**ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΛΟΝΙΚΗΣ**

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**Πτυχιακή Εργασία**

**Ανάπτυξη βιβλιοθήκης χειρισμού αρχείων dBase (dbf) στη γλώσσα Java**

****

**Του φοιτητή: Επιβλέπων Καθηγητής:**

**Ζιώγα Ευθυμίου Δρ. Σφέτσος Παναγιώτης**

**Αρ. Μητρώου: 08/3395**

**Θεσσαλονίκη 2014**

# Πρόλογος-Εισαγωγή

Η ακόλουθη πτυχιακή εργασία αφορά την ανάπτυξη μιας βιβλιοθήκης σε γλώσσα Java μέσω της οποίας θα γίνεται χειρισμός αρχείων dBase(.dbf). Επίσης, αφορά την ανάπτυξη μιας εφαρμογής με γραφικό περιβάλλον, επίσης σε γλώσσα Java, η οποία θα χρησιμοποιεί την βιβλιοθήκη χειρισμού αρχείων dBase και θα προσφέρει γραφικό χειρισμό αρχείων. Οι στόχοι της εργασίας είναι η μελέτη της τεχνολογίας dBase, η μελέτη της δομής του αρχείου .dbf και κυρίως της κεφαλίδας του αρχείου καθώς και η προσπάθεια προσπέλασης αυτού του αρχείου μέσω της γλώσσας Java. Η ανάπτυξη όλης της βιβλιοθήκης καθώς και της γραφική εφαρμογής έγινε με τη βοήθεια του Eclipse IDE.

Στο πρώτο κεφάλαιο της πτυχιακής εργασίας, γίνεται μια αναφορά στο σύστημα dBase. Αναφέρονται ιστορικά στοιχεία, καθώς και η εξέλιξη που είχε το σύστημα μέχρι τις μέρες μας. Επίσης, γίνεται αναφορά στη γλώσσα προγραμματισμού που χρησιμοποιεί η dBase για τον χειρισμό των δεδομένων, αλλά και στους τύπους αρχείων που χρησιμοποιούνται από το σύστημα. Τέλος, γίνεται αναλυτική παρουσίαση της δομής των αρχείων .dbf(κύριο αρχείο), .dbt(αρχείο Memo) και .ndx(αρχείο index).

Στο δεύτερο κεφάλαιο, παρουσιάζεται η γλώσσα προγραμματισμού Java. Γίνεται μια ιστορική αναδρομή της γλώσσας από την αρχή της μέχρι σήμερα, καθώς και μια αναφορά στην απόδοσή της, στο συντακτικό της και στην εικονική της μηχανή. Τέλος, αναφέρω κάποια στοιχεία για τα περιβάλλοντα ανάπτυξης εφαρμογών σε Java καθώς και στις ειδικές κλάσεις τις.

Στο τρίτο κεφάλαιο, αναφέρομαι στην Java Swing μέσω της οποίας ανέπτυξα την εφαρμογή με γραφικό περιβάλλον. Αναφέρονται ιστορικά στοιχεία και τα στάδια ανάπτυξης της εργαλειοθήκης καθώς και στοιχεία για τη ανταγωνιστική εργαλειοθήκη SWT η οποία αναπτύσσεται από την Eclipse Foundation.

Στο τέταρτο κεφάλαιο, παρουσιάζεται το ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού Eclipse IDE το οποίο χρησιμοποίησα για την συγγραφή της βιβλιοθήκης και της εφαρμογής. Γίνεται μια αναφορά στην ιστορία του, στην εξέλιξή του και στις εκδόσεις του. Τέλος, αναφέρω λίγα λόγια για τις πιο σημαντικές επεκτάσεις που χρησιμοποιούνται μαζί με το Eclipse.

Στο πέμπτο κεφάλαιο, παρουσιάζω αναλυτικά την βιβλιοθήκη χειρισμού αρχείων dBase. Αναλύω τις κλάσεις της βιβλιοθήκης μια-μια και παραθέτω στοιχεία για το πώς δουλεύουν οι πιο σημαντικές μέθοδοι. Τέλος, υπάρχει ένα αναλυτικό API το οποίο περιέχει όλες τις μεθόδους της βιβλιοθήκης τις οποίες μπορεί να χρησιμοποιήσει κάποιος για να χειριστεί αρχεία dBase.

Τέλος, στο έκτο κεφάλαιο παρουσιάζω το λογισμικό το οποίο χρησιμοποιεί τη βιβλιοθήκη χειρισμού αρχείων dBase. Αναφέρω όλες τις κλάσεις από τις οποίες αποτελείτε και αναφέρω περιληπτικά το τι κάνει κάθε κλάση. Επίσης, στο τέλος του κεφαλαίου υπάρχει ένας αναλυτικός οδηγός χρήσης του λογισμικού με εικόνες ο οποίος περιγράφει το λογισμικό και τις λειτουργίες οι οποίες μπορούν να εκτελεστούν μέσω αυτού.

# Περίληψη

Η dbase είναι από τα πρώτα συστήματα διαχείρισης βάσεων δεδομένων. Η πρώτη έκδοση δημοσιεύτηκε το 1980 από την Ashton-Tate και από τότε ακολούθησαν και άλλες εκδόσεις, από τις οποίες οι πιο σημαντικές είναι η dBase III και dBase IV. Η dBase περιλαμβάνει μία γλώσσα προγραμματισμού η οποία προσφέρει καλύτερο έλεγχο και χειρισμό των εγγραφών. Οι εντολές εκτελούνται μέσω dot prompt με τη χρήση διερμηνέα κάτι που κάνει τη γλώσσα γρήγορη σε απαιτητικές εφαρμογές. Η dBase ως κύριο file format έχει το .dbf, το οποίο αποδείχθηκε πολύ επιτυχημένο λόγω της απλότητάς του και πλέον χρησιμοποιείτε και από άλλες εφαρμογές. Άλλα σημαντικά file formats που χρησιμοποιεί η dBase είναι το .dbt για memo πεδία και .ndx, .mdx για αρχεία με indexes. Η πιο πρόσφατη έκδοση είναι η dBase PLUS 8 η οποία ενσωματώνει νέες λειτουργίες και μπορεί να ‘τρέξει’ στα σύγχρονα λειτουργικά συστήματα.

Η γλώσσα Java είναι μια δημοφιλής γλώσσα προγραμματισμού η οποία σχεδιάστηκε από τον James Gosling για την Sun Microsystems στις αρχές τις δεκαετίας του 1990 κατ’ αρχήν για διαδραστικές τηλεοράσεις. Η πρώτη έκδοση δημοσιοποιήθηκε το 1995 και από τότε ακολούθησαν και άλλες. Το 2006 η εταιρεία έκανε την Java open source, ενώ το 2010 η Sun Microsystems καθώς και η γλώσσα εξαγοράστηκαν από την Oracle Corporation. Το κύριο χαρακτηριστικό της γλώσσας είναι η φορητότητα, δηλαδή μια εφαρμογή σε Java ‘τρέχει’ σε όλα τα λειτουργικά συστήματα χωρίς να γίνει ξανά compile. Αυτό επιτυγχάνεται μέσω της Java Virtual Machine, η οποία μεταφράζει τον κώδικα σε Java bytecode και όχι απευθείας σε γλώσσα μηχανής. Η Java αναπτύχθηκε με βάση τη C και C++, και συγκρινόμενη με αυτές είναι περίπου κατά 44% πιο αργή. Τέλος, η Java περιέχει ειδικές κλάσεις οι οποίες δίνουν πρόσθετη λειτουργικότητα. Κάποιες από αυτές είναι τα applets, τα servlets, η swing κα.

Η Swing είναι η κύρια εργαλειοθήκη για δημιουργία γραφικού περιβάλλοντος στη γλώσσα Java. Είναι η εξέλιξη της awt και προσφέρει πιο εξελιγμένα components. Όλα τα components είναι γραμμένα σε γλώσσα Java και έτσι είναι ανεξάρτητα από κάθε πλατφόρμα. Η SWT είναι μια βιβλιοθήκη παρόμοια με την Swing η οποία αναπτύχθηκε από την IBM και την Eclipse Foundation και υποστηρίζει απεικόνιση του γραφικού περιβάλλοντος της εφαρμογής ίδια με αυτή των εφαρμογών που γράφτηκαν με το native API του λειτουργικού.

Το Eclipse είναι ένα Ολοκληρωμένο Περιβάλλον Ανάπτυξης(IDE), μέσω του οποίου μπορεί να γίνει ανάπτυξη λογισμικού σε διάφορες γλώσσες προγραμματισμού. Ο αρχικός κώδικας του Eclipse προήλθε από το IBM VisualAge. Πλέον το Eclipse αναπτύσσεται από την Eclipse Foundation η οποία δημοσιοποιεί μια νέα έκδοση κάθε χρόνο. Τέλος, για το Eclipse δημιουργούνται και διάφορες επεκτάσεις οι οποίες δίνουν νέες δυνατότητες στους προγραμματιστές. Οι πιο σημαντικές είναι οι pydev και ADT για ανάπτυξη εφαρμογών python και Android αντίστοιχα.

Η βιβλιοθήκη χειρισμού αρχείων dBase είναι μια συλλογή κλάσεων οι οποίες δίνουν τη δυνατότητα χειρισμού αρχείων dBase μέσω της γλώσσας Java. Οι κύριες κλάσεις της βιβλιοθήκης είναι οι εξής: Η DbaseFile, η οποία αναπαριστά ένα αρχείο dBase και εκτελεί διάφορες λειτουργίες όπως δημιουργία, άνοιγμα, pack του αρχείου κα. Η κλάση Header, η οποία αναπαριστά την κεφαλίδα του αρχείου και εκτελεί λειτουργίες όπως διάβασμα/γράψιμο της κεφαλίδας και παροχή διάφορων πληροφοριών για το περιεχόμενό της. Η κλάση Column, η οποία αναπαριστά ένα πεδίο του αρχείου και παρέχει λειτουργίες χειρισμού όπως δημιουργία, ορισμός ονόματος κα. Τέλος, η κλάση Record η οποία αναπαριστά μια εγγραφή του αρχείου και αναλαμβάνει τη δημιουργία, την εγγραφή και την ανάγνωσή της.

Τέλος, η εφαρμογή DBF Editor, χρησιμοποιεί την βιβλιοθήκη χειρισμού αρχείων dBase και προσφέρει χειρισμό αυτών των αρχείων μέσω ενός γραφικού περιβάλλοντος. Άλλες λειτουργίες τις οποίες υποστηρίζει είναι η εξαγωγή δεδομένων σε αρχεία MS Excel, xml, csv, html καθώς και η μαζική εισαγωγή εγγραφών στο αρχείο από αρχεία MS Excel, xml, csv. Επίσης έχει τη δυνατότητα σύνθετης αναζήτησης εγγραφών του αρχείου βάση κριτηρίων τα οποία εισάγει ο χρήστης και προβολή καθώς και εξαγωγή των αποτελεσμάτων. Τέλος το οπτικό περιβάλλον της εφαρμογής(GUI) αναπτύχθηκε μέσω του Eclipse IDE και συνολικά η εφαρμογή αποτελείται από 21 κλάσεις.

# Abstract

dBase is one of the first database management systems . The first edition was published in 1980 by Ashton-Tate and since then other versions have been created and most important are dBase III and dBase IV. The dBase includes a programming language which offers better control and handling of records. Commands are executed via dot prompt using interpreter which makes the language quick in demanding applications. The main dBase file format is .dbf, which proved very successful because of its simplicity and now it used by other applications. Other major file formats used by dBase are .dbt for memo fields and .ndx, .mdx for files with indexes. The most recent version is the dBase PLUS 8 which incorporates new functionalities and can be executed in modern operating systems .

The language Java is a popular programming language which was designed by James Gosling for Sun Microsystems in the early ‘90s primarily for interactive TVs. The first edition was published in 1995 and till today many versions have been published. In 2006 the company made ​​the Java open source, and in 2010 Sun Microsystems and the language acquired by Oracle Corporation. The main feature of the language is portability, that means an application in Java runs on all operating systems without extra compilation. This is achieved through the Java Virtual Machine, which translates the code into Java bytecode and not directly into machine language. The Java was developed based on C and C++, and compared with those it is about 44% slower. Lastly, Java contains special classes which give additional functionality. Some of them are the applets, servlets and swing etc.

The swing is the primary tool for creating graphical user interface in Java language. Swing is the evolution of awt and offers more sophisticated components. All components are written in Java language and is thus independent of any platform. The SWT is a library similar to Swing, which was developed by IBM and the Eclipse Foundation and supports display of GUI application identical to applications written with the native API functions.

The Eclipse is an Integrated Development Environment (IDE), through which can be done software development in various programming languages. The original code of Eclipse came from the IBM VisualAge. In nowadays, Eclipse developed by the Eclipse Foundation which publishes a new version every year. Finally, various extensions created for Eclipse and that provide new abilities to developers. The most important are pydev and ADT for python application development and Android respectively.

The handling dBase files library is a collection of classes that enable handling dBase files through Java language. The main classes of the library are: The DbaseFile, which represents a dBase file and performs various functions such as creating, opening, pack a file etc. The class Header, which represents the file header and performs functions such as reading / writing the header and provide a variety of information about its content. The class Column, which represents a field of file and provides a variety of functions like creating, and name the field etc . Finally, the class Record represents a record in the file and takes over the creation, the writing and reading it.

Finally, the application DBF Editor, use the library for the handling dBase files and offers handling these files through a graphical interface. Other functions which supports are the export of data to MS Excel, xml, csv, html files and mass insert of records in the file from MS Excel, xml, csv files. Also has the ability of advanced search of records in file based on criteria given by the user and view and export the results. Finally, the visual interface of the application developed using Eclipse IDE and overall application consists of 21 classes.

# Ευρετήριο Περιεχομένων

[Πρόλογος-Εισαγωγή 1](#_Toc367538106)

[Περίληψη 3](#_Toc367538107)

[Abstract 5](#_Toc367538108)

[Ευρετήριο Περιεχομένων 7](#_Toc367538109)

[Ευρετήριο Εικόνων 10](#_Toc367538110)

[Ευρετήριο Πινάκων 12](#_Toc367538111)

[1 Το σύστημα διαχείρισης δεδομένων dBase 13](#_Toc367538112)

[1.1 Εισαγωγή 13](#_Toc367538113)

[1.2 Ιστορικά στοιχεία 13](#_Toc367538114)

[1.3 Πρόσφατη ιστορία 14](#_Toc367538115)

[1.4 Η γλώσσα προγραμματισμού dBase 15](#_Toc367538116)

[1.5 Θετικά στοιχεία του συστήματος 15](#_Toc367538117)

[1.6 File formats(Τύποι αρχείων) 16](#_Toc367538118)

[1.7 Η δομή του αρχείου .DBF(Αναλυτικά) 16](#_Toc367538119)

[1.7.1 Επεξήγηση της δομής του αρχείου 19](#_Toc367538120)

[1.8 Δομή Memo αρχείου(.DBT) 21](#_Toc367538121)

[1.9 Δομή αρχείου index (.ndx) 23](#_Toc367538122)

[1.9.1 Αλγόριθμος αναζήτησης στα αρχεία index. 25](#_Toc367538123)

[1.10 Η δομή αρχείου index (.mdx) 25](#_Toc367538124)

[1.11 Προδιαγραφές-περιορισμοί της dBase 26](#_Toc367538125)

[1.12 Σημαντικές καταλήξεις αρχείων της dBase III 26](#_Toc367538126)

[2 Μια ματιά στη γλώσσα προγραμματισμού Java 27](#_Toc367538127)

[2.1 Εισαγωγή 27](#_Toc367538128)

[2.2 Ιστορικά στοιχεία 27](#_Toc367538129)

[2.3 Εικονική Μηχανή(Java Virtual Machine) 28](#_Toc367538130)

[2.4 Απόδοση της γλώσσας 28](#_Toc367538131)

[2.5 Συντακτικό της γλώσσας 28](#_Toc367538132)

[2.6 Περιβάλλον ανάπτυξης 29](#_Toc367538133)

[2.7 Ειδικές κλάσεις 29](#_Toc367538134)

[3 Η εργαλειοθήκη Java Swing 29](#_Toc367538135)

[3.1 Εισαγωγή 29](#_Toc367538136)

[3.2 Ιστορικά στοιχεία 29](#_Toc367538137)

[3.3 Standard Widget Toolkit(SWT) 30](#_Toc367538138)

[4 Το περιβάλλον ανάπτυξης Eclipse IDE 30](#_Toc367538139)

[4.1 Εισαγωγή 30](#_Toc367538140)

[4.2 Επεκτάσεις(extensions) 31](#_Toc367538141)

[4.3 Εκδόσεις του Eclipse IDE 31](#_Toc367538142)

[5 Η βιβλιοθήκη χειρισμού αρχείων dBase(.dbf) 31](#_Toc367538143)

[5.1 Εισαγωγή 31](#_Toc367538144)

[5.2 Κλάση DbaseFile 33](#_Toc367538145)

[5.3 Κλάση Header 34](#_Toc367538146)

[5.4 Κλάση Column 36](#_Toc367538147)

[5.5 Κλάση Record 36](#_Toc367538148)

[5.6 Παρουσίαση του API της βιβλιοθήκης για τον χειρισμό DBF αρχείων 40](#_Toc367538149)

[5.6.1 Κλάση DBaseFile. 40](#_Toc367538150)

[5.6.2 Κλάση Header 43](#_Toc367538151)

[5.6.3 Κλάση Column 48](#_Toc367538152)

[5.6.4 Κλάση Record 50](#_Toc367538153)

[6 Η εφαρμογή DBF Editor 53](#_Toc367538154)

[6.1 Κλάσεις OpenDbfFrame και OpenFileProgressDialog 54](#_Toc367538155)

[6.2 Κλάσεις AddDialog/EditDialog 55](#_Toc367538156)

[6.3 Κλάσεις AddDBF,AddField,EditField 55](#_Toc367538157)

[6.4 Κλάσεις ImportFromCsvProgressDialog, ImportFromExcelProgressDialog, ImportFromXMLProgressDialog 55](#_Toc367538158)

[6.5 Κλάσεις FindDIalog και ReplaceDialog 56](#_Toc367538159)

[6.6 Κλάσεις MEMODialog και EditMemoFieldDialog 56](#_Toc367538160)

[6.7 Κλάσεις AdvancedFilterDialog και FilteredDataDialog 56](#_Toc367538161)

[6.8 Κλάση RecordViewDialog 57](#_Toc367538162)

[6.9 Κλάσεις ModifyStructureDialog και ChangeStructureProgressDialog. 57](#_Toc367538163)

[6.10 Κλάσεις AboutDialog και StructureDialog 58](#_Toc367538164)

[6.11 Οδηγός χρήσης του λογισμικού DBF Editor 58](#_Toc367538165)

[6.11.1 Το κεντρικό μενού(Main Menu) 59](#_Toc367538166)

[6.11.2 Η γραμμή εργαλείων (Tool Bar) 87](#_Toc367538167)

[6.11.3 Κύρια περιοχή φόρτωσης δεδομένων(Main Area) 94](#_Toc367538168)

[6.11.4 Shortcuts(Συντομεύσεις στο λογισμικό). 96](#_Toc367538169)

[7 Συμπεράσματα-Επίλογος 97](#_Toc367538170)

[Βιβλιογραφία 98](#_Toc367538171)

[Παράρτημα 102](#_Toc367538172)

[Αναφορές 104](#_Toc367538173)

# Ευρετήριο Εικόνων

[Εικόνα 1: Το λογότυπο της dBase 13](file:///C:\Users\Efthimis\Desktop\πτυχιακη-word\πτυχιακη.docx#_Toc367538174)

[Εικόνα 2: Η δομή ενός dBase(.dbf) αρχείου 18](#_Toc367538175)

[Εικόνα 3: Η δομή του αρχείου memo(.dbt) 22](#_Toc367538176)

[Εικόνα 4: Δομή αρχείου index(.ndx) 24](#_Toc367538177)

[Εικόνα 5: Το λογότυπο της Java 27](#_Toc367538178)

[Εικόνα 6: Το λογότυπο του Eclipse IDE 30](#_Toc367538179)

[Εικόνα 7: Διάγραμμα κλάσεων 32](#_Toc367538180)

[Εικόνα 8: Το περιβάλλον του λογισμικού DBF Editor 59](#_Toc367538181)

[Εικόνα 9: Main Menu 60](#_Toc367538182)

[Εικόνα 10: File Menu 61](#_Toc367538183)

[Εικόνα 11: Άνοιγμα αρχείου dBase 61](#_Toc367538184)

[Εικόνα 12: Νέο αρχείο .dbf 62](#_Toc367538185)

[Εικόνα 13: Προσθήκη πεδίου 62](#_Toc367538186)

[Εικόνα 14: Τροποποίηση πεδίου 63](#_Toc367538187)

[Εικόνα 15: Διαγραφή πεδίου 63](#_Toc367538188)

[Εικόνα 16: Αναδιάταξη πεδίων 64](#_Toc367538189)

[Εικόνα 17: Δημιουργία του αρχείου 64](#_Toc367538190)

[Εικόνα 18: Εξαγωγή δεδομένων 65](#_Toc367538191)

[Εικόνα 19: Εισαγωγή εγγραφών από αρχείο 65](#_Toc367538192)

[Εικόνα 20: Κλείσιμο αρχείου 66](#_Toc367538193)

[Εικόνα 21: ‘Έξοδος από την εφαρμογή 66](#_Toc367538194)

[Εικόνα 22: Edit Menu 67](#_Toc367538195)

[Εικόνα 23: Προσθήκη εγγραφής 68](#_Toc367538196)

[Εικόνα 24: Τροποποίηση μιας εγγραφής 68](#_Toc367538197)

[Εικόνα 25: Διαγραφή εγγραφής 69](#_Toc367538198)

[Εικόνα 26: Επαναφορά εγγραφών 69](#_Toc367538199)

[Εικόνα 27: Pack Database 70](#_Toc367538200)

[Εικόνα 28: Αλλαγή της δομής 71](#_Toc367538201)

[Εικόνα 29: Προσθήκη πεδίου 71](#_Toc367538202)

[Εικόνα 30: Τροποποίηση πεδίου 72](#_Toc367538203)

[Εικόνα 31: Διαγραφή πεδίου 72](#_Toc367538204)

[Εικόνα 32: Αναδιάταξη των πεδίων 73](#_Toc367538205)

[Εικόνα 33: Αλλαγή της δομής 73](#_Toc367538206)

[Εικόνα 34: Αναζήτηση εγγραφής 74](#_Toc367538207)

[Εικόνα 35: Αναζήτηση και αντικατάσταση εγγραφής 74](#_Toc367538208)

[Εικόνα 36: View Menu 75](#_Toc367538209)

[Εικόνα 37: Μέτρηση εγγραφών 75](#_Toc367538210)

[Εικόνα 38: Πληροφορίες για το αρχείο 76](#_Toc367538211)

[Εικόνα 39: Η δομή του αρχείου 76](#_Toc367538212)

[Εικόνα 40: Advanced Filter 77](#_Toc367538213)

[Εικόνα 41: Επιλογή πεδίων 78](#_Toc367538214)

[Εικόνα 42: Επιλογή τελεστών και εισαγωγή τιμών για σύγκριση 78](#_Toc367538215)

[Εικόνα 43: Επιλογή των εγγραφών που θα πάρουν μέρος στην αναζήτηση 79](#_Toc367538216)

[Εικόνα 44: Εφαρμογή του φίλτρου 79](#_Toc367538217)

[Εικόνα 45: Προβολή των αποτελεσμάτων του φίλτρου 80](#_Toc367538218)

[Εικόνα 46: Καθαρισμός του φίλτρου 80](#_Toc367538219)

[Εικόνα 47: Επιλογές διαγραφής εγγραφών 81](#_Toc367538220)

[Εικόνα 48: Εφαρμογή της σύνθετης διαγραφής 81](#_Toc367538221)

[Εικόνα 49: Καθαρισμός του φίλτρου 82](#_Toc367538222)

[Εικόνα 50: Επιλογές επαναφοράς εγγραφών 82](#_Toc367538223)

[Εικόνα 51: Εφαρμογή φίλτρου για επαναφορά εγγραφών 83](#_Toc367538224)

[Εικόνα 52: Καθαρισμός του φίλτρου 83](#_Toc367538225)

[Εικόνα 53: Record View 84](#_Toc367538226)

[Εικόνα 54: Navigate Menu 85](#_Toc367538227)

[Εικόνα 55: Επόμενη εγγραφή 85](#_Toc367538228)

[Εικόνα 56: Προηγούμενη εγγραφή 86](#_Toc367538229)

[Εικόνα 57: Πρώτη εγγραφή 86](#_Toc367538230)

[Εικόνα 58: Τελευταία εγγραφή 87](#_Toc367538231)

[Εικόνα 59: Γραμμή Εργαλείων 88](#_Toc367538232)

[Εικόνα 60: Open DIalog 88](#_Toc367538233)

[Εικόνα 61: Επόμενη εγγραφή 89](#_Toc367538234)

[Εικόνα 62: Προηγούμενη εγγραφή 89](#_Toc367538235)

[Εικόνα 63: Πρώτη εγγραφή 90](#_Toc367538236)

[Εικόνα 64: Τελευταία εγγραφή 90](#_Toc367538237)

[Εικόνα 65: Απευθείας μετάβαση σε εγγραφή 91](#_Toc367538238)

[Εικόνα 66: Quick filter 91](#_Toc367538239)

[Εικόνα 67: Προσθήκη νέας εγγραφής 92](#_Toc367538240)

[Εικόνα 68: Τροποποίηση μιας εγγραφής 92](#_Toc367538241)

[Εικόνα 69: Διαγραφή εγγραφής 93](#_Toc367538242)

[Εικόνα 70: Επαναφορά εγγραφής 93](#_Toc367538243)

[Εικόνα 71: Main Area 94](#_Toc367538244)

[Εικόνα 72: Ταξινόμηση με βάση ένα πεδίο 94](#_Toc367538245)

[Εικόνα 73: Τροποποίηση εγγραφής 95](#_Toc367538246)

[Εικόνα 74: Προβολή περιεχομένου πεδίου τύπου MEMO 95](#_Toc367538247)

# Ευρετήριο Πινάκων

[Πίνακας 1: Παραδείγματα τύπων αρχείων 19](#_Toc367538248)

[Πίνακας 2: Τιμές του πεδίου και τα αντίστοιχα code pages 19](#_Toc367538249)

[Πίνακας 3: Προδιαγραφές-Περιορισμοί 26](#_Toc367538250)

[Πίνακας 4: Καταλήξεις αρχείων 26](#_Toc367538251)

[Πίνακας 5: Συντομεύσεις λογισμικού 96](#_Toc367538252)

# Το σύστημα διαχείρισης δεδομένων dBase

## Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζω το σύστημα διαχείρισης δεδομένων dBase. Παραθέτω κάποια ιστορικά στοιχεία, παρουσιάζω τη γλώσσα προγραμματισμού dBase και αναλύω τη δομή των αρχείων της.

Η dBase δημοσιεύτηκε από την εταιρία Ashton-Tate και ήταν από τα πρώτα ευρέως χρησιμοποιούμενα συστήματα διαχείρισης βάσεων δεδομένων. Η dBase περιλαμβάνει μία database engine[[1]](#endnote-2), ένα σύστημα ερωτημάτων και μια γλώσσα προγραμματισμού που «δένει» όλα αυτά τα συστατικά μεταξύ τους. Η dBase χρησιμοποιεί ως file format τα αρχεία .dbf τα οποία χρησιμοποιούνται και από άλλες εφαρμογές οι οποίες χρειάζονται να αποθηκεύσουν δομημένα δεδομένα. Ο τύπος αυτός αναλύεται παρακάτω διεξοδικά.

