workspace;
	public boolean saved = false;
	public Properties properties = new Properties();

	public CProject(CWorkspace newWorkspace, int num, int type) {
		super((TreeNode)new DefaultMutableTreeNode(defaultProjectName));
		properties.setProperty("Name", defaultProjectName);
		properties.setProperty("FileName", defaultProjectFileName);

//		sd.insertAfterPos("MainClassDeclarationEnd", "public class "+newName+"
{\n", "MainClassDeclaration", OVERWRITE);

		properties.setProperty("Location", "");
		workspace = newWorkspace;
		switch (type) {
			case TYPE_APPLICATION: {
				addProjectItem("APPCLASS");
				addProjectItem("javax.swing.JFrame");
				addProjectItem("javax.swing.JFrame");
				break;
			}
			case TYPE_APPLET: {
				addProjectItem("APPLET_HTML");
				addProjectItem("javax.swing.JApplet");
```

```
                        break;
                  }
            }
      }

      public void addProjectItem(String classID) {
            //the project will be responsible for giving unique autoincreased names
            if ( classID.equals("javax.swing.JFrame") )  {
                  JFrame f = new JFrame("Title of JFrame");
                  f.setVisible(true);
                  f.setVisible(false);
                  addProjectItem(new CProjectItem(this, f.getName(), f, new
Rectangle(100,100,300,200), CProjectItem.TYPE_JFRAME));
            } else if ( classID.equals("javax.swing.JApplet") )  {
      JApplet a = new JApplet();//"JApplet" + "1" ,
      addProjectItem(new CProjectItem(this, a.getName(), a, new
Rectangle(100,100,300,200), CProjectItem.TYPE_JAPPLET));
            } else if ( classID.equals("APPCLASS") )  {
                  String defaultProjectName =
properties.getProperty("Name").replace(' ', '_');
                  defaultProjectName = defaultProjectName.replace('.', '_');
                  addProjectItem(new CProjectItem(this, defaultProjectName , null,
null, CProjectItem.TYPE_APPCLASS));
            } else if ( classID.equals("APPLET_HTML") )  {
                  String defaultProjectName =
properties.getProperty("Name").replace(' ', '_');
                  defaultProjectName = defaultProjectName.replace('.', '_');
                  addProjectItem(new CProjectItem(this, defaultProjectName , null,
null, CProjectItem.TYPE_APPLET_HTML));
            }
      }

      public void addProjectItem(CProjectItem pi) {
            ((DefaultMutableTreeNode)
(this.getRoot())).add((DefaultMutableTreeNode)pi);
            CSystem.system.fMain.projectExplorer.updateProjectTree();
            reload();
      }

      public void run() {
            String filename = "";
            CProjectItem child = null;
            boolean cont = false;
            for(Enumeration e = ((DefaultMutableTreeNode)getRoot()).children();
e.hasMoreElements(); ) {
                  child = (CProjectItem)e.nextElement();
                  if (child.getType()==child.TYPE_APPCLASS) {
                        filename = child.getJavaFilenameForBuild();
                        if (filename!="") cont = true;
                  }
                  if (child.getType()==child.TYPE_APPLET_HTML) {
                        filename = child.getJavaFilenameForBuild();
                        if (filename!="") cont = true;
                  }
      if (cont)
        break;
            }
            if (cont) {
                  try {
                        File fl = new File(filename);
                        File flDir = new File(fl.getParent());
```

```
                        filename = fl.getName();
                    int k = filename.lastIndexOf(".");
                    if (k!=-1) filename = filename.substring(0,k);
                    System.out.println(World.getS("Starting") +" "+ filename);
                    String[] prgArgs = new String[4];
                    //prgArgs[0] = "java -classpath \"" + fl.getParent() + "\" " +
filename;
                    prgArgs[0] = "java";
                    prgArgs[1] = "-classpath";
                    prgArgs[2] = "\"" + fl.getParent() + "\"";
                    prgArgs[3] = filename;
                    //Runner runner = new Runner(prgArgs);
                        Runtime.getRuntime().exec(prgArgs, null, flDir);
                    //runner.start();
                    System.out.println(filename +" " +World.getS("started")+".");
                  } catch (Exception e) {
                    System.out.println(World.getText("Unable to execute ") +
filename);
                  }
            } else {
                  System.out.println(World.getText("Unable to execute ") +
filename);
            }
      }

