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

Πτυχιακή εργασία

Ανάπτυξη ενός αντικειμενοστρεφούς συστήματος ενός πολιτιστικού συλλόγου(φιλαρμονικής) με έμφαση στην εκπαιδευση και την ανάπτυξη με τη μέθοδο RUP

Του φοιτητή

Παπαδημητρίου Γουλιέλμου
Αρ. Μητρώου: 03/2354

Επιβλέπων καθηγητής
Ιγνάτιος Δεληγιάννης

Θεσσαλονίκη 2010

Πρόλογος

Λίγοι θα διαφωνήσουν με τη θέση ότι το λογισμικό αποτέλεσε ένα από τα σημαντικότερα εργαλεία που ο άνθρωπος κατασκεύασε τον αιώνα που πέρασε, χάρη στο οποίο έγινε δυνατή η επανάσταση της πληροφορικής και των επικοινωνιών την οποία ακόμη βιώνουμε. Είναι αλήθεια ότι πολλοί άνθρωποι έχουν έρθει εκούσια ή ακούσια στη θέση του χρήστη μιας ή περισσοτέρων εφαρμογών λογισμικού, ωστόσο δεν είναι αυτονόητο ότι όλοι αντιλαμβάνονται το Λογισμικό ως ένα μοναδικό τεχνικό κατασκεύασμα. Συνήθως, το λογισμικό γίνεται αντιληπτό ως εργαλείο, ως παιχνίδι, ή γενικότερα ως μέρος ενός μεγαλύτερου και πιο σύνθετου συστήματος, σπάνια όμως το ίδιο ως σύνθετο κατασκεύασμα με τα δικά του προβλήματα. Η εμπειρία δείχνει ότι η κατασκευή καλού λογισμικού δεν είναι καθόλου απλή υπόθεση. Συνήθως πολλές παρανοήσεις χαρακτηρίζουν αυτό που γίνεται αντιληπτό ως "κατασκευή λογισμικού" και ο αναγνώστης αναμφίβολα και αναπόφευκτα θα συναντήσει σημαντικό πλουραλισμό απόψεων κατά την ενασχόλησή του με το θέμα. Πέραν από τα καθαρά τεχνικά ζητήματα που αντιμετωπίζει ο κατασκευαστής λογισμικού, πολλά από τα επιθυμητά χαρακτηριστικά αυτού ή και της διαδικασίας κατασκευής του, οφείλονται στην υπόστασή του ως προϊόντος.

Στην πτυχιακή που θα ακολουθίσει θα ασχοληθούμε με την ανάπτυξη ενός λογισμικού - εφαρμογής οπου από τις απαιτήσεις θα φτάσουμε μέχρι το στάδιο της ανάπτυξης εφαρμογής μέχρι ότου να γίνει λειτουργικό.Σκοπός αυτού του προγράμματος είναι η ανάπτυξη μιας εφαρμογής που θα βοηθήσει έναν αρχιμουσικό να έχει μια εικόνα της φιλαρμονικής που διευθύνει.

Περίληψη

Ο σκοπός της πτυχιακής εργασίας είναι η ανάλυση ο σχεδιασμός και τέλος η υλοποίηση μιας εφαρμογής διαχείρισης φιλαρμονικής. Η ανάλυση και ο σχεδιασμός θα γίνει με τη μεθοδολογία RUP.
Ξεκινάμε από τις περιπτώσεις χρήσης και με βάση αυτές προχωράμε στην ανάπτυξη του προγράμματος. Στη συνέχεια θα αναφερθούμε στο εννοιολογικό μοντέλο του συστήματος μας εξηγώντας τις έννοιες και τα χαρακτηριστικά του. Επόμενο μας βήμα θα είναι η δημιουργία του ssd μοντέλου. Ώστε να γίνει κατανοητό η αλληλεπίδραση του χρήστη με το σύστημα. Επόμενο βήμα μας είναι η ανάθεση λειτουργιών στις κλάσεις. Ακολουθεί η το διάγραμμα ακολουθίας του συστήματος μας. Όπου βλέπουμε στη μονάδα του χρόνου τις αλληλεπιδράσεις μεταξύ των αντικειμένων. Κλείνοντας την σχεδίαση ολοκληρώνουμε με το διάγραμμα κλάσεων Όπου εμφανίζουμε τις κλάσεις με χαρακτηριστικά και τις μεθόδους.
Κάθε κεφάλαιο του προγράμματος είναι και μια φάση ανάπτυξης. Σε κάθε κεφάλαιο έχουμε το θεωρητικό κομμάτι και το πρακτικό. Το πρακτικό είναι η εφαρμογή της φάσης ανάπτυξης στο πρόγραμμα μας.
Όταν ολοκληρωθεί η ανάλυση και ο σχεδιασμός προχωράμε στη φάση της υλοποίησης του προγράμματος μας. Σε αυτή τη φάση παίρνουμε βλέποντας και έχοντας ως οδηγό την σχεδίαση που κάναμε κωδικοποιούμε το πρόγραμμα.

Abstract

The purpose of this thesis is to analyze, to design and finally to implement the management Philharmonic application. The analysis and the design will be executed with the RUP methodology.

Starting from use cases we will proceed to develop the program. Then we will refer to the conceptual model of our system by explaining concepts and features.
Our next step is to establish the SSD model for understanding user's interaction with the system and to allocate operations to classes.

Below is a sequence diagram of our system where we can see that the unit "time" interacts between objects.
To finalize the design, we have to complete the diagram with the classes which will show us the features of the classes and methods.

Each chapter of the program is a development stage. In each chapter there is the theoretical and practical part. The report is the application development phase in our program.
Once the analysis and the design are finished we continue with the implementation phase of our program.
At this stage we have as guidance the design we made to encode the program.

Περιεχόμενα

Κεφάλαιο 4

Κεφάλαιο 5

Κεφάλαιο 6

Κεφάλαιο 7

Κεφάλαιο 8

Πίνακας Εικόνων-Διαγραμμάτων

Εισαγωγή

Ο σκοπός αυτής της πτυχιακής εργασίας είναι η ανάλυση ο σχεδιασμός και τέλος ο προγραμματισμός μιας εφαρμογής που ως σκοπό θα έχει την οργάνωση μιας φιλαρμονικής πολιστικού συλλόγου.

Παράλληλα όμως δίνεται βάση και στην επεξήγηση ώστε μέσα από την ανάπτυξη αυτής της εφαρμογής να μπορεί κάποιος να καταλάβει τα στάδια και τους τρόπους ανάλυσης σχεδιασμού μιας εφαρμογής. Κάθε κεφάλαιο συνοδεύεται από ένα θεωρητικό κομμάτι ώστε να γίνει κατανοητό στον αρχάριο η όλη διαδικασία..

Στο πρώτο κεφάλαιο θα ασχοληθούμε με ορισμούς και να εξηγήσουμε το τι ειναι αντικειμενοστραφής προγραμματισμός, τι είναι ανάλυση σχεδίαση, γιατί είναι απαραίτητη η ανάλυση και ο σχεδιασμός μιας εφαρμογής, ποιες είναι οι αδυναμίες τις δομημένης ανάλυσης σχεδίασης. Επίσης θα γίνει αναφορά στις βασικές έννοιες που θα συναντήσουμε.

Στο δεύτερο κεφάλαιο θα έχουμε την περιγραφή του τρόπου λειτουργίας της φιλαρμονικής. Θα περιγράφει πώς λειτουργεί μια φιλαρμονική από τι αποτελείται ποιες είναι οι λειτουργίες της. Ωστε να γίνει κατανοητός ο τρόπος λειτουργίας της.

Στο ίδιο κεφάλαιο θα γίνει περιγραφή το τι είναι μια περίπτωση χρήσης τι μας εξυπηρετεί και τον τρόπο γραφής της. Και τέλος θα γίνει ανάλυση και δημιουργία των περιπτώσεων χρήσης του συστήματος μας.Θα αποφασίσουμε ποιες λειτουργίες θα ικανοποιεί το σύστημα και τα στοιχεία που θα κρατάει το σύστημα μας.Καθώς επίσης θα σχεδιάσουμε και το διάγραμμα περιπτώσεων χρήσης.

Στο τρίτο κεφάλαιο θα ασχοληθούμε με το εννοιολογικό μοντέλο. Θα αναφέρουμε τι είναι πού μας εξυπηρετεί και πώς δημιουργείται. Υπάρχει η ανάλυση του εννοιολογικού μοντέλου περιγραφούν ποιες και ποιος ο ρόλος των βασικών εννοιών του συστήματος μας καθώς και η περιγραφή των σχέσεων των εννοιών.

Στο τέταρτο κεφάλαιο θα ασχοληθούμε με το διάγραμμα ακολουθίας συστήματος(ssd). Θα περιγράψουμε τα μηνύματα που υπάρχουν ανάμεσα στο στον χρήστη και το σύστημα. Θα αναφέρουμε ποιος είναι ο στόχος του πώς τα δημιουργούμε.

Στο πέμπτο κεφάλαιο θα αναφερθούμε στα grasp Paterrns και τους ευρετικούς κανόνες.Πιο συγκεκριμένα θα περιγράψουμε ποιο grasp χρησιμοποιήσαμε και γιατί, ποια προβλήματα λύσαμε με τη χρήση του. Επίσης θα περιγράψουμε τη χρήση του mvc μοντέλου ανάπτυξης και γιατί κάναμε χρήση του στο προγραμμα μας.

Στο έκτο κεφάλαιο θα ασχοληθούμε με τα τα διαγράμματα Αλληλεπίδρασης. Βάρος θα δώσουμε στην θα ανάλυση των διαγραμμάτων ακολουθίας. θα εξηγήσουμε πώς το λόγο ύπαρξης του πώς δημιουργείται και ποια τα χαρακτηριστικά τους. Τέλος θα σχεδιάσουμε τα διαγράμματα ακολουθίας για κάθε περίπτωση χρήσης.

Στο έβδομο κεφάλαιο θα αναφερθούμε στο διάγραμμα κλάσεων. Θα γίνει περιγραφή των βασικών εννοιών του διαγράμματος, θα γίνει μια περιγραφή για τις συσχετίσεις μεταξύ των κλάσεων. Τέλος θα σχεδιάσουμε το διάγραμμα κλάσεων της φιλαρμονικής.

Από αυτό το κεφάλαιο μπαίνουμε στο μέρος της κωδικοποίησης του προγράμματος μας. Δημιουργούμε τον κώδικα βλέποντας τα σημαντικότερα μέρη. Χωρίζουμε τον κώδικα σε τρία μέρη(μοντέλα,controllers,views). Οπου αναλύουμε και περιγράφουμε τα βασικά μέρη – σημαντικότερα σημεία του κώδικα για κάθε μερος. Επίσης Θα υπάρχουν και screenshots απο το interface της εφαρμογής όπως επίσης και ο τρόπος δημιουργίας τους.

Τέλος υπάρχει το εγχειρίδιο χρήσης που παρουσιάζεται ο τρόπος λειτουργίας του προγράμματος.

Κεφάλαιο 1
Αντικειμενοστρεφή ανάπτυξη και σχεδιασμός

1.1 Αδυναμίες δομημένης ανάλυσης σχεδίασης

Ποιο από τα χαρακτηριστικά των εφαρμογών λογισμικού καθιστά ανεπαρκή την προσέγγιση τους με τη δομημένη ανάλυση και σχεδίαση; Η απάντηση στο ερώτημα αυτό αποτελεί κλειδί για την αναζήτηση και την κατανόηση μιας νέας φιλοσοφίας προσέγγισης του λογισμικού. Μπορεί κανείς να ανατρέξει εκτενώς στη βιβλιογραφία προκειμένου να βρει διάφορες απαντήσεις στο ερώτημα. Εμείς θα επιλέξουμε την μονολεκτική απάντηση «η πολυπλοκότητα». Η πολυπλοκότητα αφορά όχι μόνο την καθαρά υπολογιστική εργασία του λογισμικού, αλλά και την ανάγκη διαχείρισης μεγάλου όγκου δεδομένων. Μια εκτενής συζήτηση σχετικά με το θέμα περιέχεται στο βιβλίο του Booch, «Object Oriented Analysis and Design with applications», το οποίο

προτείνεται στη βιβλιογραφία. Σημεία στα οποία η πολυπλοκότητα του λογισμικού κάνει φανερές τις αδυναμίες της δομημένης ανάλυσης και σχεδίασης, μπορούν ν' αναζητηθούν σε δύο κατευθύνσεις: σ' αυτή που αφορά το θεωρητικό και σ' αυτή που αφοράτο πρακτικό μέρος αυτής.

Στο θεωρητικό επίπεδο, η δομημένη ανάλυση και σχεδίαση παρουσιάζει μια εγγενή αδυναμία στην απεικόνιση των οντοτήτων του πραγματικού κόσμου σε συστατικά λογισμικού. Επιχειρεί να αποσυνθέσει ένα πρόβλημα σε μια ιεραρχία συστατικών λογισμικού, καθένα εκ των οποίων επιτελεί ένα μικρό μέρος της λύσης του. Αυτή η εφαρμογή της αρχαίας ρήσης «διαίρει και βασίλευε», έχει νόημα όταν η λύση του προβλήματος είναι μόνο υπόθεση αριθμητικών υπολογισμών. Στον πραγματικό κόσμο, όμως, και ιδιαίτερα στο πεδίο εφαρμογών λογισμικού που σχετίζονται με την επιχειρηματική δραστηριότητα, αυτή η προσέγγιση υστερεί για δύο λόγους:

• Πρώτον, δε λάμβανε υπόψη τα δεδομένα, τα οποία έχουν τη δική τους πολύπλοκη δομή και εξαρτήσεις. Τα δεδομένα στη δομημένη ανάλυση και

σχεδίαση είναι ανεξάρτητα από τις λειτουργικές μονάδες που επιδρούν σ' αυτά, πράγμα που δεν ισχύει στο επίπεδο του πραγματικού κόσμου, όπου η διαχείριση δεδομένων δεν είναι ανεξάρτητη από αυτά.

• Δεύτερον, το υπολογιστικό μοντέλο που αποτελείται από το δίδυμο «συστατικά λογισμικού» και «ανεξάρτητα δεδομένα», δεν αντιστοιχεί σε οντότητες του πραγματικού κόσμου, δηλαδή δεν παριστά οντότητες που είναι αντιληπτές στον πραγματικό κόσμο. Με μια άλλη διατύπωση, δεν μοντελοποιεί εύκολα και φυσικά την επιχειρησιακή λογική (businesslogic). Ως εκ τούτου, είναι ανεπαρκές ως μεθοδολογικό εργαλείο για τη δόμηση λογισμικού.

Σε πρακτικό επίπεδο μπορούμε να εντοπίσουμε τα ακόλουθα προβλήματα:

• Η διαχείριση των μοντέλων παράστασης λογισμικού, είναι μια δύσκολη και επιρρεπής σε σφάλματα εργασία. Το πλήθος, η πολυπλοκότητα και οι συσχετίσεις των συστατικών αυτών, η μεταβολή τους με το χρόνο, αλλά και οι παρενέργειες κατά την πραγματοποίηση μεταβολών καθιστούν την εργασία αυτή ακόμα δυσκολότερη.

• Η συντήρηση του Λογισμικού έχει εξελιχθεί σε μια πολύ δύσκολη διαδικασία, που δυσκολεύει περισσότερο, καθώς αυξάνεται το μέγεθος του Λογισμικού. Έχει αναφερθεί ότι η συνεισφορά της συντήρησης στο συνολικό κόστος κατά τον κύκλο ζωής λογισμικού, μπορεί να ξεπεράσει το 50%.

• Μολονότι κατά την ανάπτυξη μιας νέας εφαρμογής ενδέχεται να αναγνωριστούν ομοιότητες με τα χαρακτηριστικά μιας υπάρχουσας, η επαναχρησιμοποίηση συστατικών λογισμικού που έχουν κατασκευαστεί με τη δομημένη ανάλυση και σχεδίαση δεν ενθαρρύνεται. Αυτό ισχύει στη γενική περίπτωση, διότι η πολυπλοκότητα και η φύση των συσχετίσεων ενός συστατικού λογισμικού – μέρους μιας εφαρμογής που κατασκευάστηκε με τη δομημένη φιλοσοφία, δεν επιτρέπουν τη γενίκευση και εύκολη επαναχρησιμοποίηση του. Ο κανόνας αυτός ισχύει ιδιαίτερα για λογισμικό που χρησιμοποιείται σε επιχειρηματικές εφαρμογές και γενικά σε εφαρμογές που σχετίζονται με τη διαχείριση δεδομένων, ενώ εξαίρεση αποτελούν οι βιβλιοθήκες μαθηματικών συναρτήσεων που αφορούν καθαρά υπολογιστικές εργασίες.

Όσα αναφέρθηκαν, με κανένα τρόπο δε σημαίνουν δύο πράγματα: πρώτον, ότι η δομημένη ανάλυση και σχεδίαση είναι «άχρηστη», «εσφαλμένη» ή κάτι τέτοιο και δεύτερον, ότι η αντικειμενοστρεφής φιλοσοφία έχει τη λύση σε όλα τα προβλήματα

της ανάπτυξης του λογισμικού. Όπως εξάλλου αναφέρθηκε, η συζήτηση αυτή έχει το νόημα της αντίληψης των εξελίξεων που εισήγαγαν μια νέα φιλοσοφία προσέγγισης του λογισμικού. Πολλά από τα προβλήματα που υπήρχαν στην ανάπτυξη του λογισμικού εξακολουθούν να υπάρχουν με την μία ή άλλη έννοια, ενδεχομένως σε μικρότερο βαθμό. Η αντικειμενοστρεφής τεχνολογία είναι μόνο ένα καλύτερο εργαλείο στα χέρια του ανθρώπου για τη λύση των προβλημάτων, δεν είναι η ίδια η λύση των προβλημάτων.

1.2 Βασικές έννοιες αντικειμενοστρεφούς τεχνολογίας

Αξίζει να σημειώσουμε από την αρχή ότι η ιδέα της αντικειμενοστραφούς προσέγγισης υπήρχε από τη δεκαετία του '60 και η πρώτη γλώσσα που είχε

τέτοια χαρακτηριστικά ήταν η Simula–67. Τότε, κανείς σχεδόν δεν έδινε μεγάλη σημασία στις ιδέες που εισήγαγε η Simula: η κοινότητα του Λογισμικού είχε στραμμένη την προσοχή της στις ιδέες της δομημένης ανάλυσης/σχεδίασης. Άλλες γλώσσες προγραμματισμού που είχαν τέτοια στοιχεία ήταν Alphard, η CLU, η ADA (η οποία γνώρισε μεγάλη διάδοση διότι υποστηρίχτηκε από το υπουργείο άμυνας των ΗΠΑ) και η Smalltalk.

Η προσέγγιση της αντικειμενοστρεφούς τεχνολογίας μπορεί να γίνει από διάφορους εναλλακτικούς δρόμους. Την εποχή που παρατηρήθηκε έκρηξη

ενδιαφέροντος για το θέμα, δημοσιεύτηκαν πολλές θεωρητικές αναλύσεις με παρεμφερή ορολογία και με λιγότερο ή περισσότερο συγκεκριμένες αναφορές στις πρακτικές πλευρές της αντικειμενοστρεφούς προσέγγισης στον προγραμματισμό. Ο αναγνώστης παραπέμπεται στη βιβλιογραφία για εκτενέστερες αναφορές στο θέμα.

Ορισμοί

Μέχρι το σημείο αυτό χρησιμοποιήσαμε τον όρο αντικειμενοστρεφής ως μετάφραση του αγγλικού όρου object–oriented. Καιρός είναι να δώσουμε έναν περισσότερο σαφή ορισμό και να εισάγουμε την ορολογία που σχετίζεται με την

αντικειμενοστρεφή τεχνολογία. Θα προσεγγίσουμε το θέμα ξεκινώντας από την κορυφή της πυραμίδας.

Αντικειμενοστρεφής

Αντικειμενοστρεφής (ο, η), αντικειμενοστρεφές (το), είναι χαρακτηρισμός που σημαίνει «στραμμένος (προσανατολισμένος) σε αντικείμενα» και αποδίδεται σ' εκείνο τον τρόπο σκέψης κατά την ανάπτυξη λογισμικού, στον οποίο τα «αντικείμενα» είναι οι βασικές δομικές μονάδες. Θα σημειώσουμε ότι επιλέξαμε να χρησιμοποιήσουμε το επίθεμα «–στρεφής» και όχι «–στραφής», όπως αναφέρεται σ' άλλα σημεία στη βιβλιογραφία, σύμφωνα με τη λογική που αυτό χρησιμοποιείται στο δόκιμο «εσωστρεφής» (και όχι «εσωστραφής»), που σημαίνει «στροφή προς τον εαυτό».

Μια άλλη, κατά τη γνώμη μας άστοχη, απόδοση του όρου στην ελληνική γλώσσα είναι «αντικειμενικός», λέξη που υπήρχε ήδη στην ελληνική γλώσσα και σημαίνει κάτι εντελώς διαφορετικό. Ο παραπάνω ορισμός χρησιμοποίησε τον όρο αντικείμενο, ο οποίος θα πρέπει επίσης να οριστεί στο πλαίσιο του λογισμικού.

Αντικείμενο

Ένα αντικείμενο είναι ένα δομικό συστατικό λογισμικού. Κάθε αντικείμενο έχει **κατάσταση**, **συμπεριφορά** και **ταυτότητα**. Η **κατάσταση** περιγράφει όλες τις στατικές ιδιότητες του αντικειμένου, όπως τιμές σε μεταβλητές μνήμης. Οι ιδιότητες αυτές είναι το αποτέλεσμα της **συμπεριφοράς** του αντικειμένου, δηλαδή του τρόπου με τον οποίο αυτό ανταποκρίνεται σε κλήσεις από το περιβάλλον του. Η **ταυτότητα** είναι η μοναδική διάκριση του αντικειμένου από τα ομοειδή του.Από τεχνικής πλευράς, το αντικείμενο μπορεί να θεωρηθεί ως η συγχώνευση δύο εννοιών που κατέχουν δεσπόζουσα θέση στη δομημένη ανάλυση, σχεδίαση και προγραμματισμό: της εγγραφής (record στην Pascal) και της συνάρτησης ή διαδικασίας (function, procedure στην Pascal). Η εγγραφή είναι μια συλλογή μεταβλητών μνήμης, οι οποίες αποτελούν τη δομή της και κάθε στιγμή έχουν κάποια συγκεκριμένη τιμή Το σύνολο των τιμών των μεταβλητών μιας εγγραφής αποτελεί την κατάσταση αυτής. Αν στην εγγραφή, όπως την ορίσαμε μέχρι τώρα, προσθέσουμε ενεργά συστατικά λογισμικού, δηλαδή μονάδες προγράμματος που εκτελούν υπολογισμούς, τότε μιλάμε για αντικείμενο και όχι για εγγραφή. Δηλαδή, ένα αντικείμενο περικλείει ένα σύνολο δεδομένων και ένα σύνολο συναρτήσεων

που χειρίζονται τα δεδομένα αυτά και επιτελούν τις λειτουργίες του πεδίου ευθύνης αυτού.

Κλάση

Το σύνολο των αντικειμένων που έχουν την ίδια δομή και την ίδια συμπεριφορά, ονομάζεται κλάση. Θα πρέπει να διακρίνουμε μεταξύ της έννοιας δομή και της έννοιας κατάσταση. Η δομή χαρακτηρίζεται από το ποιες και τι τύπου μεταβλητές περιγράφουν τις ιδιότητες του αντικειμένου, ενώ η κατάσταση είναι ένα σύνολο συγκεκριμένων τιμών στις μεταβλητές αυτές. Η διατύπωση ίδια συμπεριφορά σημαίνει ίδια απόκριση στο ίδιο εξωτερικό ερέθισμα. Η κλάση είναι μια αφηρημένη έννοια, όπως αφηρημένη είναι η έννοια του τύπου (type) στις γλώσσες προγραμματισμού. Ο τύπος δεν είναι κάτι που υπάρχει την ώρα τηςεκτέλεσης ενός προγράμματος. Αυτό που υπάρχει είναι οι μεταβλητές μνήμης, καθεμία εκ των οποίων λέμε ότι «είναι» κάποιου τύπου. Μπορούμε, λοιπόν, να αναγνωρίσουμε αντιστοιχία μεταξύ των εννοιών «τύπος- μεταβλητή» και «κλάση – αντικείμενο»: Όπως ισχύει ότι «κάθε μεταβλητή μνήμης είναι κάποιου τύπου», έτσι ισχύει ότι «κάθε αντικείμενο ανήκει σε μία κλάση», η οποία καθορίζει πλήρως τη δομή και τη συμπεριφορά του. Με βάση τα παραπάνω, μια κλάση μπορεί να θεωρηθεί ως η περιγραφή της δομής και της συμπεριφοράς των αντικειμένων που ανήκουν σε αυτή, τα οποία είναι τα συστατικά στοιχεία λογισμικού που έχουν τη δομή και εκδηλώνουν τη συμπεριφορά.

Στιγμιότυπο, εκδοχή

Κάθε αντικείμενο αποτελεί ένα μοναδικό και συγκεκριμένο στιγμιότυπο ή εκδοχή (instance) της κλάσης στην οποία ανήκει.

Η κατάσταση ενός αντικειμένου καθορίζεται από τις μεταβλητές κατάστα-
σης, οι οποίες αντιστοιχούν στα ιδιώματά του.

Πεδίο

Ένα πεδίο (field) είναι μια μεταβλητή η οποία παριστάνει ένα χαρακτηριστικό γνώρισμα του αντικειμένου. Το σύνολο των τιμών όλων των πεδίων ενός αντικειμένου αποτελεί την κατάσταση αυτού. Ισοδύναμα χρησιμοποιείται και ο

όρος ιδίωμα (attribute).Η συμπεριφορά ενός αντικειμένου καθορίζεται από τον τρόπο με τον οποίο

αυτό αντιδρά σε εξωτερικά ερεθίσματα, δηλαδή σε κλήσεις από τα αντικεί-

μενα που αποτελούν το περιβάλλον του.

Μέθοδος

Μια μέθοδος είναι ένα ενεργό συστατικό λογισμικού (συνάρτηση, δια-

δικασία), η οποία υλοποιεί ένα στοιχείο συμπεριφοράς ενός αντικειμένου. Το σύνολο όλων των μεθόδων ενός αντικειμένου καθορίζει τη συμπεριφορά του. Μια χαρακτηριστική ιδιότητα των αντικειμένων, βασικό στοιχείο της αντικειμενοστρεφούς φιλοσοφίας, είναι η κελυφοποίηση (encapsulation) ή, όπως ισοδύναμα αναφέρεται στη βιβλιογραφία, η απόκρυψη πληροφοριών(information hiding). Στην ελληνική βιβλιογραφία η έννοια απαντάται και με τον όρο «ενθυλάκωση».

1.3 Ένα Γενικό πλαίσιο για την αντικειμενοστρεφή ανάπτυξη Λογισμικού

Η αυξανόμενη ζήτηση για ολοένα και περισσότερο πολύπλοκες εφαρμογές

λογισμικού έχει δημιουργήσει την απαίτηση ευέλικτων και αποτελεσματικών προσεγγίσεων στην ανάπτυξή του, όπως επισημάναμε στα δύο πρώτα κεφάλαια του βιβλίου αυτού. Μία από τις προσπάθειες να ικανοποιηθεί η απαίτηση αυτή είναι η Ενοποιημένη Προσέγγιση (Unified Software Development Methodology ή Unified Process), η οποία υποστηρίζει την ανάπτυξη λογισμικού σύμφωνα με την αντικειμενοστρεφή φιλοσοφία. Τα βασικά χαρακτηριστικά αυτής, είναι τα ακόλουθα:

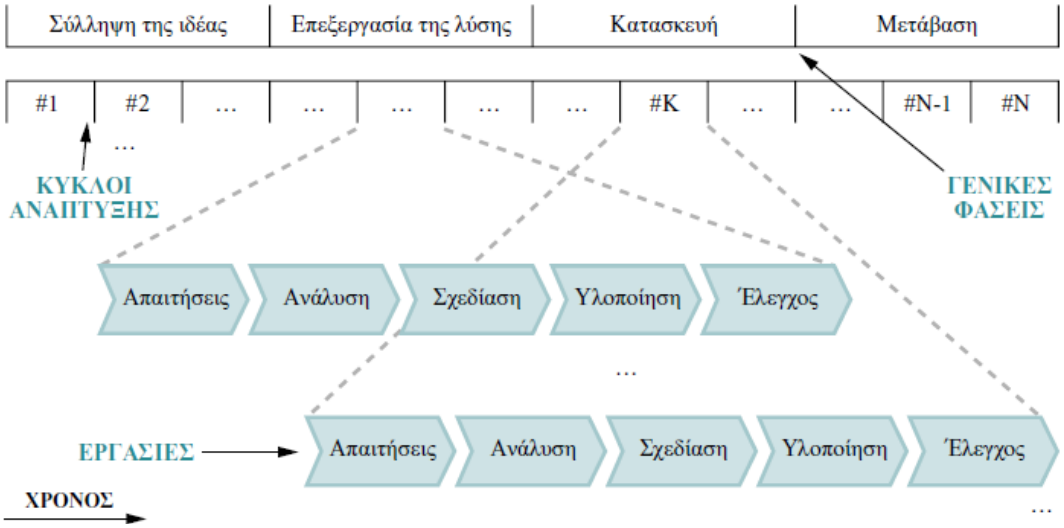
- Χρησιμοποιεί την UML για την παράσταση των μοντέλων λογισμικού που κατασκευάζονται κατά την ανάπτυξη.
- Αντιμετωπίζει το λογισμικό ως ένα σύνολο συστατικών που ικανοποιούν απαιτήσεις των χρηστών με αναφορά στις οποίες πραγματοποιεί όλες τις δραστηριότητες ανάπτυξης.
- Αντιμετωπίζει την αρχιτεκτονική του λογισμικού ως κεντρική έννοια στην ανάπτυξη, η οποία είναι δυναμικά αλληλεξαρτώμενη με τις απαιτήσεις των χρηστών, δηλαδή καθορίζεται από αυτές, αλλά και τις επηρεάζει.