Η Dbase δημοσιεύτηκε το 1980 για το λειτουργικό σύστημα CP/M και αργότερα για τους υπολογιστές της Apple και της IBM. Στα μέσα της δεκαετίας του ’80 άλλες εταιρίες δημιούργησαν τις δικές τους εκδόσεις για το προϊόν και για τη γλώσσα προγραμματισμού. Οι πιο γνωστές είναι οι Clipper και η FoxPro, οι οποίες μαζί αναφέρονται ως xBase. Παρά την επιτυχία της Dbase III, με την έλευση της έκδοσης IV τα πράγματα άλλαξαν καθώς η νέα έκδοση αποδείχθηκε ασταθής κάτι που ώθησε αρκετούς πελάτες να στραφούν σε άλλες λύσεις όπως η SQL.



Εικόνα 1: Το λογότυπο της dBase

## Ιστορικά στοιχεία

Στα τέλη της δεκαετίας του ’60 ο Fred Thompson χρησιμοποιούσε στο Jet Propulsion Laboratory(JPL)(παράρτημα της NASA), το προιόν RETRIEVE για να αποθηκεύει και να διαχειρίζεται τα δεδομένα ηλεκτρονικών υπολογισμών. Το 1971 συνεργάστηκε με τον Jack Hatfield, με σκοπό να βελτιώσουν το λογισμικό RETRIEVE, δημιουργώντας έτσι το project JPLDIS. Αργότερα ο C. Wayne Ratlif, ο οποίος εργαζόταν για την JPL, θέλοντας να χρησιμοποιήσει ένα σύστημα για να επεξεργαστεί κάποια δεδομένα στηρίχτηκε στο project JPLDIS για να δημιουργήσει μια έκδοση για το λειτουργικό PTDOS την οποία ονόμασε Vulcan.

Οι George Tate και Hal Lashlee, ήρθαν σε συμφωνία με τον Ratlif έτσι ώστε να δημιουργήσει μια εμπορική έκδοση για το Vulcan. Πράγματι ο Ratlif δημιούργησε μια έκδοση για το λειτουργικό CP/M η οποία ονομάστηκε dBase II. Το 1981 δημιουργήθηκε και μια έκδοση για τα PC της IBM κάνοντας έτσι την dBase ένα από τα λίγα επαγγελματικά προγράμματα που υπήρχαν εκείνη την εποχή, κάτι που οδήγησε στην τεράστια επιτυχία του λογισμικού.

Καθώς τα λειτουργικά συστήματα γινόταν περισσότερα και άλλαζαν η εταιρία δυσκολεύτηκε να εισάγει το λογισμικό που ήταν γραμμένο στη γλώσσα assembly στα νέα αυτά συστήματα. Αυτό οδήγησε στην εταιρία να ξαναγράψει το λογισμικό στη γλώσσα C, κάτι που έφερε ως τελικό προϊόν το 1984 μια νέα έκδοση, τη dBase III. Παρόλο που στην αρχή είχε χαμηλή απόδοση μέσω των μελλοντικών αναβαθμίσεων το λογισμικό έκανε ξανά μεγάλη επιτυχία, κάτι που φαίνεται και στην κατάσταση της εταιρίας εκείνη την εποχή η οποία απασχολούσε 500 υπαλλήλους και είχε ετήσιο τζίρο 40 εκατομμύρια δολάρια.

## Πρόσφατη ιστορία

Η dBase πλέον έχει εξελιχθεί σε μια σύγχρονη αντικειμενοστραφή γλώσσα η οποία «τρέχει» στα σύγχρονα οπτικά περιβάλλοντα και μπορεί να χρησιμοποιηθεί για μεγάλη ποικιλία εφαρμογών όπως web applications, client application κτλ. Η dBase επίσης μπορεί να έχει πρόσβαση και στις σύγχρονες database engines μέσω των ODBC drivers.

Η πιο πρόσφατη έκδοση είναι η dBASE PLUS 8, η οποία κυκλοφόρησε το Μάρτιο του 2013. Η έκδοση αυτή δεν υποστηρίζει μόνο την κλασική BDE(Borland Database Engine), αλλά έχει και υποστήριξη για ADO και ODBC συνδεσιμότητα. Επίσης έχει βελτιωθεί το οπτικό περιβάλλον έτσι ώστε να μπορεί να λειτουργήσει με τα σύγχρονα λειτουργικά συστήματα όπως είναι τα Microsoft Windows 8. Επίσης το προϊόν παρέχει και μερική υποστήριξη σε 16bit εκδόσεις και παλιότερες DOS εκδόσεις όπως οι dBase II, dBase III, dBase IV κτλ.

## Η γλώσσα προγραμματισμού dBase

Για τη διαχείριση των δεδομένων η dBase, προσφέρει διαδικαστικές εντολές και συναρτήσεις για την προσπέλαση των εγγραφών, το χειρισμό των πεδίων, το χειρισμό των αλφαριθμητικών, των αριθμών και των ημερομηνιών. Αυτές οι δυνατότητες οδήγησαν την Ashton-Tate να χαρακτηρίσει τη dBase σχεσιακή βάση δεδομένων. Παρόλα αυτά η dBase δεν είναι σχεσιακή βάση δεδομένων καθώς δεν τηρεί τα κριτήρια τα οποία έθεσε ο Dr. Edgar Codd ο οποίος περιγράφει το σχεσιακό μοντέλο.

Η dBase χρησιμοποιεί ένα διερμηνέα ο οποίος επιτρέπει την εκτέλεση εντολών σε περιβάλλον dot prompt. Μια εντολή εκτελείτε πατώντας το πλήκτρο ‘enter’. Ομοίως μπορούν να εκτελεστούν στο διερμηνέα και scripts (αρχεία με κατάληξη PRG). Αυτό κάνει την dBase γρήγορη καθώς οι εντολές εκτελούνται αμέσως χωρίς να πρέπει να περάσουν από τη διαδικασία του compiling. Με την πάροδο του χρόνου οι ανταγωνιστές της Ashton-Tate εισήγαγαν στη γλώσσα και άλλες λειτουργίες οι οποίες την έκαναν πιο δυνατή. Τέτοιες λειτουργίες ήταν οι user-defined functions(UDF)[[2]](#endnote-3), οι scoped μεταβλητές, οι πίνακες, το object-oriented συντακτικό κα. Επίσης η Ashton-Tate και οι ανταγωνιστές της άρχισαν σιγά-σιγά να ενσωματώνουν και την SQL στα προϊόντα τους.

Στα τέλη της δεκαετίας του ’80 η dBase έγινε standard(IEEE 1192), και ονομάστηκε xBase για να ξεχωρίζει από το προϊόν της Ashton-Tate.

Οι σημερινές υλοποιήσεις της dBase, έχουν επεκταθεί και πλέον περιλαμβάνουν αντικειμενοστραφή προγραμματισμό, λειτουργικότητα μέσω Internet, χειρισμό δεδομένων μέσω της SQL κα.

## Θετικά στοιχεία του συστήματος

Παρόλο που η δημοτικότητα της dBase έχει πέσει, κάποιοι συνεχίζουν να τη βρίσκουν αποτελεσματική. Για παράδειγμα είναι ευκολότερος ο χειρισμός, η διαμόρφωση, η ανάλυση και η εκτέλεση υπολογισμών σε μεμονωμένες εγγραφές σε σχέση με την χρήση της SQL. Γενικά οι λειτουργίες της dBase είναι πιο μικρές και γρηγορότερες από αυτές τις SQL.

Η dBase παρά την πτώση των πωλήσεων παρέμεινε ένα δημοφιλή εργαλείο μάθησης καθώς οι εντολές της ήταν ευκολότερες και περισσότερο κατανοητές από αυτές των ανταγωνιστών.

## File formats(Τύποι αρχείων)

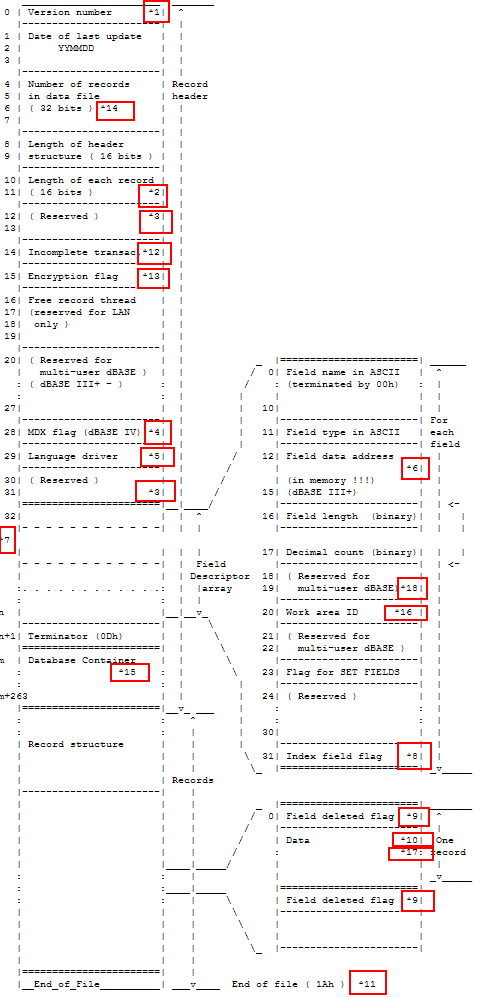
Το μεγαλύτερο επίτευγμα της dBase είναι το .dbf file format το οποίο χρησιμοποιείτε από πολλές εφαρμογές. Στην αρχή έχει ένα header(κεφαλίδα), το οποίο περιγράφει τη δομή του αρχείου και έτσι το λογισμικό δεν χρειάζεται να κρατάει άλλες πληροφορίες για τα αρχεία. Ο δεύτερος τύπος αρχείου που χρησιμοποιείτε είναι ο .dbt, για τα memo πεδία. Ο τύπος αυτός δίνει τη δυνατότητα αποθήκευσης μεγάλων κειμένων κάτι το οποίο δεν επιτρέπεται να γίνει μέσα στο αρχείο .dbf. Η dBase επίσης χρησιμοποιεί .ndx αρχεία για μονούς indexes και .mdx για πολλαπλούς indexes (από 1 μέχρι 48). Τέλος κάποια άλλα προϊόντα χρησιμοποιούν και άλλους τύπους αρχείων όπως .idx/.cdx η FoxPro και .ntx η Clipper.

## Η δομή του αρχείου .DBF(Αναλυτικά)

Τα αρχεία (\*.dbf), περιέχουν ένα header (κεφαλίδα) με πληροφορίες για τη δομή των δεδομένων και τις εγγραφές. Ένα πεδίο δεδομένων έχει ένα προκαθορισμένο τύπο ό οποίος δηλώνει το είδος των δεδομένων που μπορεί να αποθηκεύσει. Οι τύποι των δεδομένων οι οποίο συναντούνται στη dBase III+ είναι οι εξής: Character(‘C’), Numeric(‘N’), Logical(‘L’), Date(‘D’) και Memo(‘M’). Κάθε τύπος δεδομένων έχει και κάποιους περιορισμούς.

* Character(‘C’): Ένα πεδίο αυτού του τύπου μπορεί να έχει ASCII κείμενο με μέγιστο μήκος 254 χαρακτήρες. Αυτού του είδους τα πεδία μπορούν να έχουν και κείμενο μήκους 32 KB αν χρησιμοποιηθεί το πλήθος των δεκαδικών σαν high byte (το περισσότερο σημαντικό bit πρώτο) στο μήκος του πεδίου, ή 64KB αν διαβάζεται το μήκος ως unsigned (μόνο θετικές τιμές).
* Numeric(‘N’): Δέχεται αριθμούς με μέγιστο μήκος 18 ψηφία. Το μέγιστο πλήθος των δεκαδικών που μπορεί να περιέχει είναι 15 ψηφία, ενώ η διαφορά του μήκους του πεδίου με το πλήθος των δεκαδικών θα πρέπει να είναι τουλάχιστον 2 μονάδες.
* Logical(‘L’): Είναι πάντα μεγέθους 8bit (1 byte), και οι τιμές που μπορεί να πάρει είναι:
  + ? Δεν έχει αρχικοποιηθεί
  + Y, y, Yes
  + N, n, No
  + F, f, False
  + T, t, True
* Τα πεδία αυτού του τύπου πάντα αναπαρίστανται ως T/F/?.
* Date(‘D’): Η μορφοποίηση της ημερομηνίας είναι YYYYMMDD. Μια ημερομηνία της μορφής YYYY-MM-DD, δεν είναι αποδεκτή. Το μέγεθος όλων των πεδίων Date είναι πάντα 8bytes.
* Memo(‘M’): Είναι ASCII κείμενο μήκους 10 ψηφίων το οποίο αναπαριστά ένα pointer σε ένα block μέσα σε ένα DBT αρχείο.

Η μορφή της κεφαλίδας ενός dBase αρχείου(.dbf), είναι αυτή που φαίνεται στην εικόνα 2:

Εικόνα 2: Η δομή ενός dBase(.dbf) αρχείου

### Επεξήγηση της δομής του αρχείου

1. Ονομάζεται επίσης signature(υπογραφή). Δείχνει τον τύπο του αρχείου. Κάποια παραδείγματα αναφέρονται στο πίνακα 1.

Πίνακας 1: Παραδείγματα τύπων αρχείων

|  |  |
| --- | --- |
| Τιμή | Περιγραφή |
| 02h | Αρχείο τύπου FoxBase |
| 03h | Αρχείο τύπου dBase III χωρίς memo |
| 30h | Αρχείο τύπου Visual FoxPro |
| 83h | Αρχείο τύπου dBase III με memo |

1. Δίνει το άθροισμα του μήκους όλων των πεδίων + 1(deletion flag). Στην ουσία δίνει το μήκος της εγγραφής γι’ αυτό το αρχείο.
2. Γεμίζει με 00h, χρησιμοποιείτε στη dBase IV.
3. Χρησιμοποιείτε από την dBase IV και δηλώνει ή όχι την παρουσία Multiple Index File(.mdx). Οι τιμές που μπορεί να πάρει είναι: 01h υπάρχει .mdx αρχείο και 00h δεν υπάρχει .mdx αρχείο.
4. Χρησιμοποιείτε από τη FoxPro και οι τιμές που παίρνει δηλώνουν τα code pages (κωδικοποίηση χαρακτήρων) που χρησιμοποιούνται. Κάποια χαρακτηριστικά παραδείγματα φαίνονται στον πίνακα 2.

Πίνακας 2: Τιμές του πεδίου και τα αντίστοιχα code pages

|  |  |  |
| --- | --- | --- |
| Τιμή | Περιγραφή | Code Page |
| 01h | DOS USA | Code page 437 |
| 02h | DOS Multilingual | Code page 850 |
| 03h | Windows ANSI | Code page 1252 |
| 64h | EE MS-DOS | Code page 852 |
| 65h | Nordic MS-DOS | Code page 865 |
| 66h | Russian MS-DOS | Code page 866 |
| C8h | Windows EE | Code page 1250 |

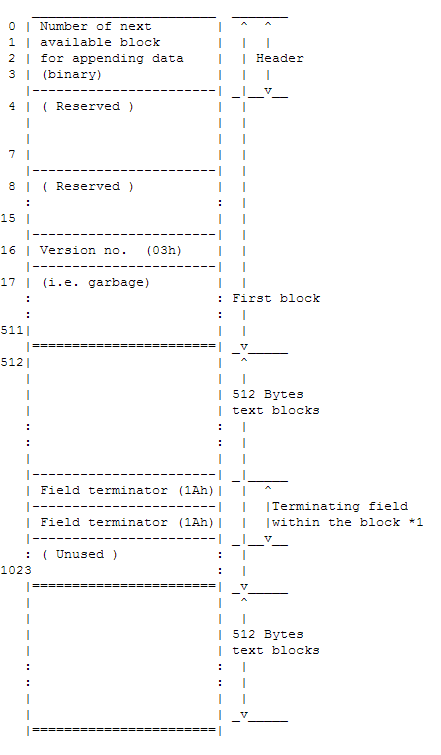
1. Δηλώνει τη διεύθυνση των δεδομένων του πεδίου. Στη dBase καταλαμβάνει τα bytes 12-15 και δηλώνει διεύθυνση στη μνήμη, ενώ στη FoxPro καταλαμβάνει τα bytes 12-13 και δηλώνει offset του πεδίου από την αρχή της εγγραφής.
2. Δείχνει το μήκος του πεδίου για τα μη αριθμητικά πεδία (FoxPro και Clipper).
3. Flag που δείχνει αν υπάρχει index για αυτό το πεδίο. 00h δεν υπάρχει index, 01h υπάρχει index στο αρχείο .mdx.
4. Περιέχει το deleted flag(δείχνει αν η εγγραφή έχει σημειωθεί για διαγραφή). Έχει τη τιμή 2Ah(\*) αν η εγγραφή είναι σημειωμένη για διαγραφή και την τιμή 20h(κενό), αν η εγγραφή δεν είναι σημειωμένη για διαγραφή.
5. Ο χώρος στον οποίο αποθηκεύονται τα δεδομένα μιας εγγραφής.
6. Είναι ο χαρακτήρας που δηλώνει το τέλος του αρχείου. Ο χαρακτήρας αυτός είναι ο 1Ah.
7. Χρησιμοποιείτε από τη dBase IV και δείχνει την κατάσταση μιας συναλλαγής. 00h - αν η συναλλαγή ολοκληρώθηκε ή έγινε roll back και 01h – αν η συναλλαγή ξεκίνησε.
8. Είναι ένα flag κρυπτογράφησης. Χρησιμοποιείτε από τη dBase IV. Δείχνει αν το περιεχόμενο είναι κρυπτογραφημένο. 00h - δεν είναι κρυπτογραφημένο, 01h – είναι κρυπτογραφημένο.
9. Δίνει το πλήθος των εγγραφών στο αρχείο.
10. Χρησιμοποιείτε στη Visual FoxPro.
11. Είναι το Work Area ID και σε όλα τα αρχεία της dBase III είναι 01h.
12. Ένα άδειο πεδίο Memo στη περιοχή των δεδομένων έχει 10 κενά.
13. Περιέχει flags για τα πεδία. Χρησιμοποιείτε με τις FoxPro, FoxBase. 01h – Πεδίο του συστήματος(μη ορατό), 02h – Το πεδίο μπορεί να έχει null τιμές, 04h – Binary πεδίο.

Σε ένα αρχείο dBase(.dbf), μπορούν κατά το άνοιγμα να γίνουν κάποιοι έλεγχοι για να ελέγξουμε την ορθότητά του.

* Για τις εγγραφές:
  + Το μήκος της εγγραφής θα πρέπει να είναι > 1 και το μέγιστο μήκος στη dBase III και dBase IV θα πρέπει να είναι <= 4000 Βytes. Επίσης ο αριθμός των εγγραφών θα πρέπει να είναι >= 0.
* Για τα πεδία:
  + Ένα αρχείο θα πρέπει να έχει τουλάχιστο ένα πεδίο. Επίσης ο αριθμός των πεδίων πρέπει να είναι μικρότερος από το μέγιστο αριθμό πεδίων που επιτρέπονται(255).
* Για το μέγεθος του αρχείου:
  + Το μέγεθος του αρχείου που αναφέρει το λειτουργικό θα πρέπει να ταιριάζει με το λογικό μέγεθος. Λογικό μέγεθος = Μήκος της κεφαλίδας + ( Πλήθος εγγραφών \* μήκος κάθε εγγραφής).

## Δομή Memo αρχείου(.DBT)

Τα dBase III Memo αρχεία περιέχουν blocks των 512 bytes, τα οποία αποθηκεύουν ASCII δεδομένα. Η δομή ενός αρχείου memo (.dbt), φαίνεται στην εικόνα 3:

Εικόνα 3: Η δομή του αρχείου memo(.dbt)

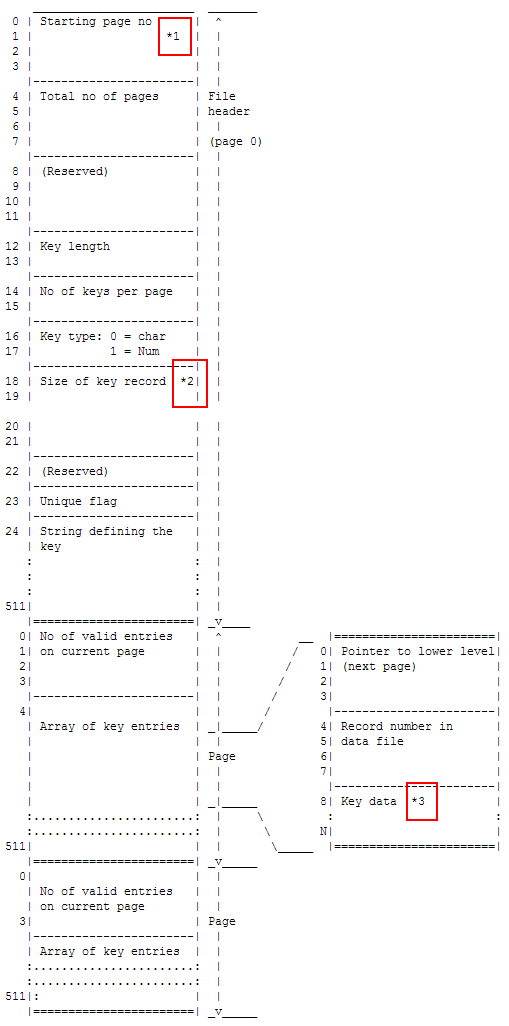
Τα πρώτα 4 bytes είναι το header(κεφαλίδα) του αρχείου και αναφέρουν τον αριθμό του επόμενου διαθέσιμου block για εγγραφή δεδομένων. Η κεφαλίδα περιλαμβάνεται στο πρώτο block του αρχείου(πρώτα 512 bytes). Από το πρώτο block τα υπόλοιπα 508 bytes δεν χρησιμοποιούνται. Ένα memo πεδίο στο αρχείο dBase(.dbf) περιέχει έναν αριθμό ο οποίος δηλώνει τον αριθμό του block στο αρχείο memo(.dbt), όπου είναι αποθηκευμένα τα δεδομένα. Ένα πεδίο μπορεί να έχει δεδομένα περισσότερα από 512 bytes (1 block). Τα δεδομένα συνεχίζουν και στο επόμενο ή επόμενα blocks. Ένα πεδίο τελειώνει όταν εμφανιστούν δύο(2) συνεχόμενοι χαρακτήρες τερματισμού. Ο χαρακτήρας τερματισμού είναι ο 1Ah. Μετά την εμφάνιση των χαρακτήρων τερματισμού ο υπόλοιπος χώρος στο block μένει αχρησιμοποίητος.

Κάθε φορά που γράφεται ένα πεδίο memo οι πληροφορίες του πεδίου γίνονται append στο τέλος του αρχείου .dbt. Στη dBase V, ο χώρος των διαγραμμένων εγγραφών ξαναχρησιμοποιείτε.

Ένα memo αρχείο(.dbt) από μόνο του δε λέει τίποτα για το πώς θα χρησιμοποιηθεί. Χρειάζεται το αντίστοιχο αρχείο DBF έτσι ώστε να διερμηνευτεί το περιεχόμενο του memo αρχείου.

## Δομή αρχείου index (.ndx)

Ένα index αρχείο είναι ένα B+ tree, αλλά περιγράφεται καλύτερα σαν εάν paged B- tree. Η δομή του είναι ένα ανεστραμμένο δέντρο με ένα anchor κόμβο, πολλαπλούς κόμβους ρίζες(roots) και κόμβους φύλλα(leafs). Η κεφαλίδα ονομάζεται anchor κόμβος. Συνήθως η κεφαλίδα διαβάζεται μια φορά από την εφαρμογή όταν το αρχείο ανοίγει. Το μέγεθος κάθε σελίδας(page) είναι πάντα 512 bytes. Η δομή ενός αρχείου .ndx φαίνεται στην εικόνα 4:

Εικόνα 4: Δομή αρχείου index(.ndx)

1. Είναι ο αριθμός της root page, είναι ένα offset που προκύπτει από τον αριθμό της σελίδας (page) \* 512(μέγεθος κάθε σελίδας).
2. Είναι το μέγεθος της εγγραφής.
3. Τα δεδομένα αποθηκεύονται σαν little endians (binary τιμές, με το λιγότερο σημαντικό byte στην αρχή).

### Αλγόριθμος αναζήτησης στα αρχεία index.

* Ο anchor node, μένει στη μνήμη όσο το αρχείο είναι ανοιχτό και καθορίζει σε ποιο root node θα αναζητήσει το key (κλείδί) που του δόθηκε.
* Ο root node διαβάζεται και σκανάρεται διαδοχικά μέχρι να βρεθεί ένα κλειδί το οποίο είναι >= με το επιθυμητό κλειδί.
* Ένας κόμβος δεύτερου επιπέδου σκανάρεται με τρόπο όμοιο με αυτό που ελέγχθηκε ο root node.
* Αυτή η διαδικασία συνεχίζεται μέχρι ο pointer για ένα κόμβο χαμηλότερου επιπέδου να έχει την τιμή 0. Αν το key που αναζητούμε έχει την ίδια τιμή με το key στον κόμβο-φύλλο, ο αριθμός της εγγραφής για το key που αναζητούσαμε επιστρέφεται.

Ένα από τα πλεονεκτήματα της χρήσης του B+ tree είναι ότι τα υψηλότερα επίπεδα του δέντρου δεν χρειάζονται να ενημερώνονται όταν κάποιος κόμβος αλλάζει – εκτός αν το δέντρο δεν είναι ισορροπημένο.

## Η δομή αρχείου index (.mdx)

Ένα αρχείο τύπου .mdx (multiple index) μπορεί να κρατάει από 1 μέχρι 48 indexes. Είναι αρχεία που χρησιμοποιούνται κυρίως από την έκδοση dBase IV και πάνω και γι’ αυτό δεν θα αναφερθώ αναλυτικά στη δομή τους.

## Προδιαγραφές-περιορισμοί της dBase

Πίνακας 3: Προδιαγραφές-Περιορισμοί

|  |  |
| --- | --- |
| Μέγιστος αριθμός εγγραφών | 1.000.000.000 |
| Αριθμός πεδίων ανά εγγραφή | 32 |
| Μέγιστος αριθμός ανοιχτών αρχείων | 10 |
| Μέγιστος αριθμός ανοιχτών αρχείων indexes | 7 |
| Χαρκτήρες ανά εγγραφή | 4000 |
| Αριθμός μεταβλητών στη μνήμη | 256 |
| Μέγιστο μέγεθος του Memo αρχείου | 5000 bytes |
| Μέγιστος αριθμός χαρακτήρων σε πεδίο τύπου Character | 254 |
| Μέγιστος αριθμός ψηφίων σε πεδίο τύπου Numeric | 18 |
| Μέγιστο μήκος πεδίου τύπου Logical | 1 |
| Μέγιστο μήκος πεδίου τύπου Date | 8 |
| Μέγιστο μήκος πεδίου τύπου Memo | 10 |
| Αριθμός τύπων πεδίων | 5 |
| Μνήμη για της μεταβλητές | 6000bytes \* 3 |

## Σημαντικές καταλήξεις αρχείων της dBase III

Πίνακας 4: Καταλήξεις αρχείων

|  |  |
| --- | --- |
| Κατάληξη | Περιγραφή |
| .$$$ | Προσωρινό αρχείο |
| .bak | Αρχείο backup |
| .db3 | Database |
| .dbf | Database αρχείο |
| .dbt | Αρχείο Memo |
| .ndx | Αρχείο index |

# Μια ματιά στη γλώσσα προγραμματισμού Java

## Εισαγωγή

Στο κεφάλαιο ‘Μια ματιά στη γλώσσα προγραμματισμού Java’ θα αναφερθώ στη γλώσσα προγραμματισμού μέσω της οποίας υλοποίησα τη βιβλιοθήκη χειρισμού αρχείων dBase. Αρχικά αναφέρω κάποια ιστορικά στοιχεία, και στη συνέχεια δίνω στοιχεία για το συντακτικό της γλώσσας, την απόδοσή της και την εικονική μηχανή της.