      public String[] getJavaFilenamesForBuild() {
            Vector filenames = new Vector();
            String addFilename = "";
            CProjectItem child;
            boolean cont = true;
            for(Enumeration e = ((DefaultMutableTreeNode)getRoot()).children();
e.hasMoreElements(); ) {
                  child = (CProjectItem)e.nextElement();
                  addFilename = child.getJavaFilenameForBuild();
                  if (addFilename!="") {
                        filenames.add(addFilename);
                  } else {
                        cont = false;
                  }
            }
            if (!cont)  {
                  //there is an error with some files
            }
            String[] sFilenames = new String[filenames.size()];
            for(int i = 0; (i < sFilenames.length) ; i++) {
                  sFilenames[i] = (String)filenames.get(i);
            }
            return sFilenames;
      }

      public boolean save(String location) {
            if (!saved) {
                  return this.saveAs(location);
            } else {
                  return true;
            }
      }

      public boolean saveAs(String location) {
            String pName = properties.getProperty("FileName");
            properties.setProperty("Location", location);
```

```
            File fl = new File(location + File.separator + pName +".jbp");
System.out.println(fl.getPath());
            boolean cont = true;
            if (fl.exists()) {
                  if (JOptionPane.showConfirmDialog(
                              null,
                              World.getText("The file \n")+ fl.getPath()
+World.getText("\n already exists. Overwrite?"),
                              World.getText("Warning"),
                              JOptionPane.YES_NO_OPTION,
                              JOptionPane.QUESTION_MESSAGE) !=
JOptionPane.YES_OPTION) {
                        cont = false;
                  } else {
                        if (!fl.delete()) {
System.err.println("file : "+fl.getPath()+"\ncould not be deleted." );
                              cont = false;
                        }
                  }
            }
            if (cont) {
                  try {
                        cont = fl.createNewFile();
                  } catch (IOException e) {
System.err.println("file : "+fl.getPath()+"\ncould not be created.\naborting..." );
                        cont = false;
                  }
                  if (cont) {
                        //save CProject properties and all CProjectItem
                        location = location + File.separator + pName;
                        fl = new File(location);
                        if (!fl.exists())
                              fl.mkdir();
                        CProjectItem child;
                        for(Enumeration e =
((DefaultMutableTreeNode)getRoot()).children(); e.hasMoreElements(); ) {
                              child = (CProjectItem)e.nextElement();
                              if (!child.save(location))
                                    cont = false;
                        }
                  }
            }
            saved = cont;
            CSystem.fMain.repaint();
            return cont;
      }
}
```

jbb\system\workspace\CProjectItem.java

```
package jbb.system.workspace;

import jbb.system.*;
import jbb.system.editors.*;
import jbb.system.editors.java.*;
import jbb.system.editors.html.*;
import jbb.international.*;
import jbb.tools.*;

import java.awt.*;
import javax.swing.tree.*;
import javax.swing.text.*;
import javax.swing.*;
import java.util.*;
import java.io.*;

public class CProjectItem extends DefaultMutableTreeNode {

	public static final boolean OVERWRITE = true;
	public static final boolean NO_OVERWRITE = false;

	public static final int TYPE_ROOT                              = 0;
	public static final int TYPE_JFRAME                            = 1;
	public static final int TYPE_JAPPLET                           = 2;
	public static final int TYPE_USERCLASS                 = 3;
	public static final int TYPE_COMPONENT                 = 4;
	public static final int TYPE_APPCLASS                          = 5;
	public static final int TYPE_APPLET_HTML        = 6;
	public static final int TYPE_OTHER                             = 998;
	public static final int TYPE_ITEM                                      = 999;

	public CProject project;

	private Object object = null;
	private Object position = null;
	private boolean visible = true;
	private CItem contentPane;

	public boolean saved = false;

	public Document source; //could be CJavaSource, CHTMLSource, ....
	public ISourceDocument sd;

	public final Properties properties = new Properties();
	/**
	 * Every CProjectItem will have a number of variables-objects. This
	 * hastable will hold their names and their associations with their
	 * methods, fields for use for the inteliCode feature.
	 *
	 * @see inteliCode
	 */
	public Hashtable variables = new Hashtable();

	public CProjectItem(CProject newProject, String newName, Object newObj,
Object newPos,int newType) {
		super( new String(newName) );
		project = newProject;
		setType(newType);
```