• Είναι μια επαναληπτική και επαυξητική προσέγγιση, δηλαδή χτίζει το τελικό προϊόν ως συσσωρευτικό αποτέλεσμα επαναλήψεων δραστηριοτήτων ανάπτυξης λογισμικού.

Οι γενικές φάσεις που αναγνωρίζονται, είναι αυτές της σύλληψης της ιδέας, της επεξεργασίας της λύσης, της κατασκευής και της μετάβασης. Κάθε γενική φάση αναλύεται σε κύκλους ανάπτυξης και σε κάθε κύκλο ανάπτυξης λαμβάνουν χώρα οι εργασίες της προδιαγραφής των απαιτήσεων, της ανάλυσης,της σχεδίασης, της υλοποίησης και του ελέγχου. Στο κεφάλαιο αυτό, θα ασχοληθούμε με την περιγραφή των εργασιών της προδιαγραφής των απαιτήσεων και της ανάλυσης λογισμικού σύμφωνα με την Ενοποιημένη Προσέγγιση.

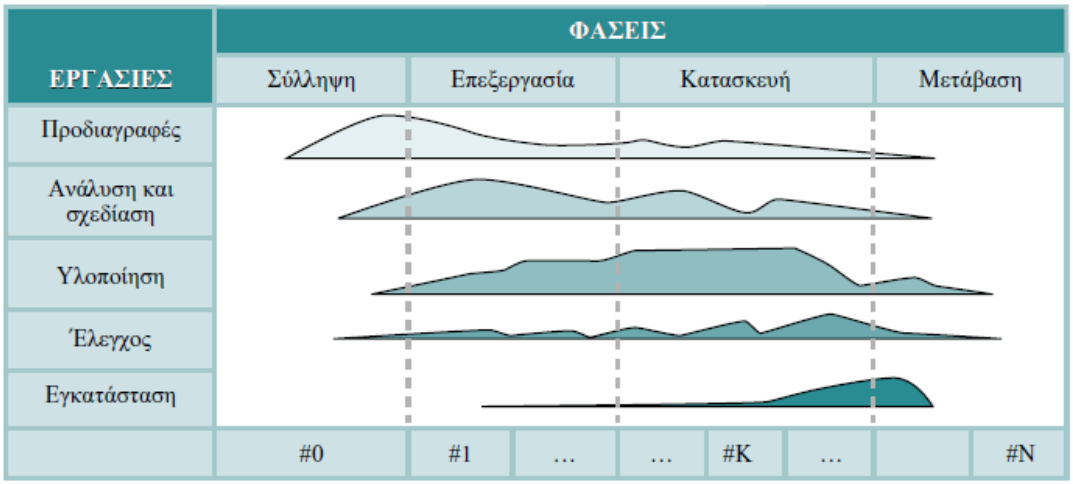
Κάθε κύκλος ανάπτυξης αφορά ένα υποσύνολο του λογισμικού υπό ανάπτυξη, το οποίο οριοθετείται είτε από το πλήθος των λειτουργιών που περιλαμβάνει,είτε από το βαθμό λεπτομέρειας με τον οποίο τις αντιμετωπίζει. Στο τέλος κάθε κύκλου ανάπτυξης έχουμε μια εκδοχή (release) του λογισμικού. Κατά τις πρώτες δύο φάσεις (σύλληψη της ιδέας και επεξεργασία της λύσης) η εκδοχή αυτή είναι ένα σύνολο από κείμενα και μοντέλα παράστασης λογισμικού, ενώ κατά τη φάση της κατασκευής η εκδοχή αυτή περιλαμβάνει πηγαίο και εκτελέσιμο κώδικα με ολοένα και περισσότερα από τα απαιτούμενα λειτουργικά χαρακτηριστικά του λογισμικού. Τέλος, κατά τη φάση της μετάβασης το λογισμικό τοποθετείται σε δοκιμαστική λειτουργία, όπου επαληθεύεται η ικανοποίηση των απαιτήσεων των χρηστών και γίνονται οι απαραίτητες διορθώσεις.

Ας σημειωθεί ότι δεν εκτελούνται όλες οι εργασίες (προδιαγραφή, ανάλυση,σχεδίαση, υλοποίηση, έλεγχος) σ' όλες τις γενικές φάσεις. Για παράδειγμα, σύμφωνα με όσα γνωρίζουμε από τη δομημένη ανάλυση και σχεδίαση, δεν είναι δυνατό κατά τη φάση της σύλληψης της ιδέας να εκτελεστεί οποιαδήποτε εργασία σχεδίασης λογισμικού. Αυτό είναι εν μέρει μόνο αληθές στην περίπτωση της αντικειμενοστρεφούς τεχνολογίας, όπου θα δούμε ότι σχετικά νωρίς στηνανάπτυξη εισάγονται έννοιες όπως «κλάση» και «συνεργασία», οι οποίες τελικά καταλήγουν να είναι πραγματικά κλάσεις και στο πεδίο της υλοποίησης.



Εικ. 1.1

Ενδέχεται, ασφαλώς, οι οντότητες που εντοπίζονται στις πρώτες φάσεις ανάπτυξης του λογισμικού να μην απεικονιστούν στη συνέχεια αυτούσιες σε κλάσεις. Αυτό, ωστόσο, δεν αναιρεί το γεγονός ότι από πολύ νωρίς στην αντικειμενοστρεφή ανάπτυξη εισάγονται όροι συστατικών στοιχείων υλοποίησης του λογισμικού. Έχοντας κατά νου τη σαφή διάκριση ανάλυσης και σχεδίασης η οποία ισχύει στη δομημένη προσέγγιση, παρατηρούμε ότι στην αντικειμενοστρεφή εκτελούμε νωρίς εργασίες που μπορούν να χαρακτηριστούν και ως εργασίες ανάλυσης και ως εργασίες σχεδίασης.



Εικ. 1.2

Στο Σχήμα φαίνεται η ποσότητα των πόρων που αποδίδονται στις εργασίες προδιαγραφής των απαιτήσεων, ανάλυσης–σχεδίασης, υλοποίησης, ελέγχου και

εγκατάστασης κατά τους κύκλους ανάπτυξης της Ενοποιημένης Προσέγγισης ανάπτυξης λογισμικού. Το εμβαδόν μεταξύ του οριζόντιου άξονα του χρόνου και κάθε καμπύλης, υποδηλώνει τους πόρους που ανατίθενται στην εκτέλεσή κάθε εργασίας, ή, ισοδύναμα, την ενέργεια που αποδίδεται σ' αυτή.

Είναι ίσως αντιληπτό ότι το διάγραμμα δεν περιέχει ποσοτικές πληροφορίες, αλλά σκοπεύει να μας δείξει ποιοτικά την αναλογία των εργασιών προδιαγραφής, ανάλυσης κλπ., σε κάθε κύκλο ανάπτυξης, καθώς και την παράλληλη εκτέλεση πολλών από αυτές. Παρατηρούμε ότι στις αρχικές φάσεις εκτελούνται περισσότερο εργασίες προδιαγραφής και λιγότερο ανάλυσης, σχεδίασης ή υλοποίησης. Όσο περνάμε σε κύκλους ανάπτυξης που ανήκουν στις επόμενες φάσεις, εκτελούνται περισσότερο κατασκευαστικές εργασίες ή έλεγχος και λιγότερο εργασίες προσδιορισμού των απαιτήσεων, οπότε μεταβάλλεται και το αντίστοιχο εμβαδόν.

Η Ενοποιημένη Προσέγγιση ανάπτυξης λογισμικού έχει το χαρακτηριστικό ότι σε περίπτωση που κατά την ανάπτυξη εντοπιστεί αδιέξοδο, το κόστος περιορίζεται σ' αυτό της επανάληψης του τελευταίου κύκλου και όχι ολόκληρης της ανάπτυξης μέχρι το σημείο εκείνο. Βέβαια, η ιδέα δεν είναι νέα και χαρακτηρίζει και άλλα μοντέλα κύκλου ζωής λογισμικού, ανεξάρτητα από το αν αυτά ακολουθούν τη δομημένη ή την αντικειμενοστρεφή προσέγγιση.

1.4 UML

Σήμερα όμως η ανάγκη για ένα καλά δομημένο διάγραμμα είναι επιτακτική. Ο πελάτης πρέπει να καταλάβει τι ακριβώς αλλαγές θα γίνουν στο σύστημα του. Θα πρέπει να προτείνει αλλαγές στο μέλλον. Το κάθε μέλος της ομάδας ανάπτυξης θα πρέπει να είναι σε θέση να καταλάβει πως θα πρέπει να λειτουργήσει το κομμάτι που αναπτύσσει και πώς επικοινωνεί με το υπόλοιπο σύστημα.

Τα υπολογιστικά συστήματα γίνονται ολοένα και πιο πολύπλοκα με βάσεις δεδομένων, δίκτυα, διαφορετικό υλικό σε διαφορετικές τοποθεσίες. Οπότε όλη αυτή η πολυπλοκότητα θα πρέπει να αντιμετωπιστεί μ κάποιο τρόπο και να γίνει κατανοητός ο τρόπος λειτουργίας σε όσους εμπλέκονται σε αυτό το σύστημα..

Αυτή την ανάγκη έρχεται να την καλύψει η UML Unified Modeling Language (Ενοποιημένη Γλώσσα Μοντελοποίησης).

Η UML δεν είναι τίποτα άλλο από μια γλώσσα. Δεν είναι ένας τρόπος σχεδίασης ενός συστήματος αλλά ένας τρόπος μοντελοποίησης του συστήματος. Η UML χρειάζεται να εφαρμόσουμε διαφόρους μεθόδους για την ανάλυση σχεδιασμό του συστήματος μας πχ(grapple).

Η UML αποτελείται από έναν αριθμό γραφικών στοιχείων που συνδυάζονται για να σχηματίσουν διαγράμματα. Τα διαγράμματα αυτά έχουν ως σκοπό να παρουσιάσουν διάφορες όψεις του συστήματος. Το σύνολο αυτών των όψεων το ονομάζουμε μοντέλο. Αυτό το μοντέλο παρουσιάζει το τι θα κάνει το σύστημα όχι όμως και το πώς θα υλοποιηθεί.

Παραδείγματα διαγραμμάτων UML

- Διαγράμματα συμπεριφοράς
    - Διαγράμματα περιπτώσεων χρήσης (use case diagrams)
    - Διαγράμματα καταστάσεων (state machine diagrams)
    - Διαγράμματα δραστηριοτήτων (activity diagrams)
    - Διαγράμματα επικοινωνίας (communication diagrams)
    - Διαγράμματα ακολουθίας (sequence diagrams)
- Δομικά διαγράμματα
    - Διαγράμματα κλάσεων (class diagrams)
    - Διαγράμματα αντικειμένων (object diagrams)
    - Διαγράμματα πακέτων (package diagrams)
- Αρχιτεκτονικά διαγράμματα
    - Διαγράμματα εξαρτημάτων (component diagrams)
    - Διαγράμματα ανάπτυξης (deployment diagrams
    - Διαγράμματα συστατικών (component diagrams)
    - Διαγράμματα συνεργασίας (collaboration diagrams)

Κεφάλαιο 2

Περιγραφή Φιλαρμονικής και Περιπτώσεις Χρήσης.

2.1 Περιγραφή Λειτουργίας Φιλαρμονικής

Το σύστημα έχει ως σκοπό την διευκόλυνση ενός αρχιμουσικού στην οργάνωση της φιλαρμονικής. Ένα σύστημα όπου θα αποθηκεύονται τα βασικά στοιχεία της φιλαρμονικής. Οι μουσικοί με τα χαρακτηριστικά τους, οι παρτιτούρες που υπάρχουν, μουσικά όργανα που έχει στην κατοχή της, στοιχεία των αρχιμουσικών που έχουν περάσει, τους εκάστοτε υπεύθυνους (ορίζονται από την δημοτική αρχή),τις εμφανίσεις με τα άτομα που συμμετείχαν, τα κομμάτια που παίχτηκαν κτλ , τις πρόβες που έγιναν.

Όταν εγγράφεται κάποιος στη φιλαρμονική θα πρέπει να δηλώσει κάποια στοιχεία. Το ονοματεπώνυμο, διεύθυνση, ημερομηνία γέννησης, τηλέφωνα, τις στολές που έχει στην κατοχή του, την ιδιότητα του και τέλος κάποιες παρατηρήσεις. Μαζί με την ημερομηνία εγγραφής θα εγγραφεί ένας μουσικός στα δεδομένα της εφαρμογής. Αυτά θα μπορούν να αλλάζουν οποιαδήποτε στιγμή και θα αποτελούν και τη καρτέλα του μέλους. Θα ανατρέχει εκεί ο αρχιμουσικός όταν θέλει να δει για κάποια χαρακτηριστικά του.

Επίσης στη φιλαρμονική υπάρχει ένα ρεπερτόριο που κάθε τόσο ανανεώνεται. Θα κρατούνται κάποια στοιχεία όπως αριθμός του έργου, τον τίτλο, το είδος(χασάπικο,Marcia religious) την κατάσταση του(πλήρες ελλιπές) και τέλος κάποιες παρατηρήσεις. Όλα αυτά συνθέτουν το ρεπερτόριο. Εδώ θα πρέπει να αναφέρουμε ότι τα μουσικά κομμάτια είναι αποθηκευμένα στη βιβλιοθήκη της φιλαρμονικής σε φακέλους αριθμημένους. Κάθε μουσικό κομμάτι έχει ένα μοναδικό αριθμό που γράφεται έξω από το φάκελο και όλοι οι φάκελοι είναι ταξινομημένοι ως προς αυτόν τον αριθμό.

Στη φιλαρμονική υπάρχουν και μουσικά όργανα. Θα πρέπει επομένως να κρατάμε μια λίστα με τα μουσικά όργανα μαζί με κάποιες απαραίτητες

πληροφορίες. Κατ’ αρχήν το μουσικό όργανο(τρομπέτα, σαξόφωνο), τον τύπο (Yamaha yts 23,premier 15”) , σειριακό αριθμό , κατάσταση(υπάρχουν μουσικά όργανα τα οποία μπορεί να μη δουλεύουν αλλά συνεχίζουν να ανήκουν στην φιλαρμονική).

Μια παρόμοια κατάσταση με τα όργανα υπάρχει και για το υλικό της φιλαρμονικής. Δηλαδή το είδος του μηχανήματος τον τύπο και ποσότητα. Σε αυτή την κατάσταση θα περιλαμβάνονται τα έπιπλα και ο εξοπλισμός.

Αρχείο θα πρέπει να κρατείται ακόμα και για τους αρχιμουσικούς που εχουν περάσει κατά διαστήματα από τη φιλαρμονική. Χαρακτηριστικά που μας ενδιαφέρουν να κρατάμε είναι το ονοματεπώνυμο, τηλέφωνο, διεύθυνση, κινητό σπουδές –πτυχία.

Ομοίως με τους αρχιμουσικούς αντίστοιχη λίστα θα πρέπει να έχουμε και για τους υπευθύνους. Υπεύθυνος είναι το άτομο που διορίζεται από την εκαστοτε δημοτική αρχή ασχολία του είναι η υποστήριξη στο έργο της φιλαρμονικής και η μεσολάβηση του με το δημοτικό συμβούλιου για τις ανάγκες που εμφανίζονται(έγκριση κονδυλίου για αγορά μουσικών οργάνων).

Για κάθε εμφάνιση-υπηρεσία κρατάμε κάποια στοιχεία. Δηλαδή το είδος(περιφορά Μ Παρασκευής, παρέλαση 25 Μαρτίου) την ακριβή ημερομηνία, ώρα προσέλευσης, τον τόπο προσέλευσης ,κίνηση(πεζή, με λεωφορείο) , επικεφαλής(συνήθως ο αρχιμουσικός) ο υπεύθυνος που θα συνοδεύει τη φιλαρμονική. Στην υπηρεσία θα πρέπει να γνωρίζουμε και τα άτομα που συμμετείχαν, όπως και τα μουσικά κομμάτια που παίχτηκαν.

Κάθε μέλος που είναι στη φιλαρμονική «ειδικεύεται» σε ένα μουσικό όργανο. Αυτό θα πρέπει να γράφετε στα στοιχεία του κάθε μουσικού ..

Σε κάθε εκδήλωση ανάλογα την εποχή του χρόνου φοράμε και την κατάλληλη στολή, χειμερινή-εαρινή. Οπότε κάθε μουσικός θα έχει και στολή ή στολές ώστε να συμμετέχει στις εκδηλώσεις. Άρα θα πρέπει να κρατείτε από το σύστημα για τον κάθε μουσικό πόσες και αν έχει στολή-ες αλλά και η κατάσταση τους.

Τέλος κάποια άτομα διαγράφονται αλλά παρόλα αυτά εμείς θέλουμε να κρατάμε τα στοιχεία τους (για τους δικούς μας λόγους) απλά να μην υπολογίζονται στις εμφανίσεις αλλά και ούτε να εμφανίζονται σαν ενεργά μέλη.

2.2 Περιπτώσεις χρήσης

Η ικανοποίηση κάθε λειτουργικής απαίτησης από μία εφαρμογή λογισμικού υλοποιείται ως μια αλληλουχία ενεργειών που εκτελούνται από το λογισμικό, αλληλεπιδρώντας είτε με κάποιον χρήστη (φυσικό πρόσωπο), είτε μ' άλλα συστήματα (λ.χ. άλλες εφαρμογές λογισμικού, εξωτερικές συσκευές, εξωτερικές πηγές δεδομένων). Μια τέτοια αλληλεπίδραση παράγει ένα αποτέλεσμα επιθυμητό για το χρήστη της εφαρμογής λογισμικού, δηλαδή ικανοποιεί μια λειτουργική απαίτησή του και ονομάζεται Περίπτωση Χρήσης.

Περίπτωση Χρήσης

Μια Περίπτωση Χρήσης (Use Case) είναι μια αλληλουχία ενεργειών που εκτελεί το λογισμικό αλληλεπιδρώντας με το χρήστη ή με εξωτερικά συστήματα, προκειμένου να ικανοποιήσει μία λειτουργική απαίτηση.

Κάθε περίπτωση χρήσης μπορεί να περιγράφεται με μεγαλύτερη ή μικρότερη λεπτομέρεια, όπως άλλωστε και κάθε απαίτηση από το λογισμικό. Για παράδειγμα, μπορούμε να ορίσουμε μια περίπτωση χρήσης γενικά ως «τήρηση αρχείου σπουδαστών», αλλά και με μεγαλύτερη λεπτομέρεια ως «εισαγωγή νέου σπουδαστή», «μεταβολή στοιχείων σπουδαστή» και «διαγραφή σπουδαστή». Στο σημείο αυτό δεν έχει μεγάλη σημασία το επίπεδο λεπτομέρειας, όσο ένα άλλο χαρακτηριστικό του ορισμού της περίπτωσης χρήσης οποίο μπορεί εύκολα να περάσει απαρατήρητο; περίπτωση χρήσης χαρακτηρίζεται τόσο από την αλληλουχία των ενεργειών που εκτελεί το λογισμικό, όσο και από το μέρος εκείνο

με το οποίο αλληλεπιδρά, δηλαδή ένα χρήστη – φυσικό πρόσωπο ή ένα εξωτερικό σύστημα. Το μέρος αυτό ονομάζεται Χειριστής

Χειριστής

Ένας Χειριστής (Actor) είναι μια κατηγορία χρηστών ή μια εξωτερική οντότητα με την οποία αλληλεπιδρά το λογισμικό κατά την εκτέλεση των ενεργειών μιας Περίπτωσης Χρήσης.

Στην περίπτωση που ένας Χειριστής αντιστοιχεί σε χρήστη–φυσικό πρόσωπο, κάνουμε λόγο για μια κατηγορία φυσικών προσώπων και όχι για κάποιο συγκεκριμένο πρόσωπο. Αυτό συμβαίνει, διότι μας απασχολεί ο καθορισμός της αλληλεπίδρασης ενός χρήστη με το λογισμικό και όχι η ταυτότητά του χρήστη αυτού ως φυσικό πρόσωπο (την οποία συνήθως δεν είμαστε σε θέση να γνωρίζουμε). Αυτή η αλληλεπίδραση μπορεί να ιδωθεί ως ρόλος που παίζει ένα φυσικό πρόσωπο όταν χρησιμοποιεί το λογισμικό, οπότε μπορεί να διατυπωθεί η ακόλουθη θέση:

Όταν ένας Χειριστής αντιστοιχεί σε κατηγορία χρηστών λογισμικού – φυσικών προσώπων, τότε η έννοια του **Χειριστή** είναι ισοδύναμη με την έννοια ενός **Ρόλου** (role) των χρηστών του λογισμικού.

Στην περίπτωση που ο Χειριστής αντιστοιχεί σε εξωτερικό σύστημα (λογισμικό, συσκευή), τότε συνήθως το σύστημα αυτό είναι συνήθως συγκεκριμένο και πρέπει, σε επόμενη φάση της ανάπτυξης, να προδιαγραφεί πλήρως η διαπροσωπεία (interface) του λογισμικού με αυτό. Εξαίρεση σ' αυτό αποτελεί η περίπτωση όπου το λογισμικό που κατασκευάζεται προορίζεται να παρέχει υπηρεσίες σε άλλα συστήματα λογισμικού, οπότε ο Χειριστής δεν αντιστοιχεί σε γνωστό εκ των προτέρων εξωτερικό σύστημα. Λαμβάνοντας υπόψη τις δύο τελευταίες παρατηρήσεις, μπορούμε να διατυπώσουμε τη θέση ότι στην Ενοποιημένη Προσέγγιση:

Το σύνολο των Χειριστών μιας εφαρμογής λογισμικού αποτελεί το περι-βάλλον λειτουργίας της

Κάθε περίπτωση χρήσης ενεργοποιείται από ένα Χειριστή. Όταν εκτελούνται οι ενέργειες που περιλαμβάνονται στην περίπτωση χρήσης, τότε μπορούμε να λέμε ότι «εκτελείται η περίπτωση χρήσης».

Η σαφής διάκριση των δύο αλληλεπιδρώντων μερών (περίπτωσης χρήσης και

χειριστής) εμπεριέχει μια ουσιώδη διαφορά της Ενοποιημένης Προσέγγισης ανάπτυξης λογισμικού από άλλες προσεγγίσεις, στον τομέα του καθορισμού των λειτουργικών απαιτήσεων από το λογισμικό. Συνήθως ο καθορισμός των λειτουργικών απαιτήσεων γίνεται απαντώντας στο ερώτημα: «Τι πρέπει να κάνει το λογισμικό;» Στην Ενοποιημένη Προσέγγιση το ερώτημα διαμορφώνεται ως «Τι πρέπει να κάνει το λογισμικό **για κάθε Χειριστή αυτού**;»
Μολονότι εκ πρώτης όψεως τα δύο ερωτήματα φαίνονται ισοδύναμα, η εισαγωγή της έννοιας του Χειριστή μας οδηγεί να σκεπτόμαστε με όρους αποτελεσματικότητας για τους χρήστες και το περιβάλλον της εφαρμογής. Αυτό σημαίνει ότι αναζητούμε τις λειτουργικές απαιτήσεις ως εργασίες συνυφασμένες με τους ωφελούμενους από αυτές και όχι γενικά και αόριστα ως εργασίες που «καλό θα ήταν να κάνει το λογισμικό».

2.2.1 Τρόπος γραφής Περίπτωσης Χρήσης

Δεν υπάρχει κάποιος τυπικός τρόπος γραφής μιας περίπτωσης χρήσης.
Αρχίζουμε συνήθως περιγράφοντας το κύριο σενάριο επιτυχίας. Ξεκινάμε μα το γράφουμε ως μια σειρά αριθμημένων βημάτων. Στη συνέχεια γράφουμε και τα υπόλοιπα σενάρια αναφέροντας σε τι διαφέρουν απο το κύριο σενάριο επιτυχίας. Αυτά τα σενάρια τα αναφέρουμε ως επεκτάσεις.
Ακόμα θα πρέπει να αναφέρουμε στην περίπτωση χρήσης τους χρήστες-ρόλους και ποιός είναι ο πρωτεύων. Ο πρωτεύων χρήστης έχει ως σκοπό την εκπλήρωση μιας περίπτωσης χρήσης.
Κάθε βήμα του σεναρίου είναι και μια αλληλεπίδραση με το σύστημα Μπορεί να είναι μια σύντομη εντολή που δείχνει το σκοπό του χρήστη και οχι το μηχανισμό που χρησιμοποιεί για να υπάρξει η επιτυχία. Μια επέκταση μέσα σε μια περίπτωση χρήσης κατονομάζει μια συνθήκη η οποια έχει ως αποτέλεσμα διαφορετικές αλληλεπιδράσεις απο αυτές που περιγράφονται στο κύριο σενάριο επιτυχίας και αναφέρει ποιες είναι οι διαφορές τους.
Ενα πολύπλοκο βήμα σε μια περίπτωση χρήσης μπορεί να είναι μια άλλη περίπτωση χρήσης. Στην ορολογία της uml λέμε ότι η πρώτη περίπτωση χρήσης περιλαμβάνει τη δεύτερη (include).

Τέλος θα πρέπει να αναφέρουμε ότι οι περιπτώσεις χρήσης είναι ένα πολύτιμο εργαλείο για την κατανόηση των λειτουργικών απαιτήσεων ενός συστήματος περιπτώσεις χρήσης αντιπροσωπεύουν μια εξωτερική άποψη του συστήματος, Άρα μην περιμένουμε συσχετίσεις μεταξύ των περιπτώσεων χρήσης.

2.2.2 Διαγράμματα περιπτώσεων Χρήσης

Η UML παρέχει και μια διαγραμματική μορφή των περιπτώσεων χρήσης .Σε ένα τέτοιο διάγραμμα μπορούμε να δούμε τους χρήστες τις περιπτώσεις χρήσης , τις σχέσεις μεταξύ τους, τις ΠΧ που συμπεριλαμβάνουνε άλλες ΠΧ, τα όρια του συστήματος. Επίσης φαίνεται καθαρά ποιο χειριστές διεκπεραιώνουν ποιες περιπτώσεις χρήσης.

2.3 Περιπτώσεις Χρήσεις Φιλαρμονικής

Στο σύστημα μας κύριο ρόλο παίζει η διαχείριση. Είτε μιλάμε για διαχείριση μέλους, οργάνου, μουσικού κομματιού, αρχιμουσικού, υπεύθυνου, πρόβας, εμφάνισης. Με τον όρο διαχείριση εννοούμε την εισαγωγή τροποποίηση διαγραφή των οντοτήτων μας. Για κάθε οντότητα θα πρέπει να έχουμε και τις αντίστοιχες περιπτώσεις χρήσης διαχείρισης Αυτό θα βοηθήσει τον αρχιμουσικό στην αποτελεσματικότερη οργάνωση και πληροφόρηση για τη φιλαρμονική που διευθύνει. Θα μπορεί εύκολα να
βρεί εύκολα τα διάφορα χαρακτηριστικά της κάθε οντότητας αλλα και να κρατάει καποιο είδος αρχείου για τις διάφορες παραστάσεις ή πρόβες.

Από την ανάλυση έχουμε τα εξής Περιπτώσεις χρήσης

- **Διαχείριση μέλους**
- **Διαχείριση μουσικού έργου**
- **Διαχείριση μουσικού οργάνου**
- **Διαχείριση Αρχιμουσικού**
- **Διαχείριση Υπεύθυνου**
- **Διαχείριση υπηρεσίας**
- **Διαχείριση πρόβας**

Όνομα Περίπτωσης Χρήσης :Διαχείριση μέλους

Ρόλοι:Αρχιμουσικός(πρωτευον) μέλος(δευτερεύον)

<u>Κύριο Σενάριο Επιτυχίας</u>

1. Εκκίνησης διαδικασίας διαχείρισης.
2. Εμφάνιση συνόλου ενεργών μελών
3. Ο αρχιμουσικός συμπληρώνει το textbox με το ονομα ή κάποιο χαρακτηριστικό του μέλους που γνωρίζει.
4. Ο αρχιμουσικός παταέι το κουμπί εύρεση.
5. εμφανίζεται μια λίστα με τις εγγραφές που ταιριάζουν στην αναζήτηση.

6. ο αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει.
7. τα κουτάκια παίρνουν τις τιμές της εγγραφής-μέλους που επιλέχθει.
8. τροποποιεί τις εγγραφές που θέλει.
9. Πατάει το κουμπί Αποθήκευση
10. Το σύστημα ενημερώνετε με τις αλλαγές.

Επεκτάσεις

3) Η εγγραφή δεν υπάρχει και πρέπει να εισαχθεί.

3.1)Ο αρχιμουσικός ρωτάει το μέλος τα στοιχεία του.

3.2)Ο αρχιμουσικός πληκτρολογεί-εισάγει τα στοιχεία στα κατάλληλα πεδία.

3.3)Όταν τελειώσει πατάει το κουμπί Εισαγωγή.

5)Ο αρχιμουσικός θέλει να κάνει εξαγωγή της λίστας σε αρχείο txt.

5.1)Ο αρχιμουσικός πατάει το κουμπί Εκτύπωση λίστας

6) Ο αρχιμουσικός θέλει να κάνει εξαγωγή στοιχείων σε αρχείο txt

6.1)Πατάει την επιλογή εκτύπωση

7.Ο αρχιμουσικός θέλει να διαγράψει μια εγγραφή

7.1)Πατάει το κουμπί διαγραφής

7.2)Το σύστημα διαγράφει την εγγραφή

Όνομα Περίπτωσης Χρήσης :Διαχείριση μουσικού εργου

Ρόλοι:Αρχιμουσικός(πρωτευον)

Κύριο Σενάριο Επιτυχίας

1. Εκκίνησης διαδικασίας διαχείρισης.
2. Εμφάνιση αυνόλου μουσικών κομματιών
3. Ο αρχιμουσικός συμπληρώνει το textbox με μέρος των δεδομένων που θελει να κανει ευρεση
4. Ο αρχιμουσικός παταέι το κουμπί εύρεση.