Η Java είναι μια γλώσσα γενικού σκοπού, αντικειμενοστραφής η οποία υλοποιήθηκε σε C και C++. Έχει σχεδιαστεί έτσι ώστε να είναι ανεξάρτητη από το λειτουργικό στο οποίο τρέχει. Έτσι ο κώδικας ο οποίος τρέχει σε μια πλατφόρμα δεν χρειάζεται να περάσει από τη διαδικασία του compiling ξανά έτσι ώστε να τρέξει σε μια άλλη. Είναι μια δημοφιλή γλώσσα η οποία, υπολογίζεται, ότι χρησιμοποιείται από 10 εκατομμύρια χρήστες παγκοσμίως. Αναπτύχθηκε από τον James Gosling στη Sun Microsystems και έχει όμοιο συντακτικό με τις C και C++, έχοντας όμως λιγότερες λειτουργίες χαμηλού επιπέδου από ότι αυτές.

Εικόνα 5: Το λογότυπο της Java

## Ιστορικά στοιχεία

Οι James Gosling, Mike Sheridan και Patrick Naughton ξεκίνησαν τη δημιουργία της γλώσσας το 1991. Αρχικά η Java δημιουργήθηκε για διαδραστικές τηλεοράσεις, αλλά αποδείχθηκε πολύ περίπλοκη για τη βιομηχανία της καλωδιακής τηλεόρασης. Στην αρχή η ονομασία της γλώσσας ήταν Oak μετέπειτα έγινε Green και τελικά ονομάστηκε Java. Η πρώτη έκδοση δημοσιοποιήθηκε το 1995 και γρήγορα έγινε δημοφιλής. Τον Νοέμβριο του 2006 η Sun δημοσιοποιεί ένα μεγάλο μέρος του κώδικα της Java και έτσι η γλώσσα γίνεται ανοιχτού κώδικα γλώσσα (GPL) όσον αφορά το μεταγλωττιστή της (javac) και το πακέτο ανάπτυξης (SDK). Τον Απρίλιο του 2010 η εταιρεία λογισμικού Oracle Corporation εξαγόρασε την Sun Microsystems και μαζί με αυτή και τη Java. Τέλος, η δημοτικότητα της γλώσσας ανέβηκε κατακόρυφα όταν η Google αποφάσισε να χρησιμοποιήσει την Java σαν κύρια γλώσσα για τη συγγραφή εφαρμογών για το λειτουργικό σύστημα Android. Το Android είναι ένα λειτουργικό ανοιχτού κώδικα το οποίο αν και είναι χτισμένο πάνω σε linux kerner που χρησιμοποιεί τη γλώσσα C, όλο το SDK του χρησιμοποιεί τη Java για την δημιουργία των εφαρμογών.

## Εικονική Μηχανή(Java Virtual Machine)

Το κύριο χαρακτηριστικό της Java είναι η φορητότητα, η οποία σημαίνει ότι τα προγράμματα που γράφονται σε αυτή τη γλώσσα, «τρέχουν» όμοια σε όλες τις πλατφόρμες. Αυτό επιτυγχάνεται μεταφράζοντας τον κώδικα σε μια ενδιάμεση αναπαράσταση η οποία ονομάζεται Java bytecode, αντί να μεταφραστεί απευθείας σε γλώσσα μηχανής. Αυτό όμως έχει και ένα μεγάλο μειονέκτημα το οποίο είναι η μείωση της ταχύτητας εκτέλεσης των προγραμμάτων.

## Απόδοση της γλώσσας

Τα προγράμματα τα οποία είναι γραμμένα σε Java έχουν τη «φήμη» ότι «τρέχουν» πιο αργά και καταναλώνουν περισσότερη μνήμη από αυτά που έχουν γραφτεί σε κάποια άλλη γλώσσα π.χ. C++. Παρόλα αυτά η ταχύτητα εκτέλεσης των προγραμμάτων έχει αυξηθεί με την προσθήκη του Just-in-Time compilation, καθώς και με βελτιστοποιήσεις οι οποίες έγιναν στην εικονική μηχανή. Αναφορικά η τελευταία έκδοση της Java (se 7), είναι κατά προσέγγιση 44% πιο αργή από τη C++. Τέλος, υπάρχουν κάποιες πλατφόρμες οι οποίες για να επιταχύνουν της εκτέλεση του κώδικα προσφέρουν άμεση υποστήριξη hardware στη Java, για παράδειγμα έχουν microcontrollers οι οποίοι εκτελούν τον κώδικα Java στο hardware αντί να τον εκτελέσουν στη virtual machine.

## Συντακτικό της γλώσσας

Το συντακτικό της Java κατά ένα μεγάλο μέρος προέρχεται από τη γλώσσα C++. Σε αντίθεση όμως με τη C++ η οποία συνθέτει το συντακτικό από δομημένο, γενικό, και αντικειμενοστραφή προγραμματισμό, η Java φτιάχτηκε εξ’ ολοκλήρου σαν μια αντικειμενοστραφής γλώσσα. Όλος ο κώδικας γράφεται μέσα σε μία κλάση και όλα είναι ένα αντικείμενο, με εξαίρεση τους κύριους τύπους δεδομένων(int, float,double κτλ) οι οποίοι δεν είναι κλάσεις. Η Java επίσης δεν υποστηρίζει υπερφόρτωση τελεστών και πολλαπλή κληρονομικότητα, αυτά όμως απλοποιούν τη γλώσσα και βοηθάνε στην αντιμετώπιση προβλημάτων.

## Περιβάλλον ανάπτυξης

Τα προγράμματα σε Java μπορούν να γραφτούν σε έναν οποιοδήποτε επεξεργαστή κειμένου, αλλά για μεγάλα έργα χρησιμοποιούνται κυρίως ολοκληρωμένα περιβάλλοντα ανάπτυξης (IDE). Ένα από αυτά είναι το Eclipse, λίγα λόγια για το οποίο θα αναφέρω παρακάτω.

## Ειδικές κλάσεις

Η Java έχει ειδικές κλάσεις όπως είναι τα applets τα οποία είναι προγράμματα τα οποία ενσωματώνονται κυρίως σε ιστοσελίδες. Επίσης έχει τη κλάση servlet η οποία προσφέρει στους web developers ένα μηχανισμό για την επέκταση της λειτουργίας ενός Web Server. Τέλος έχει και την κλάση swing η οποία επιτρέπει την δημιουργία εφαρμογών με οπτικό περιβάλλον. Αυτή την κλάση θα την παρουσιάσω παρακάτω αναλυτικότερα.

# Η εργαλειοθήκη Java Swing

## Εισαγωγή

Το κεφάλαιο αυτό αναφέρεται στο σύνολο των κλάσεων που αποτελούν την εργαλειοθήκη της Swing μέσω της οποίας ανάπτυξα το λογισμικό χειρισμού dBase αρχείων μέσα από γραφικό περιβάλλον. Παρατίθενται ιστορικά στοιχεία και γίνεται μια αναφορά στην εργαλειοθήκη SWT της Eclipse Foundation.

Η Swing είναι η κύρια εργαλειοθήκη για τη δημιουργία γραφικού περιβάλλοντος στη Java. Είναι κομμάτι της Java Foundation Classes(JFC) που είναι ένα api για τη δημιουργία γραφικών διεπαφών. Δημιουργήθηκε για να παρέχει πιο εξελιγμένα γραφικά components από τον προκάτοχό της(AWT). Προσφέρει το native look and feel κάθε πλατφόρμας, αλλά έχει και ένα γενικό το οποίο εφαρμόζεται ανεξάρτητα από την πλατφόρμα στην οποία «τρέχει» η εφαρμογή. Όλα τα components έχουν γραφτεί σε Java και έτσι είναι ανεξάρτητα από κάθε πλατφόρμα.

## Ιστορικά στοιχεία

Η Internet Foundation Classes(IFC) ήταν η γραφική βιβλιοθήκη για την Java η οποία αρχικά αναπτύχτηκε από την Netscape Communications Corporation και δημοσιοποιήθηκε το 1996. Το 1997 η Netscape Communications Corporation και η Sun Microsystems συνεργάστηκαν και στην IFC πρόσθεσαν και άλλες τεχνολογίες δημιουργώντας το framework Java Foundation Classes, που αργότερα μετονομάστηκε σε Swing. Στην αρχή διατέθηκε σαν ξεχωριστή βιβλιοθήκη αλλά στη συνέχεια ενσωματώθηκε στη Java Standard Edition 1.2. Όλες οι κλάσεις της swing πλέον περιέχονται στο πακέτο javax.swing.

## Standard Widget Toolkit(SWT)

Το SWT είναι μια γραφική open source εργαλειοθήκη για τη δημιουργία γραφικών διεπαφών. Αρχικά αναπτύχθηκε από την IBM αλλά τώρα αναπτύσσεται από την Eclipse Foundation και χρησιμοποιείτε με το Eclipse IDE. Για να απεικονίσει τα γραφικά στοιχεία προσπελαύνει τις native γραφικές βιβλιοθήκες της εκάστοτε πλατφόρμας μέσω του Java Native Interface(JNI), έτσι ώστε το interface μιας εφαρμογής που γράφτηκε σε Java να είναι ίδιο με αυτό μιας εφαρμογής που γράφτηκε χρησιμοποιώντας το API του συστήματος. Η υλοποίηση του SWT είναι διαφορετική για κάθε πλατφόρμα.

# Το περιβάλλον ανάπτυξης Eclipse IDE

## Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται το ολοκληρωμένο περιβάλλον ανάπτυξης Eclipse IDE μέσω του οποίου ανέπτυξα την βιβλιοθήκη και το λογισμικό. Αναφέρονται ιστορικά στοιχεία και γίνεται αναφορά στις επεκτάσεις του και στις εκδόσεις του.

Εικόνα 6: Το λογότυπο του Eclipse IDE

Το Eclipse είναι ένα Ολοκληρωμένο Περιβάλλον Ανάπτυξης(IDE), το οποίο προσφέρει τη δυνατότητα συγγραφής προγραμμάτων σε διάφορες γλώσσες προγραμματισμού όπως Java, Ada, C, C++, COBOL, Fortan, Javascript, Perl, PHP, Python, Ruby, Scala, Scheme και άλλες. Αποτελείτε από ένα βασικό workspace και από σύστημα επεκτάσεων για προσαρμογή του περιβάλλοντος στις ανάγκες του κάθε προγραμματιστή. Το λογισμικό Eclipse κατά ένα μεγάλο μέρος έχει γραφεί στη γλώσσα Java. Το Eclipse χρησιμοποιείται επίσης για συγγραφή πακέτων για το λογισμικό Mathematica, καθώς και για την ανάπτυξη διαφόρων περιβαλλόντων ανάπτυξης όπως είναι το Eclipse Java Development Tools(JDT) για την Java και τη Scala.

Ο αρχικός κώδικας του Eclipse προέρχεται από το IBM VisualAge που ήταν ένα IDE το οποίο υποστήριζε πολλές γλώσσες. Η Eclipse Foundation δημιουργήθηκε το 2004 και σχεδόν αμέσως δημοσιοποίησε το Eclipse 3.0.

## Επεκτάσεις(extensions)

Το Eclipse υποστηρίζει ένα μεγάλο αριθμό από επεκτάσεις οι οποίες δίνουν νέες δυνατότητες στους προγραμματιστές. Οι πιο σημαντικές είναι οι: pydev για υποστήριξη της γλώσσας Python, ADT την οποία προσφέρει η Google για ανάπτυξη εφαρμογών στο λειτουργικό Android, JavaFx για ανάπτυξη διαδικτυακών εφαρμογών.

## Εκδόσεις του Eclipse IDE

Από το 2006 η Eclipse Foundation έχει δρομολογήσει τη διάθεση μιας νέας έκδοσης του IDE κάθε χρόνο. Κάθε νέα έκδοση περιλαμβάνει την πλατφόρμα του Eclipse και άλλα projects. Μέχρι στιγμής αυτή η ετήσια διάθεση της νέας έκδοσης γίνεται κάθε τέταρτη Τετάρτη του Ιουνίου. Η τελευταία έκδοση, μέχρι τώρα, είναι η 4.3. Τέλος, κάθε έκδοση που βγαίνει από το 2006 και μετά έχει και μια κωδική ονομασία π.χ η τελευταία έκδοση ονομάζεται Kepler την οποία θα διαδεχθεί η Luna(4.4) τον Ιούνιο του 2014.

# Η βιβλιοθήκη χειρισμού αρχείων dBase(.dbf)

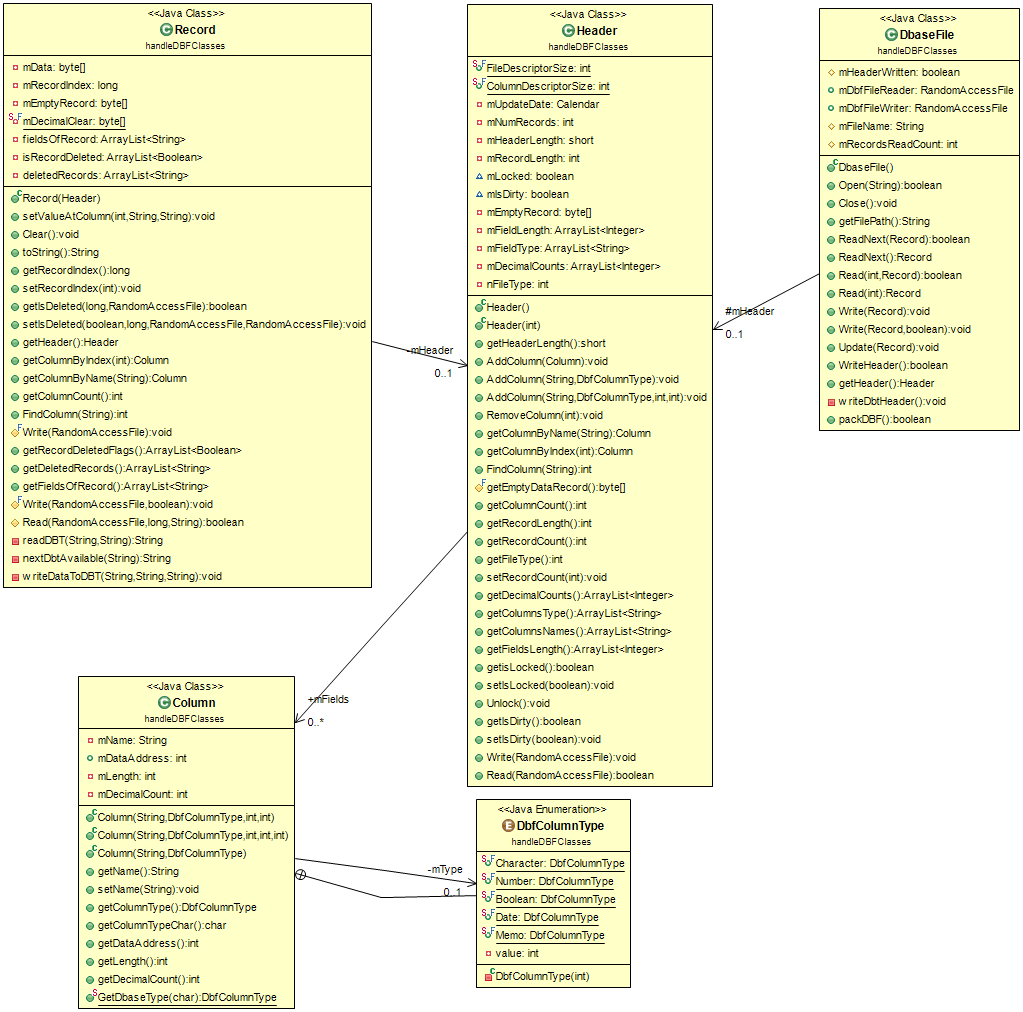
## Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται η βιβλιοθήκη χειρισμού αρχείων dBase(.dbf ). Εξετάζεται η δομή της βιβλιοθήκης και αναλύονται κάποιες σημαντικές μέθοδοι της. Επίσης στο τέλος του κεφαλαίου υπάρχει το API της βιβλιοθήκης που περιλαμβάνει όλες τις μεθόδους της.

Η βιβλιοθήκη η οποία χειρίζεται τα αρχεία dBase(.dbf), αποτελείται από τέσσερις κλάσεις:

* DbaseFile
* Header
* Column
* Record

Στην εικόνα 7 παρουσιάζεται το διάγραμμα των κλάσεων και οι σχέσεις μεταξύ τους.



Εικόνα 7: Διάγραμμα κλάσεων

Η ανάπτυξη της βιβλιοθήκης στηρίχτηκε σε μια βιβλιοθήκη ανοιχτού κώδικα την *Fast DBF,* η οποία είναι γραμμένη στη γλώσσα C-sharp(C#). Μετά την μελέτη της παραπάνω βιβλιοθήκης, μετέτρεψα κομμάτια του κώδικά της σε κώδικα Java και την προσάρμοσα στις δικές μου απαιτήσεις. Επίσης, πρόσθεσα και κάποιες λειτουργίες όπως η εγγραφή/ανάγνωση αρχείων MEMO (.dbt), η λειτουργία Pack και άλλες οι οποίες δεν υποστηρίζονταν από την βιβλιοθήκη.

## Κλάση DbaseFile

Η κλάση DbaseFile, αναπαριστά ένα αρχείο dBase (.dbf), και εκτελεί διάφορες λειτουργίες όπως δημιουργία, άνοιγμα, pack του αρχείου κα.

Η κύρια μέθοδος της κλάσης είναι η Open(), η οποία σαν παράμετρο παίρνει τη διαδρομή του αρχείου. Αν το αρχείο σ’ αυτή τη διαδρομή υπάρχει το ανοίγει και διαβάζει την κεφαλίδα (header) του, εάν δεν υπάρχει δημιουργεί ένα νέο αρχείο καθώς και μια νέα κεφαλίδα για το αρχείο αυτό. Επίσης δημιουργείται και ένα αρχείο Memo (.dbt), και μέσω της μεθόδου γράφω την κεφαλίδα του. Στην κεφαλίδα του Memo αρχείου γράφω τα τέσσερα πρώτα bytes τα οποία αναφέρονται στο επόμενο block που είναι διαθέσιμο για εγγραφή δεδομένων στο αρχείο. Επειδή το αρχείο είναι νέο ο αριθμός του πρώτου διαθέσιμου block είναι το ένα (1).

Για να διαβάσω και να γράψω το αρχείο χρησιμοποιώ το αντικείμενο RandomAccessFile της κλάσης java.io. Ένα RandomAccessFile συμπεριφέρεται σαν ένας μεγάλος πίνακας από bytes. Υπάρχει ένας cursor (file pointer), ο οποίος δείχνει τη θέση στο αρχείο από την οποία θα αρχίσει το διάβασμα ή το γράψιμο. O file pointer, μπορεί να διαβαστεί με τη μέθοδο getFilePointer() και μπορεί να μετακινηθεί με τη μέθοδο seek(). Ένα αρχείο αυτού του είδους μπορεί να δημιουργηθεί σε δύο modes, μόνο για διάβασμα(read) ή για διάβασμα και γράψιμο(read/write).

Η μέθοδος Read() παίρνει σαν παράμετρο τον αριθμό της εγγραφής που θέλω να διαβάσω και ένα αντικείμενο τύπου Record στο οποίο θα αποθηκευτεί η εγγραφή που θα διαβαστεί. Σε αυτή τη μέθοδο υπολογίζεται η θέση της εγγραφής μέσα στο αρχείο και γίνεται μετακίνηση του δείκτη (file pointer), στο σημείο αυτό. Ο υπολογισμός της θέσης γίνεται ως εξής: Μήκος της κεφαλίδας + Αριθμός της εγγραφής \* Το μήκος της εγγραφής. Στη συνέχεια καλείτε η Read() της κλάσης Record για διάβασμα της εγγραφής.

Η μέθοδος Write() παίρνει ως παράμετρο την εγγραφή που πρέπει να γραφτεί στο αρχείο και κάνει την προετοιμασία για την εγγραφή. Στην αρχή υπολογίζεται το πλήθος των εγγραφών στο αρχείο. Ο υπολογισμός γίνεται ως εξής: Το μήκος του αρχείου – Το μήκος της κεφαλίδας – 1(termination character) / το μήκος της κεφαλίδας. Το αποτέλεσμα του υπολογισμού γίνεται το record index της νέας εγγραφής. Στη συνέχεια καλείται η Update() που έχει ως όρισμα τη νέα εγγραφή. Στην μέθοδο Update() γίνεται ο υπολογισμός της θέσης στην οποία θα γραφτεί η νέα εγγραφή. Ο υπολογισμός γίνεται ως εξής: Το μήκος της κεφαλίδας + (Το record index της νέας εγγραφής \* Το μήκος της εγγραφής). Με τη μέθοδος seek() του RandomAccessFile αρχείου γίνεται η μετακίνηση του filePointer στο σημείο της εγγραφής, και στη συνέχεια καλείται η Write() της κλάσης Record για να γραφτεί η νέα εγγραφή στο αρχείο.

Με τη μέθοδο WriteHeader() ο filePointer του αρχείου μετακινείται στην αρχή (θέση 0), και καλείτε η μέθοδος Write() της κλάσης Header για να γραφτεί η κεφαλίδα στο αρχείο.

Τέλος, μια σημαντική μέθοδος της κλάσης είναι η PackDBF(), η οποία εκτελεί τη λειτουργία Pack στο αρχείο και διαγράφει όλες τις σημειωμένες για διαγραφή εγγραφές από το αρχείο. Η διαδικασία που ακολούθησα για να υλοποιήσω τη λειτουργία Pack είναι η εξής: Στην αρχή δημιουργώ ένα νέο αρχείο στο οποίο για να εξασφαλίσω τη μοναδικότητα του βάζω ως όνομα: το όνομα του αρχείου που θέλω να κάνω pack συν την τρέχουσα ημερομηνία σε milliseconds και ανοίγω το νέο αρχείο με ένα αντικείμενο τύπου RandomAccessFile. Στη συνέχεια από το ‘παλιό’ αρχείο διαβάζω μία-μία όλες τις εγγραφές και όποια εγγραφή δεν ξεκινάει με ‘\*’(σημειωμένη για διαγραφή), γράφεται στο ‘νέο’ αρχείο. Στο επόμενο βήμα προσπαθώ να διαγράψω το ‘παλιό’ αρχείο. Αν η διαγραφή αποτύχει σημαίνει ότι το αρχείο χρησιμοποιείται από κάποια άλλη εφαρμογή. Σε αυτή τη περίπτωση διαγράφω το ‘νέο’ αρχείο και επιστρέφω false (rollback). Αν η διαγραφή πετύχει στο header θέτω το νέο πλήθος εγγραφών, μετακινώ τον filePointer στην αρχή (θέση 0) και γράφω τη νέα κεφαλίδα. Τέλος, κάνω μετονομασία το ‘νέο’ αρχείο και του δίνω το όνομα του ‘παλιού’ αρχείου(που πλέον δεν υπάρχει) και επιστρέφω true.

## Κλάση Header

Η κλάση Header αναπαριστά την κεφαλίδα του αρχείου dBase(.dbf), και εκτελεί λειτουργίες όπως διάβασμα/γράψιμο της κεφαλίδας και παροχή διάφορων πληροφοριών για το περιεχόμενο της κεφαλίδας. Σημαντικά πεδία αυτής της κλάσης είναι τα ArrayLists mFields, mFieldLength, mFieldType και mDecimalCount, τα οποία περιέχουν τα πεδία, το μήκος των πεδίων, τον τύπο των πεδίων και το πλήθος των δεκαδικών που δέχονται τα πεδία αντίστοιχα.

Η μέθοδος AddColumn προσθέτει ένα νέο πεδίο στο αρχείο. Παίρνει σαν όρισμα το νέο πεδίο και το προσθέτει στη συλλογή mFields. Επίσης ενημερώνει το μήκος της εγγραφής (τρέχων μήκος + μήκος νέου πεδίου), καθώς επίσης και το μήκος της κεφαλίδας (τρέχων μήκος + 32). Το μήκος που προστίθεται είναι ο χώρος που καταλαμβάνει κάθε νέο πεδίο στην κεφαλίδα.

Με την μέθοδο RemoveColumn γίνεται διαγραφή ενός πεδίου από το αρχείο. Σαν παράμετρο παίρνει τη θέση του πεδίου στο αρχείο και διαγράφει ένα αντικείμενο από την συλλογή mFields. Επίσης ενημερώνει το μήκος της εγγραφής (τρέχων μήκος - μήκος νέου πεδίου), καθώς επίσης και το μήκος της κεφαλίδας (τρέχων μήκος - 32). Το μήκος που αφαιρείτε είναι ο χώρος που καταλαμβάνει κάθε πεδίο στην κεφαλίδα.

Η μέθοδος Write παίρνει σαν όρισμα το αντικείμενο RandomAccessFile και γράφει την κεφαλίδα στο αρχείο. Στην αρχή γίνεται έλεγχος για την ύπαρξη Memo πεδίου. Αν υπάρχει, ο τύπος του dBase αρχείου δηλώνεται ως 0x83 ενώ αν δεν υπάρχει ως 0x03. Μετά γράφονται στο αρχείο κατά σειρά, η ημερομηνία τελευταίας τροποποίησης, ο αριθμός των εγγραφών, το μήκος της κεφαλίδας, το μήκος της εγγραφής και τα reserved bytes. Στη συνέχεια γράφονται οι πληροφορίες για τα πεδία, όπως όνομα, τύπος, πλήθος δεκαδικών, offset από την αρχή της εγγραφής και το μήκος του πεδίου. Τέλος, γράφει τα reserved bytes και το byte που χαρακτηρίζει το τέλος της κεφαλίδας(0x0D). Για να γραφτούν τα δεδομένα στο αρχείο οι μέθοδοι που χρησιμοποιούνται από την κλάση RandomAccessFile είναι οι εξής: write() (γράφει ένα byte), writeInt() (γράφει 4 bytes), writeSort() (γράφει 2 bytes).