```
            setObj(newObj);
            setPos(newPos);
            properties.setProperty("Location", "");
            switch (newType) {
                  case TYPE_JFRAME:
                  case TYPE_JAPPLET:
      {
                        int tempPos;
                        source = new CJavaDocument();//newName,
"javax.swing.JFrame");
                        sd = (ISourceDocument)source;

                        sd.insertAfterPos("End", "/*|| START of imports */\n");
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "import java.lang.*;\n");
                        sd.addPosition("ImportStart", tempPos);
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "/*|| END of imports */\n");
                        sd.addPosition("ImportEnd", tempPos);

                        sd.insertAfterPos("End", "\n");
                        tempPos = (source.getEndPosition().getOffset()-1);

        if (newType==TYPE_JAPPLET)
          sd.insertAfterPos("End", "public class "+newName+" extends JApplet {\n");
        else if (newType==TYPE_JFRAME)
          sd.insertAfterPos("End", "public class "+newName+" extends JFrame {\n");
        else
          sd.insertAfterPos("End", "public class "+newName+" extends
J_ERROR_1234567 {\n");

        sd.addPosition("MainClassStart", tempPos);
                        sd.addPosition("MainClassDeclarationStart", tempPos);
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\n");
                        sd.addPosition("MainClassDeclarationEnd", tempPos);

                        sd.insertAfterPos("End", "\t/*|| START of Bean declaration
*/\n");
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\t/*|| END of Bean declaration
*/\n");
                        sd.addPosition("BeanDeclarationStart", tempPos);
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\n");
                        sd.addPosition("BeanDeclarationEnd", tempPos);

                        sd.insertAfterPos("End", "\tpublic "+newName+"() {\n");
                        sd.insertAfterPos("End", "\t\tsuper();\n");
                        sd.insertAfterPos("End", "\t\t/*|| START of creation
*/\n");
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\t\t/*|| END of creation */\n");
                        sd.addPosition("CreationStart", tempPos);
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\t}\n");
                        sd.addPosition("CreationEnd", tempPos);

                        sd.insertAfterPos("End", "\t/*|| START of methods */\n");
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\t/*|| END of methods */\n");
```

```
                        sd.addPosition("MethodsStart", tempPos);
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\n");
                        sd.addPosition("MethodsEnd", tempPos);

                        sd.insertAfterPos("End", "\tprivate void initialize()
{\n");
                        sd.insertAfterPos("End", "\t\t/*|| START of initialization
*/\n");
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\t\t/*|| END of initialization
*/\n");
                        sd.addPosition("InitStart", tempPos);
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\t}\n");
                        sd.addPosition("InitEnd", tempPos);

                        sd.insertAfterPos("End", "\t/*|| START of event handling
methods */\n");
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\t/*|| END of event handling
methods */\n");
                        sd.addPosition("EventsStart", tempPos);
                        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "\n");
                        sd.addPosition("EventsEnd", tempPos);
                        sd.insertAfterPos("End", "\tpublic void init()
{\n\n\t}\n");
                        sd.insertAfterPos("End", "\tpublic void start()
{\n\n\t}\n");
                        sd.insertAfterPos("End", "\tpublic void stop()
{\n\n\t}\n");
  //public String getAppletInfo() {
  //    return "Title: Applet Info";
  //}
                        sd.insertAfterPos("End", "}\n");

        tempPos = (source.getEndPosition().getOffset()-1);
                        sd.insertAfterPos("End", "/*|| END OF FILE */\n");
                        sd.addPosition("MainClassEnd", tempPos);