5. εμφανίζεται μια λίστα με τις εγγραφές που ταιριάζουν στην αναζήτηση.
6. ο αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει.
7. τα πεδια παίρνουν τις τιμές του μουσικου κομματιου που εχει επιλεχθεί.
8. τροποποιεί τις εγγραφές που θέλει.
9. Πατάει το κουμπί Αποθήκευση
10. Το σύστημα ενημερώνει την την βάση με τις αλλαγές

Επεκτάσεις

3) Η εγγραφή δεν υπάρχει και πρέπει να εισαχθεί.

3.1)Ο αρχιμουσικός εισάγει τα στοιχεία του μουσικού κομματιού και πατάει το κουμπί αποθήκευση

5)Ο αρχιμουσικός θέλει να κάνει εξαγωγή της λίστας σε αρχείο txt.

5.1)Ο αρχιμουσικός πατάει το κουμπί Εκτύπωση λίστας

6) Ο αρχιμουσικός θέλει να κάνει εξαγωγή στοιχείων σε αρχείο txt

6.1)Πατάει την επιλογή εκτύπωση

Όνομα Περίπτωσης Χρήσης :Διαχείριση μουσικου οργάνου

Ρόλοι:Αρχιμουσικός(πρωτευον)

Κύριο Σενάριο Επιτυχίας

1. Εκκίνησης διαδικασίας διαχείρισης.
2. Εμφάνιση του συνόλου των Μουσικών Οργάνων
3. Ο αρχιμουσικός συμπληρώνει το textbox με μέρος των δεδομένων που θέλει να κάνει εύρεση
4. Ο αρχιμουσικός πατάει το κουμπί εύρεση.
5. Εμφανίζεται μια λίστα με τις εγγραφές που ταιριάζουν στην αναζήτηση.
6. Ο αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει.

7. τα πεδια παίρνουν τις τιμές του μουσικού οργάνου που έχει επιλεχθεί.
8. τροποποιεί τις εγγραφές που θέλει.
9. Πατάει το κουμπί Αποθήκευση
10. Το σύστημα ενημερώνει την την βάση με τις αλλαγές

Επεκτάσεις

3) Η εγγραφή δεν υπάρχει και πρέπει να εισαχθεί.

3.1)Ο αρχιμουσικός εισάγει τα στοιχεία του μουσικού κομματιού

3.2)Ο αρχιμουσικός πληκτρολογεί-εισάγει τα στοιχεία στα κατάλληλα πεδία.

3.3)Όταν τελειώσει πατάει το κουμπί Αποθήκευση.

5)Ο αρχιμουσικός θέλει να κάνει εξαγωγή της λίστας σε αρχείο txt.

5.1)Ο αρχιμουσικός πατάει το κουμπί Εκτύπωση λίστας

6) Ο αρχιμουσικός θέλει να κάνει εξαγωγή στοιχείων σε αρχείο txt

6.1)Πατάει την επιλογή εκτύπωση

Όνομα Περίπτωσης Χρήσης :Διαχείριση Αρχιμουσικού

Ρόλοι:Αρχιμουσικός(πρωτευον)

Κύριο Σενάριο Επιτυχίας

1. Εκκίνησης διαδικασίας διαχείρισης.
2. Εμφάνιση Αρχιμουσικών
3. Ο αρχιμουσικός συμπληρώνει το textbox με μέρος των δεδομένων που θέλει να κάνει εύρεση
4. Ο αρχιμουσικός πατάει το κουμπί εύρεση.
5. Εμφανίζεται μια λίστα με τις εγγραφές που ταιριάζουν στην αναζήτηση.
6. Ο αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει.
7. Τα πεδία παίρνουν τις τιμές του αρχιμουσικου που έχει επιλεχθεί.
8. Τροποποιεί τις εγγραφές που θέλει.
9. Πατάει το κουμπί Αποθήκευση

10. Το σύστημα ενημερώνει την την βάση με τις αλλαγές

Επεκτάσεις

3) Η εγγραφή δεν υπάρχει και πρέπει να εισαχθεί.

3.1)Ο αρχιμουσικός εισάγει τα στοιχεία του αρχιμουσικού

3.3)Όταν τελειώσει πατάει το κουμπί Αποθήκευση.

5)Ο αρχιμουσικός θέλει να κάνει εξαγωγή της λίστας σε αρχείο txt.

5.1)Ο αρχιμουσικός πατάει το κουμπί Εκτύπωση λίστας

6) Ο αρχιμουσικός θέλει να κάνει εξαγωγή στοιχείων σε αρχείο txt

6.1)Πατάει την επιλογή εκτύπωση

Όνομα Περίπτωσης Χρήσης :Διαχείριση Υπευθυνου

Ρόλοι:Αρχιμουσικός(πρωτευον)

Κύριο Σενάριο Επιτυχίας

1. Εκκίνησης διαδικασίας διαχείρισης.
2. Εμφάνιση Συνόλου Υπεύθυνων
3. Ο αρχιμουσικός συμπληρώνει το textbox με μέρος των δεδομένων που θέλει να κάνει εύρεση
4. Ο αρχιμουσικός πατάει το κουμπί εύρεση.
5. Εμφανίζεται μια λίστα με τις εγγραφές που ταιριάζουν στην αναζήτηση.
6. Ο αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει.
7. Τα πεδία παίρνουν τις τιμές του υπεύθυνου που έχει επιλεχθεί.
8. Τροποποιεί τις εγγραφές που θέλει.
9. Πατάει το κουμπί Αποθήκευση
10. Το σύστημα ενημερώνει την την βάση με τις αλλαγές

Επεκτάσεις

3) Η εγγραφή δεν υπάρχει και πρέπει να εισαχθεί.

3.1)Ο αρχιμουσικός εισάγει τα στοιχεία του υπευθυνου

3.2)Όταν τελειώσει πατάει το κουμπί Εισαγωγή.

5)Ο αρχιμουσικός θέλει να κάνει εξαγωγή της λίστας σε αρχείο txt.

5.1)Ο αρχιμουσικός πατάει το κουμπί Εκτύπωση λίστας

6) Ο αρχιμουσικός θέλει να κάνει εξαγωγή στοιχείων σε αρχείο txt

6.1)Πατάει την επιλογή εκτύπωση

Όνομα Περίπτωσης Χρήσης :Διαχείριση εξοπλισμού

Ρόλοι:Αρχιμουσικός(πρωτευον)

Κύριο Σενάριο Επιτυχίας

1. Εκκίνησης διαδικασίας διαχείρισης.
2. Ο αρχιμουσικός συμπληρώνει το textbox με μέρος των δεδομένων που θέλει να κάνει εύρεση
3. Ο αρχιμουσικός πατάει το κουμπί εύρεση.
4. Εμφανίζεται μια λίστα με τις εγγραφές που ταιριάζουν στην αναζήτηση.
5. Ο αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει.
6. τα πεδια παίρνουν τις τιμές του εξοπλισμού που έχει επιλεχθεί.
7. τροποποιεί τις εγγραφές που θέλει.
8. Πατάει το κουμπί Αποθήκευση
9. Το σύστημα ενημερώνει την την βάση με τις αλλαγές

Επεκτάσεις

3) Η εγγραφή δεν υπάρχει και πρέπει να εισαχθεί.

3.1)Ο αρχιμουσικός εισάγει τα στοιχεία του εξοπλισμού

3.2)Όταν τελειώσει πατάει το κουμπί Εισαγωγή.

5)Ο αρχιμουσικός θέλει να κάνει εξαγωγή της λίστας σε αρχείο txt.

5.1)Ο αρχιμουσικός πατάει το κουμπί Εκτύπωση λίστας

6) Ο αρχιμουσικός θέλει να κάνει εξαγωγή στοιχείων σε αρχείο txt

6.1)Πατάει την επιλογή εκτύπωση

Όνομα Περίπτωσης Χρήσης :Διαχείριση υπηρεσίας

Ρόλοι:Αρχιμουσικός(πρωτευον)

Κύριο Σενάριο Επιτυχίας

1. Εκκίνησης διαδικασίας διαχείρισης.
2. Εμφάνιση Υπηρεσιών-Εμφανίσεων
3. Ο αρχιμουσικός συμπληρώνει το textbox με μέρος των δεδομένων που θέλει να κάνει εύρεση
4. Ο αρχιμουσικός πατάει το κουμπί εύρεση.
5. Εμφανίζεται μια λίστα με τις εγγραφές που ταιριάζουν στην αναζήτηση.
6. Ο αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει.
7. Τα πεδία παίρνουν τις τιμές της υπηρεσίας που έχει επιλεχθεί.
8. Τροποποιεί τις εγγραφές που θέλει.
9. Πατάει το κουμπί Αποθήκευση
10. Το σύστημα ενημερώνει την την βάση με τις αλλαγές

Επεκτάσεις

3) Η εγγραφή δεν υπάρχει και πρέπει να εισαχθεί.

3.1)Ο αρχιμουσικός εισάγει τα στοιχεία του υπηρεσίας

3.2)Ο αρχιμουσικός πληκτρολογεί-εισάγει τα στοιχεία στα κατάλληλα πεδία.

3.3) Ο αρχιμουσικός επιλέγει αρχιμουσικό από τη λίστα των Αρχιμουσικών.

3.4) Ο αρχιμουσικός επιλέγει αρχιμουσικό που ήταν στην υπηρεσία.

3.5) Ο αρχιμουσικός επιλέγει από τη λίστα τους αρχιμουσικούς που συμμετείχαν στην εμφάνιση

3.6) Ο αρχιμουσικός επιλέγει από τη λίστα των μουσικών κομματιών που παίχτηκαν στην εμφάνιση

3.7) Ο αρχιμουσικός πατάει το κουμπί αποθήκευση

3.8) η εγγραφή καταχωρείται στο σύστημα.

Όνομα Περίπτωσης Χρήσης :Διαχείριση πρόβας

Ρόλοι: Αρχιμουσικός(πρωτεύον)

Κύριο Σενάριο Επιτυχίας

1. Εκκίνησης διαδικασίας διαχείρισης.
2. Ο αρχιμουσικός συμπληρώνει το textbox με μέρος των δεδομένων που θέλει να κάνει εύρεση
3. Ο αρχιμουσικός πατάει το κουμπί εύρεση.
4. Εμφανίζεται μια λίστα με τις εγγραφές που ταιριάζουν στην αναζήτηση.
5. Ο αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει.
6. Τα πεδία παίρνουν τις τιμές της υπηρεσίας που έχει επιλεχθεί.
7. Τροποποιεί τις εγγραφές που θέλει.
8. Πατάει το κουμπί Αποθήκευση
9. Το σύστημα ενημερώνει την την βάση με τις αλλαγές

Επεκτάσεις

3) Η εγγραφή δεν υπάρχει και πρέπει να εισαχθεί.

3.1)Ο αρχιμουσικός εισάγει τα στοιχεία του πρόβας στα κατάλληλα πεδία.

3.2) Ο αρχιμουσικός επιλέγει Αρχιμουσικό που ήταν στην υπηρεσία.

3.3) Ο αρχιμουσικός επιλέγει τους μουσικούς που συμμετείχαν.

3.4) Ο αρχιμουσικός επιλέγει τα κομμάτια που παίχτηκαν.

3.5) Ο αρχιμουσικός πατάει το κουμπί εισαγωγή

3.6) η εγγραφή καταχωρείται στο σύστημα.

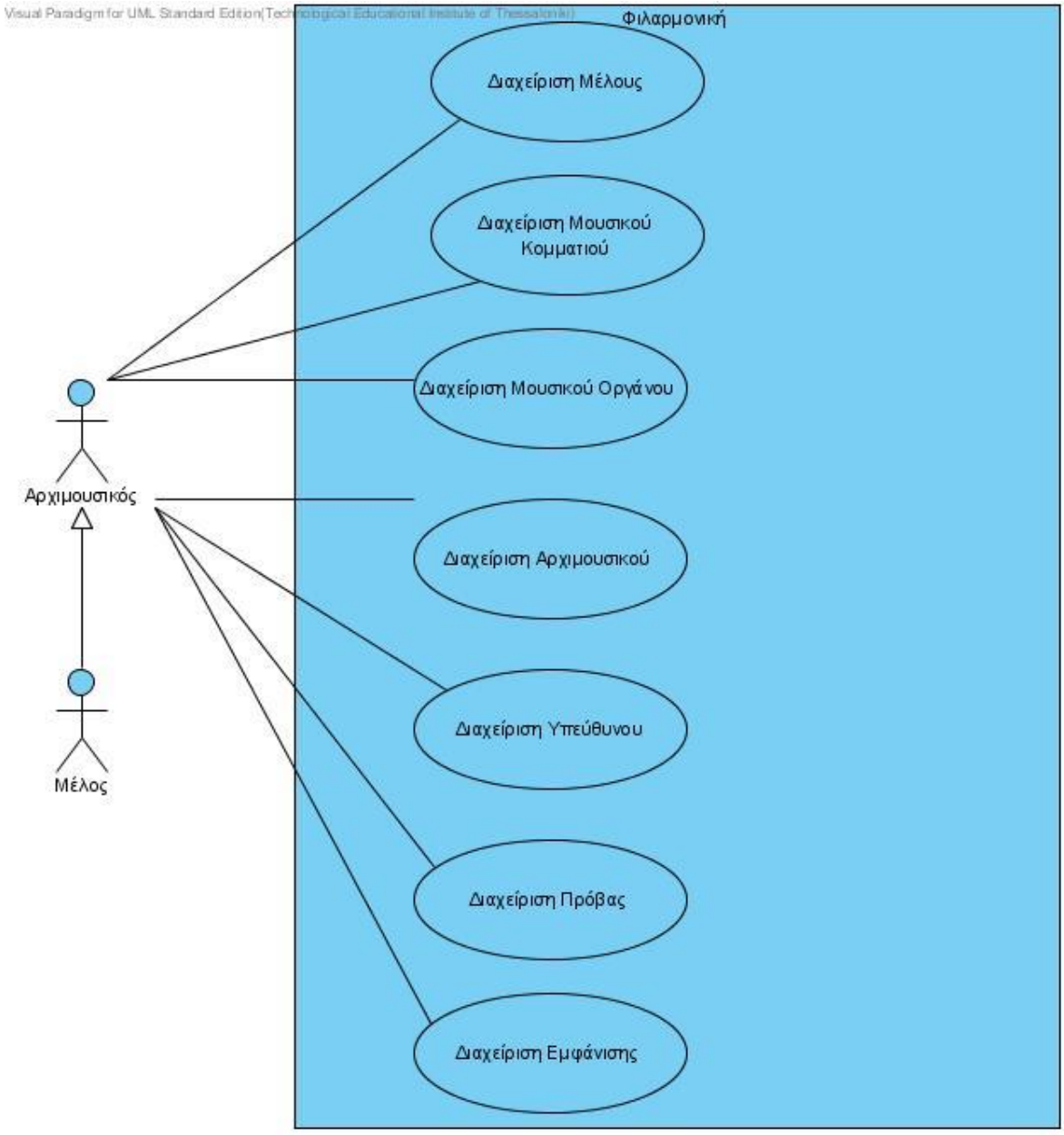
5)Ο αρχιμουσικός θέλει να κάνει εξαγωγή της λίστας σε αρχείο txt.

5.1)Ο αρχιμουσικός πατάει το κουμπί Εκτύπωση λίστας

6) Ο αρχιμουσικός θέλει να κάνει εξαγωγή στοιχείων σε αρχείο txt

6.1)Πατάει την επιλογή εκτύπωση

2.4 Διάγραμμα Περιπτώσεων Χρήσης



Εικ 2.1

Διάγραμμα Πειπρώσεων Χρήσης

Κεφάλαιο 3
Εννοιολογικό Μοντέλο

3.1 Εννοιολογικό μοντέλο

3.1.1 Εννοιολογικά μοντέλο

Ένα εννοιολογικό μοντέλο αποτελεί την κύρια πηγή έμπνευσης στην ανάλυση και σχεδιασμό του λογισμικού. Περιλαμβάνει τις σημαντικές **εννοιολογικές κλάσεις** (conceptual classes) σε μια περιοχή προβλήματος (εφαρμογής). Αποτελεί το σημαντικότερο τεχνούργημα (artifact) της αντικειμενοστραφούς (ΑΣ) ανάλυσης .Αυτό το κεφάλαιο ερευνά τις εισαγωγικές δεξιότητες στη δημιουργία εννοιολογικών μοντέλων.

Ο προσδιορισμός των αντικειμένων ή εννοιολογικών κλάσεων ενός συστήματος, αποτελεί το αντικείμενο ενδιαφέροντος της Α/Σ ανάλυσης, αλλά και τη βάση για τον μετέπειτα σχεδιασμό του συστήματος. Η ταυτοποίηση των εννοιολογικών κλάσεων αποτελεί μέρος της διερεύνησης της περιοχής προβλήματος (ΠΠ). Η UML περιέχει μια σημειογραφία παρόμοια με αυτή των διαγραμμάτων κλάσεων για την παράσταση των εννοιολογικών μοντέλων.

Κεντρική ιδέα

Ένα μοντέλο ΠΠ είναι μια αντιπροσώπευση των πραγματικών εννοιολογικών κλάσεων, όχι των τμημάτων λογισμικού. Δεν είναι επομένως ένα διάγραμμα κλάσεων ή αντικειμένων.

Εννοιολογικά μοντέλα

Το κεντρικό σημείο της ΑΣ ανάλυσης ή της διερεύνησης μιας ΠΠ, είναι η αποσύνθεση της ΠΠ σε μεμονωμένες εννοιολογικές κλάσεις, δηλ. τα πράγματα που μας ενδιαφέρουν. Ένα **εννοιολογικό μοντέλο** είναι μια **οπτική** παράσταση των εννοιολογικών κλάσεων ή πραγματικών αντικειμένων σε μια ΠΠ.

Η Ενοποιημένη Διεργασία (ΕΔ) ορίζει ένα εννοιολογικό μοντέλο ως ένα από τα τεχνουργήματα που δημιουργούνται στο γνωστικό πεδίο της 'Επιχειρησιακής Μοντελοποίησης' (Business Modeling discipline).

Στη σημειολογία της UML, ένα εννοιολογικό μοντέλο περιλαμβάνει ένα **σύνολο διαγραμμάτων κλάσεων** στα οποία δεν ορίζονται οι μέθοδοι. Μπορεί να περιλαμβάνει:

• αντικείμενα εννοιών ή εννοιολογικές κλάσεις
• σχέσεις μεταξύ των εννοιολογικών κλάσεων
• χαρακτηριστικά των εννοιολογικών κλάσεων

3.1.2Τα εννοιολογικά μοντέλα δεν είναι μοντέλα συστατικών λογισμικού

Τα εννοιολογικά μοντέλα δεν είναι μοντέλα των συστατικών λογισμικού. Ένα εννοιολογικό μοντέλο είναι μια απεικόνιση των πραγμάτων στην πραγματική ΠΠ του ενδιαφέροντος, όχι των τμημάτων λογισμικού, όπως κλάσεις java ή C++ (σχήμα 10.3), ή τα αντικείμενα με αρμοδιότητες. Επομένως, οι κάτωθι έννοιες δεν είναι κατάλληλες σε ένα εννοιολογικό μοντέλο:

- Μονάδες λογισμικού, όπως ένα παράθυρο ή μια βάση δεδομένων, εκτός εάν η έννοια που έχει γίνει μοντέλο αποτελεί μία έννοια λογισμικού, όπως ένα μοντέλο των γραφικών διεπαφών χρήστη (GUI).
- Αρμοδιότητες ή μέθοδοι

3.2 Εννοιολογικό Μοντέλο Φιλαρμονικής

Το εννοιολογικό μοντέλο του συστήματος μας αποτελείται από τις βασικές έννοιες που τις παίρνουμε από την ανάλυση – περιγραφή του συστήματος. Στο σύστημα μας έχουμε ξεχωρίσει τις εξής έννοιες¨:

- Μέλη
- Μουσικά κομμάτια
- Μουσικά ¨Όργανα
- Αρχιμουσικός
- Υπεύθυνος
- Υλικό – Εξοπλισμός
- Πρόβα
- Εμφάνιση

Αυτές τις έννοιες και επιπλέον το σύστημα που το ονομάζουμε φιλαρμονική θα τις εισάγουμε στο εννοιολογικό μας μοντέλο.

Επίσης θα πρέπει σε αυτές τις έννοιες θα πρέπει να τις συσχετίσουμε μεταξύ τους. Τις σχέσεις αυτές θα τις βρούμε από το αρχικό κείμενο που μας περιγράφει την λειτουργία της φιλαρμονικής.

Η φιλαρμονική αποτελείται από Μέλη, Μουσικά Όργανα, Αρχιμουσικούς, Υπεύθυνους, Εξοπλισμό , Πρόβες και Εμφανίσεις. Τα Μέλη πέρα από τη συμμετοχή τους στην φιλαρμονική συμμετέχουν και στις πρόβες και τις Εμφανίσεις. Ο υπεύθυνος συμμετέχει σε διάφορες εμφανίσεις ως αντιπρόσωπος της δημοτικής αρχής. Ο αρχιμουσικός συμμετέχει και στις πρόβες αλλά και στις εμφανίσεις που γίνονται. Στις πρόβες συμμετέχουν μέλη, παίζονται κάποια μουσικά κομμάτια, συμμετέχει ενας αρχιμουσικός και ένας υπεύθυνος. Στις πρόβες υπάρχει συμμετοχή μελών, μουσικών κομματιών και αρχιμουσικού.

Χαρακτηριστικά

Κάθε έννοια έχει κάποια χαρακτηριστικά. Αυτά τα χαρακτηριστικά είναι καποια δεδομένα που πρέπει να τα έχουμε αποθηκευμένα στο σύστημα μας. Επίσης αυτά τα δεδομένα μας ενδιαφέρουν γιατί περιγράφουν μια έννοια, πχ το χαρακτηριστικό ονοματεπώνυμο της έννοιας Μέλος.

Αναλυτικά σε κάθε έννοια έχει τα εξής χαρακτηριστικά

Μέλος

- Ονοματεπώνυμο
- Τηλέφωνο Σταθερό
- Τηλέφωνο Κινητό
- Διεύθυνση
- Ημερομηνία Γεννήσεως
- Ειδικότητα πχ τρομπετίστας, κλαρινετίστας
- Στολές στολές που έχει στην κατοχή του
- Ημερομηνία Εγγραφής πότε εγγράφηκε στη φιλαρμονική
- Σημειώσεις διάφορες σημειώσεις που αφορούν το μέλος

Μουσικά Κομμάτια

- Τίτλος ο τίτλος του μουσικού κομματιού
- Αρ Φακέλου Ο αριθμός του φακέλου που είναι αποθηκευμένος στη βιβλιοθήκη
- Τύπος το είδος του κομματιού πχ Marcia militair,πένθιμο
- Κατάσταση η κατάσταση στην οποία βρίσκεται καλή, έλλειπες
- Σημειώσεις

Μουσικά Όργανα

- Είδος το είδος του οργάνου πχ τρομπέτα
- Σειριακός αριθμός
- Τύπος πχ premier 15” snare
- Κατάσταση
- Σημειώσεις

Αρχιμουσικός

- Ονοματεπώνυμο
- Διεύθυνση
- Τηλέφωνο Σταθερό
- Τηλέφωνο Κινητό
- Πτυχία Τι πτυχία έχει σχετικά με τη μουσική πχ σύνθεσης
- Ημερομηνία έναρξης Η ημερομηνία που ανέλαβε αρχιμουσικός τη φιλαρμονική
- Ημερομηνία Λήξης Η ημερομηνία που τελείωσε ως αρχιμουσικός

Υπεύθυνος

- Ονοματεπώνυμο
- Διεύθυνση
- Τηλέφωνο Σταθερό
- Τηλέφωνο Κινητό
- Ημερομηνία έναρξης θητείας
- Ημερομηνία Λήξης θητείας

Υλικό-Εξοπλισμός

- Είδος Η περιγραφή του υλικού πχ φωτοτυπικό
- Τύπος ο τύπος του υλικού πχ ollivetti sp-569d
- Κατάσταση

Πρόβες

- Ημερομηνία πρόβας
- Ώρα έναρξης Πρόβας
- Ώρα λήξης πρόβας
- Μέλη
- Κομμάτια
- Σημειώσεις

Εμφάνιση

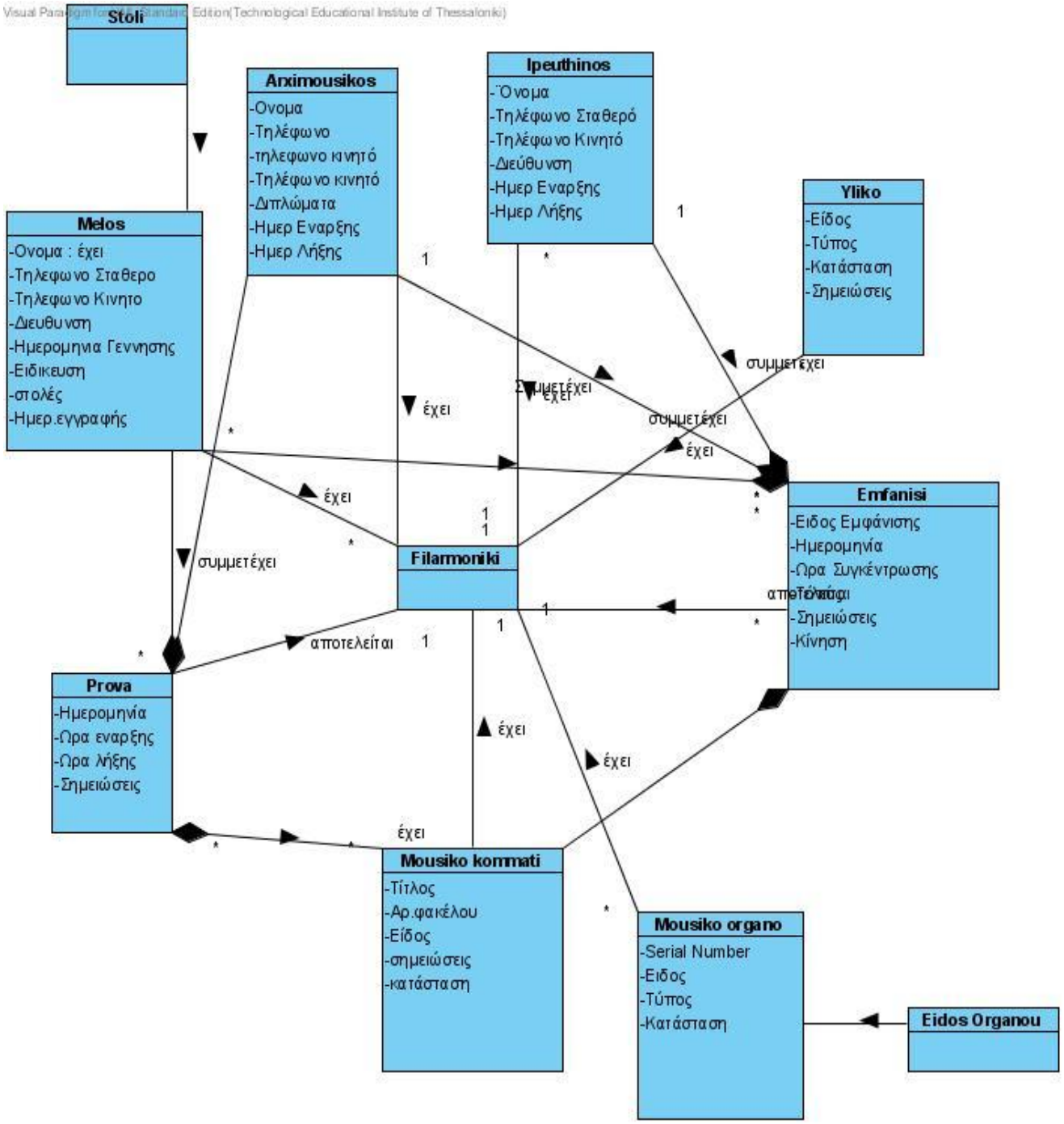
- Όνομα εμφάνισης πχ Μ Παρασκευή, Δεκαπενταύγουστος
- Ημερομηνία
- Ώρα Προσέλευσης
- Τόπος προσέλευσης
- Υπεύθυνος
- Αρχιμουσικός
- Μέλη
- Μουσικά κομμάτια
- Κίνηση
- Παρατηρήσεις

Στολές

- Τι στολές μπορεί να εχει η φιλαρμονική

Είδος Μουσικού οργάνου

- Τα διαφορετικά είδη οργάνων που υπάρχουν σε μια μπάντα



Εικ 3.1
Εννοιολογικό Μοντέλο

Στο εννοιολογικό μοντέλο όπως παρατηρούμε βασικό ρόλο στο σύστημα παίζει η εννοια της φιλαρμονικής. Η φιλαρμονική αποτελείται από το σύνολο των μελών της, από το ρεπερτόριο της (τα μουσικά κομμάτια) τα μουσικά όργανα, ο εξοπλισμός, οι αρχιμουσικοί, οι υπεύθυνοι αλλά και οι εμφανίσεις και οι πρόβες της.

Οι εμφανίσεις εκτός από τα βασικά χαρακτηριστικά τους αποτελούνται και από μέλη που συμμετείχαν, κομμάτια από το ρεπερτόριο της κάποιο αρχιμουσικό το πολύ έναν, αλλά και το πολύ έναν υπεύθυνο.

Τώρα όσον αφορά τα μέλη μπορούν να έχουν διαφορετικές στολές στην κατοχή τους αλλά πάντα το πολύ δύο(χειμερινή και καλοκαιρινή) αλλά και να είναι γνώστες ενός οργάνου.

Τα μουσικά όργανα έχουν κάποια χαρακτηριστικά. Αλλά μπορούν να έχουν και κάποιες γενική ομάδα. Οι ομάδες είναι γνωστές για μια φιλαρμονική.