Τέλος, η μέθοδος Read() διαβάζει την κεφαλίδα του αρχείου και αποθηκεύει τις πληροφορίες σε μεταβλητές. Η μέθοδος αυτή παίρνει ως παράμετρο το αντικείμενο τύπου RandomAccessFile, που χρησιμοποιείτε για την ανάγνωση του αρχείου. Στην αρχή διαβάζεται το πρώτο byte που περιέχει τον τύπο του αρχείου και το αποθηκεύει σε μια μεταβλητή τύπου integer. Αν η τιμή της μεταβλητής δεν είναι 3 ή 131 (FoxBASE+/FoxPro/dBase III plus με ή χωρίς MEMO), τότε η μέθοδος επιστρέφει false, επειδή αυτοί είναι οι μόνοι τύποι που υποστηρίζονται. Στη συνέχεια διαβάζονται τα bytes που περιέχουν τις πληροφορίες για την τελευταία τροποποίηση του αρχείου, τον αριθμό των εγγραφών, το μήκος της κεφαλίδας, το μήκος της εγγραφής, τον αριθμό των πεδίων, τα ονόματα, τους τύπους και τα πλήθη των δεκαδικών των πεδίων. Για κάθε πεδίο που διαβάζει δημιουργεί ένα νέο αντικείμενο τύπου Column και το βάζει στη συλλογή mFields. Επίσης, γεμίζει με πληροφορίες και τις συλλογές mFieldLength, mDecimalsCount, mFieldType με τα μήκη, τα πλήθη των δεκαδικών και τους τύπους των πεδίων αντίστοιχα. Για να γίνει ανάγνωση του αρχείου χρησιμοποιήθηκαν οι εξής μέθοδοι της κλάσης RandomAccessFile: read() (διαβάζει ένα byte), readInt() (διαβάζει 4 bytes), readShort() (διαβάζει 2 bytes) και η skipBytes() η οποία προσπερνάει έναν αριθμό από bytes (τα reserved).

Κατά την εγγραφή και ανάγνωση της κεφαλίδας του αρχείου στα σημεία της ανάγνωσης/εγγραφής του συνόλου των εγγραφών, του μήκους της κεφαλίδας, του μήκους της εγγραφής και του μήκους του πεδίου, πρέπει να γίνει μετατροπή της τιμής από big σε little endian ( big endian – σημαντικότερο byte στην πρώτη θέση, liitle endian – σημαντικότερο byte στην τελευταία θέση).

## Κλάση Column

Η κλάση Column, αναπαριστά ένα πεδίο και έχει τις απαραίτητες λειτουργίες για τον χειρισμό ενός πεδίου όπως δημιουργία, ορισμός ονόματος, ανάκτηση πληροφοριών (όνομα, τύπος, μήκος και πλήθος δεκαδικών).

Η κλάση περιέχει έναν enumerate τύπο ο οποίος καλύπτει όλους τους τύπους που μπορεί να έχει ένα πεδίο. Η μέθοδος getColumnType() επιστρέφει ένα enumerate που δηλώνει τον τύπο του πεδίου.

Η μέθοδος DbfColumnType() σαν παράμετρο παίρνει ένα χαρακτήρα και επιστρέφει τον enumerate τύπο που αντιστοιχεί σ’ αυτόν τον χαρακτήρα. Αντίθετα η μέθοδος getColumnTypeChar(), επιστρέφει ένα χαρακτήρα ο οποίος αντιστοιχεί στον enumerate τύπο του πεδίου.

## Κλάση Record

Η κλάση Record αναπαριστά μια εγγραφή του αρχείου dBase, και αναλαμβάνει τη δημιουργία, την εγγραφή και την ανάγνωσή της. Επίσης γράφει και διαβάζει αρχεία Memo(.dbt). Σημαντικές μεταβλητές μέσα σ’ αυτή τη κλάση είναι οι συλλογές (ArrayLists) fieldsOfRecord, deletedRecord και isRecordDeleted οι οποίες αποθηκεύουν τις μη διαγραμμένες εγγραφές, τις διαγραμμένες εγγραφές και τις καταστάσεις όλων των εγγραφών (διαγραμμένες – μη διαγραμμένες) και η μεταβλητή mData στην οποία αποθηκεύεται μια εγγραφή.

Η μέθοδος setValueAtColumn() είναι αυτή η οποία βάζει μια τιμή σε ένα πεδίο. Ως ορίσματα παίρνει τη θέση του πεδίου στο οποίο θα μπει η τιμή, την τιμή και τη διαδρομή του αρχείου στο οποίο θα αποθηκευτεί. Ανάλογα με τον τύπο του πεδίου η τιμή μπαίνει στη μεταβλητή mData. Αν η τιμή είναι κενή στο πεδίο μπαίνει μια άδεια εγγραφή. Σε όλους τους τύπους των πεδίων γίνεται έλεγχος για το μήκος της τιμής και αν δεν υπερβαίνει το μήκος του πεδίου, με την μέθοδο System.arraycopy() η τιμή μπαίνει στην μεταβλητή mData, αφού πρώτα μετατραπεί σε byte array. Επιπρόσθετα αν ο τύπος του πεδίου είναι ‘Numeric’, γίνονται επιπλέον έλεγχοι που αφορούν το μήκος του αριθμού και το πλήθος των δεκαδικών. Επίσης αν ο τύπος του πεδίου είναι ‘Date’ και η τιμή για εισαγωγή δεν μπορεί να μετατραπεί σε ημερομηνία εμφανίζεται Exception. Τέλος, αν το πεδίο είναι τύπου Memo, καλείται η μέθοδος nextDbtAvailable() η οποία επιστρέφει το επόμενο διαθέσιμο block στο .dbt αρχείο. Η τιμή αυτή αποθηκεύεται στο πεδίο (αρχείο .dbf), ενώ η τιμή που ήρθε ως παράμετρος στη μέθοδο γράφεται μέσω της writeDataToDBT() στο αρχείο MEMO(.dbt).

Η μέθοδος getIsDeleted() παίρνει ως ορίσματα τη θέση μιας εγγραφής και το stream που χρησιμοποιείτε για ανάγνωση. Μετακινεί τον filePointer του αρχείου στην θέση της εγγραφής, που δηλώνεται από την παράμετρο της μεθόδου και διαβάζει την εγγραφή. Αν το πρώτο byte της εγγραφής είναι ‘\*’ (διαγραμμένη) επιστρέφει true.

Με την μέθοδο setIsDeleted(), θέτω μια εγγραφή ως διαγραμμένη ή μη διαγραμμένη. Ως ορίσματα παίρνει μια τιμή που δηλώνει αν θέλω να διαγράψω την εγγραφή (true) ή αν θέλω να την επαναφέρω (false), την θέση της εγγραφής στο αρχείο, το stream που χρησιμοποιώ για ανάγνωση και το stream που χρησιμοποιώ για εγγραφή. Στη συνέχεια ο filePointer του αρχείου μετακινείται στη θέση της εγγραφής που υποδηλώνει η παράμετρος της μεθόδου και διαβάζω την εγγραφή μέσα στη μεταβλητή mData. Ανάλογα με το αν η εγγραφή πρέπει να διαγραφεί ή όχι το πρώτο byte στο mData array γίνεται ‘\*’ ή ‘ ‘, αντίστοιχα. Τέλος καλείτε η Write() η οποία γράφει την τροποποιημένη εγγραφή στο αρχείο.

Η μέθοδος Write παίρνει ως όρισμα το stream εγγραφής και καλώντας τη μέθοδο write() της κλάσης RandomAccessFile με παράμετρο τον mData array, γράφει μια εγγραφή στο αρχείο.

Μέσω της μεθόδου Read() γίνεται ανάγνωση μιας εγγραφής από το αρχείο. Ως ορίσματα παίρνει το stream που χρησιμοποιείται για την ανάγνωση, την θέση της εγγραφής που πρέπει να διαβαστεί στο αρχείο και τη διαδρομή του αρχείου στο οποίο ανήκει η εγγραφή. Στην αρχή διαβάζω το πρώτο byte και ελέγχω αν η εγγραφή είναι σημειωμένη για διαγραφή ή όχι. Στη συνέχεια, διαβάζω τα υπόλοιπα bytes του πρώτου πεδίου και ακολούθως διαβάζω όλη την εγγραφή πεδίο-πεδίο. Αν ένα πεδίο είναι τύπου MEMO, για να πάρω τα δεδομένα του καλώ την readDBT(). Τέλος, αν το πρώτο byte της εγγραφής είναι ‘\*’ (σημειωμένη για διαγραφή), τα δεδομένα μπαίνουν στη συλλογή deletedRecords και στη συλλογή isRecordDeleted μπαίνει η τιμή false, ενώ στην αντίθετη περίπτωση τα δεδομένα μπαίνουν στη συλλογή filedsOfRecord και στη συλλογή isRecordDeleted μπαίνει η τιμή true.

Με τη μέθοδο readDBT() γίνεται ανάγνωση από το αρχείο MEMO (.dbt). Σαν παράμετρο παίρνει τον αριθμό του block από το οποίο θα ξεκινήσει η ανάγνωση, καθώς και τη διαδρομή του αρχείου(.dbf). Μέσα στη μέθοδο γίνεται αλλαγή της κατάληξης (.dbf) σε (.dbt). Αν η τιμή που περιέχει τον αριθμό του block είναι κενή επιστρέφεται κενό String. Ανοίγω το MEMO αρχείο μέσω του αντικειμένου RandomAccessFile και μετακινώ τον filePointer στη θέση 0(αρχή). Στη συνέχεια βάση του αριθμού του block μετακινώ τον filePointer στην θέση για να αρχίσει η ανάγνωση. Η θέση αυτή υπολογίζεται ως εξής: Αριθμός του block προς ανάγνωση \* 512. Ένα πεδίο μπορεί να έχει δεδομένα σε παραπάνω από ένα blocks, γι’ αυτό το λόγο έχω μια συλλογή (ArrayList) από blocks μέσα στην οποία αποθηκεύω κάθε block που διάβασα. Ακόμη, έχω μια μεταβλητή ‘k’ η οποία δηλώνει πόσα bytes από το τελευταίο block είναι χρήσιμα. Διαβάζω 512 bytes (1 block) και τα αποθηκεύω στη μεταβλητή data. Επίσης μετακινώ το δείκτη κατά 512 bytes. Στη συνέχεια αρχίζω να διαβάζω τα δεδομένα που διάβασα αναζητώντας το τέλος των δεδομένων. Το τέλος των δεδομένων σηματοδοτείτε από δύο συνεχόμενα bytes της τιμής ‘0x1A’. Αν βρω το byte με την τιμή ‘0x1A’ στην πρώτη θέση του πίνακα από τη συλλογή blocks, παίρνω το τελευταίο block και διαβάζω το τελευταίο byte. Αν το τελευταίο byte έχει την τιμή ‘0x1A’, σημαίνει ότι έφτασα στο τέλος των δεδομένων, άρα το block που διάβασα δεν το βάζω στη συλλογή καθώς περιέχει μόνο ένα χαρακτήρα τερματισμού (‘0x1A’). Η μεταβλητή ‘k’ παίρνει τη τιμή 511 καθώς από το τελευταίο block στη συλλογή ο τελευταίος χαρακτήρας είναι ένας χαρακτήρας τερματισμού. Αν βρω το byte με την τιμή ‘0x1A’ σε κάποια άλλη θέση του πίνακα ελέγχω την τιμή της προηγούμενης και αν είναι ‘0x1A’, τότε αν η θέση που βρήκα το ‘0x1A’ ήταν η 1 το block δεν το βάζω καθόλου στη συλλογή, αφού περιέχει μόνο δύο χαρακτήρες τερματισμού. Η μεταβλητή ‘k’ παίρνει τη τιμή 512 καθώς το τελευταίο block στη συλλογή θα διαβαστεί ολόκληρο. Αν η θέση που βρέθηκε το ‘0x1A’ δεν ήταν η 1 τότε η μεταβλητή ‘k’ παίρνει την τιμή της θέσης που βρέθηκε το ‘0x1A’ –1. Τέλος, δημιουργώ ένα πίνακα με μέγεθος όσο το πλήθος των (blocks – 1)\*512 + k και με την System.arraycopy αντιγράφω τα ν-1 blocks στον πίνακα. Από το τελευταίο block παίρνω μόνο τόσα bytes όσα δηλώνει η μεταβλητή ‘k’ και επιστρέφω τα δεδομένα.

Η μέθοδος nextDbtAvailable(), διαβάζει το header (πρώτα 512 byte), του MEMO αρχείου και επιστρέφει τον αριθμό του επόμενου διαθέσιμου block, το οποίο μπορεί να χρησιμοποιηθεί για εγγραφή δεδομένων. Ως παράμετρο παίρνει την διαδρομή του αρχείου, διαβάζει τα πρώτα 4 bytes, μετατρέπει τον αριθμό από big σε little endian και επιστρέφει την τιμή. Τα υπόλοιπα 508 bytes, τα προσπερνάει καθώς δεν περιέχουν κάποια πληροφορία.

Τέλος μια σημαντική μέθοδος στην κλάση Record είναι η writeDataToDBT(), η οποία παίρνει σαν παράμετρο, τη διαδρομή του αρχείου, τον αριθμό του πρώτου διαθέσιμου block για εγγραφή και την τιμή που πρέπει για γραφτεί στο αρχείο. Η μέθοδος στην αρχή ανοίγει το αρχείο MEMO με ένα αντικείμενο τύπου RandomAccessFile για διάβασμα/εγγραφή και μετακινεί τον filePointer στην αρχή του stream. Η τιμή που είναι για εγγραφή μετατρέπεται σε έναν byte array και βρίσκω το συνολικό μήκος που θα καταλαμβάνει τελικά η τιμή στο αρχείο. Ο υπολογισμός γίνεται ως εξής: Προσθέτω στο μήκος της τιμής δύο μονάδες (στο τέλος μπαίνουν δύο χαρακτήρες τερματισμού(‘0x1A’)) και το νέο μήκος το στρογγυλοποιώ κατά πάνω έτσι ώστε να είναι πολλαπλάσιο του 512. Στη συνέχεια διαιρώντας το νέο μήκος με το 512 βρίσκω τον αριθμό των blocks που τελικά θα καταλάβει η τιμή στο αρχείο. Προσθέτω αυτή την τιμή στον αριθμό του επόμενου διαθέσιμου block που ήρθε σαν παράμετρος στη μέθοδο και έχω τον νέο αριθμό επόμενου διαθέσιμου block, τον οποίο γράφω στην κεφαλίδα του αρχείου. Τέλος, μετακινώ το δείκτη (filePointer) , μέσω της μεθόδου seek(), στο σημείο που δηλώνει ο αριθμός του επόμενου διαθέσιμου block που ήρθε ως παράμετρος και γράφω την τιμή στο αρχείο. Ο υπολογισμός της θέσης στο αρχείο γίνεται ως εξής: Αριθμός επόμενου διαθέσιμου block \* 512(μήκος κάθε block).

Αυτή ήταν η περιγραφή των σημαντικότερων μεθόδων στων κλάσεων DbaseFile, Header, Column, Record. Πληροφορίες για όλες τις μεθόδους περιλαμβάνονται στο API της βιβλιοθήκης που ακολουθεί. Παραδείγματα για τη χρήση των μεθόδων, σε μορφή κώδικα, υπάρχουν στο Παράρτημα.

## Παρουσίαση του API της βιβλιοθήκης για τον χειρισμό DBF αρχείων

### Κλάση DBaseFile.

Αναπαριστά ένα αρχείο DBF και αναλαμβάνει όλες τις λειτουργίες χειρισμού του (άνοιγμα, κλείσιμο, Pack κτλ).

public boolean **Open**(String path)

Ανοίγει ένα αρχείο DBF ή δημιουργεί ένα καινούργιο.

**Παράμετροι**

*path* - η διαδρομή για το αρχείο. Αν δεν υπάρχει δημιουργείται καινούργιο.

**Επιστρέφει**

*true* - αν το αρχείο διαβάστηκε-δημιουργήθηκε χωρίς πρόβλημα.

public void **Close**()

Κλείνει τα streams του αρχείου. Αν ο header δεν έχει γραφτεί τον γράφει.

**Throws**

*Exception*

public String **getFilePath**()

Επιστρέφει τη διαδρομή του αρχείου.(Για το αρχείο που έχει ανοιχτεί μέσω της *Open*()).

**Επιστρέφει**

Τη διαδρομή του αρχείου.

public boolean **ReadNext**(Record oFillRecord)

Διαβάζει την επόμενη εγγραφή και βάζει τα δεδομένα στη μεταβλητή *oFillRecord*.

**Παράμετροι**

*oFillRecord* - Μεταβλητή τύπου *Record* για αποθήκευση της εγγραφής.

**Επιστρέφει**

*true* - αν η εγγραφή διαβάστηκε.

public Record **ReadNext**()

Διαβάζει μια εγγραφή.

**Επιστρέφει**

Ένα αντικείμενο *Record* αν η εγγραφή διαβάστηκε ή null αν δεν διαβάστηκε τίποτα.

**Throws**

*IOException*

public boolean **Read**(int index, Record oFillRecord)

Διαβάζει την εγγραφή που καθορίζει η μεταβλητή *index* και τη βάζει στη μεταβλητή *oFillRecord*.

**Παράμετροι**

*index* - ο αριθμός της εγγραφής προς ανάγνωση

*oFillRecord* - Μεταβλητή τύπου *Record* για αποθήκευση της εγγραφής.

**Επιστρέφει**

*true* - αν η εγγραφή διαβάστηκε

**Throws**

*Exception*

public Record **Read**(int index)

Διαβάζει την εγγραφή που καθορίζει η μεταβλητή *index*.

**Παράμετροι**

*index* - ο αριθμός της εγγραφής προς ανάγνωση

**Επιστρέφει**

t*rue -* αν η εγγραφή διαβάστηκε

**Throws**

*Exception*

public void **Write**(Record orec)

Γράφει μια εγγραφή στο αρχείο.

**Παράμετροι**

*orec* - η εγγραφή που πρέπει να μπει στο αρχείο.

**Throws**

*Exception*

public void **Write**(Record orec, Boolean bClearRecordAfterWrite)

Γράφει μια εγγραφή στο αρχείο και δίνει τη δυνατότητα να ‘καθαριστεί’ το αντικείμενο.

**Παράμετροι**

*orec* - η εγγραφή που πρέπει να μπει στο αρχείο.

*bClearRecordAfterWrite -* δηλώνει αν το αντικείμενο θα καθαριστεί.

public boolean **WriteHeader**()

Γράφει τον header στο αρχείο

**Επιστρέφει**

*true* - αν ο header γράφτηκε

**Throws**

*Exception*

public Header **getHeader**()

Επιστρέφει τον header του αρχείου

**Επιστρέφει**

Ένα αντικείμενο τύπου *Header*.

public boolean **packDBF**()

Μέθοδος που κάνει pack ένα αρχείο DBF. **ΠΡΟΣΟΧΗ**: Όλες οι εγγραφές που είναι σημειωμένες για διαγραφή διαγράφονται από το αρχείο.

***Throws***

*Exception*

### Κλάση Header

Αναπαριστά την κεφαλίδα του αρχείου DBF και παρέχει διάφορες πληροφορίες όπως το πλήθος των πεδίων, το πλήθος των εγγραφών, οι τύποι των πεδίων, οι τύπο των πεδίων κτλ.

public **Header**()

Δομητής της κλάσης *Header*. Δημιουργείτε ένα αντικείμενο τύπου *Header*.

public **Header**(int nFieldCapacity)

Δομητής της κλάσης *Header*. Δημιουργείτε ένα αντικείμενο τύπου *Header*.

**Παράμετροι**

*nFieldCapacity* - δηλώνει το μέγιστο πλήθος των πεδίων που θα αποθηκευτούν στο header.

public short **getHeaderLength**()

**Επιστρέφει**

Το μήκος της κεφαλίδας.

public void **AddColumn**(Column oNewCol)

Προσθέτει μια νέα στήλη στο αρχείο.

**Παράμετροι**

*oNewCol* - η στήλη που θα προστεθεί στο αρχείο

public void **AddColumn**(String sName, Column.DbfColumnType type)

Προσθέτει μια νέα στήλη στο αρχείο.

**Παράμετροι**

*sName -* το όνομα του νέου πεδίου

*type* - o τύπος του νέου πεδίου

public void **AddColumn**(String sName, Column.DbfColumnType type, int nLength, int nDecimals)

Προσθέτει μια νέα στήλη στο αρχείο.

**Παράμετροι**

*sName* - το όνομα του νέου πεδίου

*type* - o τύπος του νέου πεδίου

*nLength* - το μήκος του νέου πεδίο

*nDecimals* - το πλήθος των δεκαδικών του πεδίου

public void **RemoveColumn**(int nIndex)

Διαγράφει ένα πεδίο

**Παράμετροι**

*nIndex* - η θέση του πεδίου προς διαγραφή.

public Column **getColumnByName**(String sName)

Επιστρέφει ένα πεδίο βάση του ονόματός του.

**Παράμετροι**

*sName - τ*o όνομα του πεδίου προς επιστροφή.

**Επιστρέφει**

Ένα αντικείμενο *Column* αν βρέθηκε το πεδίο ή null.

public Column **getColumnByIndex**(String nIndex)

Επιστρέφει ένα πεδίο βάση της θέσης του.

**Παράμετροι**

*nIndex* - η θέση του πεδίου.

**Επιστρέφει**

Ένα αντικείμενο *Column* αν βρέθηκε το πεδίο ή null.

public int **FindColumn**(String sName)

Αναζητά ένα πεδίο βάση του ονόματος του.

**Παράμετροι**

*sName* - το όνομα του πεδίου

**Επιστρέφει**

Τη θέση του πεδίου αν υπάρχει ή -1.

public int **getColumnCount**()

**Επιστρέφει**

Το πλήθος των πεδίων.

public int **getRecordLength**()

**Επιστρέφει**

Το μήκος της εγγραφής

public int **getRecordCount**()

**Επιστρέφει**

Το πλήθος των εγγραφών

public int **getFileType**()

**Επιστρέφει**

Τον τύπο του αρχείου DBF

public void **setRecordCount**(int value)

Θέτει το πλήθος των εγγραφών

**Παράμετροι**

*value* - το νέο πλήθος των εγγραφών

public ArrayList<Integer> **getDecimalsCounts**()

**Επιστρέφει**

Ένα *ArrayList* που περιέχει τα πλήθη των δεκαδικών για κάθε πεδίο.

public ArrayList<String> **getColumnsType**()

**Επιστρέφει**

Ένα *ArrayList* που περιέχει τον τύπο κάθε πεδίου.

public ArrayList<String> **getColumnsNames**()

**Επιστρέφει**

Ένα *ArrayList* που περιέχει τα ονόματα όλων των πεδίων.

public ArrayList<Integer> **getFieldsLength**()

**Επιστρέφει**

Ένα *ArrayList* που περιέχει τα μήκη όλων των πεδίων.

public Boolean **getisLocked**()

**Επιστρέφει**

*true* - αν ο header δεν δέχεται τροποποίηση.

public void **setIsLocked**(Boolean value)

Θέτει τον header ως κλειδωμένο ή ξεκλείδωτο

**Παράμετροι**

*value -* ένα flag που ορίζει τον header ως κλειδωμένο ή ξεκλείδωτο.

public void **Unlock**()

Θέτει τον header ως ξεκλείδωτο (δέχεται αλλαγές).

public Boolean **getIsDirty**()

**Επιστρέφει**

*true* - αν ο header έχει τροποποιηθεί και δεν έχει γραφτεί στο αρχείο.

public void Write(RandomAccessFile writer)

Γράφει το header στο αρχείο DBF.

**Παράμετροι**

Το stream της εγγραφής.

**Throws**

*Exception*

public Boolean **Read**(RandomAccessFile reader)

Διαβάζει το header από το DBF αρχείο.

**Παράμετροι**

Το stream της ανάγνωσης.

**Throws**

*IOException*

### Κλάση Column

Αναπαριστά ένα πεδίο του αρχείου DBF και δίνει τη δυνατότητα δημιουργίας, μετονομασίας ενός πεδίου. Επίσης παρέχει πληροφορίες για τον τύπο του, για το μήκος του, για το πλήθος των δεκαδικών που δέχεται κα.

public **Column**(String sName, DbfColumnType type, in nLength, int nDecimals)

Δομητής της κλάσης *Column*. Δημιουργεί ένα νέο αντικείμενο τύπου *Column*.

**Παράμετροι**

*sName* - το όνομα του νέου πεδίου.

*type* - ο τύπος του νέου πεδίου.

*nLength -* το μήκος του νέου πεδίου.

*nDecimals -* το πλήθος των δεκαδικών του νέου πεδίου.

public **Column**(String sName, DbfColumnType type)

Δομητής της κλάσης *Column*. Δημιουργεί ένα νέο αντικείμενο τύπου *Column*. **ΠΡΟΣΟΧΗ**: Χρήση μόνο για πεδία τύπου *Logical, Date, Memo*

**Παράμετροι**

*sName* - το όνομα του νέου πεδίου.

*type* - ο τύπος του νέου πεδίου.

public String **getName**()

**Επιστρέφει**

Το όνομα του πεδίου.

public void **setName**(String value)

Θέτει το όνομα ενός πεδίου.

**Παράμετροι**

*value -* το όνομα του πεδίου. Δεν μπορεί να είναι κενό ή πάνω από 11 χαρακτήρες.

public DbfColumnType **getColumnType**()

**Επιστρέφει**

Τον τύπο του πεδίου.

public char **getColumnTypeChar**()

**Επιστρέφει**

Ένα χαρακτήρα ανάλογα με τον τύπο του πεδίου. (*Numeric*-N, *Character*-C, *Logical*-L, *Date*-D, *Memo*-M).

public int **getDataAddress**()

**Επιστρέφει**

Το offset του πεδίου από την αρχή της εγγραφής.

public int **getLength**()

**Επιστρέφει**

Το μήκος του πεδίου.

public int **getDecimalCount**()

**Επιστρέφει**

Το πλήθος των δεκαδικών του πεδίου

public static DbfColumnType **GetDbaseType**(char c)

**Επιστρέφει**

Ένα enumerate type με βάση τον χαρακτήρα που πήρε ως παράμετρο.

**Παράμετροι**

*c* - χαρακτήρας ο οποίος δηλώνει τον τύπο πεδίου. Δεκτές τιμές (C,L,M,D,N).

### Κλάση Record

Αναπαριστά μια εγγραφή του αρχείου DBF και δίνει τη δυνατότητα προσθήκης νέων εγγραφών, σημείωση μιας εγγραφής ως διαγραμμένη ή μη διαγραμμένη. Επίσης είναι η κλάση μέσα από την οποία διαβάζονται οι εγγραφές από το αρχείο.

public **Record**(Header oHeader)

Δομητής της κλάσης Record. Δημιουργείτε ένα νέο αντικείμενο τύπου *Record*.

**Παράμετροι**

*oHeader -* ένα αντικείμενο τύπου *Header*.

public void **setValueAtColumn** (int nColIndex, String value, String path)

Βάζει μια τιμή σε ένα πεδίο.

**Παράμετροι**

*nColIndex* - η θέση του πεδίου στο οποίο θα μπει η τιμή.

*value* - η τιμή η οποία θα μπει στο πεδίο.

*path* - διαδρομή του αρχείου στο οποίο θα γραφτεί η νέα τιμή.

**Throws**

IOException

public void **Clear**()

‘Καθαρίζει’ όλα τα δεδομένα από μία εγγραφή.

@**override**

public String **toString**()

**Επιστρέφει**

Τα δεδομένα μιας εγγραφής σε μορφή *String*.

public long **getRecordIndex**()

**Επιστρέφει**

Το record index της εγγραφής.

public void **seRecordIndex**(int value)

Θέτει το record index της εγγραφής.

**Παράμετροι**

*value* - το record index της εγγραφής.

public Boolean **getIsDeleted**(long recordIndex, RandomAccessFile reader)

**Επιστρέφει**

*true -* αν η εγγραφή είναι σημειωμένη για διαγραφή.

**Παράμετροι**

*recordIndex -* η θέση της εγγραφής την οποία θέλουμε να ελέγξουμε. Δεν μπορεί να είναι < 0 ή > από το συνολικό πλήθος των εγγραφών στο αρχείο.