                        //make the contentPane a CItem //could be a scollpane
                        contentPane = new CItem(this, "contentPane", new JPanel(new
BorderLayout()), "ContentPane");
                        this.add(contentPane);
                        String temp, temp2;
                        temp = Tools.getClassRightName(contentPane.getObj());
                        temp2 = "\tpublic " + temp + " " + contentPane.getNam() +
";\n";
                        sd.insertAfterPos("BeanDeclarationStart", temp2, temp2,
OVERWRITE);
                        temp2 = "\t\t" + contentPane.getNam() + " = new " + temp +
"();\n";
                        sd.insertAfterPos("CreationStart", temp2, temp2,
OVERWRITE);
                        temp2 = "\t\tthis.setContentPane(" + contentPane.getNam() +
");\n";
                        sd.insertAfterPos("CreationStart", temp2, temp2,
OVERWRITE);
                        break;
                }
```

```
                    case TYPE_APPCLASS: {
                          int tempPos;
                          source = new CJavaDocument();//newName,
"javax.swing.JFrame");
                          sd = (ISourceDocument)source;

                          sd.insertAfterPos("End", "/*|| START of imports */\n");
                          tempPos = (source.getEndPosition().getOffset()-1);
                          sd.insertAfterPos("End", "import java.lang.*;\n");
//                        sd.insertAfterPos("End", "import javax.swing.*;\n");
                          sd.addPosition("ImportStart", tempPos);
                          tempPos = (source.getEndPosition().getOffset()-1);
                          sd.insertAfterPos("End", "/*|| END of imports */\n");
                          sd.addPosition("ImportEnd", tempPos);

                          sd.insertAfterPos("End", "\n");
                          tempPos = (source.getEndPosition().getOffset()-1);
                          sd.insertAfterPos("End", "public class "+ newName + "
{\n");
                          sd.addPosition("MainClassStart", tempPos);
                          sd.addPosition("MainClassDeclarationStart", tempPos);
                          tempPos = (source.getEndPosition().getOffset()-1);
                          sd.insertAfterPos("End", "\n");
                          sd.addPosition("MainClassDeclarationEnd", tempPos);

                          sd.insertAfterPos("End", "\tpublic static void
main(String[] args) {\n");
                          tempPos = (source.getEndPosition().getOffset()-1);
                          sd.insertAfterPos("End", "\t\tJFrame mainFrame = new
frame0();\n");
                          sd.addPosition("CreationStart", tempPos);
                          tempPos = (source.getEndPosition().getOffset()-1);
                          sd.insertAfterPos("End", "\t\tmainFrame.pack();\n");
                          sd.insertAfterPos("End",
"\t\tmainFrame.setVisible(true);\n");
                          sd.addPosition("CreationEnd", tempPos);
                          sd.insertAfterPos("End", "\t}\n");
                          sd.insertAfterPos("End", "}\n");
                          tempPos = (source.getEndPosition().getOffset()-1);
                          sd.insertAfterPos("End", "/*|| END OF FILE */\n");
                          sd.addPosition("MainClassEnd", tempPos);
                          break;
                    }
                    case TYPE_APPLET_HTML: {
                          int tempPos;
                          source = new CHTMLDocument();//newName,
"javax.swing.JFrame");
                          sd = (ISourceDocument)source;
          sd.insertAfterPos("End", "<html>\n  <body>\n      <applet
code=applet0.class>\n      </applet>\n  </body>\n</html>\n");
                          sd.insertAfterPos("End", "<!--/*|| END OF FILE */-->\n");
                          break;
                    }
              }
              // JBB Bannner
              Date d = new Date();
      if (newType==TYPE_APPLET_HTML)
        sd.insertAfterPos("Start", "\n<!--\n");
      else
        sd.insertAfterPos("Start", "\n");
              sd.insertAfterPos("Start", " */\n");
```

```
            sd.insertAfterPos("Start", " * Date:  "+d+" \n");
            sd.insertAfterPos("Start", " *\n");
            sd.insertAfterPos("Start", " * "+World.getS("JBBVersion")+" \n");
            sd.insertAfterPos("Start", "/** File generated by Java Begginers
Builder.\n");
            sd.insertAfterPos("Start", "\n");
//          sd.insertAfterPos("BeanDeclarationStart", "//testing3a insert\n");
    if (newType==TYPE_APPLET_HTML) {