Οι πρόβες ομοίως με τις εμφανίσεις αποτελούνται από τα κομμάτια που παίχθηκαν στην πρόβα, τα μέλη που συμμετείχαν αλλά και τον αρχιμουσικό που οργάνωσε την πρόβα.

Επίλογος

Το εννοιολογικό μοντέλο μας δέιχνει τις γενικές εννοιες που υπάρχουν στο συτημα μας.Κα΄ποιες από αυτές ενδεχομένος να υλοποιηθούν ως κλασεις λογισμικού.Επιπλέον μας βοηθάει να κατανοήσουμε καλύτερα το συστημα μας.Και να κατανοήσουμε κάποιους περιορισμούς που ενδεχομένως να μην ήταν εμφανεις στην περιγραφή του συστήματος

Κεφάλαιο 4
Διάγραμμα Ακολουθίας Συστήματος

4.1.1 Τι είναι ένα διάγραμμα ακολουθίας;

Ένα διάγραμμα ακολουθίας είναι ένα γρήγορα και εύκολα σχεδιαζόμενο τεχνούργημα που απεικονίζει τα συμβάντα εισόδου και εξόδου της σχεδίασης του συστήματος,. Η UML περιλαμβάνει σημειολογία της μορφής Διαγράμματος Ακολουθίας για να αναπαραστήσει τα συμβάντα (events) των εξωτερικών χρηστών (actors) προς το σύστημα.

4.1.2 Διάγραμμα Ακολουθία Συστήματος

Οι Περιπτώσεις Χρήσης περιγράφουν πώς αλληλεπιδρούν οι εξωτερικοί χρήστες με το υπό ανάπτυξη σύστημα. Κατά την αλληλεπίδραση αυτή ο χρήστης παράγει συμβάντα προς το σύστημα, ζητώντας ουσιαστικά την απόκριση λειτουργιών του συστήματος. Για παράδειγμα, όταν ο ταμίας εισάγει τον κωδικό ενός προϊόντος, ζητάει από το POS σύστημα να καταχωρήσει το συγκεκριμένο προς πώληση προϊόν. Το αιτούμενο συμβάν ενεργοποιεί μια λειτουργία του συστήματος.
Είναι επιθυμητό να απομονώσουμε και να απεικονίσουμε τις λειτουργίες που κάποιος εξωτερικός χρήστης αιτείται προς το σύστημα, διότι αυτές αποτελούν σημαντικό μέρος της κατανόησης της συμπεριφοράς του συστήματος. Η UML περιλαμβάνει το **Διάγραμμα Ακολουθίας** (sequence diagram), σαν σημειολογία που μπορεί να απεικονίσει τις αλληλεπιδράσεις των χρηστών και τις λειτουργίες του συστήματος που ενεργοποιούνται από αυτές.

Ένα Διάγραμμα Ακολουθίας Συστήματος (SSD) είναι μια εικόνα που δείχνει, για ένα συγκεκριμένο σενάριο μιας περίπτωσης χρήσης, τα συμβάντα που παράγουν οι εξωτερικοί χρήστες, την σειρά τους, και τα συμβάντα μεταξύ των συστημάτων. Όλα τα συστήματα τα απεικονίζουμε σαν μαύρα κουτιά. Η έμφαση που δίνεται από το διάγραμμα είναι τα συμβάντα των χρηστών που διαπερνούν τα όρια του συστήματος, από τους χρήστες στα συστήματα.. Ένα Διάγραμμα Ακολουθίας

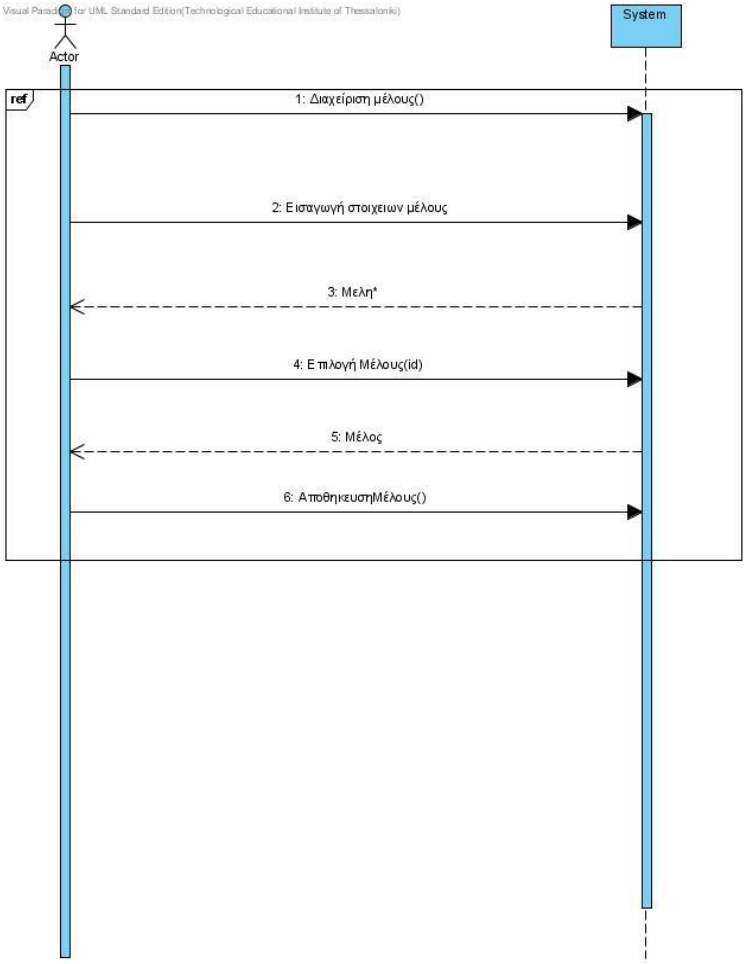
Συστήματος (SSD) θα πρέπει να δημιουργείτε για το βασικό σενάριο επιτυχίας της περίπτωσης χρήσης, καθώς και για τα συχνά ή πολύπλοκα εναλλακτικά σενάρια.

Η UML δεν ορίζει ξεχωριστά ένα Διάγραμμα Ακολουθίας Συστήματος, αλλά απλά ένα Διάγραμμα Ακολουθίας. Η συμβολή τους συνίσταται στο να δοθεί έμφαση στην εφαρμογή τους σε συστήματα προσεγγίζοντάς τα σαν μαύρα κουτιά. Αργότερα, το Διάγραμμα Ακολουθίας θα χρησιμοποιηθεί στην σχεδίαση της αναπαράστασης της αλληλεπίδρασης των αντικειμένων λογισμικού, για την υλοποίηση λειτουργιών.

4.2 Διάγραμμα Ακολουθίας Συστήματος Φιλαρμονικής

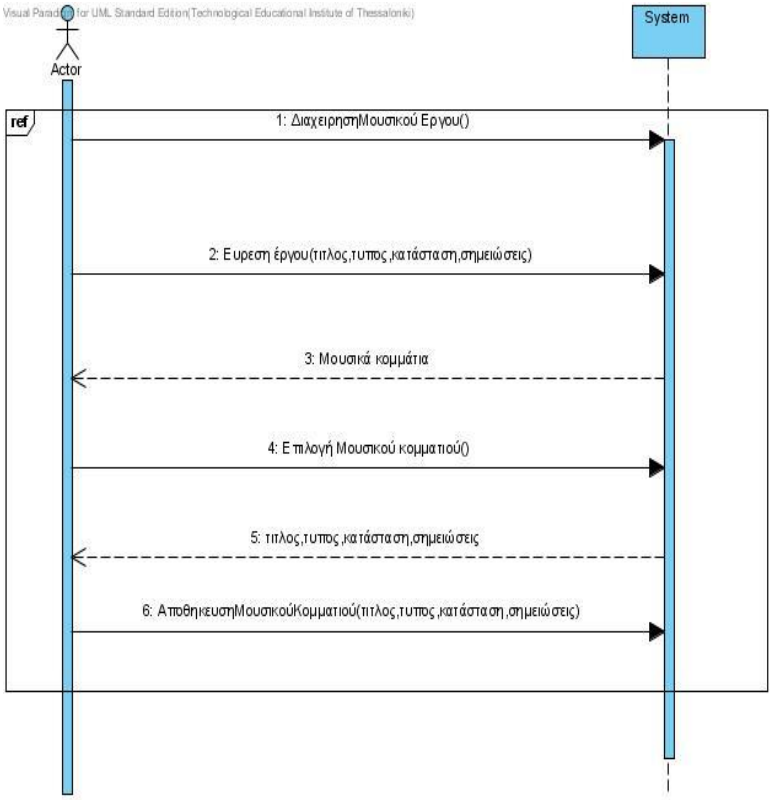
Διαχείριση Μέλους

Στην περίπτωση χρήσης Διαχείριση Μέλους ο αρχιμουσικός ψάχνει να βρει από τη λίστα των μελών το μέλος που θέλει να τροποποιήσει. Δίνει ένα μέρος του ονόματος του ή κάποιο άλλο χαρακτηριστικό πχ τηλέφωνο. Οταν το πληκτρολογήσει πατάει το κουμπί της εύρεσης. Το σύστημα ψάχνει να βρέι ποιες εγγραφές ταιριάζουν με το προς εύρεση κείμενο. Αφού βρει ποιες εγγραφές ταιριάζουν εμφανίζει μόνο αυτές-ή στη λίστα αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει. Αφου την επιλέξει κάνει τις αλλαγές που θέλει και πατάει το πλήκτρο της αποθήκευσης. Το σύστημα αμέσως ενημερώνεται με τις αλλαγές



4.1ssd Διαχ. Μελους

Διαχείριση Μουσικού Έργου

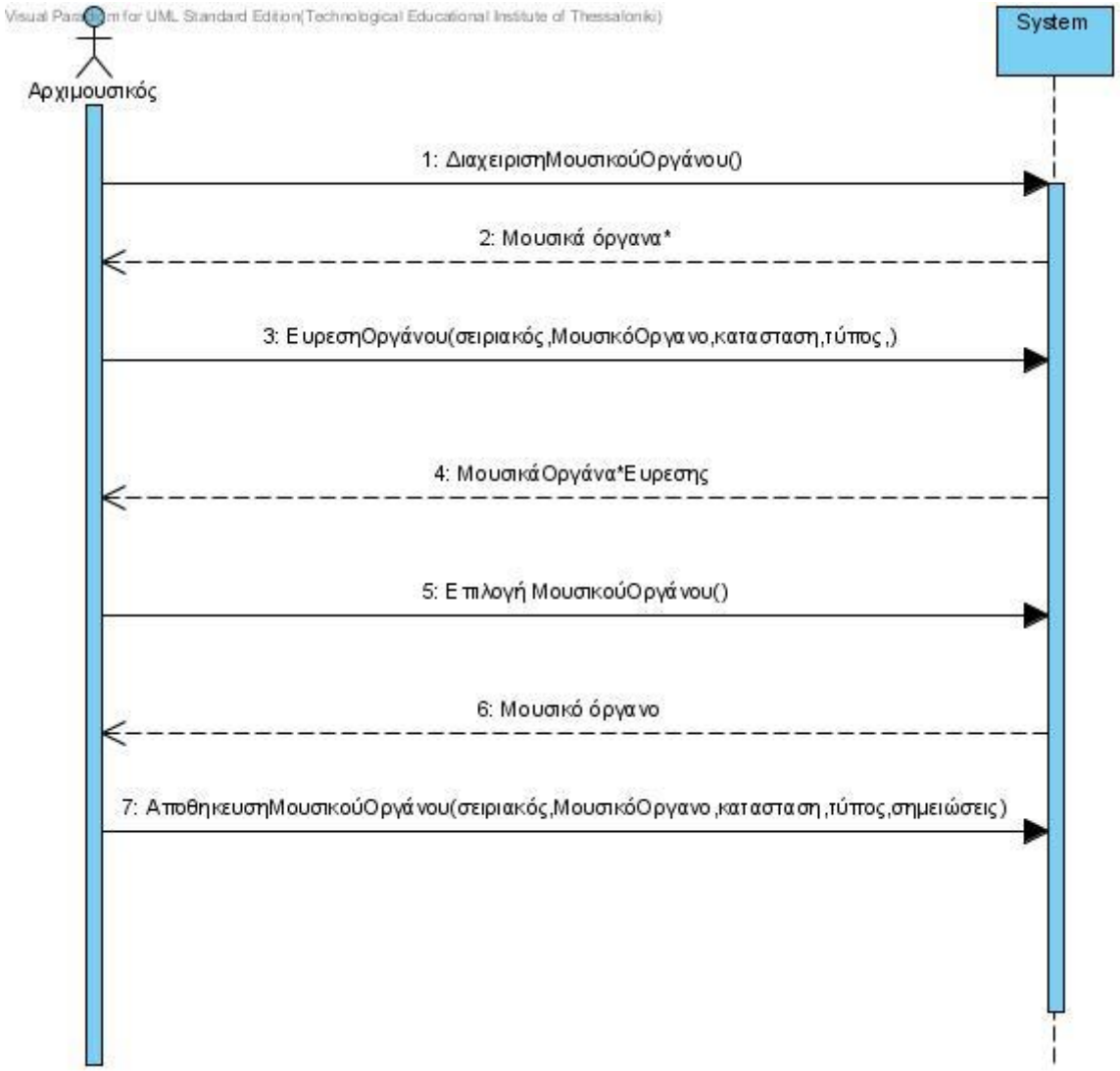


Διαγ 4.2 ssd Διαχ μουσικού έργου

Στην περίπτωση χρήσης Διαχείριση Μουσικού Εργου ο αρχιμουσικός ψάχνει να βρει από τη λίστα των κομματιών το κομματι που θέλει να τροποποιήσει. Δίνει ένα μέρος του ονόματος του ή κάποιο άλλο χαρακτηριστικό . Οταν το πληκτρολογήσει πατάει το κουμπί της εύρεσης. Το σύστημα ψάχνει να βρέι ποιες εγγραφές ταιριάζουν με το προς εύρεση κείμενο. Αφού βρει ποιες εγγραφές ταιριάζουν εμφανίζει μόνο αυτές-ή στη λίστα αρχιμουσικός επιλέγει αυτή που θέλει να τροποποιήσει. Αφου την επιλέξει κάνει τις αλλαγές που θέλει και πατάει το πλήκτρο της αποθήκευσης. Το σύστημα αμέσως ενημερώνεται με τις αλλαγές.

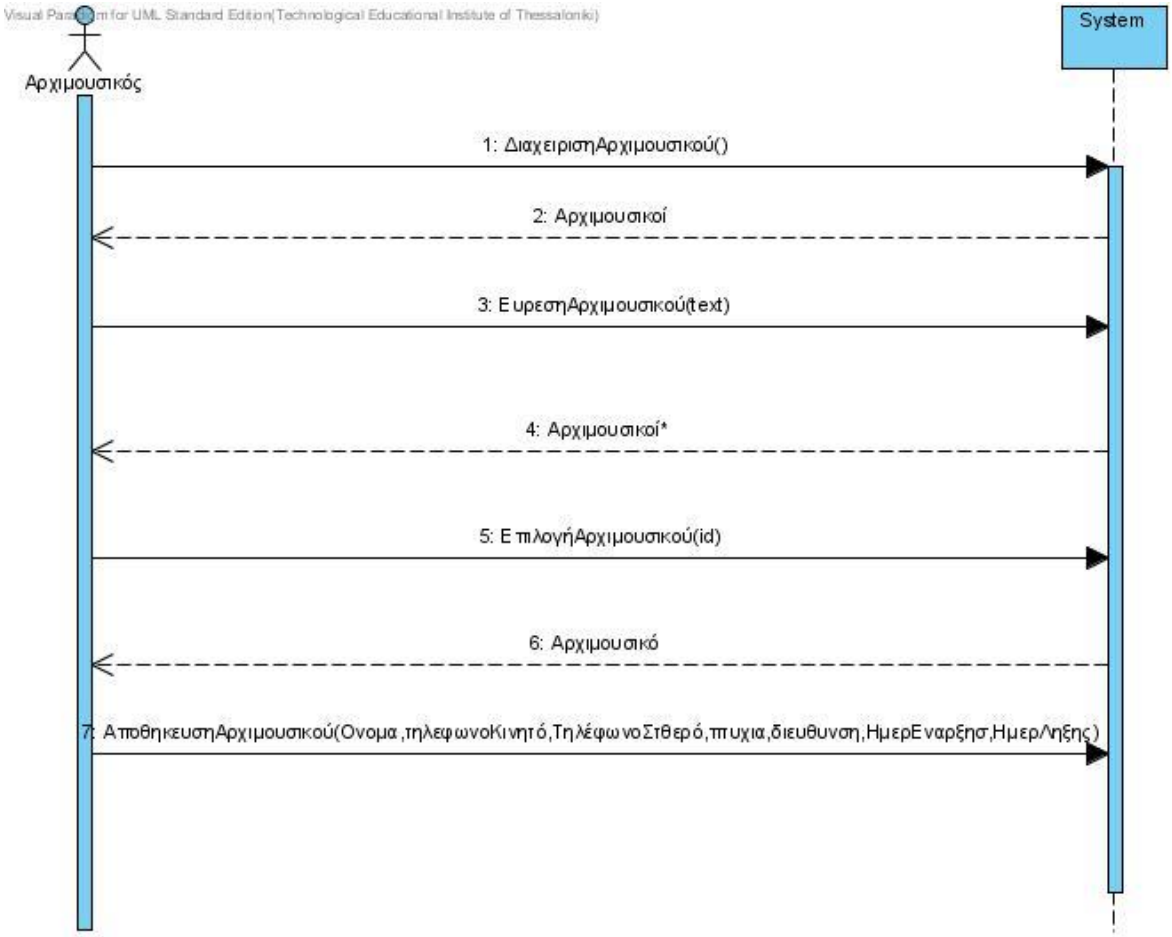
Διαχείριση μουσικού οργάνου

Ομοίως με τις παραπάνω περιπτώσεις χρήσης ακολουθεί και αυτή την ίδια λογική της εύρεσης τροποποίησης και αποθήκευσης.Ομοίως και οι Διαχείριση Αρχιμουσικού και Υπεύθυνου.



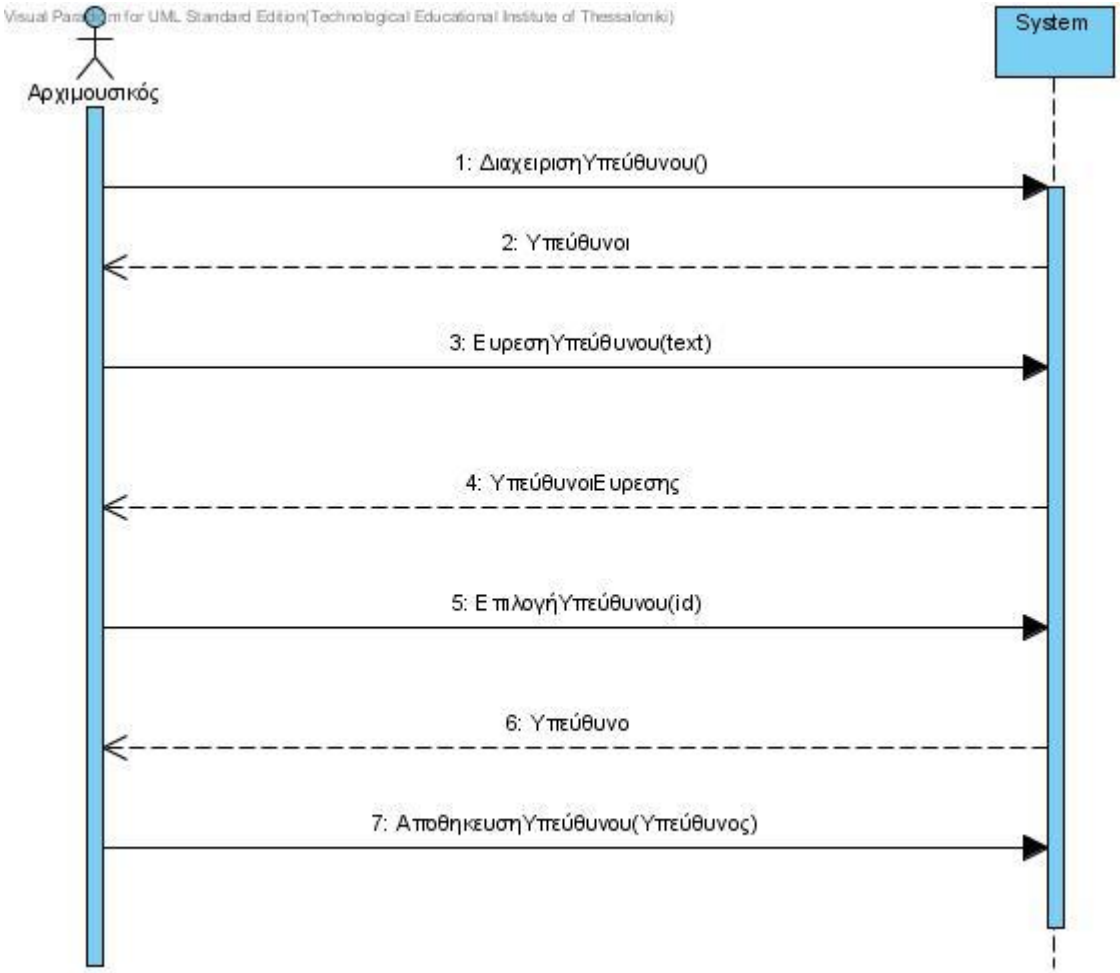
Διαγ 4.3 ssd Διαχ μουσικού οργάνου

Διαχείριση Αρχιμουσικού



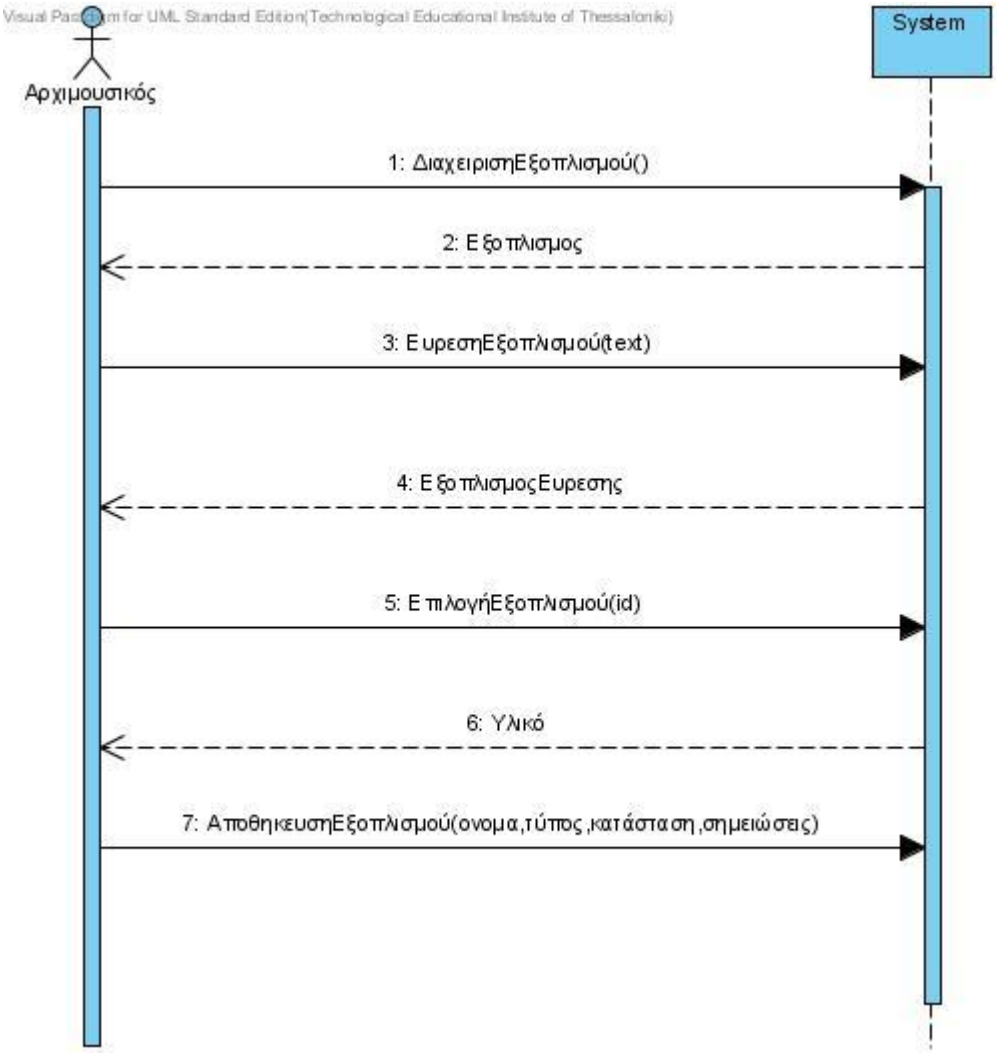
Διαγ 4.4 ssd Διαχ Αρχιμουσικού

Διαχείριση Υπεύθυνου



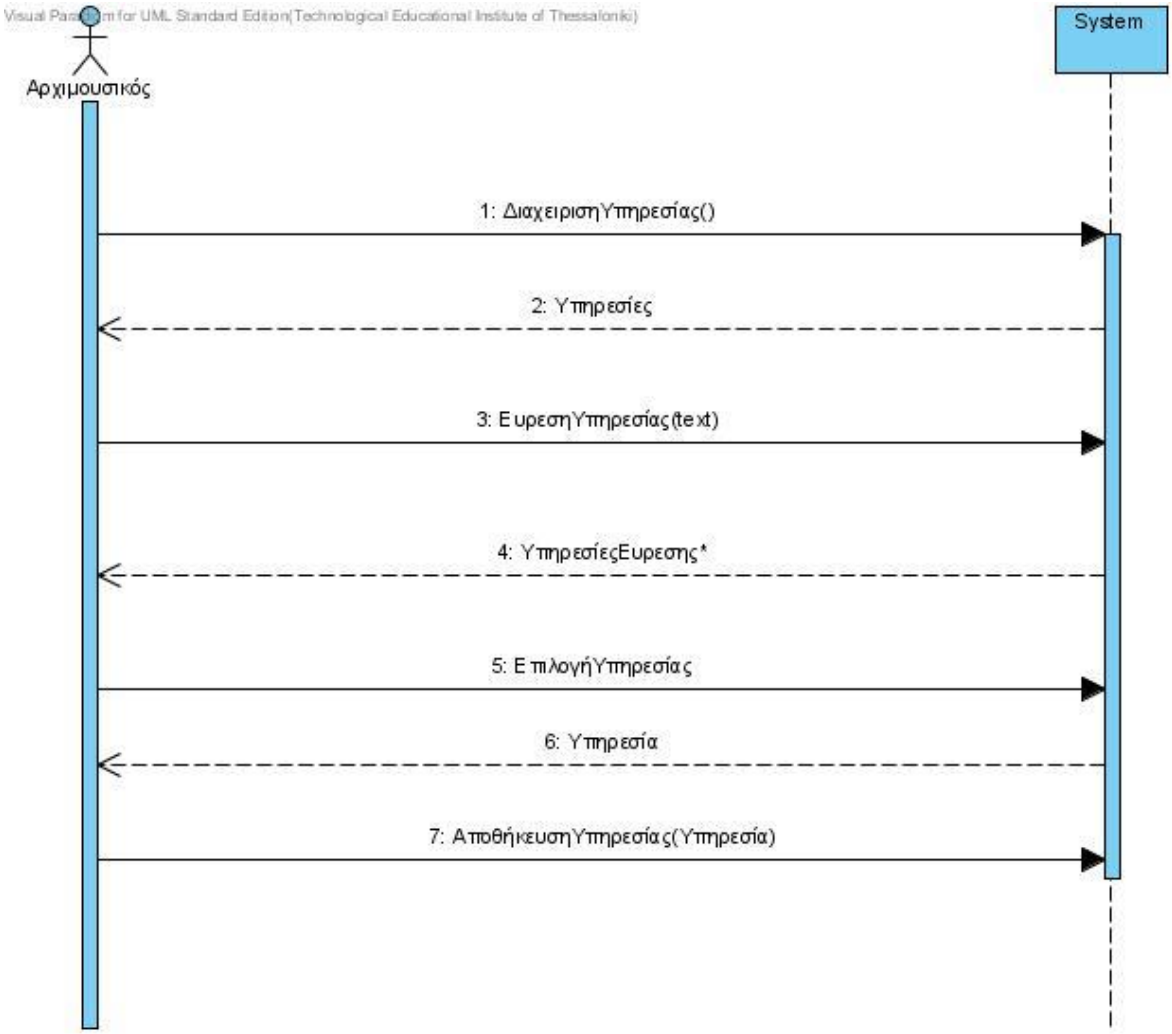
Διαγ 4.5 ssd Διαχ Υπεύθυνου

Διαχείριση εξοπλισμού



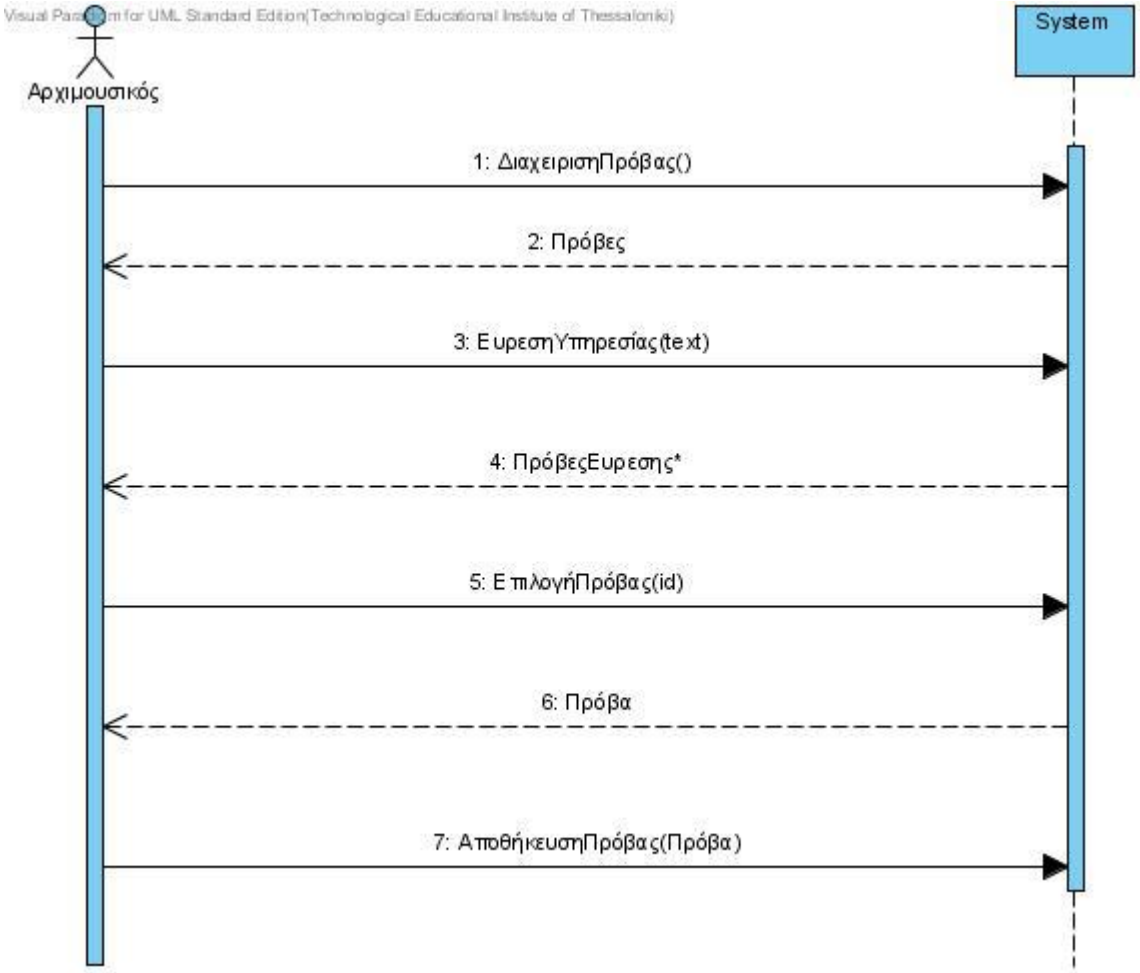
Διαγ 4.6 ssd Διαχ Εξοπλισμού

Διαχείριση Εμφάνισης



Διαγ 4.7 ssd Διαχ Εμφάνισης

Διαχείριση πρόβας



Διαγ 4,8 Διαχ Πρόβας

Κλεινωντας αυτό το κεφάλαιο θα πρέπει να πούμε ότι μέχρι τώρα έχουμε ολοκληρώσει και έχουμε δέι την ανταλλαγή μηνυμάτων μεταξύ του χρήστη και του συστήματος.Εχουμε δέιτ ι ζητάει αλλά και τι επισρτέφει.Αυτο θα μας βοηθήσει να συνθέσουμε το διάγραμμα ακολουθίας σε επόμενο κεφάλαιο.