*reader -* το stream της ανάγνωσης.

public void **settIsDeleted**(Boolean value, long recordIndex, RandomAccessFile reader, RandomAccessFile writer)

Σημειώνει μια εγγραφή ως διαγραμμένη ή μη διαγραμμένη. **ΠΡΟΣΟΧΗ**: Οι εγγραφές που σημειώνονται ως διαγραμμένες δεν διαγράφονται από το αρχείο. Για να γίνει αυτό κάλεσε την *Pack*() της κλάσης *DbaseFile*.

**Παράμετροι**

*value* ένα flag που δηλώνει πως θα σημειωθεί η εγγραφή.

*recordIndex* η θέση της εγγραφής την οποία θέλουμε να σημειώσουμε. Δεν μπορεί να είναι < 0 ή > από το συνολικό πλήθος των εγγραφών στο αρχείο.

*reader* - το stream της ανάγνωσης.

*writer* - το stream της εγγραφής.

public **Header** getHeader()

**Επιστρέφει**

Ένα αντικείμενο τύπου *Header*.

public Column **getColumnByName**(String sName)

Επιστρέφει ένα πεδίο βάση του ονόματός του.

**Παράμετροι**

*sName* - τo όνομα του πεδίου προς επιστροφή.

**Επιστρέφει**

Ένα αντικείμενο *Column* αν βρέθηκε το πεδίο ή null.

public Column **getColumnByIndex**(String nIndex)

Επιστρέφει ένα πεδίο βάση της θέσης του.

**Παράμετροι**

*nIndex* - η θέση του πεδίου.

**Επιστρέφει**

Ένα αντικείμενο *Column* αν βρέθηκε το πεδίο ή null.

public int **FindColumn**(String sName)

Αναζητά ένα πεδίο βάση του ονόματος του.

**Παράμετροι**

*sName -* το όνομα του πεδίου

**Επιστρέφει**

Τη θέση του πεδίου αν υπάρχει ή -1.

public int **getColumnCount**()

**Επιστρέφει**

Το πλήθος των πεδίων.

public ArrayList<Boolean> **getRecordDeletedFlags**()

**Επιστρέφει**

Ένα *ArrayList* που περιέχει flags για όλες τις εγγραφές ανάλογα με το αν έχουν σημειωθεί για διαγραφή ή όχι.

public ArrayList<String> **getDeletedRecord**()

**Επιστρέφει**

Ένα *ArrayList* που περιέχει ΜΟΝΟ τις εγγραφές οι οποίες είναι σημειωμένες για διαγραφή.

public ArrayList<String> **getFieldsOfRecord**()

**Επιστρέφει**

Ένα *ArrayList* που περιέχει ΜΟΝΟ τις εγγραφές οι οποίες ΔΕΝ είναι σημειωμένες για διαγραφή.

# Η εφαρμογή DBF Editor

Παρακάτω παρουσιάζεται η εφαρμογή DBF Editor που δίνει τη δυνατότητα χειρισμού των αρχείων dBase μέσα από ένα γραφικό περιβάλλον. Αναφέρονται περιληπτικά όλες οι κλάσεις της εφαρμογής και στο τέλος υπάρχει ένας αναλυτικός οδηγός χρήσης του λογισμικού.

Η εφαρμογή DBF Editor, χρησιμοποιεί την βιβλιοθήκη που χειρίζεται τα αρχεία dBase και προσφέρει ένα τρόπο χειρισμού αυτών των αρχείων μέσα από ένα γραφικό περιβάλλον. Το οπτικό περιβάλλον της εφαρμογής αναπτύχθηκε με την τεχνολογία Java Swing με τη βοήθεια του Eclipse IDE. Η εφαρμογή αποτελείται από τις εξής 21 κλάσεις:

* AboutDialog
* AddDBF
* AddDialog
* AddField
* AdvancedFilterDialog
* ChangeStructureProgressDialog
* EditDialog
* EditField
* EditMenoFieldDialog
* FilteredDataDialog
* FindDialog
* ImportFromCsvProgressDialog
* ImportFromExcelProgressDialog
* ImportFromXMLProgressDialog
* MEMODialog
* ModifyStructureDialog
* OpenDbfFrame
* OpenFileProgressDialog
* RecordViewDialog
* ReplaceDialog
* StructureDialog

## Κλάσεις OpenDbfFrame και OpenFileProgressDialog

Η OpenDbfFrame είναι η κεντρική κλάση της εφαρμογής. Αποτελείται από ένα μενού (JMenu), μια γραμμή εργαλείων (JToolBar) και ένα JTable στο οποίο φορτώνονται τα δεδομένα του αρχείου. Η κλάση δίνει τη δυνατότητα να εξαχθούν τα δεδομένα σε MS Excel, csv, xml και html. Η εξαγωγή σε αρχεία MS Excel γίνεται με τη χρήση εξωτερικής βιβλιοθήκης (apache poi) η οποία χειρίζεται αρχεία MS Office μέσα από τη Java. Για να γεμίσει το JTable χρησιμοποιείται το DefaultTableModel το οποίο ως παράμετρο παίρνει ένα array με τα δεδομένα και ένα array με τα ονόματα των στηλών. Για την φόρτωση των δεδομένων καλείται η κλάση OpenFileProgressDialog η οποία διαβάζει τα δεδομένα από το αρχείο μέσω της βιβλιοθήκης χειρισμού dBase αρχείων και τα φορτώνει στο JTable. Η κλάση αυτή επίσης έχει και μια JProgressBar που δείχνει την πρόοδο της εργασίας. Η γραμμή εργαλείων περιλαμβάνει τις λειτουργίες οι οποίες χρησιμοποιούνται συνήθως. Στο JTextField ‘Go To Record’, μπαίνει ένας αριθμός και γίνεται απευθείας μετάβαση σε αυτήν την εγγραφή, ενώ στο JTextField ‘Quick Filter’ μπαίνει μια κανονική έκφραση και μένουν στο JTable μόνο όσες εγγραφές ταιριάζουν με αυτή. Επίσης, με την επιλογή του Table Info από το Edit menu, προσφέρονται πληροφορίες για το μέγεθος του αρχείου, το πλήθος των εγγραφών, την τελευταία τροποποίηση του αρχείου κα. Τέλος, στο JTable μέσω της ιδιότητας του setAutoCreateRowSorter(true), με το πάτημα σε κάποια κεφαλίδα (header) μιας στήλης γίνεται ταξινόμηση του JTable με βάση αυτό το πεδίο.

## Κλάσεις AddDialog/EditDialog

Μέσω αυτών των κλάσεων γίνεται προσθήκη και διαγραφή εγγραφών. Σχεδιαστικά είναι παρόμοιες καθώς αποτελούνται από JLabels και JTextFields τα οποία περιέχουν τα ονόματα των πεδίων και το περιεχόμενο αυτών. Κατά την εισαγωγή ή τροποποίηση μιας εγγραφής γίνεται έλεγχος των τιμών έτσι ώστε να ταιριάζουν με τους περιορισμούς των πεδίων (μήκος, πλήθος δεκαδικών, τύπος δεδομένων). Η αποθήκευση μιας τροποποιημένης ή μιας νέας εγγραφής γίνεται απευθείας στο αρχείο. Δηλαδή δεν υπάρχει κάποιο κουμπί Save στο μενού της εφαρμογής για αποθήκευση των αλλαγών στον πίνακα.

## Κλάσεις AddDBF,AddField,EditField

Οι κλάσεις αυτές χρησιμοποιούνται για τη δημιουργία ενός νέου αρχείου dBase. Η κλάση AddField δίνει τη δυνατότητα να προστεθεί ένα νέο πεδίο στο αρχείο. Περιέχει ένα JComboBox το οποίο περιέχει όλους τους διαθέσιμους τύπους για ένα πεδίο. Επίσης έχει JTextFields για την δήλωση του μήκους, του ονόματος και του πλήθους των δεκαδικών του πεδίου. Η δομή της κλάσης EditField είναι παρόμοια με αυτή της κλάσης AddField και δίνει τη δυνατότητα για τροποποίηση ενός πεδίου. Τέλος, η κλάση AddDBF έχει τη δυνατότητα αλλαγής της θέσης των πεδίων, διαγραφή διαφόρων πεδίων και τελικά τη δυνατότητα δημιουργίας του νέου αρχείου.

## Κλάσεις ImportFromCsvProgressDialog, ImportFromExcelProgressDialog, ImportFromXMLProgressDialog

Οι τρείς αυτές κλάσεις αναλαμβάνουν τη μαζική φόρτωση εγγραφών στο αρχείο από αρχεία τύπου MS Excel (97-2003 και 2007), csv και xml. Έχουν παρόμοια δομή και αποτελούνται από μία JProgressBar που δείχνει την πρόοδο της διαδικασίας. Προτού γίνει η εισαγωγή των εγγραφών στο Table γίνονται οι κατάλληλοι έλεγχοι οι οποίοι επιτρέπουν να μπουν στο αρχείο μόνο οι εγγραφές που ταιριάζουν με τους περιορισμούς των πεδίων (μήκος, πλήθος δεκαδικών, τύπος δεδομένων) του αρχείου. Για την ταυτόχρονη φόρτωση εγγραφών και την παρακολούθηση της εξέλιξης της διαδικασίας χρησιμοποιείται η κλάση SwingWorker (multi-threaded). Η κλάση SwingWorker σχεδιάστηκε για καταστάσεις όπου μια χρονοβόρα εργασία εκτελείται στο background και θέλουμε να έχουμε ενημέρωση σε ένα component του γραφικού περιβάλλοντος είτε κατά την διάρκεια αυτής της εργασίας είτε όταν αυτή τελειώσει. Η χρονοβόρα εργασία εκτελείτε μέσα στη μέθοδο doInBackground() παράλληλα με το main thread της εφαρμογής.

## Κλάσεις FindDIalog και ReplaceDialog

Η κλάση FindDialog κάνει σειριακή αναζήτηση με τη βοήθεια κανονικών εκφράσεων (regular expressions) βάση ενός λήμματος. Κατά την αναζήτηση σημειώνεται το πεδίο στο οποίο έχει ταιριάξει η κανονική έκφραση. Επίσης η κλάση μπορεί να κάνει και καταμέτρηση των συσχετίσεων που βρέθηκαν. Η υλοποίηση του παραθύρου γίνεται με ένα JDialog. Η κλάση ReplaceDialog είναι δομικά παρόμοια με την FindDialog και μέσω αυτής δίνεται η δυνατότητα σειριακής αναζήτησης και αντικατάστασης της τιμής ενός πεδίου. Κατά τη διάρκεια της αντικατάστασης ελέγχεται αν η τιμή που θα προκύψει μετά την αντικατάσταση τηρεί τους περιορισμούς του πεδίου στο οποίο ανήκει (μήκος, πλήθος δεκαδικών, τύπος δεδομένων) και τελικά η αντικατάσταση γίνεται μόνο αν δεν υπάρχει παραβίαση της νέας τιμής στη δομή του αρχείου. Τέλος, οι τιμές που προήλθαν από την αντικατάσταση αποθηκεύονται απευθείας στο αρχείο.

## Κλάσεις MEMODialog και EditMemoFieldDialog

Οι κλάσεις αυτές αποτελούνται από ένα JTextArea και δίνουν τη δυνατότητα παρουσίασης του κειμένου ενός Memo πεδίου και την τροποποίησή του αντίστοιχα. Η κλάση MEMODialog χρησιμοποιείται από την κλάση OpenDbfFrame όταν πατηθεί δεξί κλικ πάνω σε ένα πεδίο τύπου Memo για να παρουσιαστεί το περιεχόμενο του πεδίου. Η κλάση EditMemoFieldDialog χρησιμοποιείτε από τις κλάσεις AddDialog και EditDialog για την προσθήκη ή τροποποίηση του περιεχομένου Memo πεδίων.

## Κλάσεις AdvancedFilterDialog και FilteredDataDialog

Η κλάση AdvancedFilterDialog κάνει σύνθετη αναζήτηση, διαγραφή ή επαναφορά εγγραφών του αρχείου βάση μιας συνθήκης. Υλοποιείται με ένα JDialog και περιέχει διάφορα components όπως JList, JTable, JRadioButtons, JComboBoxes κα. Ανάλογα με την τιμή και τον τελεστή σύγκρισης που έχει επιλεχθεί από τον χρήστη, γίνεται η αναζήτηση εγγραφών που ταιριάζουν σε αυτά τα κριτήρια. Αν γίνεται αναζήτηση τα αποτελέσματα εμφανίζονται στο JTable του FilteredDataDialog και από εκεί υπάρχει η δυνατότητα εξαγωγής των αποτελεσμάτων σε αρχεία τύπου MS Excel, csv, xml ή html. Αν ο χρήστης έχει επιλέξει διαγραφή ή επαναφορά εγγραφών, δεν εμφανίζονται αποτελέσματα αλλά οι εγγραφές που ταιριάζουν με τα κριτήρια σημειώνονται ως διαγραμμένες ή επανέρχονται αντίστοιχα.

## Κλάση RecordViewDialog

Μέσω αυτής της κλάσης δίνεται η δυνατότητα να παρουσιάσουν οι εγγραφές μία-μία. Αποτελείτε από ένα JTable το οποίο περιέχει τα ονόματα των πεδίων και τα περιεχόμενά τους. Υπάρχει ένα JTooBar μέσω του οποίου γίνεται η πλοήγηση στην επόμενη, προηγούμενη, πρώτη, τελευταία εγγραφή του αρχείου. Τέλος, είναι δυνατόν η εγγραφή να σημειωθεί ως διαγραμμένη ή ως μη διαγραμμένη ή να τροποποιηθεί.

## Κλάσεις ModifyStructureDialog και ChangeStructureProgressDialog.

Με την κλάση ModifyStructureDialog αλλάζει η δομή του αρχείου. Μπορεί να διαγράψει, να εισάγει, να τροποποιήσει, να ανακατανέμει ένα πεδίο. Για να διατηρήσω τη συνέπεια στο αρχείο κατά την αλλαγή της δομής υπάρχουν οι εξής κανόνες:

* 1. Επιτρέπεται να γίνει μια μετονομασία (rename) κάθε φορά. Για να γίνει δεύτερη πρέπει πρώτα να αποθηκευτούν οι αλλαγές (να αλλάξει η δομή).
  2. Αν γίνει μετονομασία (rename) δεν επιτρέπεται να γίνει αλλαγή τύπου, μήκους, πλήθους δεκαδικών στο ίδιο πεδίο.
  3. Αν γίνει μετονομασία (rename), δεν επιτρέπεται να γίνει εισαγωγή (insert) ή διαγραφή (delete) πεδίων.
  4. Κατά την εισαγωγή (insert) ή διαγραφή (delete) πεδίων, επιτρέπεται να γίνει αλλαγή τύπου, μήκους, πλήθους δεκαδικών.
  5. Επιτρέπεται να γίνουν πολλές εισαγωγές (insert) ή διαγραφές (delete) πεδίων.
  6. Αν γίνει εισαγωγή (insert) δεν επιτρέπεται να γίνει διαγραφή (delete) και το αντίστροφο.
  7. Σε κάθε περίπτωση επιτρέπεται η αναδιάταξη των πεδίων.

Για να γίνει προσθήκη ή τροποποίηση ενός πεδίου, καλούνται οι κλάσεις AddField και EditField αντίστοιχα. Η εκτέλεση της αλλαγής της δομής γίνεται με την κλάση ChangeStructureProgressDialog η οποία προσφέρει ένα JProgressBar για την παρακολούθηση της εξέλιξης της διαδικασίας.

## Κλάσεις AboutDialog και StructureDialog

Τα δύο αυτά Dialogs δεν προσφέρουν λειτουργικότητα στην εφαρμογή. Το AboutDialog παρέχει κάποιες πληροφορίες για την εφαρμογή (έκδοση, δημιουργός κτλ), ενώ το StructureDialog παρέχει πληροφορίες για το αρχείο dBase, όπως έκδοση, πλήθος πεδίων, τύπος κάθε πεδίου, μήκος κάθε πεδίου κτλ.

Στην εφαρμογή ως εξωτερικές βιβλιοθήκες χρησιμοποίησα την Apache POI η οποία χειρίζεται τα αρχεία του MS Office και την Joda η οποία προσφέρει ευκολότερο έλεγχο των ημερομηνιών.

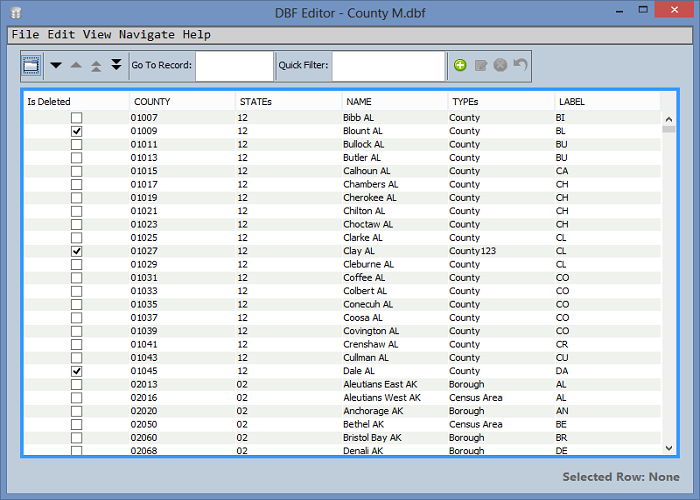
Αυτή ήταν η παρουσίαση της δομής της εφαρμογής DBF Editor και των κλάσεών της, παρακάτω ακολουθεί ένας αναλυτικός οδηγός χρήσης του λογισμικού.

## Οδηγός χρήσης του λογισμικού DBF Editor

Τι είναι το λογισμικό DBF Editor;

Το DBF Editor είναι ένα λογισμικό για την δημιουργία, τροποποίηση, προβολή αρχείων dBase(.dbf).

Κύριες λειτουργίες του DBF Editor:

* Υποστήριξη αρχείων FoxBase/FoxBase+/dBase III plus με και χωρίς memo.
* Άνοιγμα και προβολή των δεδομένων ενός αρχείου dBase (.dbf).
* Δημιουργία νέου αρχείου dBase (.dbf).
* Εισαγωγή νέων εγγραφών σε ένα αρχείο.
* Διαγραφή εγγραφών από το αρχείο.
* Εξαγωγή εγγραφών σε μορφή αρχείων xml, csv, html, xls, xlsx.
* Μαζική εισαγωγή εγγραφών από αρχεία xml, csv, xls, xlsx.
* Σύνθετη αναζήτηση εγγραφών (βάση συνθήκης) μέσα στο αρχείο και εξαγωγή των αποτελεσμάτων σε αρχεία τύπου xml, csv, html, xls, xlsx.
* Μαζική διαγραφή εγγραφών βάση μιας συνθήκης.
* Αλλαγή της δομής του αρχείου (προσθήκη/διαγραφή πεδίων, αλλαγή τύπων, μήκους πεδίων κτλ).
* Λειτουργίες αναζήτησης και αντικατάστασης εγγραφών μέσα στο αρχείο.
* Διάβασμα, δημιουργία, τροποποίηση αρχείων τύπου memo (.dbt).

Εικόνα 8: Το περιβάλλον του λογισμικού DBF Editor

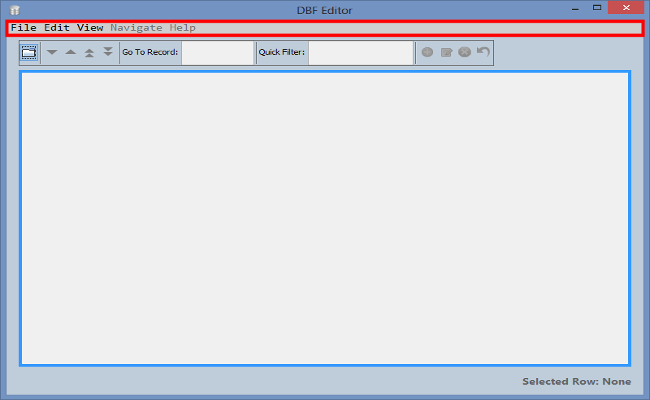
Τα κύρια μέρη της εφαρμογής είναι 3:

1. Το κυρίως menu (Main Menu).
2. Η γραμμή εργαλίων (Tool Bar).
3. Η περιοχή στην οποία φορτώνονται τα δεδομένα (Main Area).

### Το κεντρικό μενού(Main Menu)

Το μενού αυτό περιλαμβάνει τα εξής υπομενού:

* File Menu
* Edit Menu
* View Menu
* Navigate Menu
* Help Menu

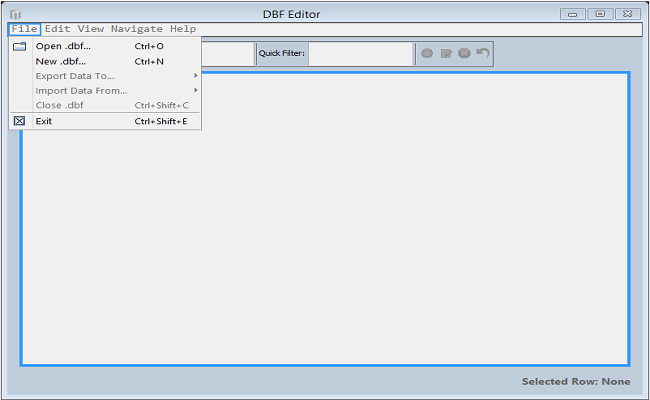


Εικόνα 9: Main Menu

#### File Menu

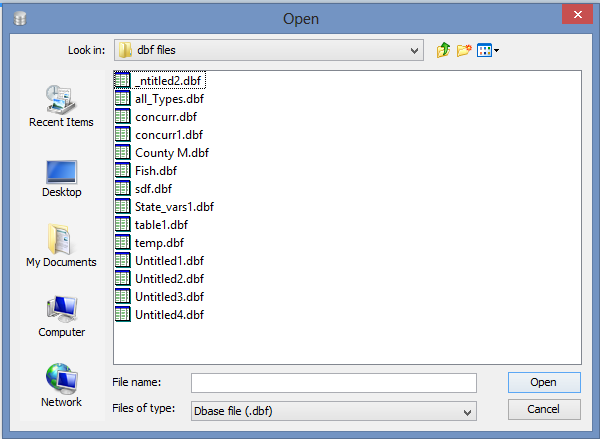
Το File Menu της εφαρμογής περιλαμβάνει τις εξής λειτουργίες:

* Open .dbf
* New .dbf
* Export Date To
* Import Data From
* Close .dbf
* Exit



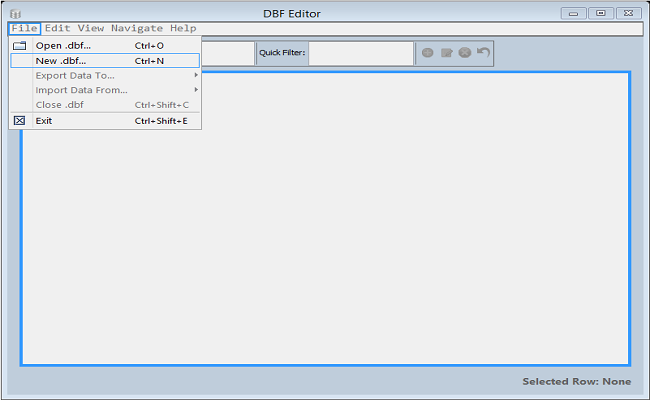
Εικόνα 10: File Menu

1. Με την επιλογή του Open .dbf, εμφανίζεται ένα dialog για την επιλογή dBase (.dbf) αρχείων. Το αρχείο που επιλέγεται φορτώνεται στην Main Area.

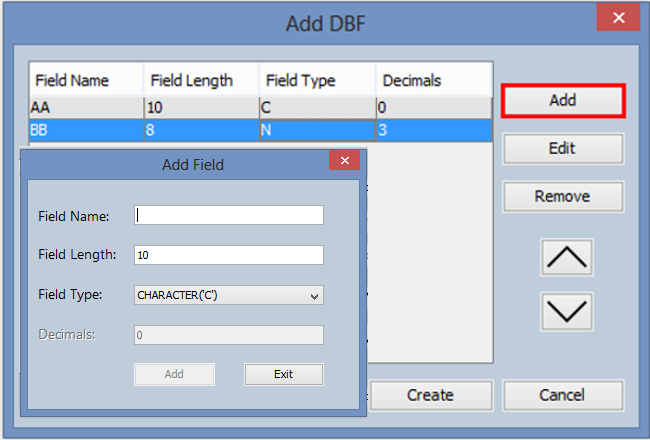


Εικόνα 11: Άνοιγμα αρχείου dBase

1. Με την επιλογή New .dbf, εμφανίζεται ένα dialog το οποίο δίνει τη δυνατότητα δημιουργίας νέου αρχείου.

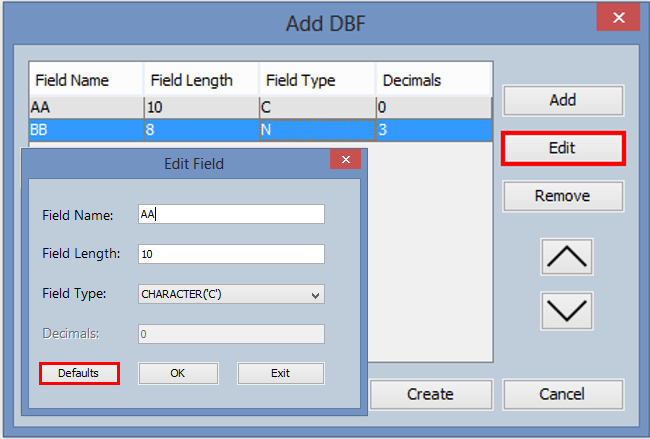


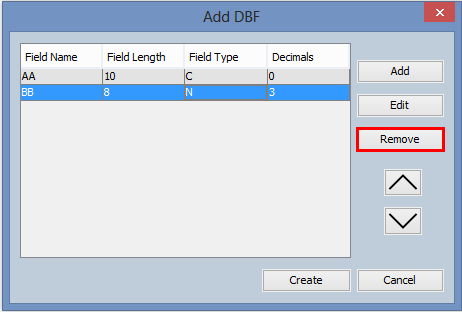
Εικόνα 12: Νέο αρχείο .dbf

* 1. Με το πάτημα του Add, εμφανίζεται το dialog για την προσθήκη ενός νέου πεδίου στο Table[[3]](#endnote-4).

Εικόνα 13: Προσθήκη πεδίου

* 1. Με το πάτημα του Edit, εμφανίζεται το dialog για την τροποποίηση ενός πεδίου3 .( Το κουμπί Defaults, βάζει τις τιμές στα πεδία που υπήρχαν πριν την τροποποίηση).

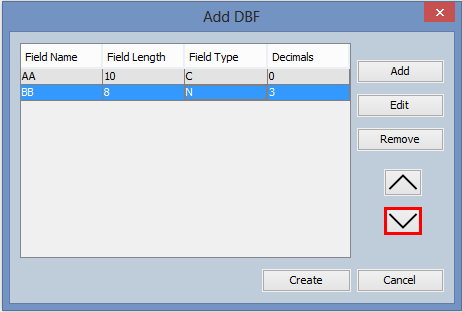
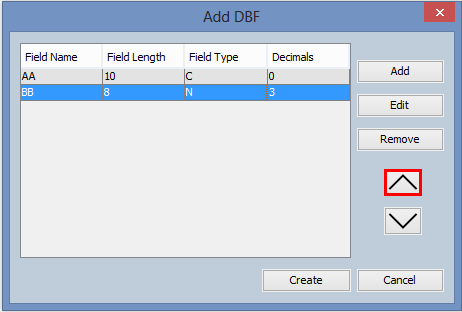


Εικόνα 14: Τροποποίηση πεδίου

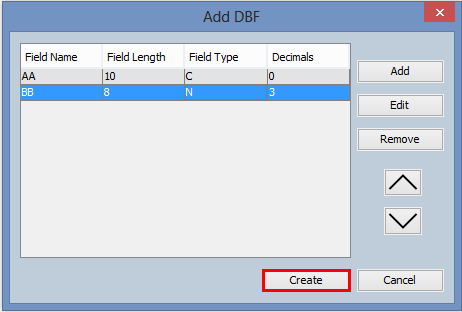
* 1. Το κουμπί Remove, διαγράφει ένα πεδίο.