    } else {
            sd.insertAfterPos("ImportEnd", "import javax.swing.*;\n",
                        "import javax.swing.*;\n", NO_OVERWRITE);
    }
            setNam(newName);
      }
      public final Document getDoc() {
            return source;
      }
      public final String getNam() {
            return (String)properties.getProperty("Name");
      }
      public final void updateJavaSource(CItem ci) {
            String newMark = "import " + Tools.getPackage(ci.getObj())+".*;\n";
            sd.insertAfterPos("ImportEnd", newMark, newMark, NO_OVERWRITE);
            //++ others (and in CItem) (also in CProject)
      }
      public final void setNam(String newName) {
            properties.setProperty("Name", newName);
            properties.setProperty("FileName", newName);
            if (object instanceof Component)
                  ( (Component)object ).setName(newName);
      }
      public final int getType() {
            return Integer.parseInt((String)properties.getProperty("Type"));
      }
      public final void setType(int newType) {
            properties.setProperty("Type", String.valueOf(newType));
      }
      public final Object getPos() {
            return position;
      }
      public final void setPos(Object newPosition) {
            position = newPosition;
      }
      public final Object getObj() {
            return object;
      }
      public final void setObj(Object newObject) {
            object = newObject;
      }
      public final CItem getContentPane() {
            return contentPane;
      }
      public final String toString() {
            return getNam();
      }
      /** Java source code generation
        */
      public String getJavaFilenameForBuild() {
            String pName = properties.getProperty("FileName");
            String location = properties.getProperty("Location");
```

```
            return (location + File.separator + "Java" + File.separator + pName +
".java");
      }
      public final boolean save(String location) {
            String pName = properties.getProperty("FileName");
            properties.setProperty("Location", location);
            File fl;
            String javaLocation = location + File.separator + "Java";
            fl = new File(javaLocation);
            if (!fl.exists())
                  fl.mkdir();
            //create only if is needed(for JFrame, JApplet, etc)
            fl = new File(location + File.separator + pName +".jbpi");
System.out.println(fl.getPath());
            boolean cont = true;
            if (fl.exists()) {
/*                if (JOptionPane.showConfirmDialog(
                              null,
                              World.getText("The file \n")+ fl.getPath()
+World.getText("\n already exists. Overwrite?"),
                              World.getText("Warning"),
                              JOptionPane.YES_NO_OPTION,
                              JOptionPane.QUESTION_MESSAGE) !=
JOptionPane.YES_OPTION) {
                        cont = false;
                  } else {
*/                      if (!fl.delete()) {
System.err.println("file : "+fl.getPath()+"\ncould not be deleted." );
                              cont = false;
                        }
/*                }*/
            }
            if (cont) {
                  try {
                        cont = fl.createNewFile();
                  } catch (IOException e) {
System.err.println("file : "+fl.getPath()+"\ncould not be created.\naborting..." );
                        cont = false;
                  }
                  if (cont) {//save CProjectItem properties and source
                        fl = new File(javaLocation + File.separator + pName
+".java");
                        if (fl.exists())
                              if (!fl.delete()) {
System.err.println("file : "+fl.getPath()+"\ncould not be deleted." );
                                    cont = false;
                              }
                        if (cont) {
                              try {
                                    cont = fl.createNewFile();
                              } catch (IOException e) {
System.err.println("file : "+fl.getPath()+"\ncould not be created.\naborting..." );
                                    cont = false;
                              }
                        }
                        if (cont) {
                              FileWriter fw = null;
                              try {
                                    fw = new FileWriter(fl);
                                    fw.write(source.getText(0,
source.getLength()));
```