Κεφάλαιο 5
Grasp Patterns και ανάθεση λειτουργιών σε κλάσεις

5.1.1 Πρότυπα Ορισμού Αρμοδιοτήτων σε αντικείμενα

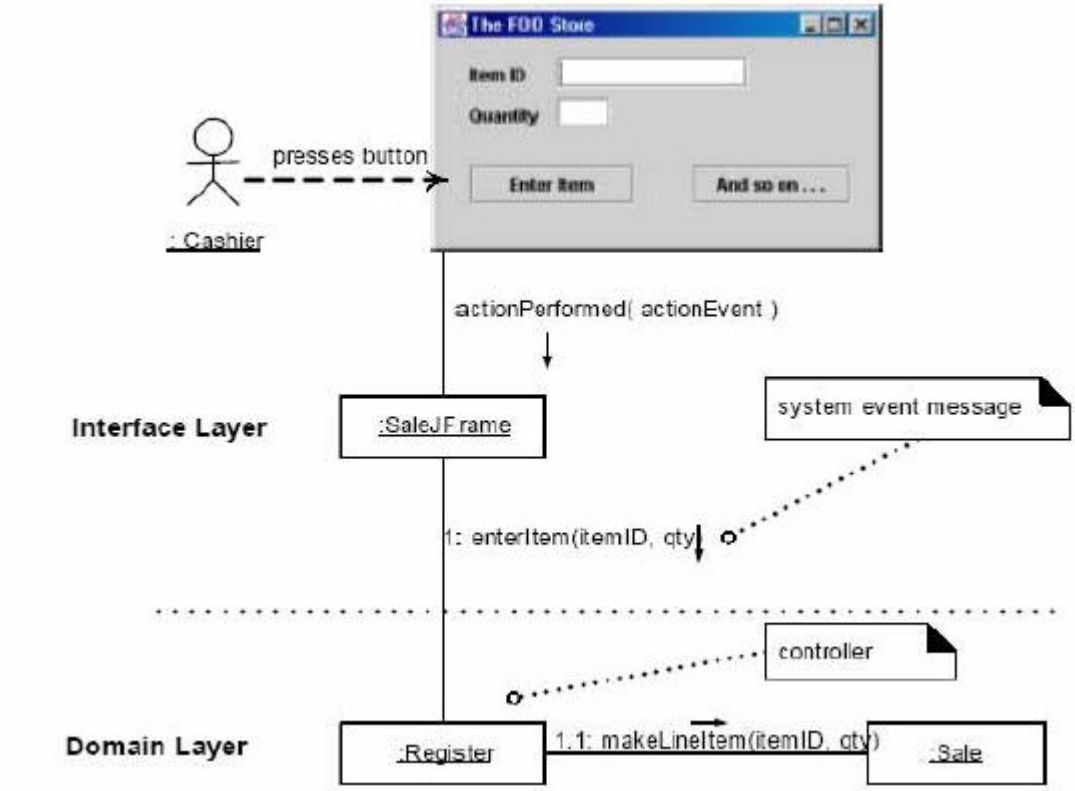
Εισαγωγή

Αφού δημιουργήσουμε το Διάγραμμα Ακολουθίας Συστήματος (SSD – System Sequence Diagram) έχουμε ολοκληρώσει την βασική αλληλεπίδραση μεταξύ χρήστη – συστήματος. Βλέποντας το σύστημα σαν «μαύρο κουτί». Έχουμε ουσιαστικά καταγράψει τα συμβάντα που διαπερνούν το σύστημα ζητώντας απ' αυτό κάποια λειτουργία. Το σύστημα με το που δέχεται το μήνυμα (συμβάν) αναλαμβάνει να εκτελέσει την επιθυμητή λειτουργία του χρήστη.
Εμείς μετά θα πρέπει να ορίσουμε τις μεθόδους και τις κλάσεις λογισμικού που θα υπάρχουν στο σύστημα μας αλλά και θα πρέπει να ορίσουμε και τον τρόπο αλληλεπίδρασης τους.

Design Patterns

Πεπειραμένοι μηχανικοί λογισμικού ανάπτυξης αντικειμενοστραφών συστημάτων, έχουν αναπτύξει ένα σύνολο γενικών αρχών και ιδιωματικών λύσεων, οι οποίες τους καθοδηγούν στην ανάπτυξη του λογισμικού. Οι αρχές αυτές και οι ιδιωματισμοί, κωδικοποιημένοι με δομημένη μορφή, που περιγράφει το πρόβλημα και την λύση, και με κατάλληλο δηλωτικό όνομα, μπορούν να αναφέρονται ως πρότυπα.Τα πρότυπα GRASP αποτελούν ένα βοήθημα καθοδήγησης και κατανόησης του ουσιαστικού σχεδιασμού των αντικειμένων, αλλά και στην εφαρμογή της λογικής του σχεδιασμού με έναν μεθοδικό, λογικό, και επεξηγηματικό τρόπο. Αυτή η προσέγγιση της κατανόησης και της εφαρμογής των σχεδιαστικών αρχών, βασίζεται στα πρότυπα ανάθεσης αρμοδιοτήτων λίγα λόγια με τη βοήθεια αυτών των προτύπων Αποφασίζοντας ποιες μέθοδοι ανήκουν που, και πως πρέπει τα αντικείμενα να αλληλεπιδρούν μεταξύ τους.

Ένα σημαντικό πρόβλημα είναι στην ανάπτυξη της εφαρμογής είναι η σύνδεση της διεπαφής χρήστη (User Interface) με το υπόλοιπο πρόγραμμα.
Αυτή τη λύση μπορεί να μας τη δώσει το πρότυπο controller.
Σύμφωνα με αυτό το πρότυπο θα πρέπει να δημιουργήσουμε μια κλάση για κάθε περίπτωση χρήσης. Αυτή η κλάση θα αποφασίζει ποιες μέθοδοι θα εκτελεστούν .Δηλαδή ο ελεγκτής θα παραλαμβάνει τα αιτήματα από το UI και θα αποφασίζει ποιες μέθοδοι θα εκτελεστούν και θα προωθεί τα αιτήματα στις κατάλληλες κλάσεις. Ενα παράδειγμα εφαρμογής αυτού του GRASP Pattern



Εικ 5.1
Παράδειγμα GRASP PATTERN μιας περίπτωσης χρήσης διενέργεια πώλησης.

Οπότε έχοντας τις βασικές μας έννοιες (Μέλη, Μουσικά κομμάτια, Μουσικά όργανα, Εξοπλισμός-υλικά, Αρχιμουσικό, Υπεύθυνο, Εμφανίσεις, Πρόβες.) και δημιουργώντας μία κλάση controller για κάθε περίπτωση χρήσης θα έχουμε και τις αντίστοιχες κλάσεις controller(μια για κάθε βασική έννοια) επίσης θα πρέπει να προσθέσουμε ότι θα έχουμε και τις κλάσεις που είναι υπεύθυνες για τις διεπαφές του προγράμματος μας. Οπότε θα ήταν χρήσιμο και καλό έχοντας ως οδηγό τον ευρετικό κανόνα που λέει «Σε εφαρμογές που αποτελούνται από ένα

αντικειμενοστραφές μοντέλο που αλληλεπιδρά με τη διασύνδεση χρήστη το μοντέλο θα πρέπει να είναι ανεξάρτητο από τη διασύνδεση. Η διασύνδεση θα πρέπει να εξαρτάται από το μοντέλο».Παρόλο όμως που αναφέρεται ως ευρετικός κανόνας η σχεδίαση που προκύπτει από την εφαρμογή του αποτελεί ένα από τα πιο κλασικά πρότυπα σχεδιασμού λογισμικού. Το πρότυπο Μοντέλου-Οψης-Ελεγκτή(Model-View-Controller,MVC).

5.1.2Ευρετικός Κανόνας (MVC)

Ο στόχος του μορφήματος αυτού είναι η αρχιτεκτονική σχεδίαση ενός συστήματος που θα κατανέμει τις ευθύνες της εφαρμογής σε ορισμένα σημαντικά συνθετήματά του, βάσει ορισμένων κανόνων. Ορισμένες από τις ευθύνες αυτές είναι η λήψη των αιτήσεων των χρηστών του συστήματος, ο προσδιορισμός της μονάδας που θα επεξεργαστεί την αίτηση του χρήστη, η ενημέρωση των δεδομένων του συστήματος, ο προσδιορισμός του τρόπου με τον οποίο το σύστημα θα ανταποκριθεί στον χρήστη και άλλες μικρότερης σημασίας.

Το μόρφημα MVC όπως προκύπτει και από το όνομά του στηρίζεται σε τρία βασικά συνθετήματα. Το Μοντέλο, την Όψη και τον Ελεγκτή.

Το Μοντέλο αποτελεί ένα συνθέτημα το οποίο αναπαριστά το μοντέλο της εφαρμογής. Ως μοντέλο της εφαρμογής εννοούνται τα δεδομένα του μοντέλου οι δομές δεδομένων στις οποίες αποθηκεύονται αυτά και οι διαδικασίες πρόσβασης, ενημέρωσης και αποθήκευσης των δεδομένων. Θα μπορούσε κάποιος να φανταστεί το μοντέλο ως μία οντότητα, που παρέχει μία διαπροσωπεία. Με την διαπροσωπεία αυτή άλλες οντότητες μπορούν να χρησιμοποιήσουν τα δεδομένα που το μοντέλο περιέχει.

Η Όψη όπως πολύ εύκολα μπορεί να φανταστεί κάποιος δεν είναι τίποτε διαφορετικό από την όψη του συστήματος στον εξωτερικό χρήστη. Το συνθέτημα αυτό αποτελεί μία γέφυρα μεταξύ του συστήματος και των χρηστών του. Οποιαδήποτε αίτηση του χρήστη εκφράζεται μέσω της όψης σε αίτηση προς το σύστημα και οποιαδήποτε αντίδραση του συστήματος προσφέρεται στον χρήστη μέσω της όψης. Αν και ο χρήστης δεν το αντιλαμβάνεται αυτό, η όψη δεν γνωρίζει

καμία λεπτομέρεια σχετικά με το επιχειρηματικό μοντέλο του συστήματος και ούτε με τον τρόπο υλοποίησης των υπηρεσιών που του προσφέρονται. Στην ιδανική περίπτωση το συνθέτημα της όψης αρκείται στο να λαμβάνει πληροφορίες από το μοντέλο και να τις παρουσιάζει στον χρήστη.

Στην προηγούμενη παράγραφο αναφέρθηκε η φράση «επιχειρηματικό μοντέλο». Πρόκειται για τον «αλγόριθμο» του συστήματος, σύμφωνα με τον οποίο το σύστημα λειτουργεί. Ο αλγόριθμος αυτός αποτυπώνεται στο τρίτο συνθέτημα του μορφήματος MVC, τον ελεγκτή. Ο ελεγκτής αποτελεί την γέφυρα μεταξύ της όψης και του μοντέλου. Μετά την αίτηση του χρήστη ποιες αλλαγές και με ποια σειρά πρέπει να γίνουν στο μοντέλο του συστήματος. Έπειτα προωθεί στον χρήστη το αποτέλεσμα, δίνοντας την σωστή εντολή στην Όψη.

Όπως είναι προφανές ένα σύστημα σχεδιασμένο σύμφωνα με το μόρφημα MVC έχει συνθετήματα με ξεκάθαρες αρμοδιότητες. Η συντήρηση και η περαιτέρω ανάπτυξη της εφαρμογής είναι ευκολότερη. Δεδομένων μάλιστα της επεκτασιμότητας και ευελιξίας που είναι αναγκαίο να χαρακτηρίζουν την εφαρμογή, η επιλογή του μοντέλου MVC υπήρξε μονόδρομος.

Όταν ένα σύστημα καλείται να αλλάζει τον τρόπο λειτουργίας του, ή την δομή δεδομένων του για να προσαρμοστεί σε άλλες εφαρμογές είναι εύλογη η ανάγκη σαφούς κατανομής των αρμοδιοτήτων των συνθετημάτων του. Ο προγραμματιστής πρέπει ανά πάσα στιγμή να γνωρίζει το τι ακριβώς πρέπει να αλλάξει και που σε ποια ψηφίδα θα βρει αυτό που αναζητά.

5.2 Εφαρμογή του MVC μοντέλου στο στο σύστημα μας.

Στο συστημα μας έως τώρα έχουμε τις εξής οντότητες:

Μέλη, Μουσικά Κομμάτια, Μουσικά Όργανα, Εξοπλισμός, Αρχιμουσικό, Υπεύθυνο, Εμφανίσεις, Πρόβες.

Αρα θα έχουμε από τις περιπτώσεις χρήσης και τους αντίστοιχους controllers.

Επομένως θα έχουμε

Τις εξής κλάσεις

Member

Piece

Instrument

Material

BandMaster

Director

Appearance

Training

Όπως επίσης και τα εξής enums: Suits(στολές),Movement(κινηση), Type(τύπος οργάνου) επειδή οι τιμές που θα εχουν αυτές οι οντότητες θα είναι προκαθορισμένες.

Επίσης για κάθε περίπτωση χρήσης θα έχουμε και τον αντίστοιχο controller

MemberController

PieceController

InstrumentController

MaterialController

BandMasterController

DirectorController

AppearanceController

TrainingController

Κεφάλαιο 6
Διάγραμμα Αλληλεπίδρασης-Ακολουθίας

6.1 Διαγράμματα Αλληλεπίδρασης

Τα διαγράμματα αλληλεπίδρασης (interaction diagrams) χρησιμοποιούνται για την οπτική αναπαράσταση των διαφόρων σεναρίων των περιπτώσεων χρήσης του συστήματος . Απεικονίζουν ένα σενάριο σαν στιγμιότυπα αντικειμένων που αλληλεπιδρούν μεταξύ τους ανταλλάσοντας μηνύματα.

Η διαδικασία απεικόνισης της συνεργασίας των αντικειμένων του συστήματος μέσω των αλληλεπιδράσεων τους βρίσκεται στο επίκεντρο της διαδικασίας σχεδίασης μια και μέσω αυτής ο σχεδιαστής του συστήματος ουσιαστικά ανακαλύπτει τις λειτουργίες που θα πρέπει να υποστηρίζουν τα διάφορα αντικείμενα του συστήματος ή ακόμα και τα ίδια τα αντικείμενα και τις κλάσεις στις οποίες ανήκουν. Ο βασικός κανόνας είναι ότι θα πρέπει να έχουμε τα αντικείμενα εκείνα με τις λειτουργίες εκείνες που θα μπορούν να ανταπεξέλθουν σε όλα τα σενάρια όλων των περιπτώσεων χρήσης του συστήματος.

Υπάρχουν δύο τύποι διαγραμμάτων αλληλεπίδρασης οι οποίοι είναι ισοδύναμοι

- Το Διάγραμμα Ακολουθίας (sequence diagram)
- Το Διάγραμμα Συνεργασίας (collaboration diagram)

Το διάγραμμα ακολουθίας δίνει έμφαση στη χρονική ακολουθίας των μηνυμάτων, ενώ το διάγραμμα συνεργασίας απεικονίζει τους συνδέσμους μεταξύ των αντικειμένων σε αυτό και για να γίνει προφανής η χρονική σειρά των μηνυμάτων απαραίτητη είναι η αρίθμηση τους. Το διάγραμμα συνεργασίας απαιτεί λιγότερο σχεδιαστικό χώρο μια και η διάταξη των αντικειμένων είναι ελεύθερη στο χώρο και επομένως προτιμάται όταν έχουμε πολλά αντικείμενα σε μια αλληλεπίδραση ή όταν σχεδιάζουμε ένα διάγραμμα με το χέρι. Επειδή τα διαγράμματα ακολουθίας προσφέρουν καλύτερη απεικόνιση της χρονικής αλληλουχίας των μηνυμάτων θα τα προτιμήσουμε στην σχεδίαση του συστήματος μας.

6.2 Διάγραμμα Ακολουθίας

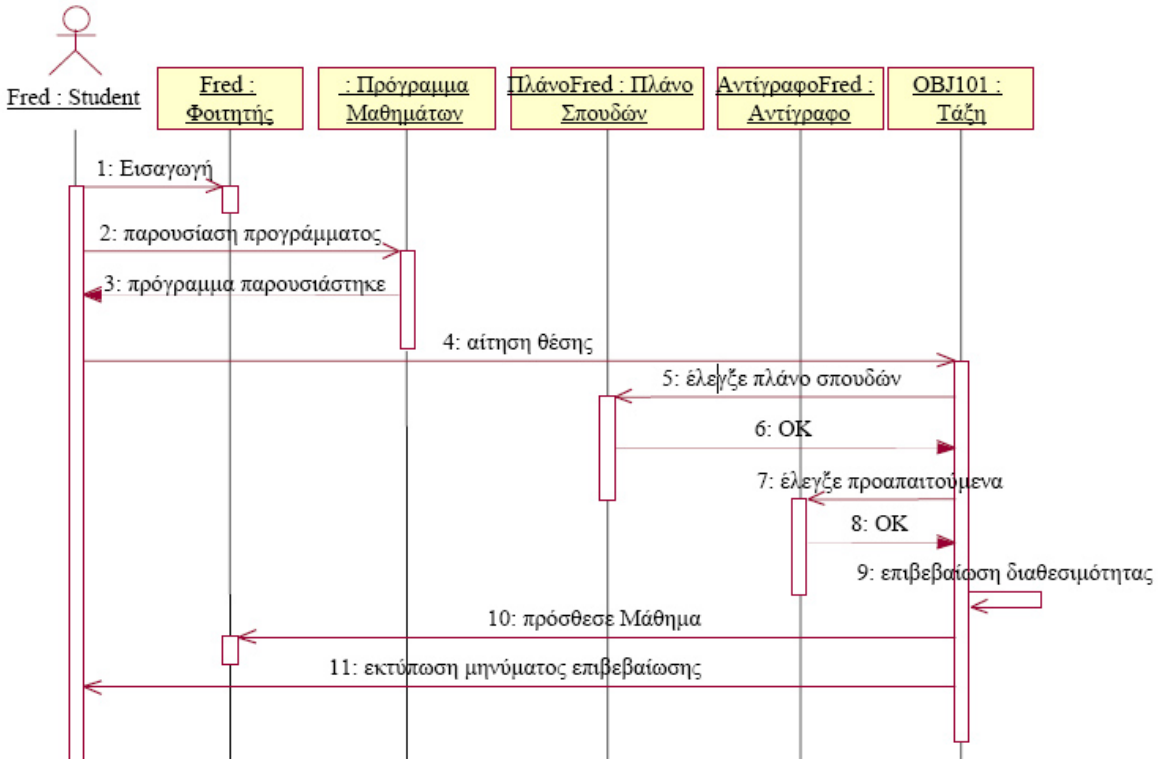
Τα διαγράμματα ακολουθίας χρησιμοποιούνται κυρίως για να αναπαραστήσουν την αλληλεπίδραση μεταξύ των αντικειμένων του συστήματος και συγκεκριμένα την ακολουθιακή σειρά με την οποία ανταλλάσσονται τα μηνύματα. Συνήθως οι σχεδιαστές συστημάτων θεωρούν ότι τα διαγράμματα ακολουθίας χρησιμοποιούνται αποκλειστικά από αυτούς. Ωστόσο, τα διαγράμματα ακολουθίας χρησιμοποιούνται και σε άλλες περιπτώσεις όπως για παράδειγμα την περιγραφή ανταλλαγής πληροφοριών σε μια επιχείρηση.

Ένα διάγραμμα ακολουθίας παρουσιάζει την αλληλεπίδραση μεταξύ των αντικειμένων σε δύο διαστάσεις. Η κάθετη διάσταση αντιστοιχεί στην κλίμακα του χρόνου ενώ στην οριζόντια διάσταση συμβολίζονται ανεξάρτητα αντικείμενα. Τα αντικείμενα συμβολίζονται ως ορθογώνια παραλληλόγραμμα μέσα στα οποία μπορεί να σημειωθεί το όνομα του αντικειμένου που συμμετέχει στο σενάριο που απεικονίζεται(καθώς πρόκειται για κάποιο στιγμιότυπο και το όνομα του μπορεί να είναι οτιδήποτε) .Αυτό όμως που σημειώνεται είναι το όνομα της κλάσης στην οποία ανήκει το κάθε αντικείμενο. Το όνομα του αντικειμένου και το όνομα της κλάσης διαχωρίζονται με μια άνω-κάτω τελεία(:) . Στην περίπτωση που το σενάριο εμπλέκει και εξωτερικούς χρήστες εκτός από αντικείμενα τότε συμβολίζονται ως μικρά σχηματικά ανθρωπάκια.

Για κάθε αντικείμενο αντιστοιχεί μια κάθετη γραμμή που ονομάζεται γραμμή ζωής(lifeline). Όσο χρόνο ένα αντικείμενο απλά υπάρχει τοτε η γραμμή είναι διακεκομμένη και όσο χρόνο είναι ενεργό τότε αυτή σχεδιάζεται ως διπλή. Ενα μήνυμα αποστέλλεται μεταξύ αντικειμένων, συμβολίζεται με ένα βέλος από τη μια γραμμή ζωής ενός αντικειμένου προς μια άλλη γραμμή ζωής. Κάθε μήνυμα αντιστοιχεί σε μια λειτουργία που την υλοποιεί το αντικείμενο παραλήπτης και μέσα σε παρένθεση αναφέρονται οι τυχόν παράμετροι που απαιτούνται θέση των ακμών είναι αντίστοιχη με το χρόνο .

Η λήψη ενός μηνύματος από ένα αντικείμενο σηματοδοτεί την πραγματοποίηση ενός γεγονότος που αποτελεί το εξωτερικό ερέθισμα. Για το αντικείμενο. Ένα γεγονός μπορεί να είναι:

- Ενεργοποιούμενο από το χρήστη του συστήματος
- Ενεργοποιούμενο από κάποιο άλλο υπολογιστικό σύστημα.
- Ενεργοποιούμενο από κάποιο άλλο αντικείμενο του ίδιου συστήματος
- Ενεργοποιούμενο από την πάροδο του χρόνου.

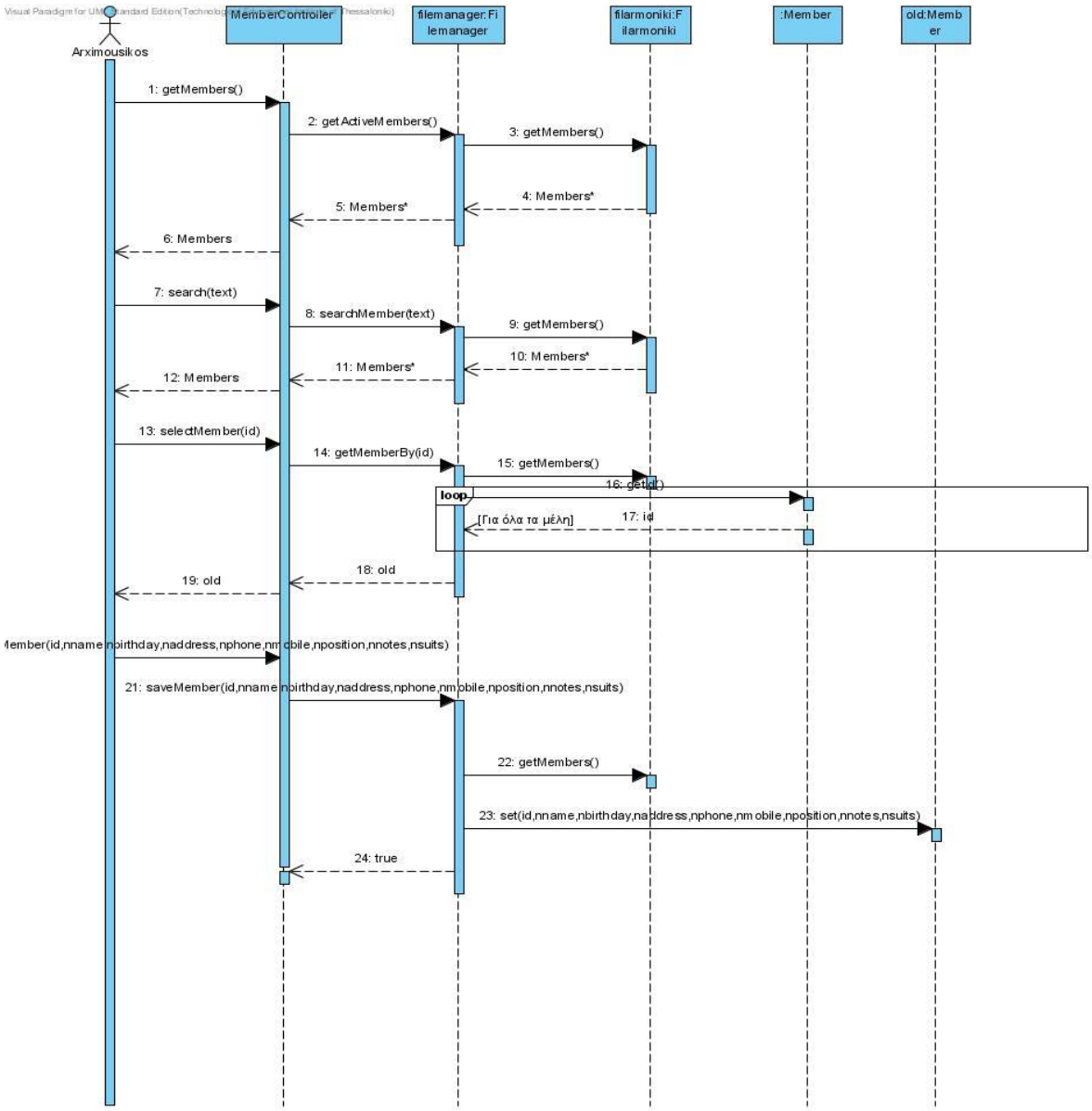


Εικ 6.1

Παράδειγμα διαγράμματος ακολουθίας

6.3 Διαγράμματα Ακολουθίας Φιλαρμονικής

Διαχείριση Μέλους



Διαγ 6.2 Διαγ Διαχειρισης Μέλους

Στη διαχείριση μέλους ξεκινάει η διαδικασία με την εμφάνιση των μελών στην λίστα του παραθύρου. Με την μέθοδο του Controller getMembers() παίρνουμε τα μέλη από την filemanger.Με τη σειρά της η filemanager παίρνει τα μέλη από την filarmoniki.Αυτά τα μέλη τα επιστρέφει στον χρήστη και να του τα εμφανίσει.
Επόμενο βήμα είναι να εισάγει ο χρήστης ένα String και να το δώσει για αναζήτηση. Αυτό το String text θα αποσταλεί στον Controller του μέλους και μετά

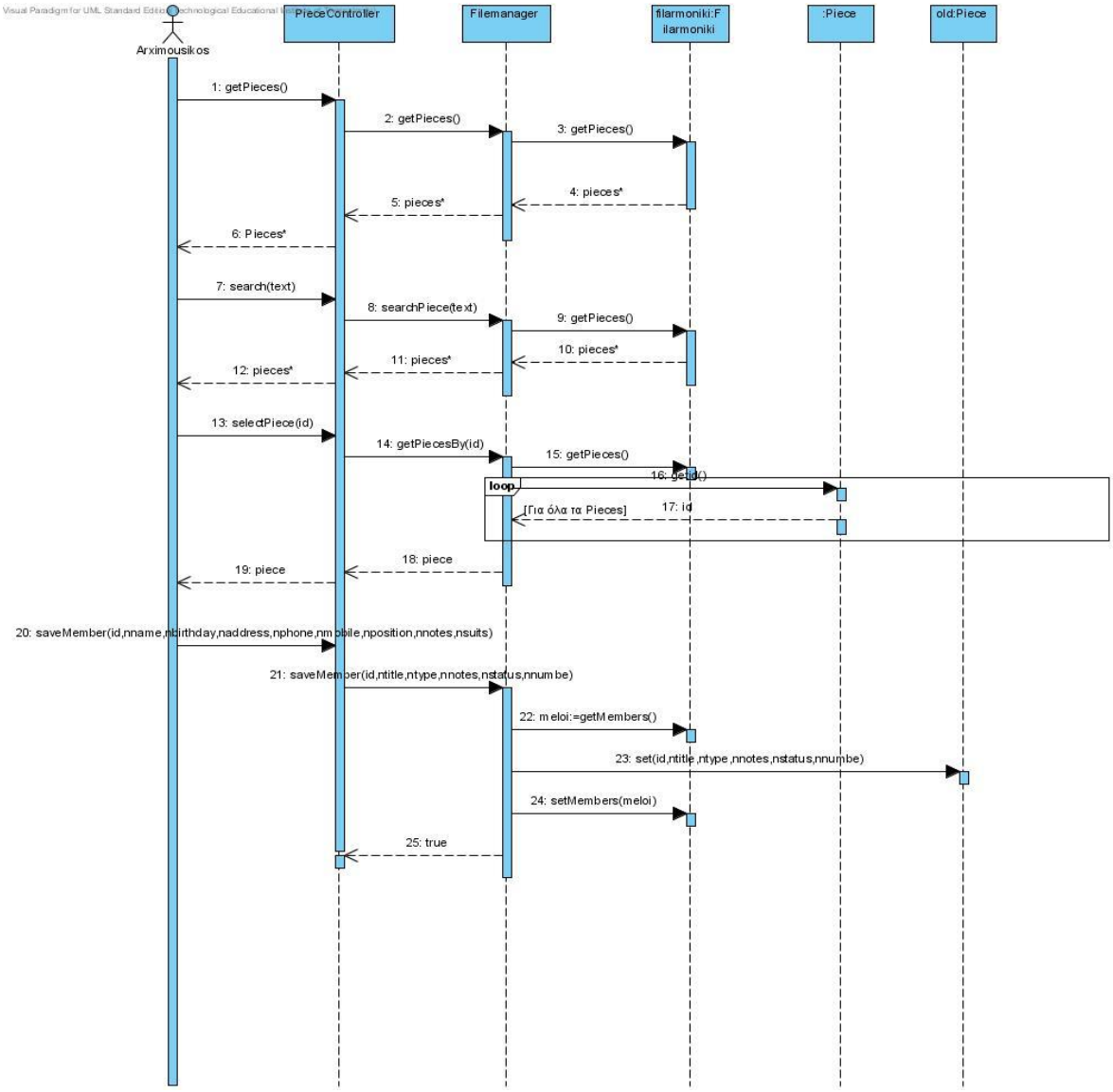
θα προωθηθεί στην file manager με τη μέθοδο searchMember(text).Η filemanager θα πάρει όλα τα μέλη από την filarmoniki και θα δει με ποια μέλη ταιριάζει το text.Τα μέλη που θα ταιριάζουν θα τα επιστρέψει στον Controller για να τα προωθήσει στον χρήστη.