Εικόνα 15: Διαγραφή πεδίου

* 1. Τα κουμπιά με τα βελάκια Up και Down, επιτρέπουν την αναδιάταξη των πεδίων.

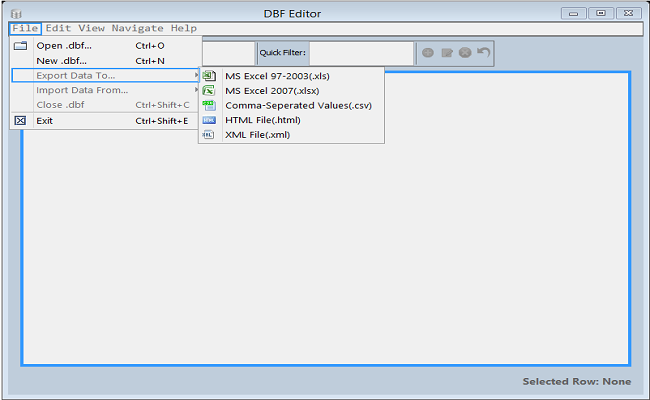
Εικόνα 16: Αναδιάταξη πεδίων

* 1. Με το κουμπί Create, δημιουργείτε το νέο Table.

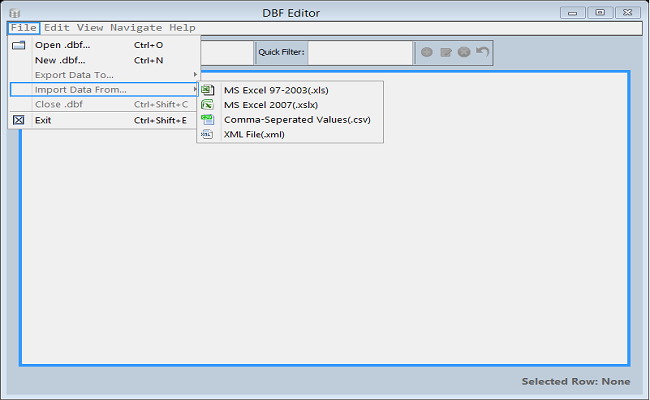


Εικόνα 17: Δημιουργία του αρχείου

1. Με την επιλογή του Export Data To, δίνεται η δυνατότητα να εξαχθούν τα δεδομένα του αρχείου σε αρχεία xml, csv, html, xls, xlsx. (ΠΡΟΣΟΧΗ: Τα όρια του αρχείου .xls (MS Excel 97-2003) είναι 255 πεδία και 65535 γραμμές. Αν τα δεδομένα του αρχείου, προς εξαγωγή, υπερβαίνουν τα όρια χρησιμοποίησε MS Excel 2007 (.xlsx).

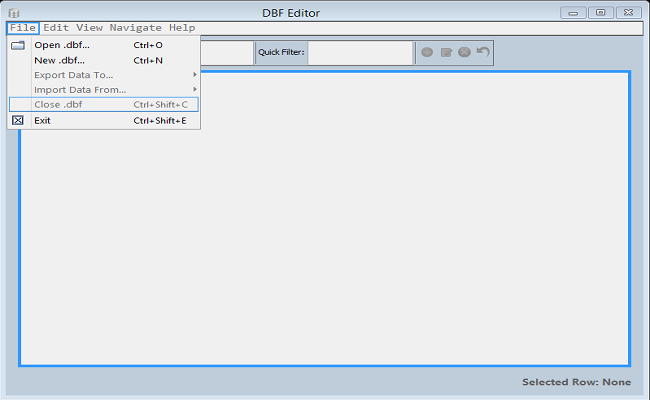


Εικόνα 18: Εξαγωγή δεδομένων

1. Με την επιλογή του Import Data From, δίνεται η δυνατότητα να εισαχθούν μαζικά εγγραφές στο Table από αρχεία τύπου xls, xlsx, csv και xml.

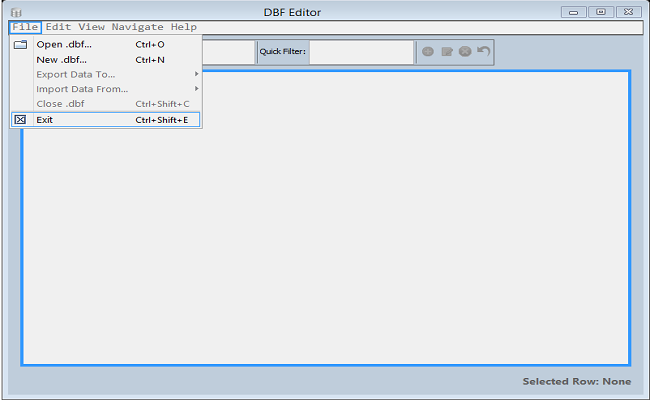
Εικόνα 19: Εισαγωγή εγγραφών από αρχείο

1. Με την επιλογή του Close .dbf, κλείνει ένα αρχείο .dbf.



Εικόνα 20: Κλείσιμο αρχείου

1. Με την επιλογή του Exit, γίνεται έξοδος από την εφαρμογή.

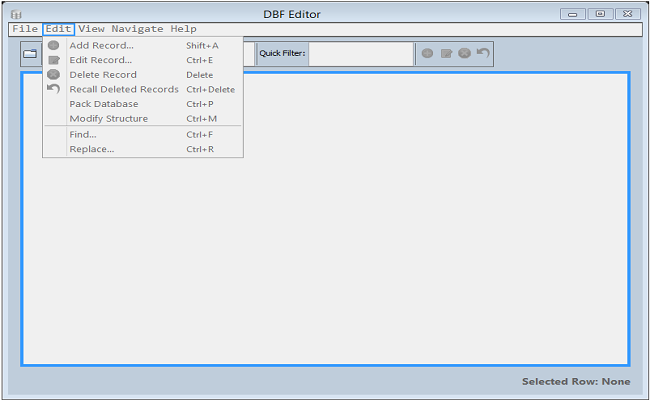


Εικόνα 21: ‘Έξοδος από την εφαρμογή

#### Edit Menu

Το edit menu της εφαρμογής περιλαμβάνει τις εξής λειτουργίες:

* Add Record
* Edit Record
* Delete Record
* Recall Deleted Records
* Pack Database
* Modify Structure
* Find
* Replace

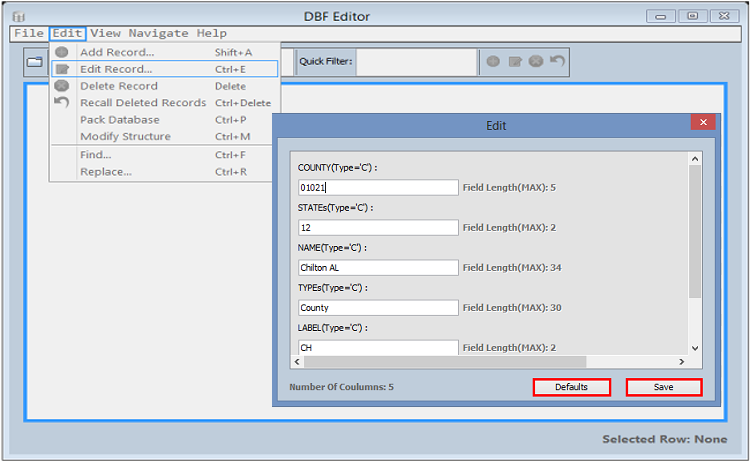


Εικόνα 22: Edit Menu

1. Με την επιλογή του Add Record, εμφανίζεται ένα dialog για προσθήκη μιας νέας εγγραφής. Με το πάτημα του κουμπιού Add γίνεται προσθήκη της εγγραφής στο Table και παράλληλη αποθήκευση στο αρχείο.

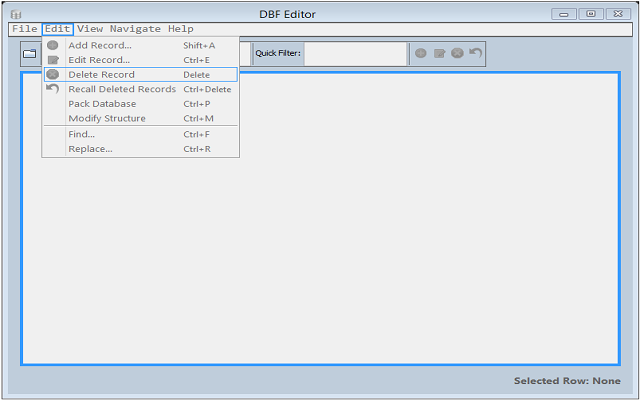


Εικόνα 23: Προσθήκη εγγραφής

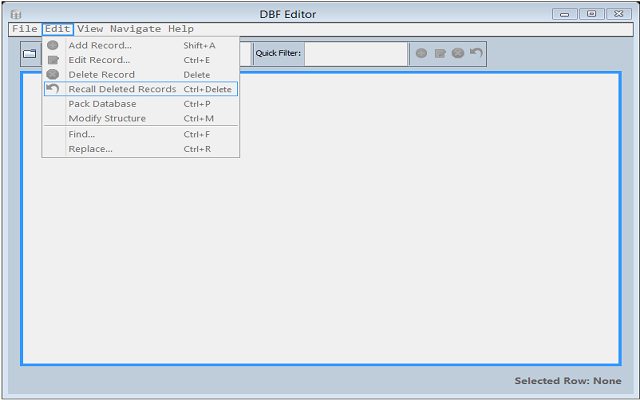
1. Με την επιλογή Edit Record, εμφανίζεται ένα dialog το οποίο δίνει τη δυνατότητα τροποποίησης της επιλεγμένης εγγραφής. Το κουμπί Defaults, επαναφέρει τις αρχικές τιμές στα πεδία, ενώ με το κουμπί Save, αποθηκεύεται η επεξεργασμένη εγγραφή στο Table και στο αρχείο.

Εικόνα 24: Τροποποίηση μιας εγγραφής

1. Με την επιλογή Delete Record, σημειώνεται μια εγγραφή ως διαγραμμένη. ΠΡΟΣΟΧΗ: Η εγγραφή απλά σημειώνεται ως διαγραμμένη, ΔΕΝ διαγράφεται ούτε από το αρχείο, ούτε από το Table. Για να γίνει αυτό πρέπει να εκτελεστεί η Pack!!.

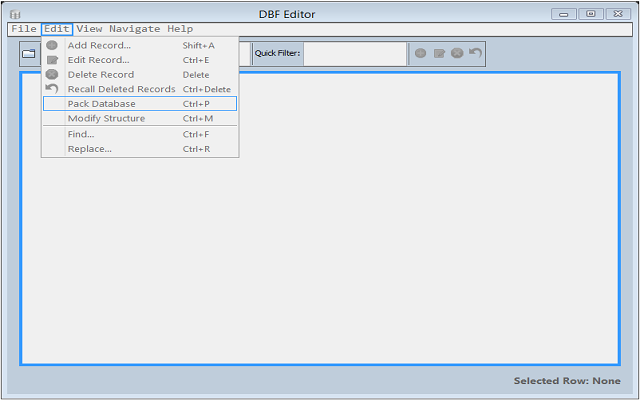


Εικόνα 25: Διαγραφή εγγραφής

1. Με την επιλογή του Recall Deleted Records, όλες οι εγγραφές οι οποίες είναι σημειωμένες για διαγραφή επανέρχονται.

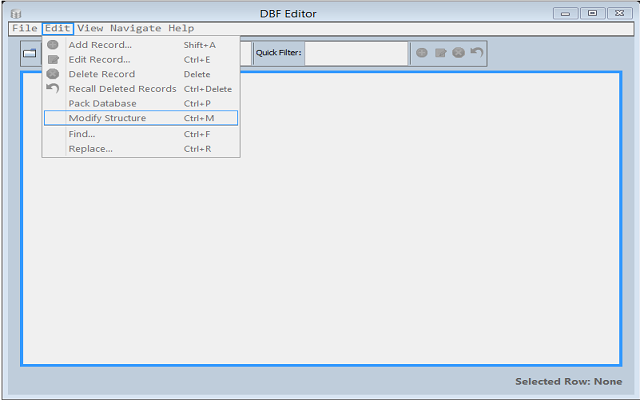
Εικόνα 26: Επαναφορά εγγραφών

1. Με την επιλογή του Pack Database, όλες οι εγγραφές που είναι σημειωμένες για διαγραφή, διαγράφονται οριστικά από το αρχείο και το Table. ΠΡΟΣΟΧΗ: Μετά την εκτέλεση της Pack, ΔΕΝ υπάρχει η δυνατότητα επαναφοράς (undo) των διαγραμμένων εγγραφών.



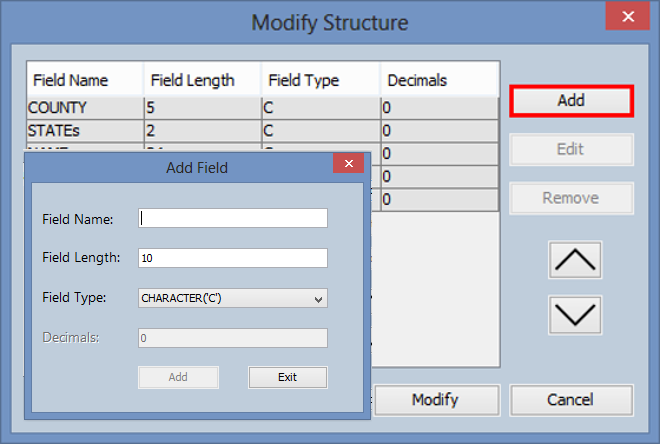
Εικόνα 27: Pack Database

1. Με την επιλογή του Modify Structure, εμφανίζεται ένα dialog το οποίο δίνει τη δυνατότητα για αλλαγή της δομής του αρχείου. Για να διατηρηθεί η συνέπεια του αρχείου κατά την αλλαγή της δομής υπάρχουν οι εξής κανόνες:
   1. Επιτρέπεται να γίνει μια μετονομασία (rename) κάθε φορά.
   2. Αν γίνει μετονομασία (rename) δεν επιτρέπεται να γίνει αλλαγή τύπου, μήκους, πλήθους δεκαδικών στο ίδιο πεδίο.
   3. Αν γίνει μετονομασία (rename), δεν επιτρέπεται να γίνει εισαγωγή (insert) ή διαγραφή (delete) πεδίων.
   4. Κατά την εισαγωγή (insert) ή διαγραφή(delete) πεδίων, επιτρέπεται να γίνει αλλαγή τύπου, μήκους, πλήθους δεκαδικών.
   5. Επιτρέπεται να γίνουν πολλές εισαγωγές (insert) ή διαγραφές (delete) πεδίων.
   6. Αν γίνει εισαγωγή (insert) δεν επιτρέπεται να γίνει διαγραφή (delete) και το αντίστροφο.
   7. Σε κάθε περίπτωση επιτρέπεται η αναδιάταξη των πεδίων.



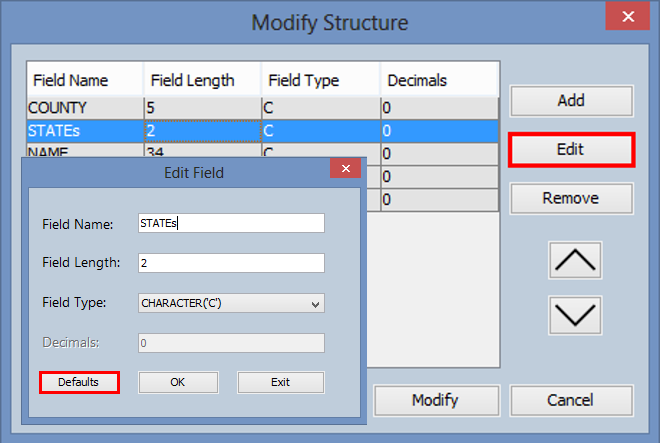
Εικόνα 28: Αλλαγή της δομής

* + 1. Με το πάτημα του Add, εμφανίζεται το dialog για τη προσθήκη ενός νέου πεδίου στο Table3.



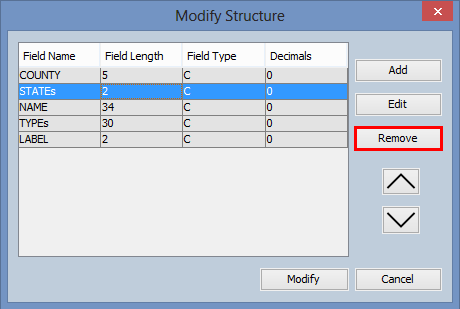
Εικόνα 29: Προσθήκη πεδίου

* + 1. Με το πάτημα του Edit, εμφανίζεται το dialog για την τροποποίηση του επιλεγμένου πεδίου3. Το κουμπί Defaults, βάζει τις τιμές στα πεδία που υπήρχαν πριν την τροποποίηση.

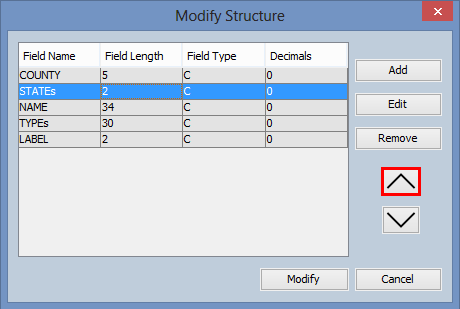
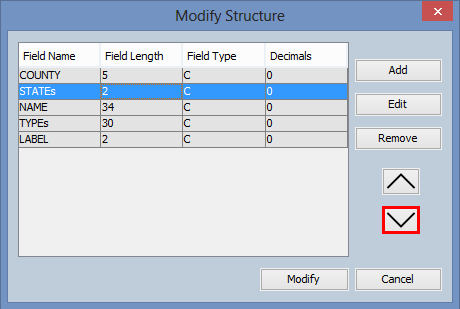


Εικόνα 30: Τροποποίηση πεδίου

* + 1. Το κουμπί Remove, διαγράφει ένα πεδίο.

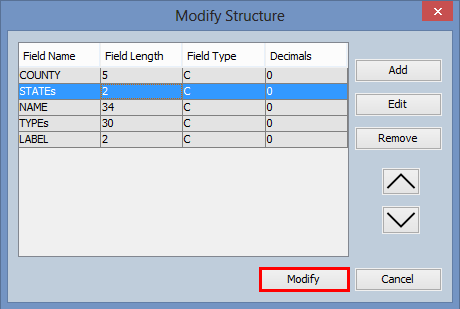


Εικόνα 31: Διαγραφή πεδίου

* + 1. Τα κουμπιά με τα δύο βελάκια Up και Down, επιτρέπουν την αναδιάταξη των πεδίων.

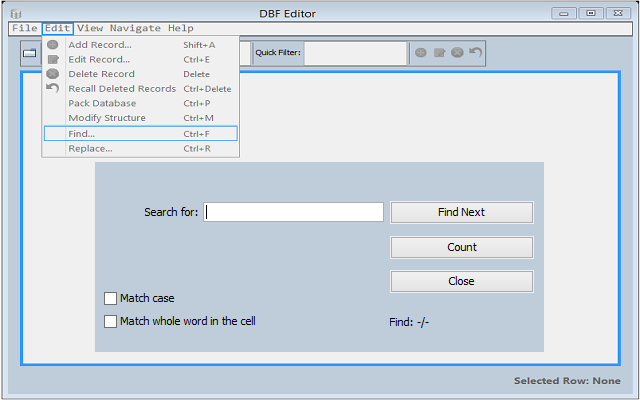
Εικόνα 32: Αναδιάταξη των πεδίων

* + 1. Με το κουμπί Modify, αλλάζει η δομή του αρχείου.



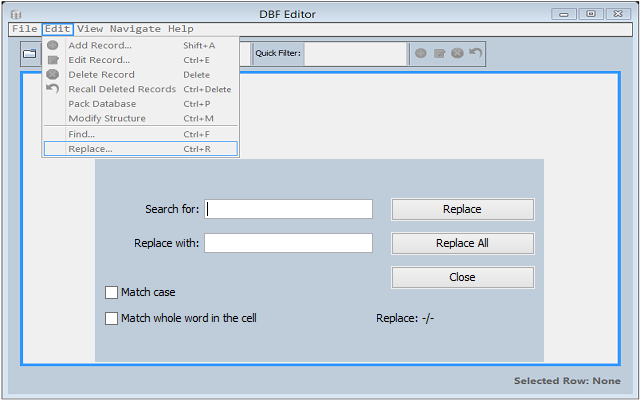
Εικόνα 33: Αλλαγή της δομής

1. Με την επιλογή του Find, δίνεται η δυνατότητα αναζήτησης στις εγγραφές του αρχείου.



Εικόνα 34: Αναζήτηση εγγραφής

1. Με την επιλογή του Replace, δίνεται η δυνατότητα να γίνει αναζήτηση και αντικατάσταση στις εγγραφές του αρχείου.



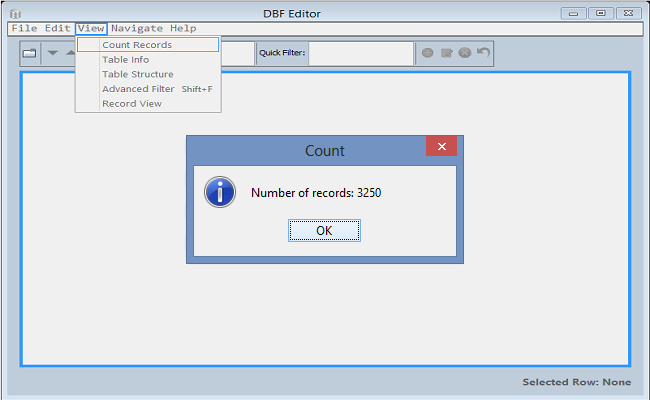
Εικόνα 35: Αναζήτηση και αντικατάσταση εγγραφής

#### View Menu

Το view menu της εφαρμογής περιλαμβάνει τις εξής λειτουργίες:

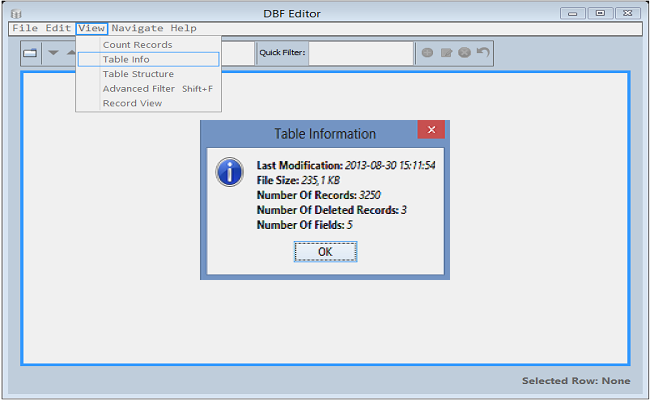
* Count Records
* Table Info
* Table Structure
* Advanced Filter
* Record View

Εικόνα 36: View Menu

1. Με την επιλογή του Count Records, εμφανίζεται ένα dialog του δείχνει το πλήθος των εγγραφών στο αρχείο.

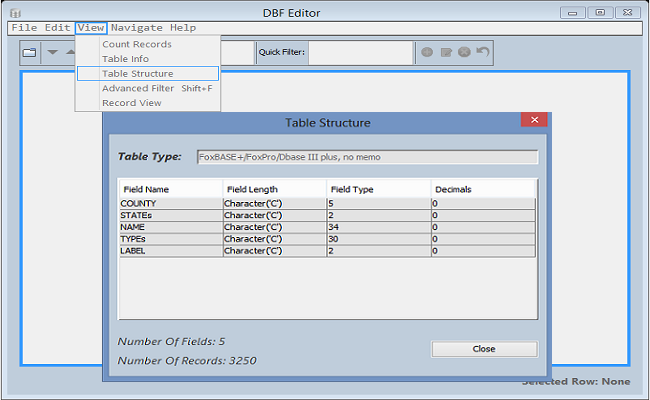
Εικόνα 37: Μέτρηση εγγραφών

1. Με την επιλογή του Table Info, εμφανίζεται ένα dialog το οποίο περιέχει πληροφορίες για το αρχείο.



Εικόνα 38: Πληροφορίες για το αρχείο

1. Με την επιλογή του Table Structure, εμφανίζεται ένα dialog το οποίο περιέχει πληροφορίες για τη δομή και τον τύπο του αρχείου.



Εικόνα 39: Η δομή του αρχείου

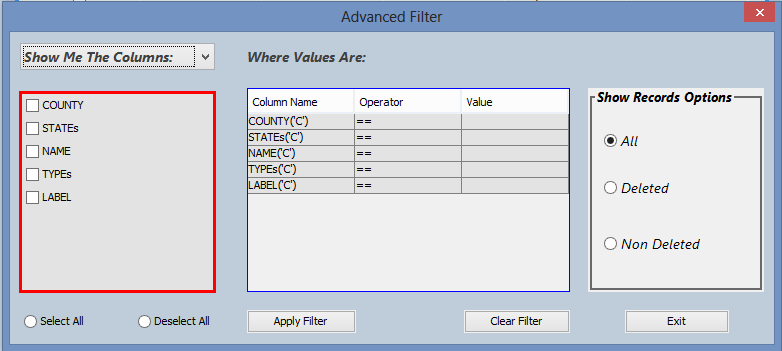
1. Με την επιλογή του Advanced Filter, εμφανίζεται ένα dialog μέσα από το οποίο μπορεί να γίνει σύνθετη αναζήτηση, διαγραφή και επαναφορά εγγραφών.



Εικόνα 40: Advanced Filter

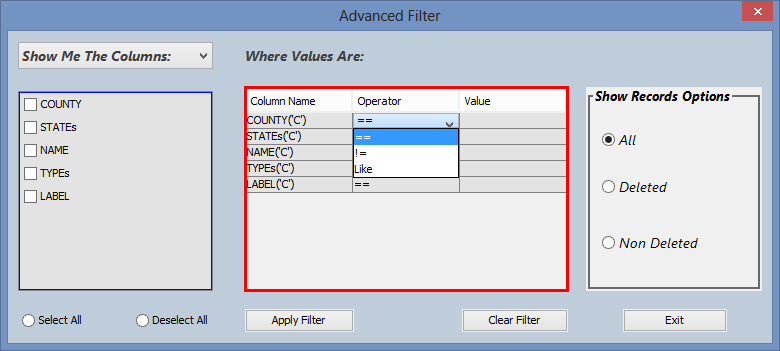
* 1. Το Advanced Filter(Συνθετο Φίλτρο), προσφέρει τριών ειδών λειτουργίες:
* Φιλτράρισμα των εγγραφών βάση σύνθετης συνθήκης και εξαγωγή των αποτελεσμάτων σε αρχείο τύπου xml, csv, html, xls, xlsx.
* Διαγραφή εγγραφών βάση σύνθετης συνθήκης.
* Σημείωση εγγραφών ως μη διαγραμμένες βάση σύνθετης συνθήκης.