```
                                } catch (IOException e) {
System.err.println(e);
                                      cont = false;
                                } catch (BadLocationException e) {
System.err.println(source.getLength() + "\n" + source);
                                      cont = false;
                                }
                                if (fw!=null)
                                      try {
                                            fw.close();
                                      } catch (IOException e) {
System.err.println(e);
                                      }
                          }
                    }
              }
              saved = cont;
              CSystem.fMain.repaint();
              return cont;
        }
}
```

jbb\system\workspace\CWorkspace.java

```
package jbb.system.workspace;

import jbb.system.*;
import jbb.system.io.*;
import jbb.international.*;

//import java.awt.*;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import java.io.File;
//import java.util.*;
//import java.beans.*;
import java.util.Vector;
import java.util.Properties;
import java.io.IOException;


/**
* keeps many projects
*/
public class CWorkspace extends Vector {

	private int selectedProject = 0;
	private int totalProjects = 0;

	/**
	* True if saved after last change and false if not.
	*/
	public boolean saved = false;
	/**
	* Holds all CWorkspace related options - properties
	*/
	public Properties properties = new Properties();


	/**
	* Main and only constructor
	*/
	public CWorkspace() {
		super();
	}
	/**
	*/
	public CProject getSelectedProject() {
		if (selectedProject == 0) {
			return null;
		} else {
			return (CProject)this.get(selectedProject - 1);
		}
	}

/**

*/
	public boolean setSelectedProject(int SP) {
		if ( (SP < 0) || (SP > totalProjects) ) {
			return false;
		} else {
```

```
                  selectedProject = SP;

      CSystem.system.fMain.projectExplorer.setProject(getSelectedProject());
                  return true;
            }
      }

/* public boolean setSelectedProject(String str) {
                  return false;
                  return true;
      }
*/

      public void addProject() {
            totalProjects++;
            CProject p = new CProject(this, totalProjects,
CProject.TYPE_APPLICATION);
            this.add(p);
            setSelectedProject(totalProjects);
      }

      public void addAppletProject() {
            totalProjects++;
            CProject p = new CProject(this, totalProjects, CProject.TYPE_APPLET);
            this.add(p);
            setSelectedProject(totalProjects);
      }

  public void addProject(CProject p) {
            totalProjects++;
            this.add(p);
            setSelectedProject(totalProjects);
      }
//should add code for savedName, status(changed or not) on save
      public void save() {
            if (!saved) this.saveAs();
      }
      public void saveAs() {
            JFileChooser chooser = new JFileChooser();
            // I could parameterize the workspace extension ".jbw" as all others
too
            CFileFilter filter =
                        new CFileFilter("jbw", World.getText("JBB Workspace files
(*.jbw)"));
            chooser.setFileFilter(filter);
            int returnVal = chooser.showSaveDialog(CSystem.system.fMain);
            File fl;
            if(returnVal == JFileChooser.APPROVE_OPTION) {
                  fl = chooser.getSelectedFile();
            int i1, i2;
            i1 = fl.getName().indexOf(".jbw");
            i2 = fl.getName().lastIndexOf(".jbw");
            if ((i1!=i2) || (i1<0))
                  fl = new File(fl.getParent() + File.separator + fl.getName() +
".jbw");
System.out.println("file : "+fl.getPath());
                  boolean cont = true;
                  if (fl.exists()) {
                        if (JOptionPane.showConfirmDialog(
                                          null,
```

```
                                                World.getText("The file \n")+
fl.getPath()+World.getText("\n already exists. Overwrite?"),
                                                World.getText("Warning"),
                                                JOptionPane.YES_NO_OPTION,
                                                JOptionPane.QUESTION_MESSAGE) !=
JOptionPane.YES_OPTION) {
                                    cont = false;
                              } else {
                                    if (!fl.delete()) {
System.err.println("file : "+fl.getPath()+"\ncould not be deleted." );
                                          cont = false;
                                    }
                              }
                        }
                        if (cont) {
                              try {
                                    cont = fl.createNewFile();
                              } catch (IOException e) {
System.err.println("file : "+fl.getPath()+"\ncould not be created." );
                                    cont = false;
                              }
                              //save all CWorkspace(only) properties and close file
                        }
                        if (cont) {
                              Object objs[] = toArray();
                              CProject p;
                              boolean allDone = true;
                              for(int i=0; i<objs.length;i++) {
                                    p = (CProject)objs[i];
                                    //should check for errors in each save(and maybe stop
the save)
//System.err.println("Project location =  "+fl.getParent());
                                    if (!p.saveAs(fl.getParent())) {
System.err.println("Project : "+p+"\ncould not be saved." );
                                          if (allDone)
                                                allDone = false;
                                    }
                              }
                              //if everything wend ok then make workspace flag saved to
true
                              saved = allDone;
                        }
                  }
                  CSystem.system.fMain.repaint();
            }
}
```

jbb\tools\Runner.java