Τώρα ο χρήστης έχει μπροστά τα μέλη που ικανοποιούν το search οπότε το μέλος θα βρίσκεται ανάμεσα στα μέλη που επεστράφησαν από το search.

Ο χρήστης επιλέγει την εγγραφή που τον ενδιαφέρει. Όταν την επιλέξει θα μπορεί να επέμβει σε αυτήν και να αλλάξει κάποιο χαρακτηριστικό του μέλους ή και όλα. Όταν αποφασίσει για τις αλλαγές που θα κάνει και καταλήξει τότε αποθηκεύει τις αλλαγές που έχουν γίνει μεταφέρονται μέσο του controllerMέλους στο filemanager.Apo ton filemanager παίρνουμε τις εγγραφές των μελών της filarmonikis και την εγγραφή που τροποποιήσαμε την τροποποιούμε με τα νέα χαρακτηριστικά και την αποθηκεύουμε στο αρχείο δεδομένων μας.

ΔιαχείρισηΜουσικών

Κομματιών



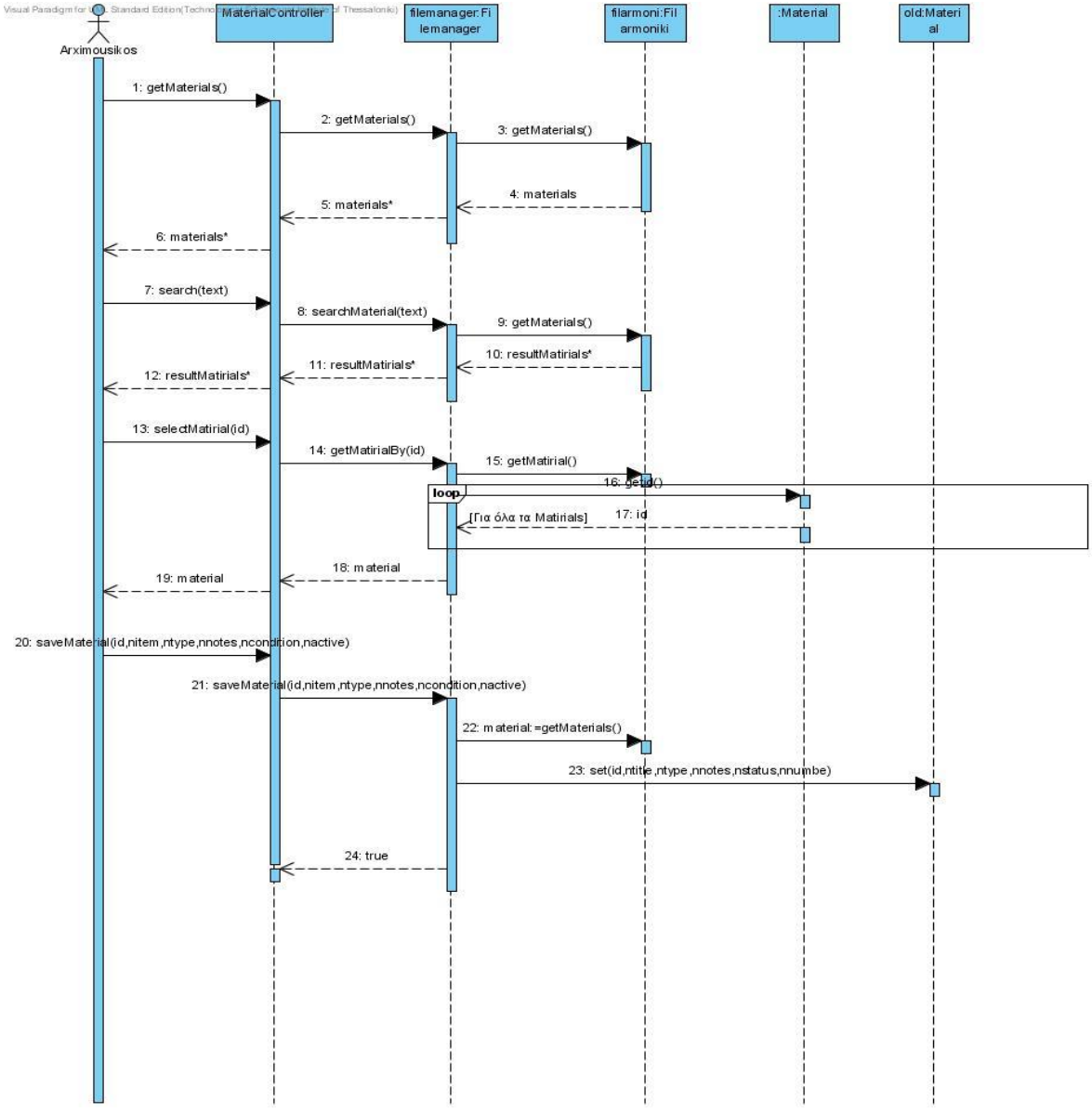
Διαγ 6.3 Διαγ Διαχειρισης Κομματιών

Η διαχείριση μουσικών κομματιών είναι υπεύθυνη για την τροποποίηση των μουσικών κομματιών. Όταν ξεκινάμε τη διαχείριση εμφανίζεται το σύνολο των κομματιών της φιλαρμονικής. Επειδή ο αριθμός τους μπορεί να είναι αρκετά μεγάλος ο χρήστης – αρχιμουσικός για να διευκολυνθεί μπορεί να δώσει κάποια λέξη κλειδί και να βρει την εγγραφή από το σύνολο των κομματιών. Εισάγει το String text όπου μέσω της PieceController kaitης μεθόδου search(text) καλούμε την searchPiece(text) από τον filemanager.Στον filemanager παίρνουμε από τη filarmoniki το σύνολο των κομματιών (getPieces()) και από εκεί ψάχνουμε τις εγγραφές που ταιριάζουν στο text. Όταν βρεθούνε τις επιστρέφουμε στον χρήστη.

Ο χρήστης έχοντας μονο τις εγγραφές που ταιριάζουν επιλέγει αυτήν που θέλει. Όταν την επιλέξει μπορεί να τροποποιήσει τα χαρακτηριστικά που τον ενδιαφέρουν.

Ο χρήστης όταν τροποποιήσει τις εγγραφές που τον ενδιαφέρουν αποθηκεύει τις αλλαγές.Με την αποθήκευση τα στοιχεία μέσω του controller και της μεθόδου savePiece(id,ntitle,number,mtype,nnotes,nstatus) πηγαίνουν στον filemanager για να τροποποιηθεί η εγγραφή που άλλαξε ο χρήστης και να αποθηκευτεί στο σύστημα.

Διαχείριση Εξοπλισμού

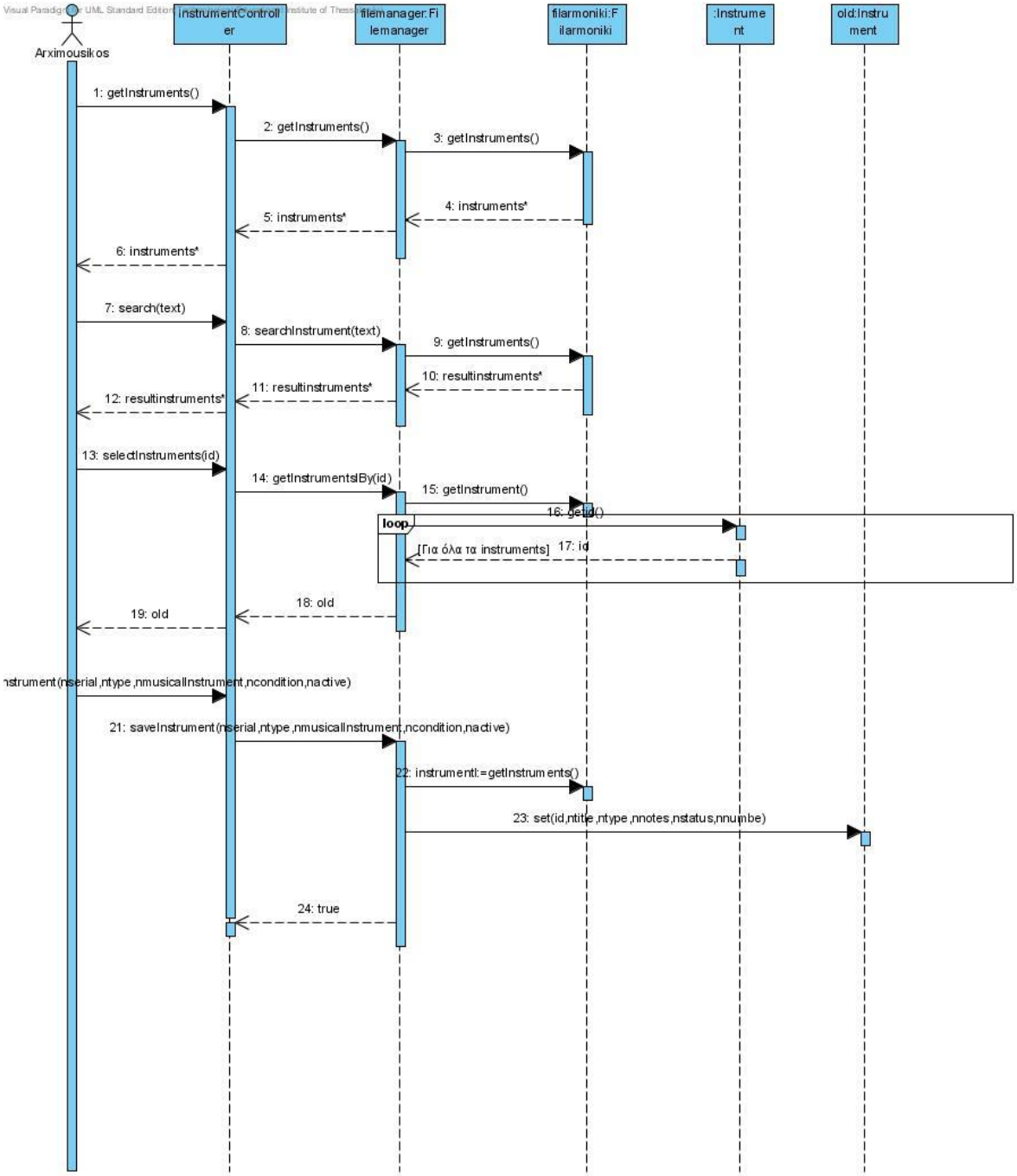


Διαγ 6.4 Διαγ Διαχειρισης Υλικού

Με την εκκίνηση της διαχείρισης εξοπλισμού ο χρήστης θα πρέπει να εχει ανάκτηση το σύνολο των υλικών-εξοπλισμού της φιλαρμονικής. Απο τον Material Controller με τη μέθοδο getMaterials() και στη συνέχεια από τις κλάσεις FileManager και filarmoniki παίρνουμε το σύνολο του υλικού και το επιστρέφουμε στο χρήστη. Ο χρήστης επομένως με το που δέι τη διαχείριση εξοπλισμού έχει το σύνολο του εξοπλισμού. Όταν ο χρήστης δώσει ένα text αυτό το to text με τη μεθοδο search(text) του Controller θα προωθηθεί με τη μέθοδο searchMaterial(text) του file manager.Η file manager παίρνει το σύνολο των υλικών από τη filamroniki. Γίνεται η σύγκριση των υλικών με το text και επιστρέφεται ένα σύνολο από υλικά. Απο αυτά τα υλικά ο χρήστης θα επιλέξει την εγγραφή που τον ενδιαφέρει και θα πάρει τα χαρακτηριστικά του υλικού.

Αυτά τα χαρακτηριστικά ο χρήστης μπορεί να τα τροποποιήσει. Όταν ο αρχιμουσικός τελειώσει με την τροποποίηση της εγγραφής τα νέα στοιχεία του εξοπλισμού μέσω της μεθόδου save και του controller.Στον filemanager η μέθοδος SaveMaterial παίρνει τα στοιχεία και αφού δεί σε ποια εγγραφή της filamronikis αναφέρονται τροποποιεί την αντίστοιχη εγγραφή.

Διαχείριση οργάνων



Διαγ 6.5 Διαγ Διαχειρισης Οργάνων

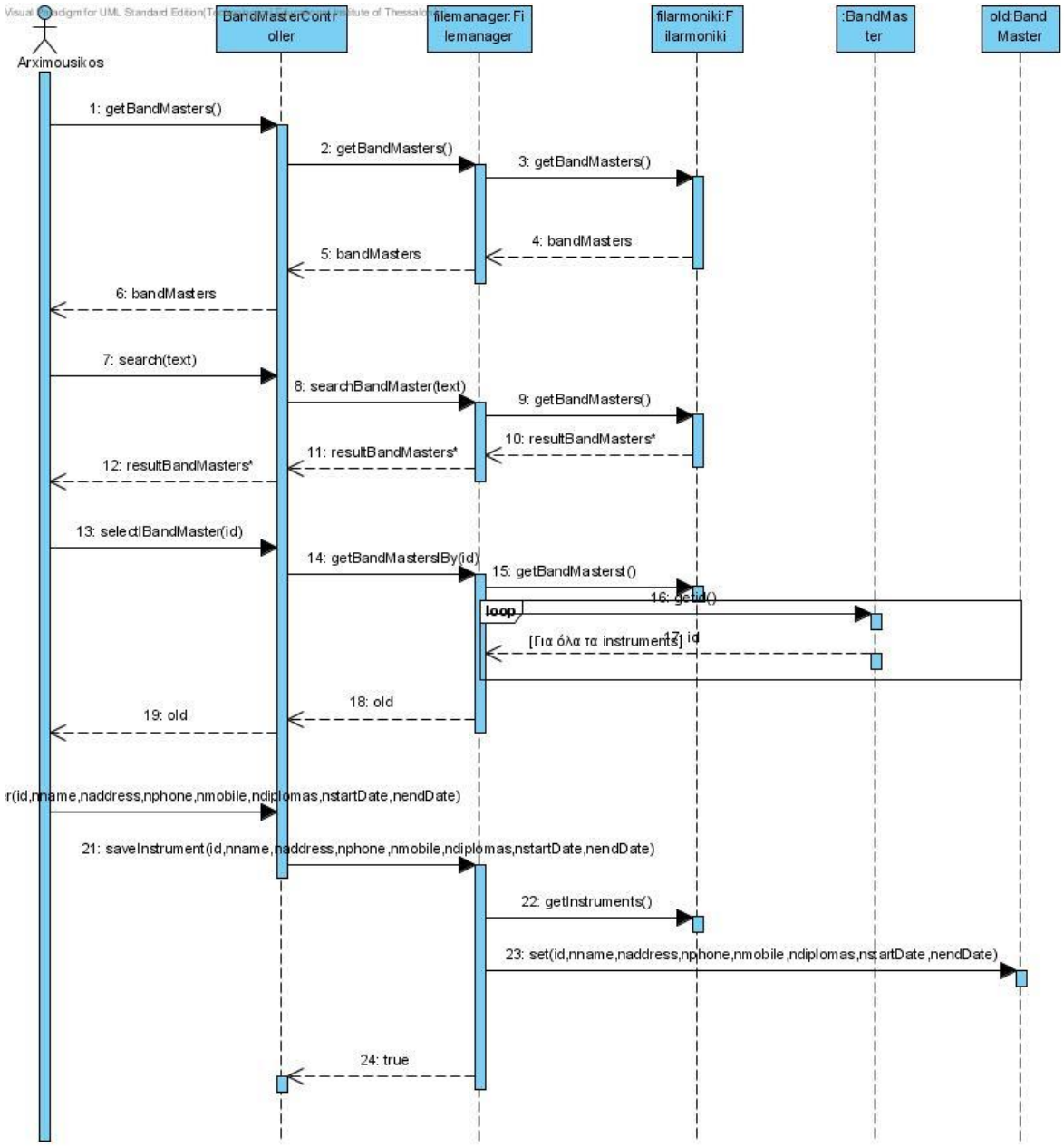
Με την εκκίνηση της διαδικασίας διαχείρισης οργάνων εμφανίζεται η οθόνη με το σύνολο των μουσικών οργάνων της φιλαρμονικής. Αφού πρώτα ο controller ανακτήσει το σύνολο τους από τον filemanager.Όταν ο αρχιμουσικός έχει τη λίστα με τα μουσικά όργανα μπροστά του .Όταν όμως θέλει να βρει ένα συγκεκριμένο οργανο θα πρέπει να το ψάξει μέσω της εύρεσης. Στην εύρεση δίνουμε ένα text. Μέσω της μεθόδου search(text) του controller στέλνουμε ένα «αίτημα» με τη μέθοδο searchInstrument(text) στον filemanager. Στον filemanager παίρνει το

σύνολο των μουσικών οργάνων της filarmonikis και συγκρίνει ποιες εγγραφές έχουν το text.Οταν βρεθούνε οι εγγραφές που μας ενδιαφέρουν τότε τις επιστρέφουμε στον χρήστη – αρχιμουσικό.

Ο χρήστης από αυτό το μικρότερο σύνολο εγγραφών είανι πιο εύκολο να εντοπίσει την εγγραφή που τον ενδιαφέρει. Όταν την βρει την επιλέγει. Τότε μπορεί να την τροποποιήσει αφου ανακτούνται και τα υπόλοιπα χαρακτηριστικά.

Από τα χαρακτηριστικά τροποποιεί τα χαρακτηριστικά που τον ενδιαφέρουν. Τα νέα χαρακτηριστικά αποστέλλονται στον InstrumentController με τη μέθοδο saveInstrument(nserial,ntype,nmusicalInstrument,ncondition,nactive). Τα ιδια χαρακτηριστικά στέλνονται στην κλάση filamanager για αποθήκευση. Στην κλάση filemanager βρήσκουμε την εγγραφή που έχει αλλάξει ο χρήστης από το σύνολο των αποθηκευμένων εγγραφών και την αλλάζουμε με τη μέθοδο set(id,ntitle,ntype,nnotes,nstatus,nnumbe).

Διαχείριση Αρχιμουσικού



Διαγ 6.6 Διαγ Διαχειρισης Αρχιμουσικού

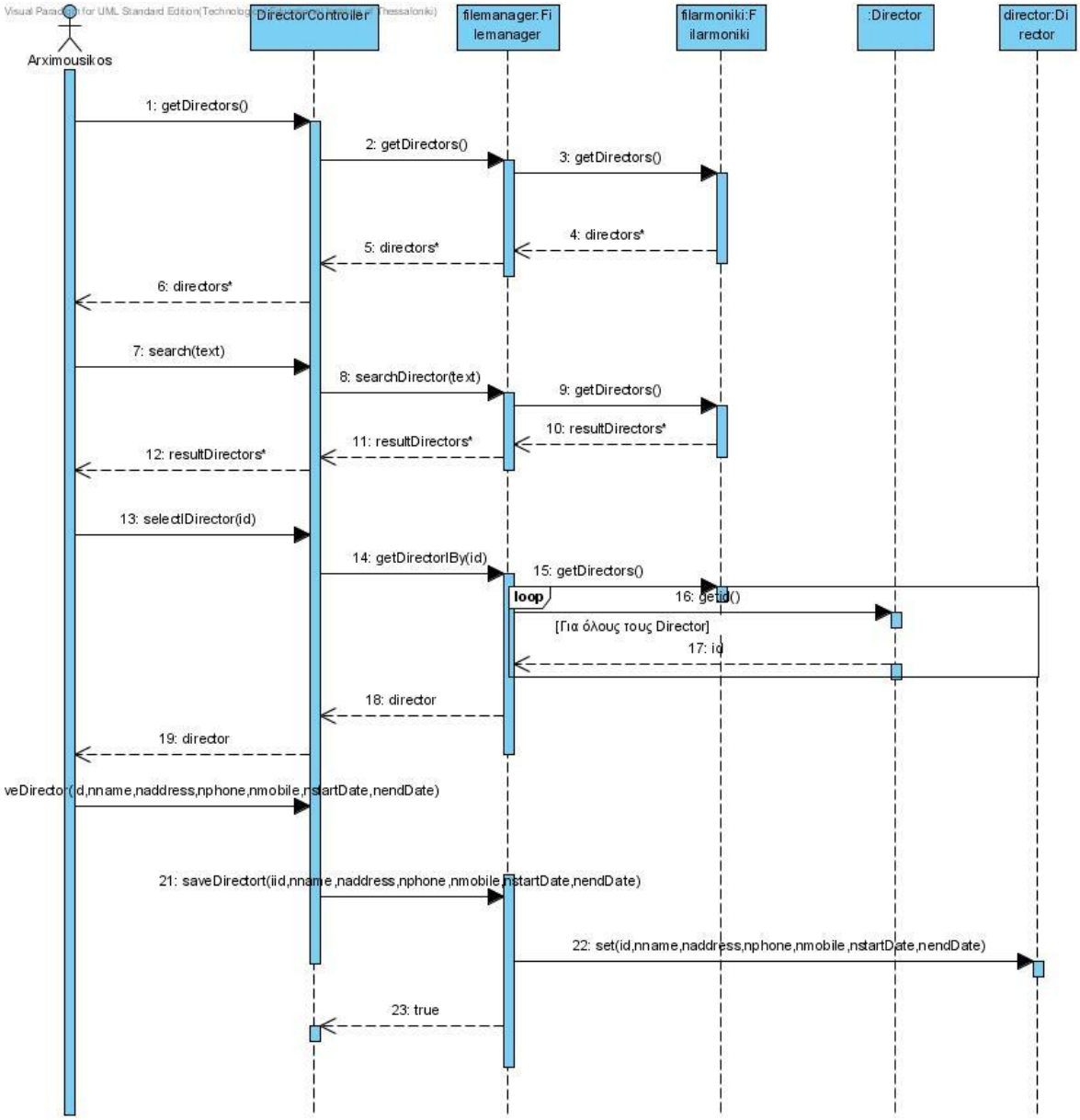
Η διαχείριση αρχιμουσικών είναι υπεύθυνη για την τροποποίηση των αρχιμουσικών και των χαρακτηριστικών τους. Όταν ξεκινάμε τη διαχείριση εμφανίζεται το σύνολο των αρχιμουσικών που «υπηρέτησαν» στη φιλαρμονική. Ο αρχιμουσικός για να διευκολυνθεί μπορεί να δώσει κάποια λέξη κλειδί και να βρεί την εγγραφή από το σύνολο των αρχιμουσικών. Εισάγει το String text οπου μέσω της BandMasterController καιτης μεθόδου search(text) καλούμε την

searchBandMaster (text) από τον filemanager.Στον filemanager περνουμε από τη filarmoniki το σύνολο των αρχιμουσικών (getBandMaster()) και από εκεί ψάχνουμε τις εγγραφές που ταιριάζουν στο text.Όταν βρεθούνε τις επιστρέφουμε στον χρήστη.

Ο χρήστης έχοντας μόνο τις εγγραφές που ταιριάζουν επιλέγει αυτήν που θέλει. Όταν την επιλέξει μπορεί να τροποποιήσει τα χαρακτηριστικά που τον ενδιαφέρουν.

Ο χρήστης όταν τροποποιήσει τις εγγραφές που τον ενδιαφέρουν αποθηκεύει τις αλλαγές. Με την αποθήκευση τα στοιχεία μέσω του controller και της μεθόδου saveBandMaster πηγαίνουν στον filemanager για να τροποποιηθεί η εγγραφή που άλλαξε ο χρήστης και να αποθηκευτεί στο σύστημα.

Διαχείριση Υπεύθυνου



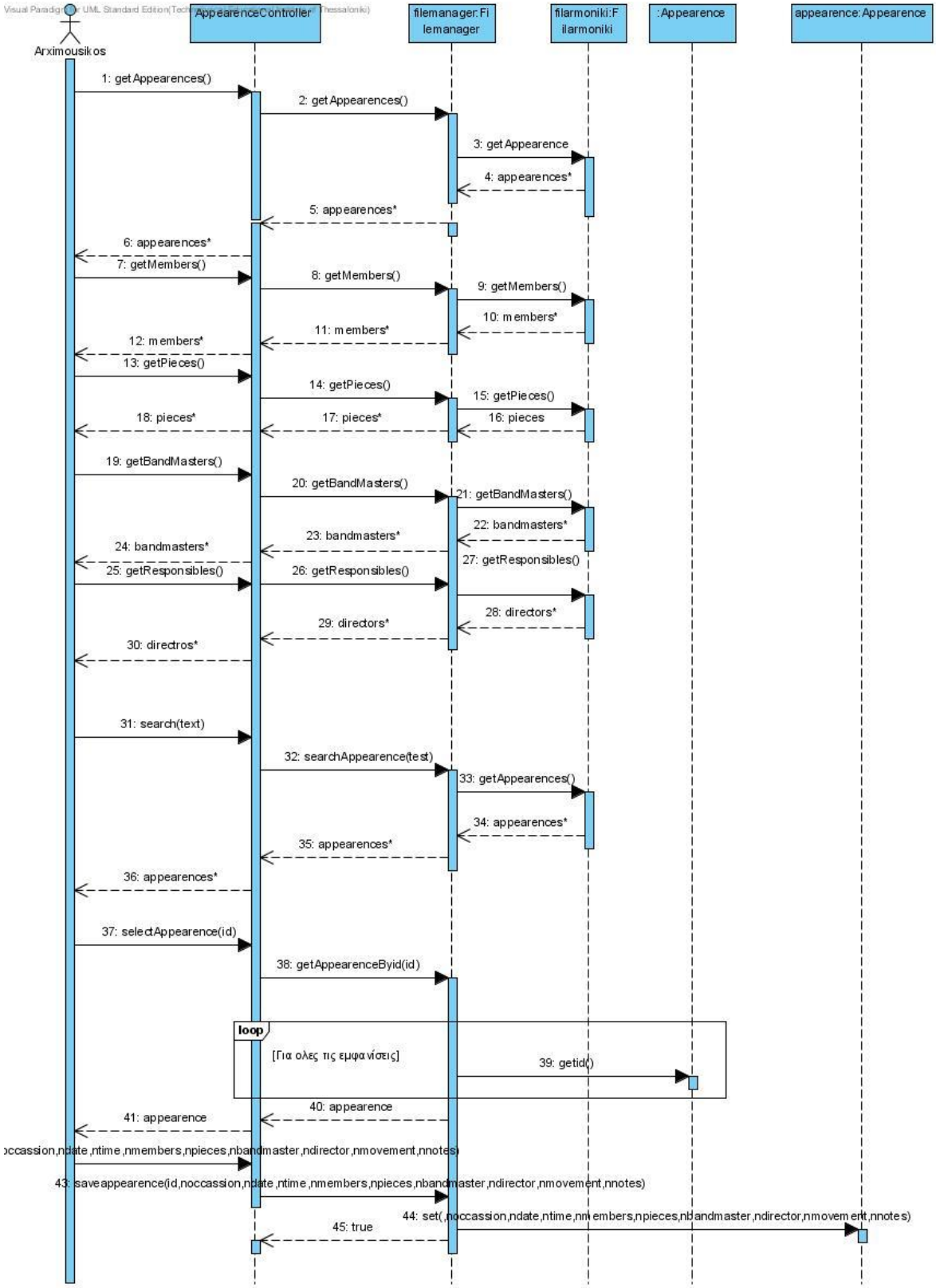
Διαγ 6.7 Διαγ Διαχειρισης Υπεύθυνου

Με την εκκίνηση της διαχείρισης υπεύθυνου ο χρήστης θα πρέπει να εχει ανάκτηση το σύνολο των υλικών υπεύθυνων της φιλαρμονικής. Από τον Material Controller με τη μέθοδο getResponsible() και στη συνέχεια από τις κλάσεις FileManager και filarmoniki παίρνουμε το σύνολο των υπεύθυνων και το επιστρέφουμε στο χρήστη. Ο χρήστης επομένως με το που δέι τη διαχείριση υπεύθυνου έχει το σύνολο των υπεύθυνων. Όταν ο χρήστης δώσει ένα text αυτό το text με τη μέθοδο search(text) του Controller θα προωθηθεί με τη μέθοδο searchResponsiblel(text) του file manager.H file manager παίρνει το σύνολο των υπεύθυνων από τη filamroniki. Γίνεται η σύγκριση των υπεύθυνων με το text και

επιστρέφεται ένα σύνολο από υπεύθυνους. Από αυτούς τους υπεύθυνους ο χρήστης θα επιλέξει την εγγραφή που τον ενδιαφέρει και θα πάρει τα χαρακτηριστικά του .