1. Φιλτράρισμα εγγραφών με σύνθετη συνθήκη
   1. Στην περιοχή κάτω από το Show Me The Columns:, γίνεται η επιλογή των πεδίων που θα εμφανιστούν μετά το φιλτράρισμα των εγγραφών.



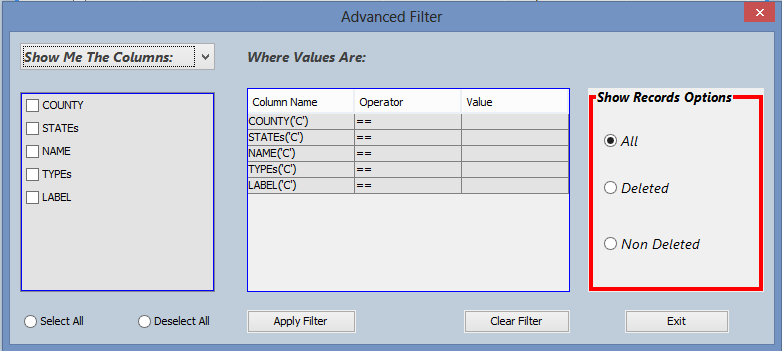
Εικόνα 41: Επιλογή πεδίων

* 1. Στην περιοχή κάτω από το Where Values Are:, εμφανίζονται όλα τα πεδία του Table. Στη στήλη Operator εμφανίζεται ένα combo box μέσω του οποίου μπορεί να επιλεχθεί ο τελεστής για τη συνθήκη. Στη στήλη Value, μπαίνει η τιμή σύμφωνα με την οποία θα γίνει η σύγκριση μέσω του τελεστή που έχει επιλεχθεί.



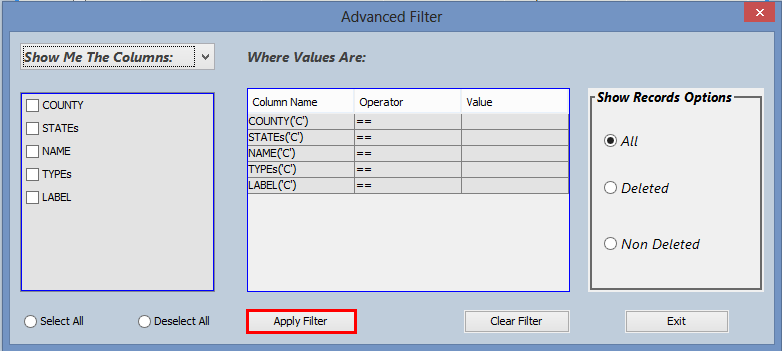
Εικόνα 42: Επιλογή τελεστών και εισαγωγή τιμών για σύγκριση

* 1. Στην περιοχή Show Record Options, επιλέγετε ποιες εγγραφές θα πάρουν μέρος στο φιλτράρισμα (όλες, μόνο οι σημειωμένες για διαγραφή, μόνο οι μη διαγραμμένες).

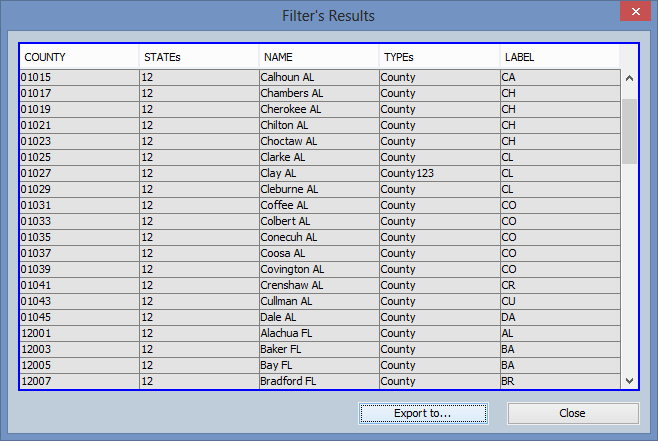


Εικόνα 43: Επιλογή των εγγραφών που θα πάρουν μέρος στην αναζήτηση

* 1. Το κουμπί Apply, εφαρμόζει το φίλτρο και εμφανίζει τα αποτελέσματα(εάν υπάρχουν). Το κουμπί Export, εξάγει τα δεδομένα σε αρχεία xml, csv, html, xls, xlsx.

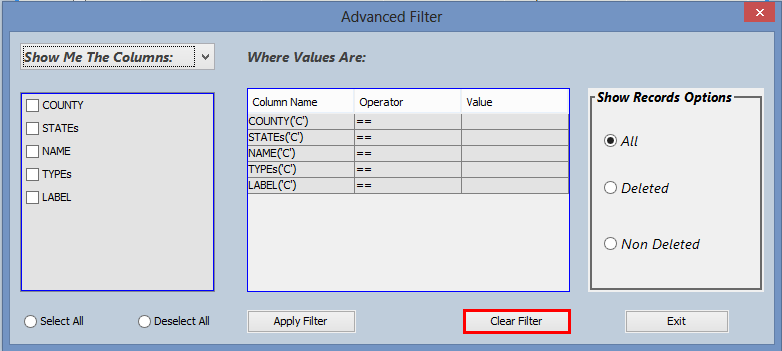


Εικόνα 44: Εφαρμογή του φίλτρου



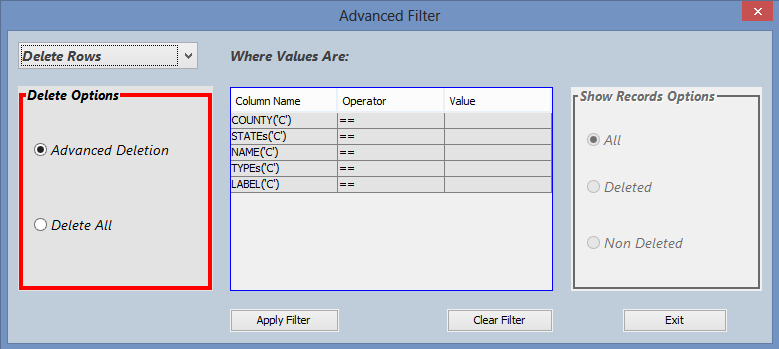
Εικόνα 45: Προβολή των αποτελεσμάτων του φίλτρου

* 1. Το κουμπί Clear Filter, επαναφέρει το φίλτρο στην αρχική κατάσταση.



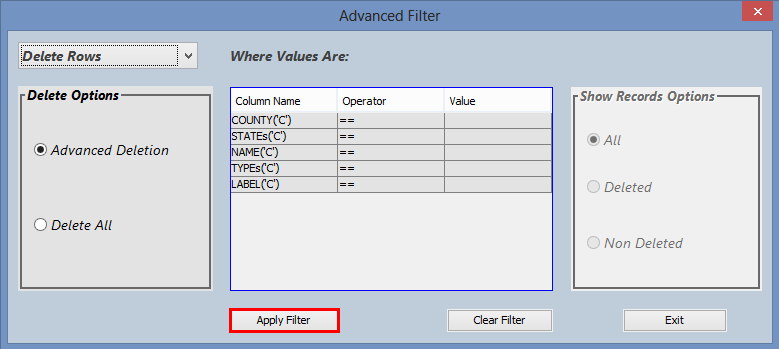
Εικόνα 46: Καθαρισμός του φίλτρου

1. Διαγραφή εγγραφών με σύνθετη συνθήκη.
   1. Στην περιοχή Delete Options, επιλέγοντας Advanced Deletetion, διαγράφονται εγγραφές βάση μιας σύνθετης συνθήκης όπως περιγράφεται στο νούμερο 1. Με την επιλογή Delete All, διαγράφονται όλες οι εγγραφές στο Table.



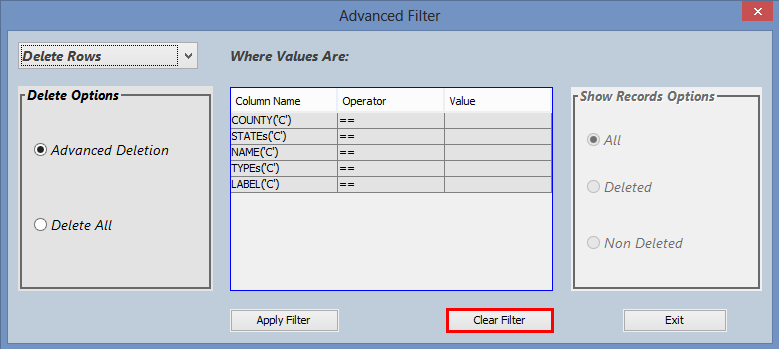
Εικόνα 47: Επιλογές διαγραφής εγγραφών

* 1. Το κουμπί Apply, εφαρμόζει το φίλτρο και διαγράφει τις εγγραφές(αν βρεθούν).



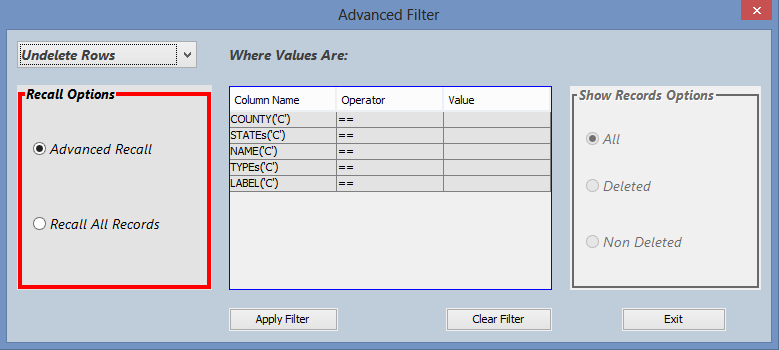
Εικόνα 48: Εφαρμογή της σύνθετης διαγραφής

* 1. Το κουμπί Clear Filter, επαναφέρει το φίλτρο στην αρχική του κατάσταση.



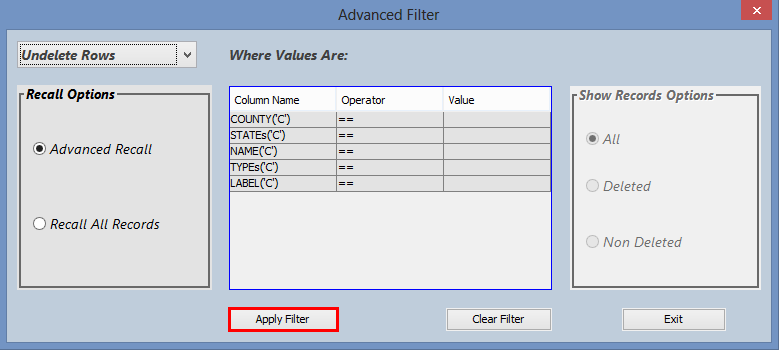
Εικόνα 49: Καθαρισμός του φίλτρου

1. Σημείωση εγγραφών ως μη διαγραμμένες με σύνθετη συνθήκη.
   1. Στην περιοχή Recall Options, επιλέγοντας την Advanced Recall, σημειώνονται ως μη διαγραμμένες οι εγγραφές βάση μιας συνθήκης (όπως περιγράφεται στο νούμερο 1). Με την επιλογή Recall All Records, σημειώνονται ως μη διαγραμμένες όλες οι εγγραφές στο Table.



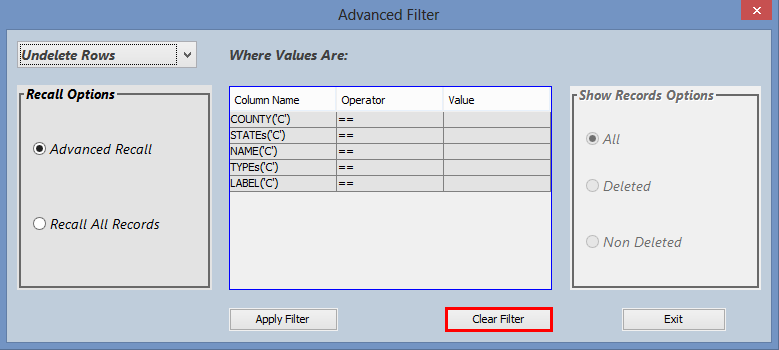
Εικόνα 50: Επιλογές επαναφοράς εγγραφών

* 1. Το κουμπί Apply, εφαρμόζει το φίλτρο και σημειώνει ως μη διαγραμμένες τις εγγραφές(όσες βρεθούν).



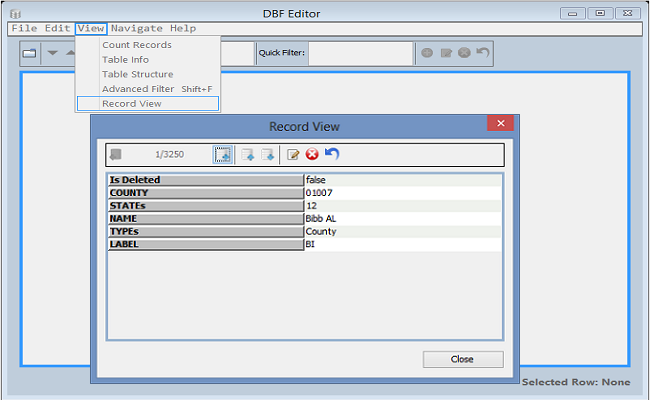
Εικόνα 51: Εφαρμογή φίλτρου για επαναφορά εγγραφών

* 1. Το κουμπί Clear Filter επαναφέρει το φίλτρο στη αρχική του κατάσταση.



Εικόνα 52: Καθαρισμός του φίλτρου

1. Με την επιλογή του Record View, ανοίγει ένα dialog που δίνει τη δυνατότητα προβολής των εγγραφών μία-μία. Μέσα από το Record View, δίνεται η δυνατότητα πλοήγησης στην επόμενη, προηγούμενη, πρώτη και τελευταία εγγραφή, καθώς και η τροποποίηση και διαγραφή μιας εγγραφής.

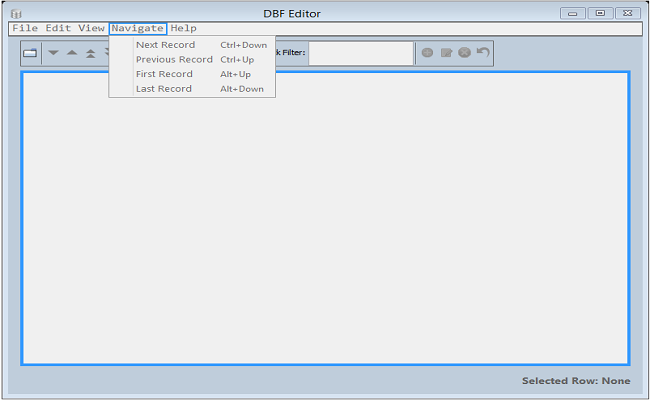


Εικόνα 53: Record View

#### Navigate Menu

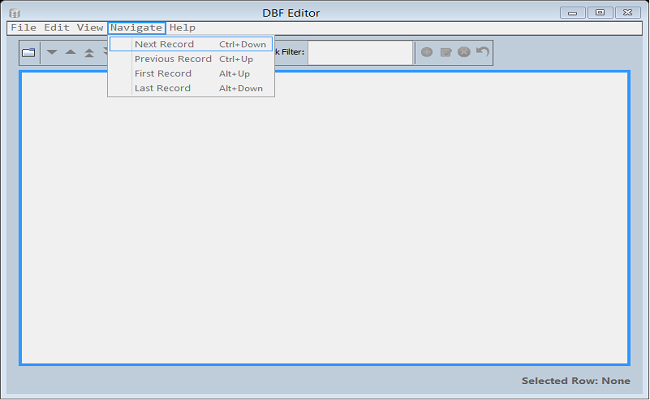
Το navigate menu της εφαρμογής περιλαμβάνει τις εξής λειτουργίες:

* Next Record
* Previous Record
* First Record
* Last Record



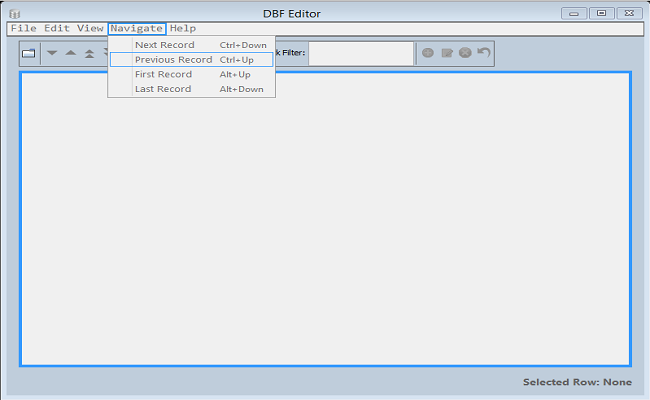
Εικόνα 54: Navigate Menu

1. Με την επιλογή του Next Record, γίνεται μετάβαση στην αμέσως επόμενη εγγραφή.



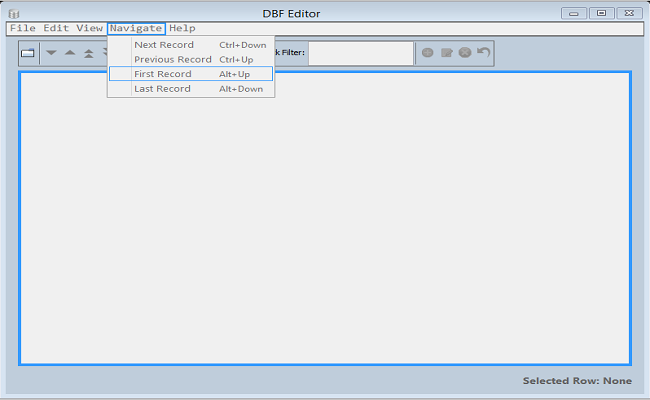
Εικόνα 55: Επόμενη εγγραφή

1. Με την επιλογή του Previous Record, γίνεται μετάβαση στην αμέσως προηγούμενη εγγραφή.



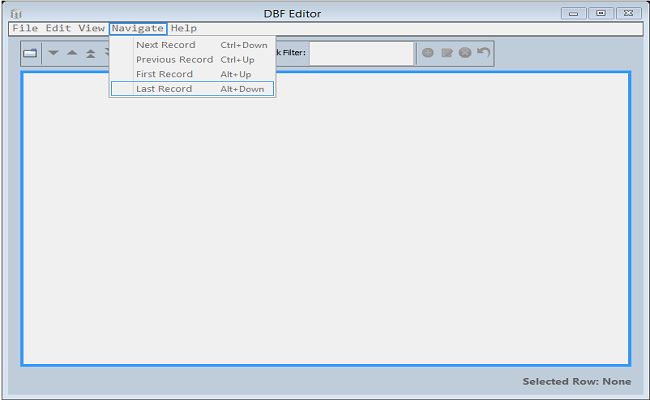
Εικόνα 56: Προηγούμενη εγγραφή

1. Με την επιλογή First Record, γίνεται μετάβαση στην πρώτη εγγραφή.



Εικόνα 57: Πρώτη εγγραφή

1. Με την επιλογή του Last Record, γίνεται μετάβαση στην τελευταία εγγραφή.



Εικόνα 58: Τελευταία εγγραφή

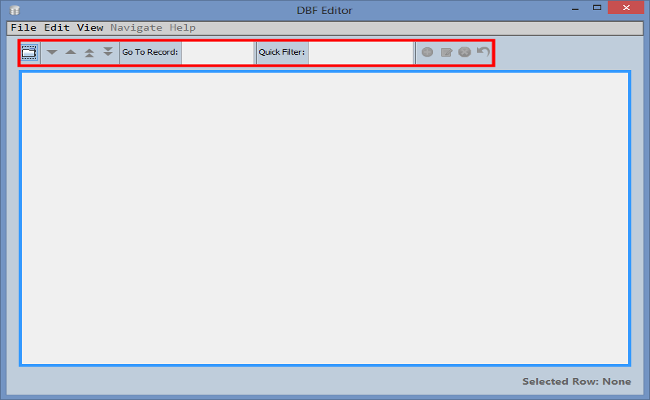
#### Help Menu

Το help menu περιλαμβάνει πληροφορίες για το λογισμικό καθώς και μια σελίδα html η οποία περιέχει τον οδηγό χρήσης της εφαρμογής.

### Η γραμμή εργαλείων (Tool Bar)

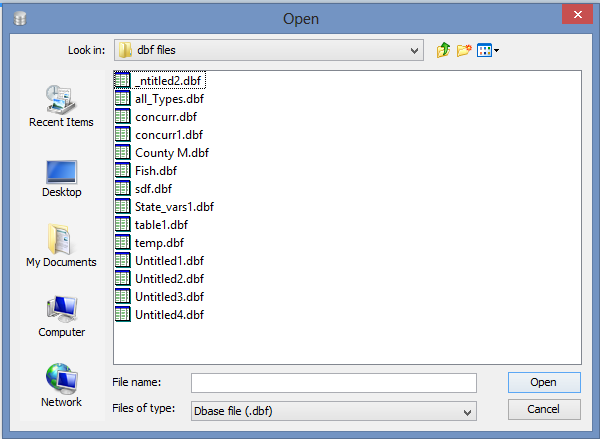
Η γραμμή εργαλείων περιλαμβάνει τις συντομεύσεις των λειτουργιών που χρησιμοποιούνται πιο συχνά:

* Open .dbf
* Next Record
* Previous Record
* First Record
* Last Record
* Go To Record
* Quick Filter
* Add Record
* Edit Record
* Delete Record
* Recall Deleted Record



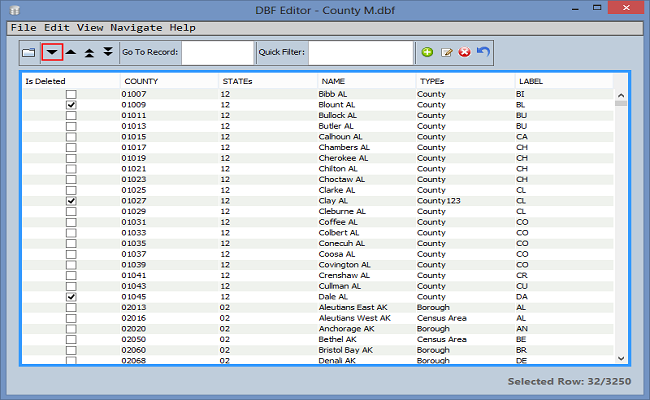
Εικόνα 59: Γραμμή Εργαλείων

1. Με την επιλογή του κουμπιού directory.gif, εμφανίζεται ένα dialog για την επιλογή dBase (.dbf) αρχείων. Το αρχείο που επιλέγεται φορτώνεται στην Main Area.



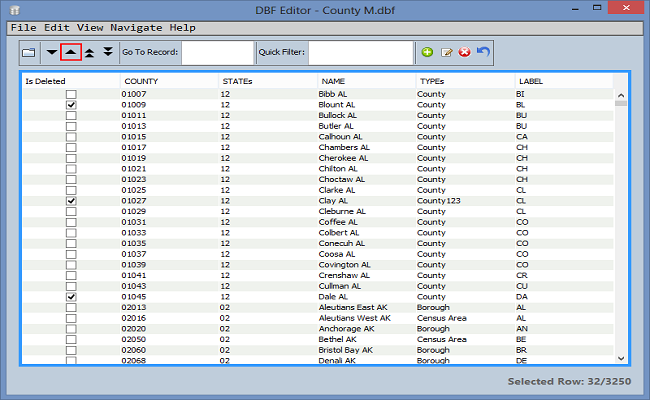
Εικόνα 60: Open DIalog

1. Με την επιλογή του κουμπιού down_arrow.png, γίνεται μετάβαση στη αμέσως επόμενη εγγραφή.



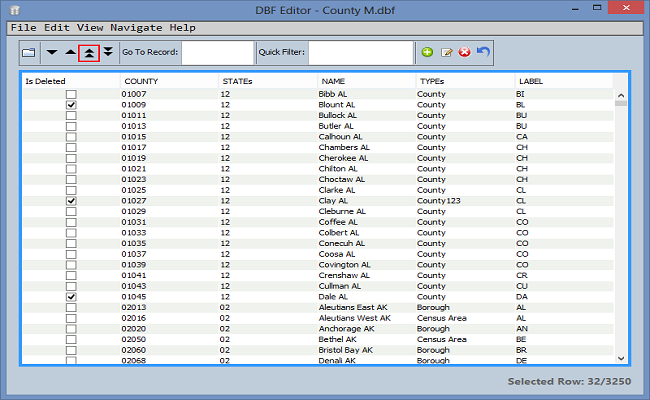
Εικόνα 61: Επόμενη εγγραφή

1. Με την επιλογή του κουμπιού up_arrow.png, γίνεται μετάβαση στη αμέσως προηγούμενη εγγραφή.



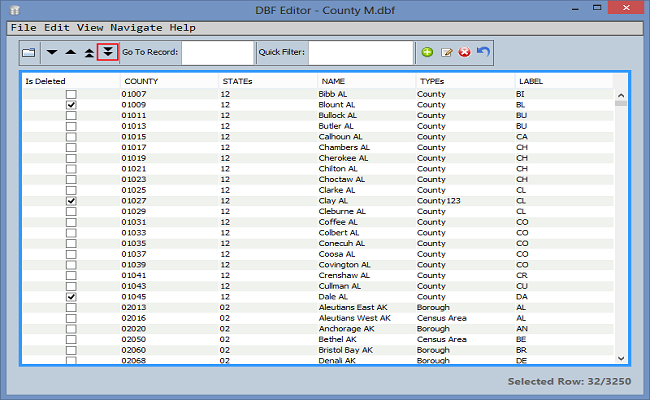
Εικόνα 62: Προηγούμενη εγγραφή

1. Με την επιλογή του κουμπιού first_Fin.png, γίνεται μετάβαση στην πρώτη εγγραφή.



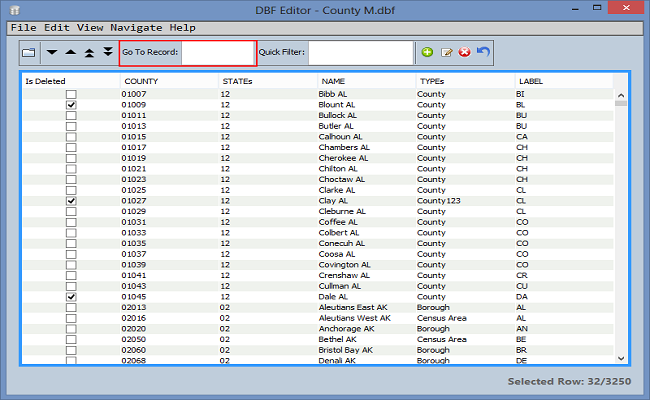
Εικόνα 63: Πρώτη εγγραφή

1. Με την επιλογή του κουμπιού last_Fin.png, γίνεται μετάβαση στην τελευταία εγγραφή.



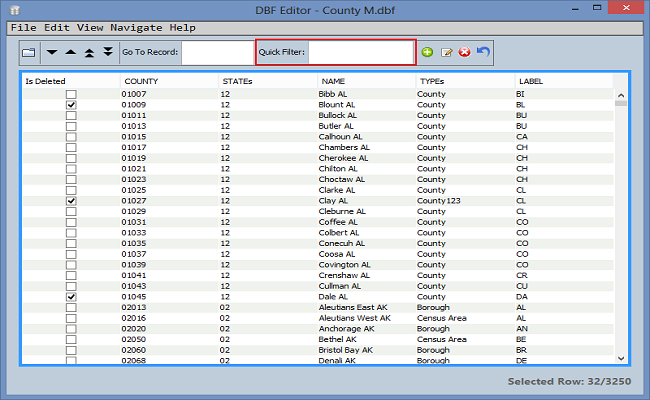
Εικόνα 64: Τελευταία εγγραφή

1. Στο πεδίο Go To Record:, επιτρέπεται η πληκτρολόγηση ενός αριθμού για απευθείας μετάβαση σε μια εγγραφή.