```
package jbb.tools;

import jbb.international.*;
import java.io.*;

public class Runner extends java.lang.Thread {

	private String[] PrgArgs;

	public Runner (String[] args) {
		PrgArgs = args;
	}

	public void run () {
		try {
			Process runningProcess = Runtime.getRuntime().exec(PrgArgs);
			InputStream is = runningProcess.getInputStream(); // Yes, it IS
InputStream. Although we're looking for the apps OUTPUT stream, to us, its an
InputStream.
			InputStream es = runningProcess.getErrorStream();
			RunningAppWatcher appWatcher = new RunningAppWatcher(is);
			RunningAppWatcher appErrWatcher = new RunningAppWatcher(es);
			appWatcher.start();
			appErrWatcher.start();
			try {
				runningProcess.waitFor();
			} catch ( InterruptedException ie ) {
				System.out.println(World.getText("Interrupted when waiting
for your program. Output monitoring terminated."));
			}
			int exitVal = runningProcess.exitValue();
			if ( exitVal == 0 ) {
				System.out.println(World.getText("Application terminated
normally."));
			}
			else {
				System.out.println(World.getText("Potentially Abnormal
exit. Exit status == " + exitVal + "."));
			}
		} catch (IOException e) {
			System.out.println (World.getText("I/O Error during exec(...).")
+ e.toString());
		}
	}
}
```

jbb\tools\RunningAppWatcher.java

```
package jbb.tools;

import jbb.international.*;
import java.io.*;

public class RunningAppWatcher extends java.lang.Thread {
	private boolean stayRunning = true;
	private BufferedReader appOutput;

	/** This is the constructor. It takes an InputStream to watch.
	*
	* @param is The stream (an OUTPUT stream from the application, which of
course is an inputStream to us)
	*/
	public RunningAppWatcher(InputStream is) {
		appOutput = new BufferedReader(new InputStreamReader(is));
	}
	/** Standard run method for any thread ;-) If you wish to stop this thread
	* running by using an external class (ie you programmatically want to stop
	* the thread rather than wait for the running application to finish, use
	* stopRunning() method.
	*/
	public void run() {
		while ( stayRunning ) {
			// code to watch process here.
			try {
				String str = appOutput.readLine();
				if ( str == null ) {
					// Throwing an exception prevents the rest of the
processing, which is exactly
					// what we want really - no point displaying 'null'
to the output window!
					throw new
NullPointerException(World.getText("Application finished"));
				}
				System.out.println(str);
			} catch ( IOException ioe ) {
				System.out.println(World.getText("IOException reading
Output from executing process. Not good."));
				stayRunning = false;
			} catch ( NullPointerException npe ) {
				stopRunning();
			}
		}
	}
	/** This method will cleanly close the thread. Plus you get no ThreadDeath
exception with this method!
	*/
	public void stopRunning() {
		stayRunning = false;
	}
}
```

jbb\tools\Tools.java

```
package jbb.tools;

import jbb.international.*;

import java.awt.*;
import java.beans.*;
import javax.swing.*;
import java.awt.Point;
import java.awt.Dimension;
import java.awt.Toolkit;


public class Tools {

	private static Dimension screenSize;
	private static Point screenCenter;