Αυτά τα χαρακτηριστικά ο χρήστης μπορεί να τα τροποποιήσει. Όταν ο αρχιμουσικός τελειώσει με την τροποποίηση της εγγραφής τα νέα στοιχεία του υπεύθυνου μέσω της μεθόδου save και του controller.Στον filemanager η μέθοδος SaveResponsible παίρνει τα στοιχεία και αφού δει σε ποια εγγραφή της filamronikis αναφέρονται τροποποιεί την αντίστοιχη εγγραφή.

Διαχείριση Εμφάνισης



Διαγ 6.7 Διαγ Διαχειρισης Εμφάνισης

Κατά την εναρξη της διαχείρισης καλούμε από τον AppearanceController τη μέθοδο getAppearence().Με τη σειρά της αυτή η μέθοδος καλεί την

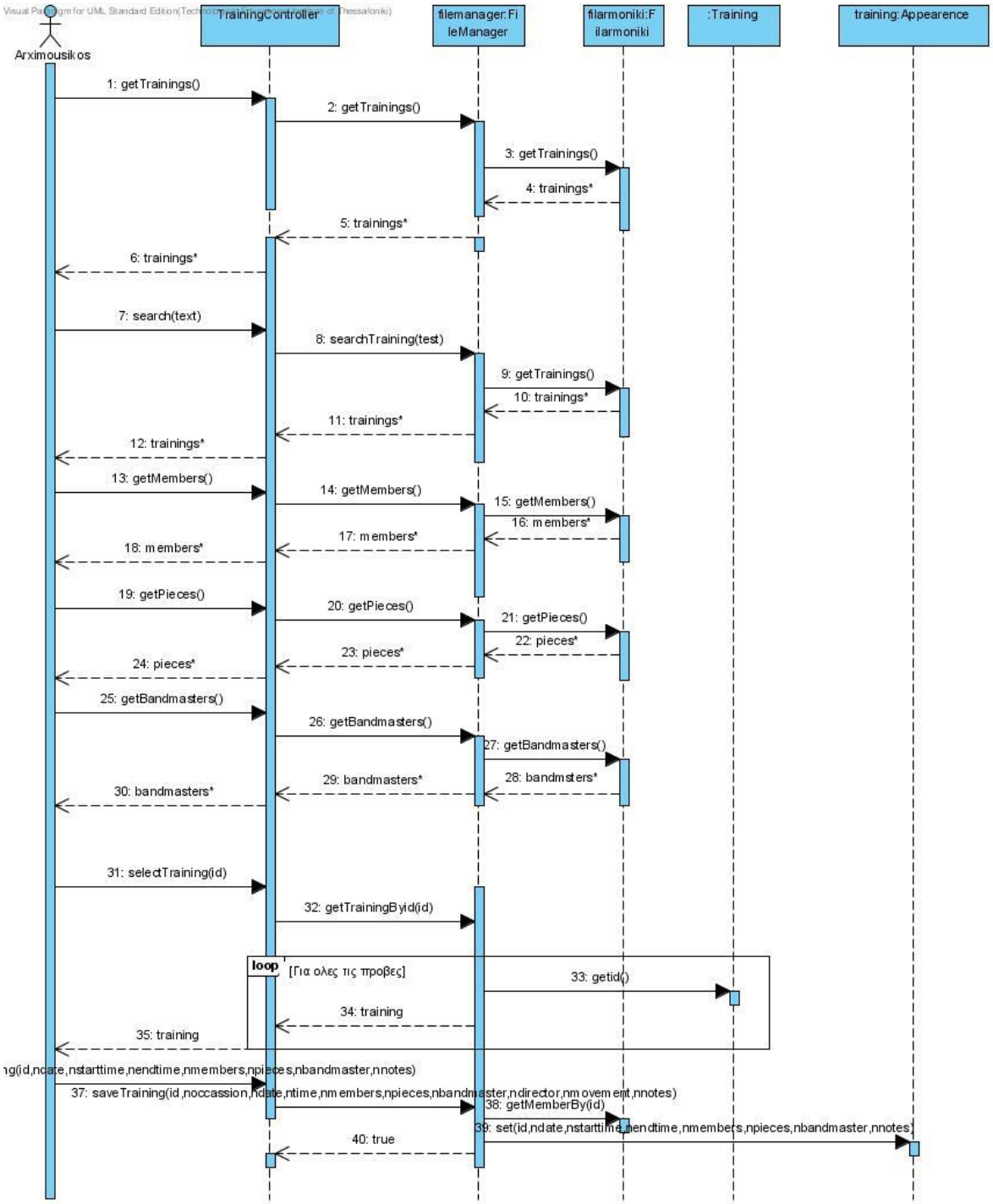
getAppearence() toy filemanager.Στον filemanager καλούμε τη μέθοδο getAppearence από το αντικείμενο filarmoniki για να μας επιστρέψει το σύνολο των εμφανίσεων. Επειδή στις εμφανίσεις συμμετέχουν και άλλες οντότητες θα πρέπει να τις ανακτήσουμε και αυτές (μέλη, μουσικά κομμάτια, αρχιμουσικοί, υπεύθυνοι) οπότε με τις μεθόδους getMembers(), getPieces, getBandMaster(),getResponsible() ανακτούμε τα δεδομένα μας.

Όταν εμφανιστούν στην οθόνη του χρήστη οι εμφανίσεις τότε μπορεί να ψάξει κάποια συγκεκριμένη εγγραφή δίνοντας ένα text για εύρεση. Καλώντας την μέθοδο search(text) από τον Controller.Η search επιστρέφει ένα σύνολο από εμφανίσεις.

Από τις επιστρεφόμενες εμφανίσεις ο χρήστης επιλέγει αυτή που θέλει να τροποποιήσει. Αφού γίνει επιλογή της εγγραφής τότε μπορεί να την τροποποιήσει. Μπορεί να αλλάξει τιμές στα πεδία ή ακόμα να προσθαφαιρέσει κάποιο μέλος ή μουσικό κομμάτι. Επίσης μπορεί να αλλάξει αρχιμουσικό ή υπεύθυνο.

Τελειώνοντας με τις αλλαγές που είναι απαραίτητες αποθηκεύει τις αλλαγές. Τα νέα χαρακτηριστικά της εμφάνισης μέσω της μεθόδου saveappearence(id,noccassion,ndate,ntime,nmembers,npieces,nbandmaster,ndirector,nmovement,nnotes).Στην κλάση filemanger βρίσκουμε την εμφάνιση που αναφερόμαστε από το σύνολο των εμφανίσεων και την τροποποιούμε.

Διαχείριση Πρόβας



Διαγ 6.8 Διαγ Διαχειρισης Πρόβας

Με το άνοιγμα του παραθύρου της διαχείρισης Πρόβας εμφανίζεται θα πρέπει να εμφανιστεί μια λίστα με τις πρόβες που έχουν γίνει. Οπότε θα πρέπει να τις ανακτήσουμε από το αρχείο των προβών. Καλούμε τη μέθοδο getTrainings από τον controller.Ο controller με τη σειρά του καλεί την getTrainings() από την filemanager.Η file manager ανακτά το σύνολο των εγγραφών από τη filarmoniki

και τις προωθεί προς τα πίσω στην filemanager και στον controller που στο τέλος θα εμφανιστούν στον χρήστη.
Επίσης στη φόρμα θα πρέπει να εμφανιστούν από την αρχή και το σύνολο των μελών, μουσικών κομματιών και αρχιμουσικών.
Η ανάκτηση τους γίνεται με την κλάση των μεθόδων getMembers(), getPieces(), getBandMasters().
Όταν γίνει η εμφάνιση των προβών ο αρχιμουσικός θα χρειαστεί να ψάξει από ένα μεγάλο πλήθος προβών μια πρόβα. Θα δώσει στο πεδίο της εύρεσης μια λέξη κλειδί για την αναζήτηση μιας-πολλών εγγραφών. Με τη μέθοδο search(text) και στη συνέχεια με τη μέθοδο searchTraining(text) από την filemanager.Αφού πάρουμε το σύνολο των προβών που ταιριάζουν στη σύγκριση μας από τη filamroniki τις επιστρέφουμε στον χρήστη.
Ο χρηστής επιλέγει την εγγραφή που θέλει. Τα πεδία της φόρμας παίρνουν τις τιμές των χαρακτηριστικών της πρόβας.
Ο χρήστης τροποποιεί τις εγγραφές που τον ενδιαφέρουν. Όταν ολοκληρώσει τις αλλαγές αποθηκεύει τις εγγραφές. Οι νέες τιμές μέσω της μεθόδου save προωθούνται στον filemanager.Ο filemanager βρίσκει την εγγραφή που άλλαξε από την filamroniki και τροποποιεί την εγγραφή από στο σύνολο των αποθηκευμένων προβών και αποθηκεύει το νέο σύνολο.

Επίλογος

Αφού έχουμε δει την ανταλλαγή των μηνυμάτων μεταξύ των αντικειμένων που υπάρχουν στο συστημα μας.Επίσης έχουμε αποφασίσει που ανήκει η κάθε μεθοδος. Επομένως έχουμε μια γενική εικόνα του σχεδιασμού του προγραμματος μας. Επόμενο βήμα μας είναι το διάγραμμα κλάσεων οπου θ έχουμε σε ένα σεδιάγραμμα τις κλάσεις τις μεθόδους αλλά και τα χαρακτηριστικά του συστηματος μας.Οπως επίσης και τις συσχτισεις μεταξυ των κλάσεων

Κεφάλαιο 7
Διαγράμματα Κλάσεων

7.1.1 Διαγράμματα Κλάσεων

Τα αντικειμενοστραφή συστήματα όμως λειτουργούν ως μια συλλογή συνεργαζόμενων αντικειμένων. Τα αντικείμενα αποτελούν στιγμιότυπα κλάσεων, η κλάση είναι ένας τύπος από τον οποίο δημιουργούνται κατά τη διάρκεια της εκτέλεσης του προγράμματος αντικείμενα που ανήκουν στον τύπο αυτό. Είναι λοιπόν χρήσιμο να καταγράψουμε τις κλάσεις που ανήκουν σε ένα σύστημα και τις μεταξύ τους συσχετίσεις. Αυτό ακριβώς επιτυγχάνουμε με ένα διάγραμμα κλάσεων.

Στη UML τα διαγράμματα κλάσεων (class diagrams) αποτελούν βασικό στοιχείο της διαδικασίας μοντελοποίησης και αναπαριστούν τη στατική δομή του συστήματος. Ανάλογα με την πολυπλοκότητα του συστήματος μπορούμε να χρησιμοποιήσουμε ένα διάγραμμα για να μοντελοποιήσουμε ολόκληρο το σύστημα ή πολλά διαγράμματα κλάσεων κάθε ένα από τα οποία μοντελοποιεί συστατικά του συστήματος.

Τα διαγράμματα κλάσεων μπορούν να χρησιμοποιηθούν επίσης για να μοντελοποιήσουμε τα αντικείμενα από τα οποία αποτελείται το σύστημα, για να αναπαραστήσουμε τις μεταξύ τους συσχετίσεις και για να περιγράψουμε τα αντικείμενα και τις υπηρεσίες που προσφέρουν.

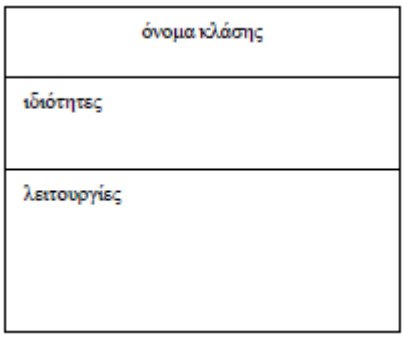
Τα διαγράμματα κλάσεων χρησιμοποιούνται σε διαφορετικά τμήματα της ανάπτυξης του συστήματος. Στην ανάλυση μπορούν να μας βοηθήσουν να κατανοήσουμε τις απαιτήσεις των χρηστών και να προσδιορίσουμε τα βασικά δομικά συστατικά του συστήματος. Σε συστήματα που περιέχουν μόνο λογισμικό, τα διαγράμματα κλάσεων που χρησιμοποιούνται στα αρχικά τμήματα της μεθοδολογίας συχνά μπορούν να μεταφραστούν απευθείας σε κώδικα. Τα μοντέλα του συστήματος που παράγονται στα τμήματα της μεθοδολογίας ανάπτυξης που αφορούν στη συγκέντρωση απαιτήσεων και στην ανάλυση, μπορούν να μετατραπούν σε διαγράμματα κλάσεων που αναπαριστούν συγκεκριμένα

υποσυστήματα, διασυνδέσεις με το χρήστη κτλ. Σε αυτή την περίπτωση τα διαγράμματα κλάσεων αποτελούν μια φωτογραφία του τρόπου λειτουργίας του υπό ανάπτυξη συστήματος, των συστατικών του και των μεταξύ τους σχέσεων σε διάφορα επίπεδα αφαίρεσης καθώς και τον τρόπο υλοποίησης τους.
Με τα διαγράμματα κλάσεων μπορούμε να αναπαραστήσουμε , να ορίσουμε και να τεκμηριώσουμε τα στατικά δομικά χαρακτηριστικά του συστήματος μας. Κατά την ανάλυση και τον σχεδιασμό για παράδειγμα μπορούμε να δημιουργήσουμε διαγράμματα κλάσεων για

- Να ορίσουμε τη δομή των κλάσεων
- Να ορίσουμε τις συσχετίσεις
- Να αναπαραστήσουμε τη δομή ενός μοντέλου χρησιμοποιώντας ιδιότητες , λειτουργίες και μηνύματα.
- Να αναπαραστήσουμε ρόλους και υπευθυνότητες που καθορίζουν τη συμπεριφορά του συστήματος.
- Να αναπαραστήσουμε τη δομή και τη συμπεριφορά μιας ή περισσοτέρων κλάσεων.
- Να αναπαραστήσουμε ιεραρχίες κληρονομικότητας μεταξύ των κλάσεων.

7.1.2ΚΛΑΣΗ, ΙΔΙΟΤΗΤΕΣ ΚΑΙ ΛΕΙΤΟΥΡΓΙΕΣ

Οι κλάσεις αποτελούν τη βάση της κατασκευής οποιουδήποτε αντικειμενοστραφούς συστήματος. Ενσωματώνουν τα δεδομένα καθώς και τις λειτουργίες που επενεργούν στα δεδομένα αυτά. Συμβολίζονται με ένα παραλληλόγραμμο που έχει τρία διαμερίσματα. Στο πάνω διαμέρισμα αναγράφεται το όνομα της κλάσης, στο μεσαίο διαμέρισμα απεικονίζονται οι ιδιότητες και στο κάτω διαμέρισμα οι λειτουργίες. Ανάλογα με τη οπτική γωνία σχεδίασης που χρησιμοποιούμε, η κλάση μπορεί να είναι μία κλάση λογισμικού ή μία έννοια του πεδίου του προβλήματος. Αντίστοιχα, οι λειτουργίες μπορεί να είναι μέθοδοι ή υποχρεώσεις υψηλού επιπέδου.



7.1 Σχήμα Κλάσης

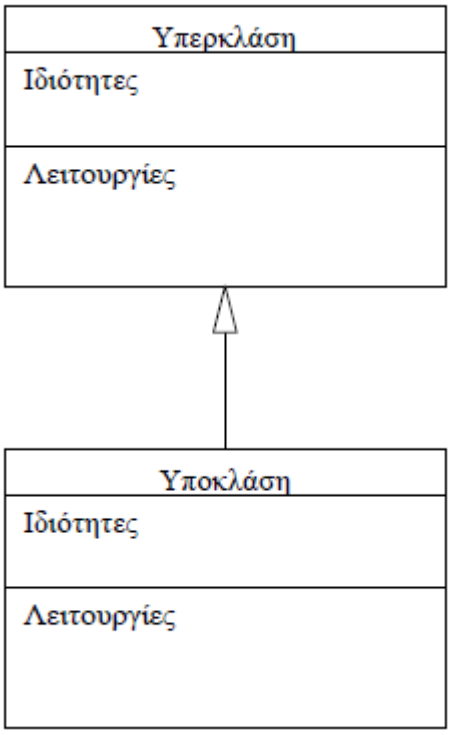
7.1.3 ΣΥΣΧΕΤΙΣΕΙΣ ΚΛΑΣΕΩΝ

Μια συσχέτιση (association) μεταξύ δύο κλάσεων απεικονίζει μια στατική σχέση μεταξύ των δύο κλάσεων. Η ανάγκη μιας συσχέτισης προκύπτει από τη διαπίστωση πως για τη λειτουργία μιας κλάσης απαιτείται η συνεργασία της με μία ή περισσότερες άλλες κλάσεις. Αν αυτή η συνεργασία απαιτείται να είναι σε μόνιμη βάση, τότε χρησιμοποιούμε συσχέτιση, αν είναι παροδική τότε χρησιμοποιούμε εξάρτηση. Για την αναπαράσταση μια συσχέτισης χρησιμοποιούμε ε μία γραμμή μεταξύ των δύο κλάσεων. Κάθε συσχέτιση έχει δύο πέρατα συσχέτισης, κάθε πέρας είναι συνδεδεμένο με μία από τις τάξεις της συσχέτισης. Ένα πέρας μπορεί να ονομαστεί με μία ετικέτα. Το πέρας μιας συσχέτισης έχει επίσης πολλαπλότητα (multiplicity), η οποία είναι μία ένδειξη για το πόσα αντικείμενα μπορούν να συμμετέχουν σε αυτή τη σχέση. Οι γραμμές συσχέτισης έχουν βέλη τα οποία δείχνουν τη δυνατότητα πλοήγησης (navigability). Αν μια δυνατότητα πλοήγησης υπάρχει μόνο στην μια κατεύθυνση, ονομάζουμε τη συσχέτιση μονοδρομη (unidirectional association), ενώ αν υπάρχει δυνατότητα πλοήγησης και προς τις δύο κατευθύνσεις τότε η συσχέτιση ονομάζεται αμφίδρομη.

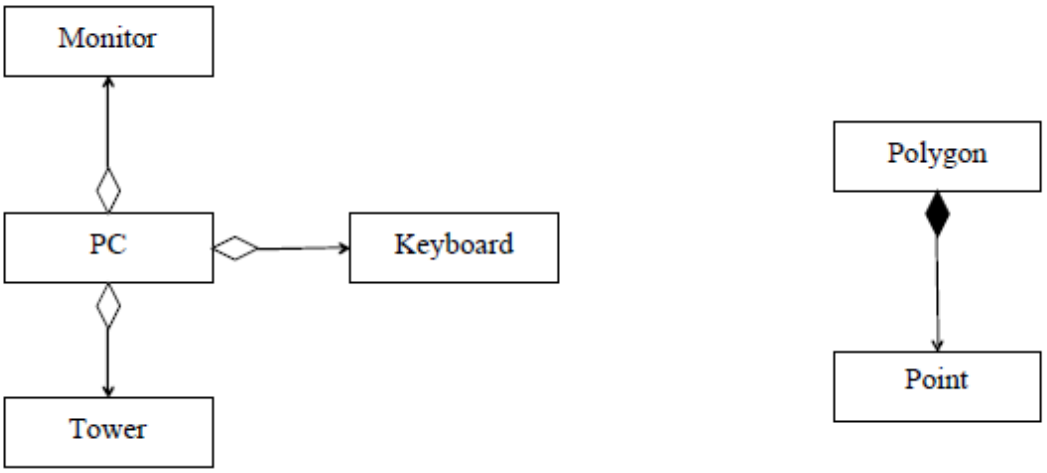
Η γενίκευση είναι μία ειδική μορφή συσχέτισης, κατά την οποία μια γενική κλάση αποτελεί τη βάση για τη δήλωση μίας ή περισσοτέρων ειδικότερων κλάσεων. Η γενική κλάση ονομάζεται υπερκλάση και ειδικές κλάσεις ονομάζονται υποκλάσεις. Η γενίκευση στις περισσότερες γλώσσες προγραμματισμού υλοποιείται με το μηχανισμό της κληρονομικότητας ή της επέκτασης. Η ιδέα είναι πως η γενική κλάση παρέχει λειτουργίες, ιδιότητες και συσχετίσεις που είναι χρήσιμες σε όλες τις υποκλάσεις. Από την άλλη μεριά, οι υποκλάσεις επεκτείνουν τη λειτουργικότητα

της υπερκλάσης και παρέχουν επιπλέον λειτουργίες όπου αυτό είναι απαραίτητο ή εξειδικεύουν τη συμπεριφορά τους.

Η συναρμολόγηση (aggregation) και η σύνθεση (composition) είναι δύο ειδικές περιπτώσεις συσχετίσεων. Και ο δύο υποδηλώνουν τη συσχέτιση μιας κλάσης με κάποια άλλη κλάση που αποτελεί μέρος της. Είναι δηλαδή και οι δύο συσχετίσεις όλου και μερών. Η συναρμολόγηση έχει μόνο έναν επιπλέον περιορισμό σε σχέση με μία συνηθισμένη συσχέτιση, δεν επιτρέπεται η κυκλική συσχέτιση του μέρους με το όλο, άλλα μόνο μία συσχέτιση από το όλο προς το μέρος. Αντιθέτως, η σύνθεση είναι μία ισχυρή μορφή συσχέτισης με δύο επιπλέον σημασιολογικά στοιχεία που τη διαφοροποιούν από τις συνηθισμένες συσχετίσεις, το πρώτο είναι πως το όλο περιέχει αποκλειστικά τα μέρη του, δεν μπορεί δηλαδή κάποιο άλλο όλο να περιέχει το ίδιο αντικείμενο. Το δεύτερο είναι πως δημιουργούνται και καταστρέφονται ταυτόχρονα με το όλο. Μια συναρμολόγηση συμβολίζεται με μία συσχέτιση από το όλο προς το μέρος, στην οποία τοποθετείται ένας άσπρος ρόμβος στην πλευρά του όλου. Η σύνθεση συμβολίζεται με μία συσχέτιση από το όλο προς το μέρος στην οποία τοποθετείται ένας μαύρος ρόμβος στην πλευρά του όλου.



7. 2 Παράδειγμα Γενίκευσης



7.3 Παραδείγματα Συναρμολόγησης και Σύνθεσης

7.2 Διάγραμμα κλάσεων Φιλαρμονικής

Το διάγραμμα κλάσεων της φιλαρμονικής εξάγεται από δύο προηγούμενα διαγράμματα τα διάγραμμα ακολουθίας και το εννοιολογικό μοντέλο.
Οι κλάσεις του προγράμματος μας όπως βγαίνουν είναι οι εξής:

Κλάση Member –Περιγράφει ένα μέλος με χαρακτηριστικά

- Ονοματεπώνυμο→name
- Τηλέφωνο Σταθερό→phone
- Τηλέφωνο Κινητό→mobile
- Διεύθυνση→address
- Ημερομηνία Γεννήσεως→birthday
- Ειδικότητα πχ τρομπετίστας, κλαρινετίστας→position
- Στολές στολές που έχει στην κατοχή του→suits
- Ημερομηνία Εγγραφής πότε εγγράφηκε στη φιλαρμονική→registration day
- Σημειώσεις διάφορες σημειώσεις που αφορούν το μέλος→notes

Επίσης θα προσθέσουμε και μια μεταβλητή id και μία active που θα λειτουργεί ως μοναδικό η id για το μέλος και η active θα ενεργοποιεί ή θα απενεργοποιεί το μέλος από τη λίστα γιατί ένα μέλος δεν μπορεί να διαγραφεί από τη λίστα γιατί μπορεί να συμμετείχε σε μια εμφάνιση.
Οι μέθοδοι του θα είναι οι get set για κάθε χαρακτηριστικό όπως φαίνονται στο διάγραμμα μας.

Κλάση Piece-Μουσικά κομμάτια με χαρακτηριστικά

- Τίτλος →Title
- Αρ Φακέλου →number
- Τύπος→ Type
- Κατάσταση→ Status

- Σημειώσεις→notes

Επίσης θα προσθέσουμε και μια μεταβλητή id που θα λειτουργεί ως μοναδικό χαρακτηριστικό για το κομμάτι.

Κλάση Instrument—Μουσικά Όργανα

- Είδος →instrument(τύπου Musical Instrument)
- Σειριακός αριθμός→serial
- Τύπος→type
- Κατάσταση→condition
- Σημειώσεις→notes

Επίσης θα προσθέσουμε και μια μεταβλητή id και μία active που θα λειτουργεί ως μοναδικό η id για το οργανο και η active θα ενεργοποιεί ή θα απενεργοποιεί το οργανο από τη λίστα γιατί ένα όργανο δεν μπορεί να διαγραφεί από τη
Οι μέθοδοι του θα είναι οι get set για κάθε χαρακτηριστικό όπως φάινονται στο διάγραμμα μας.

Κλάση Material –Υλικό –Εξοπλισμός

- Είδος →item
- Τύπος →type
- Κατάσταση →condition
- Σημειώσεις→notes

Επίσης θα προσθέσουμε και μια μεταβλητή id και μία active που θα λειτουργεί ως μοναδικό η id για το υλικό και η active θα ενεργοποιεί ή θα απενεργοποιεί το υλικό από τη λίστα γιατί το υλικό δεν μπορεί να διαγραφεί από τη λίστα μας καθώς θέλουμε να γνωρίζουμε τι υπήρχε στην κατοχή της φιλαρμονικής
Οι μέθοδοι του θα είναι οι get set για κάθε χαρακτηριστικό όπως φαίνονται στο διάγραμμα μας.

Κλάση BandMaster—Αρχιμουσικός

- Ονοματεπώνυμο→name

- Διεύθυνση→address
- Τηλέφωνο Σταθερό→phone
- Τηλέφωνο Κινητό→mobile
- Πτυχία→ArrayList<String> diplomas
- Ημερομηνία έναρξης →startDate
- Ημερομηνία Λήξης →EndDate

Τέλος προσθέτουμε μια μεταβλητή id ως μοναδικό χαρακτηριστικό του αρχιμουσικού

Κλαση Director –Υπεύθυνος

- Ονοματεπώνυμο→name
- Διεύθυνση→address
- Τηλέφωνο Σταθερό→phone
- Τηλέφωνο Κινητό→mobile
- Ημερομηνία έναρξης →startDate
- Ημερομηνία Λήξης →EndDate

Τέλος προσθέτουμε μια μεταβλητή id ως μοναδικό χαρακτηριστικό του υπεύθυνου

Κλάση Appearance—Εμφάνιση

- Όνομα →occassion
- Ημερομηνία→date
- Ώρα Προσέλευσης→startTime
- Τόπος προσέλευσης→place
- Υπεύθυνος→responsible
- Αρχιμουσικός→bandmaster
- Μέλη→ArrayList members
- Μουσικά κομμάτια→ArrayList piece
- Κίνηση→movement
- Παρατηρήσεις→notes

Τα πεδία υπεύθυνος ,μέλη , κομμάτια , αρχιμουσικός συνδέονται και είναι αντίστοιχου τύπου με τις ομώνυμες κλάσεις.

Κλάση Training— Πρόβα

- Ημερομηνία πρόβας→date
- Ώρα έναρξης Πρόβας→startTime
- Ώρα λήξης πρόβας→endTime
- Μέλη→members
- Κομμάτια→pieces
- Σημειώσεις→notes
- Αρχιμουσικός→bandMaster

Enum suits—στολές

Μπορεί να πάρει δύο τιμές

- Χειμερινή
- Καλοκαιρινή

Enum MusicalInstrument

Είναι αποθηκευμένα τα είδη των μουσικών οργάνων

- Τρομπέτα
- Κλαρινέτο
- Τύμπανο
- Πιατινια
- Σαξόφωνο
- Φλάουτο
- …

Enum movement—Κινηση

Ο τρόπος κίνησης περιέχει τις εξής τιμές

- Πεζή
- Λεωφορείο
- Πλοίο
- Αεροπλάνο

- Αμάξι

FileManager¨--Είναι η κλάση που είναι υπεύθυνη για την αποθήκευση και την ανάκτηση των δεδομένων μας.
Περιέχει τις κλάσεις φια ανάκτηση από αρχείο των δεδομένων μας αλλά καιυπεύθυνη για την επιστροφή πινάκων-συνόλου στιγμιοτύπων από τα αντικείμενα μας(πχ getMember—επιστροφή των μελών).