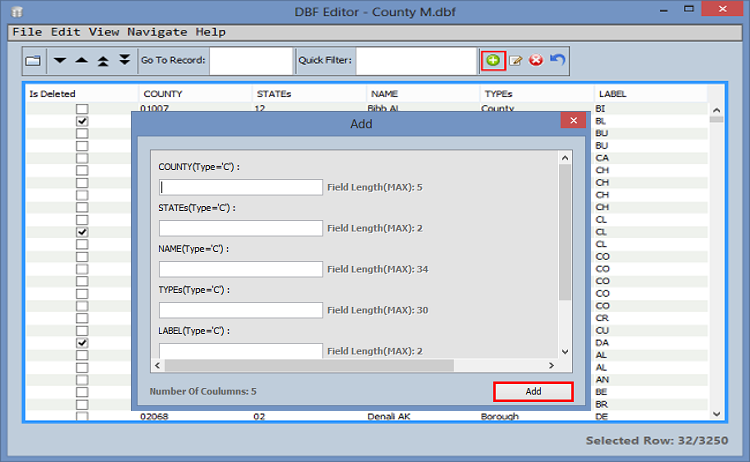
Εικόνα 65: Απευθείας μετάβαση σε εγγραφή

1. Στο πεδίο Quick Filter: επιτρέπεται η πληκτρολόγηση ενός αλφαριθμητικού έτσι ώστε να εμφανίζονται μόνο οι εγγραφές που το περιέχουν.

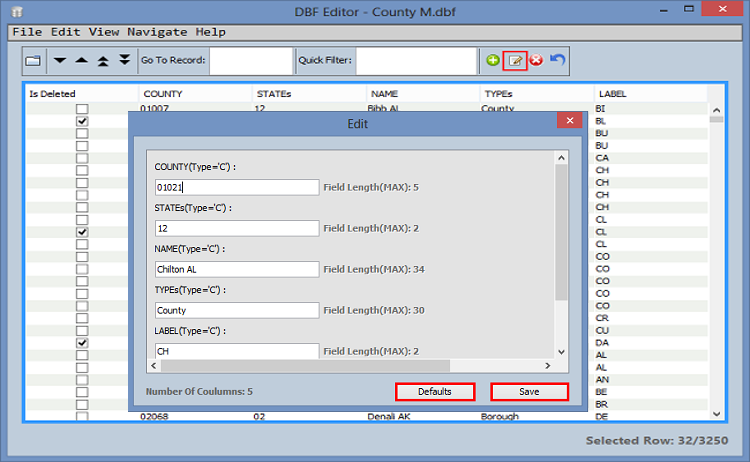


Εικόνα 66: Quick filter

1. Με την επιλογή του κουμπιού add.png, εμφανίζεται το dialog για προσθήκη μιας νέας εγγραφής στο αρχείο. Με το πάτημα του κουμπιού Add, γίνεται αποθήκευση της νέας εγγραφής στο αρχείο και στο Table.

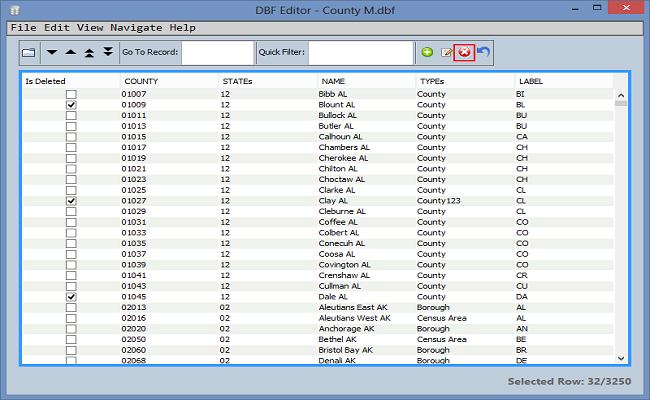


Εικόνα 67: Προσθήκη νέας εγγραφής

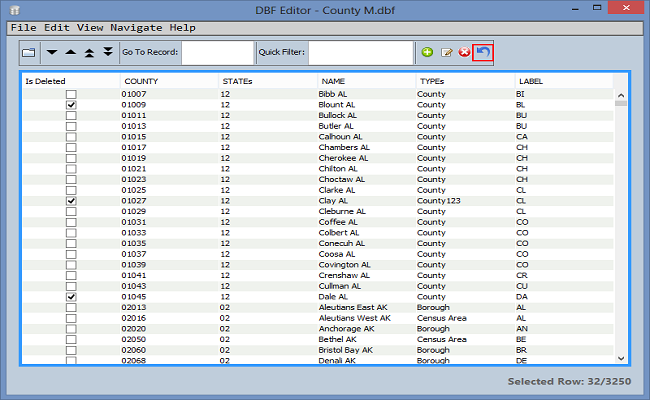
1. Με την επιλογή του κουμπιού edit.png, εμφανίζεται ένα dialog το οποίο δίνει τη δυνατότητα τροποποίησης της επιλεγμένης εγγραφής. Το κουμπί Defaults επαναφέρει τις αρχικές τιμές στα πεδία, ενώ το κουμπί Save αποθηκεύει την τροποποιημένη εγγραφή στο Table και στο αρχείο.

Εικόνα 68: Τροποποίηση μιας εγγραφής

1. Με τη επιλογή του κουμπιού icon-delete.gif, η επιλεγμένη εγγραφή σημειώνεται για διαγραφή. ΠΡΟΣΟΧΗ: Η εγγραφή απλά σημειώνεται για διαγραφή, ΔΕΝ διαγράφεται. Για να διαγραφεί από το Table και το αρχείο πρέπει να εκτελεστεί η Pack!!



Εικόνα 69: Διαγραφή εγγραφής

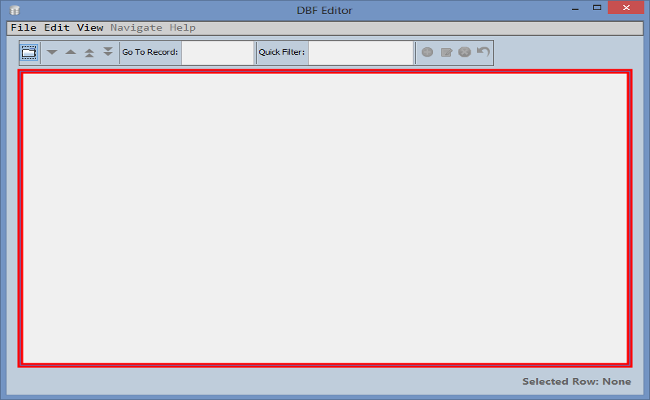
1. Με την επιλογή του κουμπιού Undo-icon.png, σημειώνεται η επιλεγμένη εγγραφή ως μη διαγραμμένη.

Εικόνα 70: Επαναφορά εγγραφής

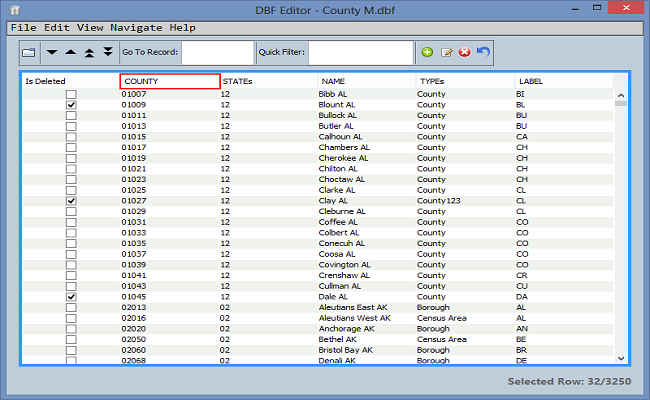
### Κύρια περιοχή φόρτωσης δεδομένων(Main Area)

Στην περιοχή που φορτώνονται τα δεδομένα του dBase(.dbf) αρχείου(Main Area), μπορούν να εκτελεστούν οι παρακάτω λειτουργίες:

* Ταξινόμηση με βάση ένα πεδίο.
* Επεξεργασία της επιλεγμένης εγγραφής.
* Εμφάνιση του περιεχομένου των πεδίων τύπου Memo.

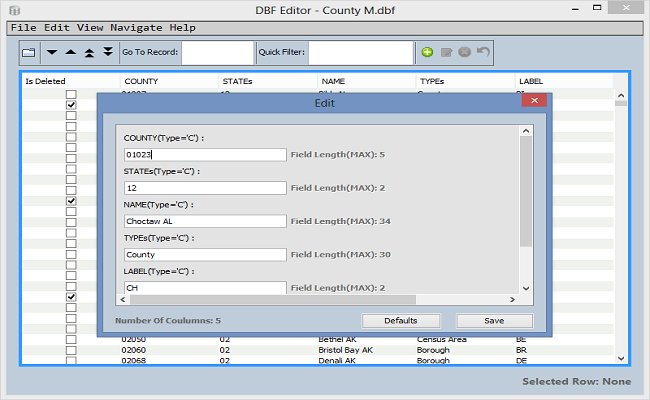


Εικόνα 71: Main Area

1. Με το πάτημα στο όνομα του πεδίου(header), γίνεται ταξινόμηση με βάση αυτό το πεδίο.

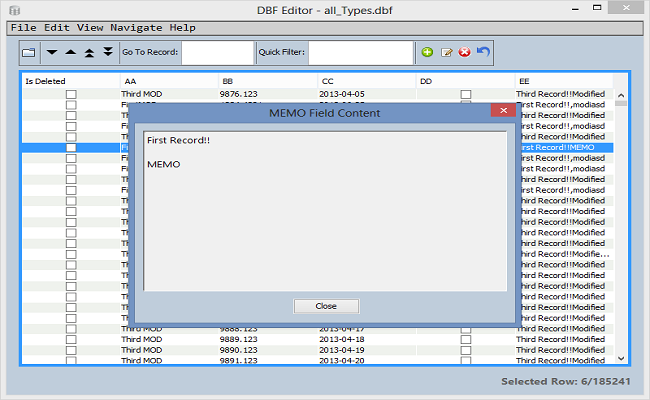
Εικόνα 72: Ταξινόμηση με βάση ένα πεδίο

1. Με διπλό κλικ πάνω σε μια εγγραφή του Table, εμφανίζεται το dialog το οποίο δίνει τη δυνατότητα τροποποίησης της επιλεγμένης εγγραφής.



Εικόνα 73: Τροποποίηση εγγραφής

1. Με δεξί κλικ πάνω σε ένα πεδίο τύπου MEMO, εμφανίζεται ένα dialog με το περιεχόμενο του πεδίου.



Εικόνα 74: Προβολή περιεχομένου πεδίου τύπου MEMO

### Shortcuts(Συντομεύσεις στο λογισμικό).

Πίνακας 5: Συντομεύσεις λογισμικού

|  |  |  |
| --- | --- | --- |
| Menu | Shortcut | Περιγραφή |
| File Menu | Ctrl+O | Εμφανίζει ένα dialog για την επιλογή dBase(.dbf) αρχείων. |
| Ctrl+N | Εμφανίζει ένα dialog το οποίο δίνει τη δυνατότητα δημιουργίας νέου αρχείου. |
| Ctrl+Shift+C | Κλείνει ένα αρχείο .dbf |
| Ctrl+Shift+E | Γίνεται έξοδος από την εφαρμογή. |
| Edit Menu | Shift+A | Εμφανίζετια ένα dialog για προσθήκη νέας εγγραφής. |
| Ctrl+E | Εμφανίζεται ένα dialog το οποίο δίνει τη δυνατότητα τροποποίησης μιας εγγραφής. |
| Delete | Σημειώνει μια εγγραφή ως διαγραμμένη. |
| Ctrl+Delete | Όλες οι εγγραφές οι οποίες είναι σημειωμένες για διαγραφή, επανέρχονται. |
| Ctrl+P | Όλες οι εγγραφές οι οποίες είναι σημειωμένες για διαγραφή, διαγράφονται από το αρχείο. |
| Ctrl+M | Δίνεται η δυνατότητα για αλλαγή της δομής του αρχείου. |
| Ctrl+F | Δίνεται η δυνατότητα αναζήτησης στις εγγραφές του αρχείου. |
| Ctrl+R | Δίνεται η δυνατότητα να γίνει αναζήτηση και αντικατάσταση. |
| View Menu | Shift+F | Εμφανίζεται ένα dialog μέσα από το οποίο μπορεί να γίνει σύνθετη αναζήτηση, διαγραφή και επαναφορά εγγραφών. |
| Navigate Menu | Ctrl+Down | Γίνεται μετάβαση στην αμέσως επόμενη εγγραφή. |
| Ctrl+Up | Γίνεται μετάβαση στην αμέσως προηγούμενη εγγραφή. |
| Alt+Up | Γίνεται μετάβαση στην πρώτη εγγραφή. |
| Alt+Down | Γίνεται μετάβαση στην τελευταία εγγραφή. |
| Help Menu | F1 | Εμφανίζεται η βοήθεια της εφαρμογής. |

# Συμπεράσματα-Επίλογος

Η πτυχιακή εργασία *Ανάπτυξη βιβλιοθήκης χειρισμού αρχείων dBase με τη γλώσσα Java,* ήταν μια εργασία απαιτητική και επίπονη. Χρειάστηκαν γνώσεις από όλα τα μαθήματα προγραμματισμού, από τα μαθήματα βάσεων δεδομένων καθώς και από τα μαθήματα της μηχανικής λογισμικού για την διαχείριση του κώδικα. Κατά την ανάπτυξη της βιβλιοθήκης συνάντησα αρκετά προβλήματα κάποια από τα οποία δεν έχω λύσει. Ένα σημαντικό πρόβλημα το οποίο δεν έχει λυθεί είναι η υποστήριξη των κωδικοποιήσεων. Αυτή τη στιγμή η βιβλιοθήκη υποστηρίζει μόνο λατινικούς χαρακτήρες, γιατί δεν μπόρεσα να βρω κάποιο τρόπο αναγνώρισης της κωδικοποίησης των δεδομένων κατά το άνοιγμα του αρχείου. Επίσης, ένα μεγάλο πρόβλημα που έχει η εφαρμογή και δεν το έχω λύσει είναι η απόδοσή της κατά την φόρτωση αρχείων με πολλές εγγραφές(πάνω από 100.000). Κατά τη φόρτωση τέτοιων αρχείων το πρόγραμμα δεσμεύει πάρα πολύ μνήμη σε σημείο που να εξαντλεί την μνήμη την οποία μπορεί να προσφέρει η JVM. Αρχικά η σκέψη ήταν να φορτώνω τα δεδομένα στο JTable τμηματικά π.χ 10.000 εγγραφές κάθε φορά. Σε αυτή την περίπτωση όμως διαπίστωσα ότι θα υπήρχε πρόβλημα κατά τις διαδικασίες της αναζήτησης, της ταξινόμησης και της εξαγωγής δεδομένων καθώς οι λειτουργίες αυτές δεν θα εκτελούνταν πάνω στο σύνολο των δεδομένων αλλά σε ένα τμήμα τους (π.χ στις πρώτες 10.000 εγγραφές). Μια άλλη σκέψη ήταν να φορτώνω αρχικά όλα τα δεδομένα σε μια βάση πχ MySql, αλλά αυτό θα είχε ως αποτέλεσμα καθυστέρηση στο φόρτωμα της εφαρμογής.

Τελειώνοντας, παρά τα όποια προβλήματα συνάντησα στη δημιουργία της βιβλιοθήκης και της εφαρμογής, θεωρώ ότι η ενασχόληση με αυτή την πτυχιακή εργασία με βοήθησε να κατανοήσω καλύτερα τα προβλήματα τα οποία συναντάει κάποιος στις βάσεις δεδομένων, όπως είναι η συνέπεια της βάσης μετά από εγγραφή ή update εγγραφών καθώς και τα προβλήματα που υπάρχουν στην δημιουργία και διαχείριση ενός αρκετά μεγάλου λογισμικού. Τέλος, με την μελέτη του συστήματος *dBase*, το οποίο ήταν ένα από τα πρώτα ολοκληρωμένα συστήματα διαχείρισης δεδομένων, μου δόθηκε η ευκαιρία να δω με ποιο τρόπο ξεκίνησε η διαχείριση και η αποθήκευση μεγάλου όγκου δεδομένων πριν ακόμα την δημιουργία των σχεσιακών βάσεων δεδομένων.

# Βιβλιογραφία

 Deitel Paul J.,Deitel Harvey M.,( 2010), Java™ how to program Eighth Edition

Herbert Schildt(2005), Java™: A Beginner’s Guide, Third Edition

<http://stackoverflow.com/questions/7693994/how-to-convert-ascii-code-0-255-to-the-associated-character>

<http://javarevisited.blogspot.gr/2011/08/enum-in-java-example-tutorial.html>

<http://stackoverflow.com/questions/3954616/java-look-and-feel-lf>

<http://stackoverflow.com/questions/258099/how-to-close-a-java-swing-application-from-the-code>

<http://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html>

<http://stackoverflow.com/questions/1675912/how-to-close-a-modal-jdialog-when-user-clicks-outside-of-jdialog>

<http://stackoverflow.com/questions/1301874/how-do-i-get-a-reference-to-the-currently-displayed-java-swing-jdialog-box>

<http://stackoverflow.com/questions/9477354/how-to-allow-introducing-only-digits-in-jtextfield>

<http://stackoverflow.com/questions/5884353/insert-a-character-in-a-string-at-a-certain-position>

<http://www.dbf2002.com/dbf-file-format.html>

<http://www.clicketyclick.dk/databases/xbase/format/dbf.html>

<http://ulisse.elettra.trieste.it/services/doc/dbase/DBFstruct.htm>

<http://msdn.microsoft.com/en-us/library/st4a0s68(v=vs.80).aspx>

<http://www.delphipages.com/forum/showthread.php?t=122342>

<http://www.promotic.eu/en/pmdoc/Subsystems/Db/dBase/DataTypes.htm>

<http://www.cs.cmu.edu/~varun/cs315p/xbase.txt>

<http://msdn.microsoft.com/en-us/library/t2704cs4(v=vs.80).aspx>

<http://www.dbase.com/help/IDE/IDH_IDE_MODIFY_STRUCTURE.htm>

<http://stackoverflow.com/questions/2408541/align-the-values-of-the-cells-in-jtable>

<http://stackoverflow.com/questions/7391877/how-to-add-checkboxes-to-jtable-swing>

<http://www.java-forums.org/awt-swing/29040-programmatically-starting-cell-editing-jtable.html>

<http://www.java-tips.org/java-se-tips/javax.swing/sorting-and-filtering-tables.html>

<http://www.coderanch.com/t/458410/GUI/java/RowFilter-dont-show-rows-JTable>

<http://docs.oracle.com/javase/tutorial/uiswing/components/table.html#sorting>

<http://stackoverflow.com/questions/2131702/populating-empty-tables-in-java-swing>

<http://stackoverflow.com/questions/3549206/how-to-add-row-in-jtable>

<http://www.java2s.com/Tutorial/Java/0240__Swing/ListeningtoJTableEventswithaTableModelListener.htm>

<http://stackoverflow.com/questions/9919230/disable-user-edit-in-jtable>

<http://stackoverflow.com/questions/1117888/how-to-remove-a-row-from-jtable>

<http://mycodepage.blogspot.gr/2006/09/how-to-create-double-click-event-on.html>

<http://rickproctor.wordpress.com/2006/08/29/exporting-jtable-data-to-excel/>

<http://tips4java.wordpress.com/2008/12/12/table-stop-editing/>

<http://www.roseindia.net/java/example/java/swing/RemoveColumn.shtml>

<http://www.snippetit.com/2009/11/java-loading-large-data-into-jtable-or-jlist/>

<http://stackoverflow.com/questions/856888/getting-selected-row-through-abstracttablemodel>

<http://stackoverflow.com/questions/1517396/moving-a-row-in-jtable>

<https://forums.oracle.com/thread/1350854>

<http://stackoverflow.com/questions/1936566/how-do-you-get-the-screen-width-in-java>

<http://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html>

<http://stackoverflow.com/questions/3717936/how-to-add-jmenubar-shortcuts>

<http://stackoverflow.com/questions/11893345/set-jmenubars-background-color>

<http://stackoverflow.com/questions/16254640/setselectionbackground-doesnt-set-color-for-custom-rendered-cells>

<http://www.leghumped.com/2008/02/08/java-close-a-jframe-with-escape/>

<http://stackoverflow.com/questions/1726525/is-there-a-way-to-collapse-all-code-blocks-in-eclipse>

<http://docs.oracle.com/javase/tutorial/uiswing/components/formattedtextfield.html>

<http://stackoverflow.com/questions/3263892/format-file-size-as-mb-gb-etc>

<http://stackoverflow.com/questions/4795586/how-can-i-check-which-jtable-cell-has-been-clicked-selected>

<http://stackoverflow.com/questions/9289617/eclipse-how-to-link-a-jar-containing-javadocs-source-with-its-binary>

<http://stackoverflow.com/questions/8112657/drawing-border-around-jlabel-when-selected-like-the-buttons>

<http://poi.apache.org/spreadsheet/quick-guide.html#CreateCells>

<http://stackoverflow.com/questions/4033190/autogenerate-function-comments-in-eclipse>

<https://sites.google.com/site/ramprosoftcodelibrary/java-code-library/checkboxes-in-jlist>

<http://stackoverflow.com/questions/7766944/having-several-jcomboboxes-with-different-values-in-a-jtable>

<http://stackoverflow.com/questions/9609673/date-only-comparison-using-isbefore-in-java-joda-time>

<http://stackoverflow.com/questions/2523284/java-string-replace-and-the-nul-null-ascii-0-character>

<http://stackoverflow.com/questions/388461/how-can-i-pad-a-string-in-java>

<http://www.coderanch.com/t/391314/java/java/read-write-binary-file>

<http://stackoverflow.com/questions/3597704/java-io-filenotfoundexception-when-using-randomaccessfile-to-create-file>

<http://stackoverflow.com/questions/2885173/java-how-to-create-and-write-to-a-file>

<http://stackoverflow.com/questions/6142901/how-to-create-a-file-in-a-directory-in-java>

<http://www.webdeveloper.com/forum/showthread.php?56818-Getting-next-highest-multiple-of-10-for-a-number>

<http://stackoverflow.com/questions/652161/how-do-i-convert-between-iso-8859-1-and-utf-8-in-java>

<http://stackoverflow.com/questions/5597671/how-to-convert-joda-localdate-to-java-util-date>

<http://stackoverflow.com/questions/10630738/how-to-set-strings-font-size-style-in-java-using-font-class>

<http://stackoverflow.com/questions/3148535/how-to-read-excel-cell-having-date-with-apache-poi>

<http://stackoverflow.com/questions/5148995/problem-making-a-jprogressbar-update-values-in-loop-threaded>

<http://stackoverflow.com/questions/4637215/can-a-progress-bar-be-used-in-a-class-outside-main/4637725#4637725>

<http://stackoverflow.com/questions/691647/why-does-my-xpath-expression-in-java-return-too-many-children>

<http://tunatore.wordpress.com/2011/05/20/how-to-use-xpath-i-java-simple-example/>

<http://stackoverflow.com/questions/2081159/launching-a-url-in-a-java-swing-application>

# Παράρτημα

Τα παρακάτω κομμάτια κώδικα δείχνουν πως μπορούν να εκτελεστούν βασικές λειτουργίες σε ένα αρχείο dBase(.dbf), καλώντας μεθόδους της βιβλιοθήκης χειρισμού αρχείων dBase.

*Δημιουργία αρχείου dBase:*

DbaseFile odbf = **new** DbaseFile();

odbf.Open(“temp.dbf”);

*Προσθήκη πεδίων*

odbf.getHeader().AddColumn(**new** Column(“String”, Column.*GetDbaseType*(‘C’),

15, 0));

odbf.getHeader().AddColumn(**new** Column(“Numeric”, Column.*GetDbaseType*(‘N’),

10, 5));

odbf.getHeader().AddColumn(**new** Column(“Date”, Column.*GetDbaseType*(‘D’)));

odbf.getHeader().AddColumn(**new** Column(“Logical”, Column.*GetDbaseType*(‘L’)));

odbf.getHeader().Unlock();

odbf.WriteHeader();

*Προσθήκη εγγραφής:*

Record orec = **null**;

orec = **new** Record(*odbf*.getHeader());

orec.setValueAtColumn(0, “First Record” ,*odbf*.getFilePath());

orec.setValueAtColumn(1, “123.32” ,*odbf*.getFilePath());

orec.setValueAtColumn(2, “20130915” ,*odbf*.getFilePath());

orec.setValueAtColumn(3, “T” ,*odbf*.getFilePath());

*odbf*.Write(orec, **true**);

*odbf*.getHeader().Unlock();

*odbf*.WriteHeader();

*Σημείωση εγγραφής ως διαγραμμένη*

Record orec = **null**;

orec = **new** Record(*odbf*.getHeader());

orec.setRecordIndex(0);

**try**

{

**if**(!orec.getIsDeleted(orec.getRecordIndex(), *odbf*.mDbfFileReader))

{

orec.setIsDeleted(**true**, 0, *odbf*.mDbfFileReader, *odbf*.mDbfFileWriter);

}

}

**catch** (IOException e) {}

*Διαγραφή εγγραφών που είναι σημειωμένες για διαγραφή:*

odbf.packDBF();

# Αναφορές

1. Μια database engine, είναι ένα underlying συστατικό λογισμικού μέσω του οποίου τα συστήματα διαχείρισης βάσεων δεδομένων, δημιουργούν, διαβάζουν, ενημερώνουν και διαγράφουν δεδομένα από τη database. Τα περισσότερα συστήματα διαχείρισης δεδομένων έχουν ένα δικό τους API, το οποίο επιτρέπει τους χρήστες να αλληλεπιδράσουν με τη database engine χωρίς να χρειάζεται να έχουν πρόσβαση στο User Interface. [↑](#endnote-ref-2)
2. Μία User-Defined Function (UDF), είναι μια μέθοδος την οποία προσφέρει ένας χρήστης του προγράμματος, σε ένα πλαίσιο όπου συνήθως αυτοί οι μέθοδοι είναι ενσωματωμένες στο λογισμικό. Στις βάσεις δεδομένων μια UDF μέθοδος, προσφέρει ένα μηχανισμό για επέκταση της λειτουργικότητας της database engine προσθέτοντας νέες λειτουργίες. [↑](#endnote-ref-3)
3. Το μέγιστο μήκος στο όνομα του πεδίου είναι 10 χαρακτήρες (Ίδια ονόματα δεν γίνονται δεκτά).

   Ο τύπος ‘Character’ μπορεί να έχει μέγιστο μήκος πεδίου ‘Field Length’ 255.

   Ο τύπος ‘Numeric’ μπορεί να έχει μέγιστο μήκος πεδίου ‘Field Length’ 19 και μέγιστο πλήθος δεκαδικών ‘Decimals’ 15.

   Επίσης το πλήθος των δεκαδικών ‘Decimals’, πρέπει να είναι μικρότερο, κατά δύο(2) τουλάχιστο μονάδες από το μήκος του πεδίου ‘Field Length’. [↑](#endnote-ref-4)