/**
initialization of Tools.
*/
	public static void initialize() {
		screenSize = Toolkit.getDefaultToolkit().getScreenSize();
		screenCenter=new Point(screenSize.width/2, screenSize.height/2);
	}
/**
getScreenCenter returns a point located at the center of the screen.
*/
	public static Point getScreenCenter() {
			return screenCenter;
	}
/**
getScreenSize returns the Dimension of the screen.
*/
	public static Dimension getScreenSize() {
		return screenSize;
	}
/**
getScreenCenter returns a point located in such way so that
the component, with size cSize will, be centered at the center of the screen.
*/
	public static Point getScreenCenter(Dimension cSize) {
		return new Point(screenCenter.x - cSize.width/2, screenCenter.y -
cSize.height/2);
	}
	public static String getNewName(String beanName) {
		beanName = beanName.replace('.', '_');
		//search a hash for next available name
		return World.getS("renameMe_") + beanName;
	}
	public static String getPackage(Object obj) {
		String packageName = obj.getClass().getPackage().getName();
		if (packageName==null)
			packageName = "null";
		return packageName;
	}
	public static String getClassRightName(Object obj) {
		String className = obj.getClass().getName();
		if (className==null)
```

```
                  className = "null";
            int k = className.lastIndexOf(".");
            if (k!=-1)
                  className = className.substring(k+1);
            return className;
      }
      public static Object instantiate(ClassLoader cls, String beanName, boolean
isXY) {
            Object obj = null;
            try {
                  obj = Beans.instantiate(cls, beanName);
            } catch (Exception e) {
                  System.err.println(e);
                  return null;
            }
            if (obj instanceof Component) {
                  Component c = (Component)obj;
                  c.setName(Tools.getNewName(beanName));
                  if (c instanceof JLabel) {
                        ((JLabel)c).setText(World.getText("New Label"));
                  } else if (c instanceof JButton) {
                        ((JButton)c).setText(World.getText("New Button"));
                  } else if (c instanceof JList) {
                        String str = World.getText("List Item");
                        Object[] objs = { str + "1",
                                                                  str + "2",
                                                                  str + "3"};
                        ((JList)c).setListData(objs);
                  } else if (c instanceof JComboBox) {
                        String str = World.getText("ComboBox Item");
                        ((JComboBox)c).addItem(str + "1");
                        ((JComboBox)c).addItem(str + "2");
                        ((JComboBox)c).addItem(str + "3");
                  } else if (c instanceof JPasswordField) {
                        ((JPasswordField)c).setText(World.getS("password"));
                  } else if (c instanceof JPanel) {
                        //((JPanel)c).setMinimumSize(new Dimension(50,50));
                        ((JPanel)c).setBackground(Color.yellow);
                        //and border
                        if (isXY)
                              ((JPanel)c).setPreferredSize(new Dimension(50,50));
System.out.println("created a JPanel");
                  }
                  if (isXY) {
                        Dimension dm = c.getPreferredSize();
                        int insetHeight = 5;//i should calculate this(and width)
                        c.setBounds(0, 0, dm.width, dm.height + insetHeight);
                  }
                  return (Object)c;
            } else {
                  return obj;
            }
      }
}
```

jbb\tools\compilers\CJavaCompiler.java

```
package jbb.tools.compilers;

import jbb.international.*;

import java.io.*;
import java.util.*;
import javax.swing.*;
//import java.awt.event.*;

public class CJavaCompiler extends JFrame /* implements ActionListener */ {
	private PrintStream psOut = System.out;

	private sun.tools.javac.Main javaCompiler =
			new sun.tools.javac.Main(psOut, "Javac");

	public CJavaCompiler() {
		super(World.getS("Java Compiler"));
		//!!
		this.getContentPane().add(new JLabel(World.getS("Compiling")+"..."));
		this.pack();
		this.setVisible(true);//!!
	}
	public void javaCompile (String[] Files) {
		psOut.println(World.getText("Starting compiling."));
		if (Files.length == 0)
			return;
		int i;
		String[] Params = new String[Files.length];
		for (i = 0; i < Files.length; i++) {
			psOut.println(Files[i]);
			Params[i] = Files[i];
		}
		//plus any other parameters
		if (javaCompiler.compile(Params)) //in a new seperated thread
			psOut.println(World.getText("Compilation successfull."));
		else
			psOut.println(World.getText("Compilation failed."));
	}
}
```