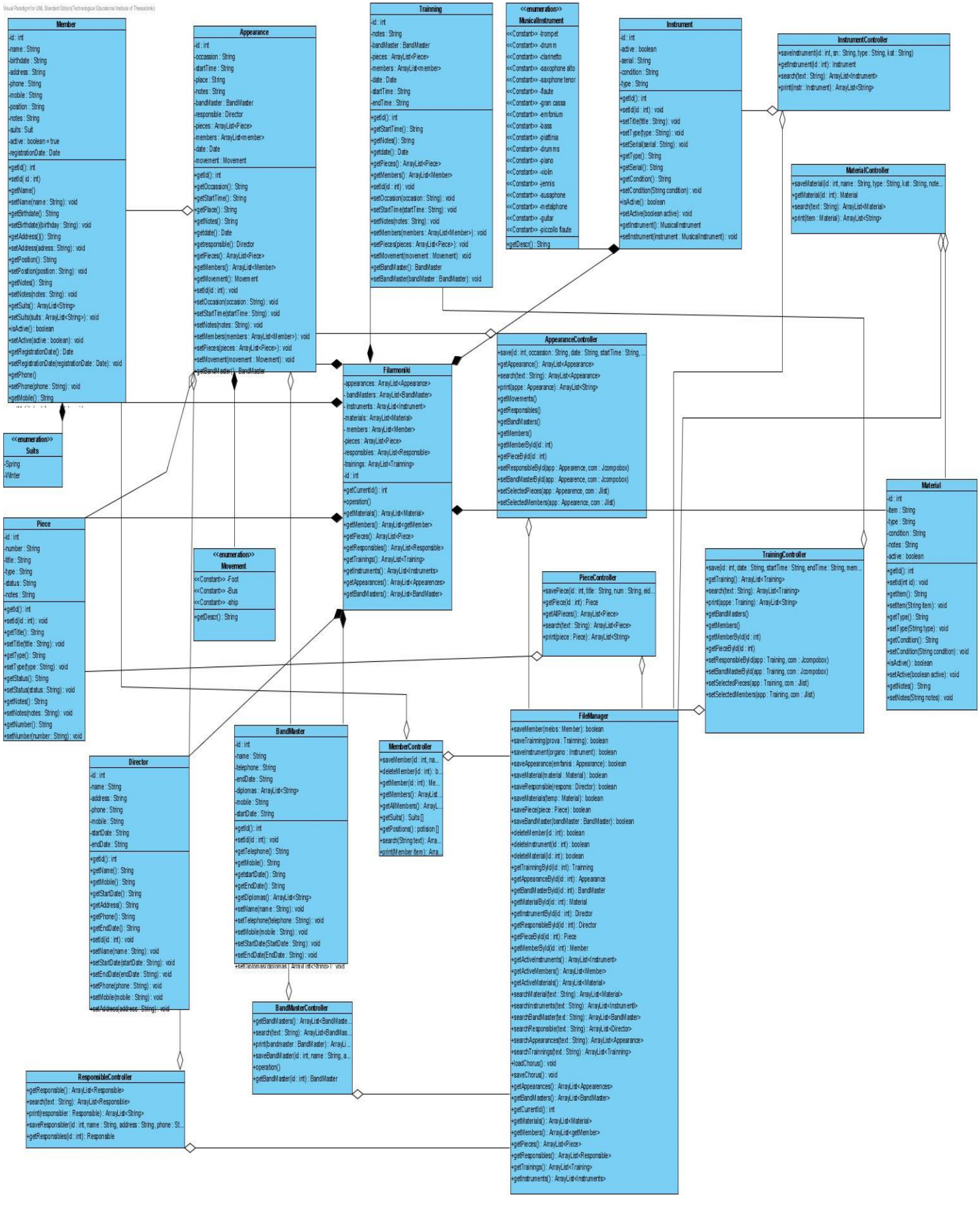
Controllers
(Member,Piece,Material,Appearance,BandMaster,Director,Training,Instrument) Δημιουργήσαμε Controllers χρήσης ώστε να μπορούμε να συνδέσουμε το μοντέλο-κλαση με το interface ώστε το υπόλοιπο πρόγραμμα να είανι οσο το δυνατόν πιο ανεξάρτητο από το interface.Ωστε ανα κανουμε αλλαγές στο interface να μην επηρεάσουν το πρόγραμμα μας αλλά και το αντίστροφο.
Τέλος συντονίζουν τις ενέργειες που πρέπει να γίνουν και με ποια σειρά.

Filarmoniki—Είναι η κλάση που περιγράφει τη φιλαρμονική. Η φιλαρμονική αποτελείται από το σύνολο των μοντέλων και του filemanager.

Συσχετίσεις Φιλαρμονικής

Η κλάση Filarmoniki αποτελείται από αντικείμενα τύπου Member,Piece,Material,Instrument,BandMaster,Director,Appearance,Training οπότε έχουμε Σύνθεση μεταξυ των κλάσεων και της κλάσης Filarmoniki.
Ενας Controller από αντικείμενα τύπου filemanager και της αντίστοιχης κλάσης(πχ στον MemberController από την κλάση Member) οπότε έχουμε συσσωμάτωση. Επίσης στις κλάσεις Aappearence και training περα από τα χαρακτηριστικά τους έχουν και χαρακτηριστικά τύπου Member,Piece,Bandmaster kai Director οπότε εχουμε και εκεί συσσωμάτωση.

7.4 Διαγραμμα Κλάσεων



Κεφάλαιο 8

Κωδικοποίηση Προγράμματος

8.1 Σχολιασμός Κλάσεων βασικών εννοιών(Models)

Κλάση Member

Η κλάση Member είναι υπεύθυνη για την διαχείριση των μελών της φιλαρμονικής. Περιέχει μεταβλητές οπου κρατούνται τα στοιχεία των μελών που θέλουμε να έχουμε καταχωρημένα στο σύστημα μας. Σε αυτήν περιέχονται οι μέθοδοι που είναι υπεύθυνοι για την επικοινωνία με τις υπόλοιπες κλασεις(get,set).

private int id;→ το μοναδικό ιδ-αναγνωριστικό του κάθε μέλους
private String name;→το ονοματεπώνυμο του μέλουε
private String birthdate; →η ημερομηνία γέννησης
private String address;→η διεύθυνση κατοικίας
private phone;→ το σταθερό τηλέφωνο του
privete mobile;→το κινητό του τηλέφωνο
private Position position;→το όργανο στο οποιο ειδικεύεται
private String notes;→καταχωρούνται διάφορες παρατηρήσεις πχ ασθένειες
private ArrayList<Suit> suits;→εδώ μπορούμε να δούμε τι στολές έχει
private boolean active;→Θα μας χρησιμέψει στη διαγραφή του μέλους
private Date registrationDate;→Ημερομηνία εγγραφής

Στο παραπάνω κομμάτι κώδικα δηλώνουμε τις μεταβλητές μας

```
public Member() {
	}
```

```
	public Member(int cod, String on, String im, String die, String
phones, String mobile,
			String idio, String parat, ArrayList<String> suits,
boolean active, Date regDate) {
		id = cod;
		name = on;
		birthdate = im;
		address = die;
		this.phone = phones;
		this.suits = suits;
		position = idio;
```

```
            notes = parat;
            registrationDate = regDate;
            this.active = active;
            this.mobile = mobile;
      }
```

Ο παραπάνω δομητής εισάγει ένα νέο μέλος στο σύστημα μας.Τα στοιχεία τα έχει δώσει ο χρήστης-αρχιμουσικός μοναδικό πεδίο που δεν το δίνει ο χρήστης είναι η ημερομηνία που δινεται αυτόματα από το σύστημα και το id.βάζουμε προσωρινά το id =0 οπου αυτό θα αλλάξει στο επόμενο μεγαλύτερο ελεύθερο αριθμό.

Επειδή οι μεταβλητές στις κλασεις τις έχουμε δηλώσει ως private, αρα δεν μπορούμε να τις καλέσουμε παρα μόνο μέσα στην κλάση. Δημιουργούμε μεθόδους get για να δούμε μια τιμή και set για να αλλάξουμε την τιμή σε μία μεταβλητή.

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

Επιστρέφει το όνομα του μέλους.Θα χρειαστεί στην εμφάνιση της λιστας στο αντίστοιχο view.

Κλάση Piece

Η κλάση Piece περιγράφει το μουσικό κομμάτι. Η φιλαρμονική έχει στην κατοχή της μουσικά κομμάτια. Αυτά τα μουσικά κομμάτια είναι ταξινομημένα σε φακέλους στη βιβλιοθήκη της. Κάθε μουσικό κομμάτι βρίσκεται σε έναν αριθμημένο φάκελος αυτόν τον φάκελο βρίσκονται τα μουσικά κομμάτια ή παρτιτούρες των μουσικών οργάνων. Για να περιγράψουμε ένα μουσικό κομμάτι θα κρατάμε ένα σύνολο από στοιχεία..Αυτά είναι το όνομα του κομματιού, είδος(Marcia funebre),κατάσταση (ελλιπές),διάφορες σημειώσεις.

Οι μεταβλητές της κλάσης

private int id;→Μοναδικό id
private String title;→τίτλος του μουσικού κομματιού
private String type;→τύπος του κομματιού
private String status;→κατάσταση του

```
private String notes;→σημειώσεις
```

Δομητές

```
public Piece() {
}

public Piece(int ide, String tit, String eid, String stat, String para) {
        id = ide;
        title = tit;
        type = eid;
        status = stat;
        notes = para;
}
```

Επειδή οι μεταβλητές στις κλάσεις τις έχουμε δηλώσει ως private, άρα δεν μπορούμε να τις καλέσουμε παρά μόνο μέσα στην κλάση. Δημιουργούμε μεθόδους get για να δούμε μια τιμή και set για να αλλάξουμε την τιμή σε μία μεταβλητή.

Κλάση Instrument

Αυτή η κλάση περιγράφει τα μουσικά όργανα που υπάρχουν στη φιλαρμονική. Ένα μουσικό όργανο έχει ένα όνομα (sax-alto Yamaha 54-9), σειριακό αριθμό, τύπο(σαξόφωνο) επίσης κρατάμε την κατάσταση στην οποία βρίσκεται (λείπει κουμπί χειρισμού).

```
private int id;→Μοναδικό id
private String serial;→σειριακός αριθμός οργάνου
private String type;→τύπος μουσικού οργάνου
private String condition;→κατάσταση
private String name;→ονομα-περιγραφή
```

Δομητής

```
public Instrument(int id1, String sn, Position instrument, String nam, String
typ, String kat) {
        id = id1;
        serial = sn;
        this.instrument = instrument;
        type = typ;
        condition = kat;
        name=nam;}
```

Επειδή οι μεταβλητές στις κλάσεις τις έχουμε δηλώσει ως private, άρα δεν μπορούμε να τις καλέσουμε παρά μόνο μέσα στην κλάση. Δημιουργούμε μεθόδους get για να δούμε μια τιμή και set για να αλλάξουμε την τιμή σε μία μεταβλητή.

Κλαση Material

Η κλάση Material είναι υπεύθυνη για τα αντικείμενα τύπου εξοπλισμού.Δηλαδή τον εξοπλισμό που έχει η φιλαρμονική στη διαθεσή της όπως φωτοτυπικά, καρέκλες, ηχεία, μικρόφωνα εκτός των μουσικά οργανα.

private int id;→το μοναδικό id για κάθε είδος

private String item;→αναγράφεται ο σειριακός αριθμός του μηχανήματος

private String type;→το είδος του μηχανήματος πχ ηχειο,μικρόφωνο

private String condition;→η κατάσταση που βρίσκεται ή προβήματα που έχει

private String name;→η περιγραφή του είδος αναλυτικά πχ olliveti AK-500F

```
public Material(int id1, String sn, String nam, String typ, String kat) {
		id = id1;
		item=sn;
		name=nam;
		type=typ;
		condition=kat;
	}
```

Ο δομητής του της κλάσης με πεδία

Επειδή οι μεταβλητές στις κλάσεις τις έχουμε δηλώσει ως private, άρα δεν μπορούμε να τις καλέσουμε παρά μόνο μέσα στην κλάση. Δημιουργούμε μεθόδους get για να δούμε μια τιμή και set για να αλλάξουμε την τιμή σε μία μεταβλητή.

Κλάση BandMaster

Η κλάση BandMaster περιγράφει τον αρχιμουσικό. Ο αρχιμουσικός είναι ο δάσκαλος-μαέστρος και υπεύθυνος για τη διαχείριση, πρόβες, εμφανίσεις τις φιλαρμονικής. Κράτάει στοιχεία όπως το όνομα του, τηλέφωνα, διεύθυνση,

διπλώματα(φούγκας,σύνθεσης) ημερομηνία έναρξης συνεργασίας, ημερομηνία λήξης συνεργασίας.

private int id;→μοναδικό id
private String name;→το ονοματεπώνυμο του αρχιμουσικού
private String telephone;→Το σταθερό τηλέφωνο του
private String mobile;→το κινητό του τηλέφωνο
private String address;→Διεύθυνση κατοικίας
private ArrayList<String> diplomas;→τα διπλώματα που έχει
private Date startDate;→ημερομηνία εναρξης της συνεργασίας
private Date endDate; →ημερομηνία λήξης συνεργασίας

Δομητής

```
public BandMaster(int i, String nam, String ts, String tk, String di,ArrayList<String> pt, Date ie, Date il) {
		id = i;
		name = nam;
		telephone = ts;
		mobile = tk;
		address = di;
		diplomas = pt;
		startDate = ie;
		endDate = il;
	}
```

Επειδή οι μεταβλητές στις κλάσεις τις έχουμε δηλώσει ως private, άρα δεν μπορούμε να τις καλέσουμε παρά μόνο μέσα στην κλάση. Δημιουργούμε μεθόδους get για να δούμε μια τιμή και set για να αλλάξουμε την τιμή σε μία μεταβλητή.

Κλάση Responsible

Στην κλάση Responsible περιγράφουμε τον υπεύθυνο.Ο υπέυθυνος είναι το άτομο που είναι διορισμένο από τη δημοτική αρχή και είναι υπέυθυνο να μεταφέρει τις ανάγκες της φιλαρμονικής αλλά και να διαχειρίζεται τα οικονομικά.Τα στοιχεία που

πρέπει να κρατούνται για τον υπεύθυνο είναι το ονομα του,διεύθυνση,τηλέφωνα,ημερ εναρξης θητείας,ημερ. λήξης θητείας.

Μεταβλητές

int id;→Μοναδικό id
String name;→ονοματεπώνυμο υπεύθυνου
String phone;→Σταθερό τηλέφωνο
String mobile;→κινητό τηλέφωνο
String address;→διεύθυνση
Date startDate;→ημερομηνία έναρξης θητείας
Date endDate;→ημερομηνία λήξης θητείας

Δομητές

```
public Responsible() {
	}

	public Responsible(int i, String na, String ts, String tk, String di,
			Date ie, Date il) {
		id = i;
		name = na;
		phone = ts;
		mobile = tk;
		address = di;
		startDate = ie;
		endDate = il;
	}
```

Επειδή οι μεταβλητές στις κλάσεις τις έχουμε δηλώσει ως private, άρα δεν μπορούμε να τις καλέσουμε παρά μόνο μέσα στην κλάση. Δημιουργούμε μεθόδους get για να δούμε μια τιμή και set για να αλλάξουμε την τιμή σε μία μεταβλητή.

Κλάση Appearance

Η κλάση Appearance περιγράφει μια εμφάνιση της φιλαρμονικής. Κάθε εμφάνιση πρέπει να καταγράφεται σε ένα αρχείο. Θα καταγράφονται στοιχεία όπως ημερομηνία ώρα προσέλευσης, το είδος της εμφάνισης(παρέλαση, συναυλία) τα άτομα που συμμετείχαν, ποιος ήταν ο υπεύθυνος κομμάτια που παίχθηκαν, τόπος συγκέντρωσης, και τρόπο που κινηθήκαν για να πάνε στην εμφάνιση.

Μεταβλητές

int id;
String occassion;→είδος
Date date;→ημερομηνία
String startTime;→ώρα προσέλευσης
String place;→τόπος προσέλευσης
Responsible responsible;→υπεύθυνος
BandMaster bandmaster; →αρχιμουσικός
ArrayList<Piece> pieces;→μουσικά κομμάτια
ArrayList<Member> members;→μέλη που συμμετείχαν
String notes;→σημειώσεις
Movement movement→κινηση

Δομητές

```
public Appearance() {
	}

	public Appearance(int i, String ei, Date im, String ora, String top,
			Responsible ip, BandMaster a, String para, ArrayList<Piece> pieces,
	ArrayList<Member> members, Movement movement) {
		id = i;
		occasion = ei;
		date = im;
		startTime = ora;
		place = top;
		responsible = ip;
		bandmaster = a;
		notes = para;
		this.pieces = pieces;
		this.members = members;
		this.movement = movement;
	}
```

Κρατάμε ένα σύνολο από κομμάτια. Αυτά τα κομμάτια υπάρχουν στο ρεπερτόριο της φιλαρμονικής άρα έχουμε τα στοιχεία τους.

```
	public void setMembers(ArrayList<Member> members) {
		this.members = members;
	}
```

Ένα σύνολο από αντικείμενα τύπου Member.Ειναι κάποια από τα μέλη που είναι εγγραμμένα στη φιλαρμονική

```
	public void setMovement(Movement movement) {
	this.movement = movement;
	}
```

Επειδή οι μεταβλητές στις κλάσεις τις έχουμε δηλώσει ως private, άρα δεν μπορούμε να τις καλέσουμε παρά μόνο μέσα στην κλάση. Δημιουργούμε μεθόδους get για να δούμε μια τιμή και set για να αλλάξουμε την τιμή σε μία μεταβλητή.

Κλάση Training

Στην κλάση Training περιγράφει μια πρόβα πρόβα περιλαμβάνει γενικές πληροφοριές όπως η ημερομηνία διεξαγωγής, ώρα έναρξης, ώρα λήξης, σημειώσεις. Επίσης περιλαμβάνει τον αρχιμουσικό που τη διεύθυνε τα μέλη που συμμετείχαν όπως και τα κομμάτια που παίχθησαν

Μεταβλητές

```
	private int id;→μοναδικός κωδικός
	private String date;→ημερομηνία πρόβας
	private String startTime;→ώρα εναρξης
	private String endTime;→ωρα λήξης
	private BandMaster bandMaster;→αρχιμουσικός
	private ArrayList<Piece> pieces;→κομματια που παίχθηκαν
	private ArrayList<Member> members;→μέλη που συμμετείχαν
	private String notes;→σημειώσεις παρατηρήσεις
```

Δομητές

```
public Trainning(int id,  String date, String startTime,String endTime,
BandMaster bandMaster, String notes, ArrayList<Piece> pieces,
	ArrayList<Member> members) {
		this.id = id;
		this.date = date;
		this.startTime = startTime;
		this.endTime = endTime;
		this.bandMaster = bandMaster;
		this.notes = notes;
		this.pieces = pieces;
		this.members = members;
	}
```

```
public void setPieces(ArrayList<Piece> pieces) {
```

```
            this.pieces = pieces;
      }
```

Κρατάμε ένα σύνολο από κομμάτια. Αυτά τα κομμάτια υπάρχουν στο ρεπερτόριο της φιλαρμονικής άρα έχουμε τα στοιχεία τους.

```
      public void setMembers(ArrayList<Member> members) {
            this.members = members;
      }
```

Ένα σύνολο από αντικείμενα τύπου Member.Ειναι κάποια από τα μέλη που είναι εγγραμμένα στη φιλαρμονική

Επειδή οι μεταβλητές στις κλάσεις τις έχουμε δηλώσει ως private, άρα δεν μπορούμε να τις καλέσουμε παρά μόνο μέσα στην κλάση. Δημιουργούμε μεθόδους get για να δούμε μια τιμή και set για να αλλάξουμε την τιμή σε μία μεταβλητή.

Κλάση File Manager

Η FileMAnager είναι υπεύθυνη για την διαχείριση των δεδομένων μας.

```
private void loadFilarmoniki(){
            ObjectInputStream in = null;

            try {
                  in          =          new          ObjectInputStream(new
FileInputStream("filarmoniki.dat"));
                  filarmoniki = (Filarmoniki)in.readObject();

            } catch (FileNotFoundException e) {
                  e.printStackTrace();
            } catch (EOFException e) {
                  System.out.println("No  filarmoniki  found.  creating  a
new one.");
                  filarmoniki = new Filarmoniki();
            } catch (IOException e) {
                  e.printStackTrace();
            } catch (ClassNotFoundException e) {
                  e.printStackTrace();
            }finally{
                  if(in != null){
                        try {
                              in.close();
                        } catch (IOException e) {
                              e.printStackTrace();
                        }
                  }
            }
      }
```

Η loadChorus() μέθοδος είναι υπεύθυνη για την ανάκτηση των δεδομένων μας από το αρχείο. Σε περίπτωση που δεν βρεθεί το αρχείο επιστρέφει μηνυμα λάθους

```
private void saveFilarmoniki(){
            ObjectOutputStream out = null;

            try {
                  out         =         new         ObjectOutputStream(new
FileOutputStream("filarmoniki.dat"));
                  out.writeObject(filarmoniki);

            } catch (FileNotFoundException e) {
                  e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            }finally{
                  if(out != null){
                        try {
                              out.close();
                        } catch (IOException e) {
                              e.printStackTrace();
                        }
                  }
            }
      }
```

Η saveChorus() παίρνει το αντικείμενο filarmoniki και το αποθηκεύει στο αρχείο δεδομένων μας.

```
public boolean saveMember(Member newItem) {
            ArrayList<Member> list = getMembers();

            if(newItem.getId()  ==  0){  //  An  to  id  einai  0  exoume
dhmiourgia neou
                  newItem.setId(filarmoniki.getCurrentId());  //  bazoume
ena egkuro id
                  newItem.setRegistrationDate(new Date()); //bazoume thn
twrinh hmeromhnia ws hmeromhnia eggrafhs
                  list.add(newItem); // to pros8etoume sthn lista

            }else{ // An to id den einai 0 exoume tropopoihsh
                  Member temp;
                  for(int i=0; i<list.size(); i++) { //diatrexei olh th
lista
                        temp = list.get(i);
                        if(temp.getId()==newItem.getId()) { // An bre8ei
to melos
                              //list.set(i, newItem); // To antika8ista
me auto pou periexei ta nea stoixeia
                              temp.setActive(newItem.isActive());
                              temp.setAddress(newItem.getAddress());
                              temp.setBirthdate(newItem.getBirthdate());
                              temp.setMobile(newItem.getMobile());
                              temp.setName(newItem.getName());
                              temp.setNotes(newItem.getNotes());
                              temp.setPhone(newItem.getPhone());
                              temp.setPosition(newItem.getPosition());

      temp.setRegistrationDate(newItem.getRegistrationDate());
```

```
                                temp.setSuits(newItem.getSuits());
                          }
                    }
              }
              saveFilarmoniki();
              return true;
        }
```

Η saveMember πένρει μια εγγραφή ενός μέλους και την αποθηκεύει στο σύστημα. Αφού πρώτα ελέγξει αν η εγγραφή υπάρχει τότε όταν τη βρέι αντικαθιστά τα χαρακτηριστικά με τα νέα χαρακτηριστικά .

```
public ArrayList<Member> searchMembers(String text) {
            ArrayList<Member> list = getActiveMembers();
            ArrayList<Member> resultList = new ArrayList<Member>();
            Member member = new Member();
            boolean alreadyAdded = false;

            //an den psaxnoume gia to keno
            if(text!=null && !text.equals("")) {

                  for(int i=0; i< list.size(); i++) { //diatrexoume th
lista
                        alreadyAdded = false; //"mhdenizoume" to flag
                        member = list.get(i); //pairnoume to iosto
stoixeio ths listas
                        if(member.getName().contains(text))     {     //
elegxoume an periexetai to string sto name
                              resultList.add(member);
                              alreadyAdded = true; // an bre8ei to
pros8etoume sth lista pou 8a epistrepsoume kai to shmeiwnoume, wste na
mhn to 3anabaloume
                        }
                        if(member.getAddress().contains(text)          &&
!alreadyAdded) { // elegxoume an periexetai to string sto address, efoson
den to exoume hdh sth lista
                              resultList.add(member);
                              alreadyAdded = true;
                        }
                        if(member.getNotes().contains(text)            &&
!alreadyAdded) { // elegxoume an periexetai to string sto notes, efoson
den to exoume hdh sth lista
                              resultList.add(member);
                              alreadyAdded = true;
                        }
                        if(member.getPhone().contains(text)            &&
!alreadyAdded) { // elegxoume an periexetai to string sto phone, efoson
den to exoume hdh sth lista
                              resultList.add(member);
                              alreadyAdded = true;
                        }
                        if(member.getPosition().contains(text)         &&
!alreadyAdded) { // elegxoume an periexetai to string sto position,
efoson den to exoume hdh sth lista
                              resultList.add(member);
                              alreadyAdded = true;
```

```
                        }
                  }
                  return resultList;
            } else { //an psaxnoume gia to keno string tote epistrefetai
olh h lista
                  return list;
            }
      }
```

Η `searchMembers` δέχεται ένα String text και ελένχει σε όλα τα χαρακτηριστικά του αν υπάρχει αυτό το text.Σε όσα υπάρχει τα εισάγει σε μια λίστα και τα επιστρέφει .

```
public Member getMemberById(int id) {
            ArrayList<Member> list = getMembers();

            Member temp;
            for(int i=0; i<list.size(); i++) { //diatrexei olh th lista
                  temp = list.get(i);
                  if(temp.getId()==id) {
                        return temp; //an to brei to epistrefei
                  }
            }
            return null; //an eftase ws edw tote den exei bre8ei kati,
opote epistrefei null
      }
```

Η getMemberById(int id) επιστρέφει το μέλος με ένα συγκεκριμένο id αφού πρώτα σαρώσει ολη τη λίστα των μελών.

Αντίστοιχες μεθόδοι υπάρχουν για τις υπόλοιπες κλάσεις(Piece, Material, Instrument, BandMaster, Director, Appearance, Training)

8.1.2 Λόγοι επιλογής Αρχείων από ΣΔΒΔ για αποθήκευση Δεδομένων

Το πρόγραμμα έχει κατασκευαστεί ώστε η αποθήκευση των δεδομένων να γίνεται σε αρχεία αντικειμένων. Αυτό έγινε για τους εξής λόγους:

1)Το πρόγραμμα γίνεται αυτόνομο. Δηλαδή δεν υπάρχει η ανάγκη εγκατάστασης προγράμματος διαχείρισης βάσης δεδομένων. Όπως επίσης θα υπάρχουν και λιγότερα προβλήματα(θέματα σύνδεσης στη βάσης δεομένων)

2)Μειώνεται η πολυπλοκότητα. Το πρόγραμμα διαχειρίζεται μονό του τα δεδομένα και τα αποθηκεύει σε αρχεία αντικειμένων. Όποτε έχουμε απόλυτο έλεγχο της μορφής των δεδομένων μας.

3)Δε θα χρειαστεί να γίνει εγκατάσταση επιπλέον προγράμματος ή παραμετροποίηση του λειτουργικού συστήματος για τη δημιουργία σύνδεσης βάσης δεδομένων-προγράμματος που ενδεχομένως να δυσκολέψει το χρήστη.

4)Το πρόγραμμα θα είναι εύκολα μεταφέρσιμο. Επειδή τα δεδομένα θα αποθηκεύονται μαζί με το πρόγραμμα στο φάκελο του προγράμματος τότε με την απλή αντιγραφή του φακέλου θα έχουμε πρόγραμμα και τα δεδομένα .Αυτό θα είναι πολύ χρήσιμο για τον αρχάριο χρήστη του προγράμματος γιατί θα μπορεί να το μεταφέρει σε διάφορους υπολογιστές και να έχει πάντα την πληροφόρηση που θέλει αποθηκεύοντας το πρόγραμμα σε ένα απλό usb stick.Οπότε θα το τρέχει και θα διαχειρίζεται το πρόγραμμα σε οποιοδήποτε Η\Υ θέλει πολύ εύκολα.

5)Λιγότερες απαιτήσεις στο συστήματος. Επειδή δεν θα τρέχει κάποιο ΣΔΒΔ που χρησιμοποιεί αρκετούς πόρους του συστήματος θα μπορεί η εφαρμογή να τρέχει ακόμα και σε παλιότερους Η/Υ ή και υπολογιστές netbook.

6) Ο όγκος των δεδομένων δεν είναι τέτοιος ώστε να κάνει επιτακτική την ανάγκη χρήσης βάσης δεδομένων. Οπότε με τη χρήση βάσης δεδομένων θα χρειαζόμασταν περισσότερο ελεύθερο χώρο στον σκληρό δίσκο για την εγκατάσταση του προγράμματος αλλά και επιπλέον χώρο για τα δεδομένα μας.

8.2 Σχολιασμός Controllers

Controller Μέλους

Ο Controller του μέλους είανι η κλάση που συνδέει το μοντέλο-κλάση του μέλους με το View και την διαχείριση αρχείων(File Manager).

```
public boolean saveMember(int id, String name, String birthdate, String
address, String phone, String mobile, String position, String
paratiriseis, ArrayList<String> suits, boolean active, Date regDate) {

            melos = new Member(id, name, birthdate, address, phone,
mobile, position, paratiriseis, suits, active, regDate);

            return fm.saveMember(melos);
      }
```

Η μέθοδος saveMember είναι η αποθήκευση-τροποποίηση ενός μέλους. Παίρνουμε τα στοιχεία από την διεπαφή δημιουργούμε ένα αντικείμενο μέλος και αυτό το μέλος το αποθηκεύουμε στο αρχείο δεδομένων μας.

```
public Member getMember(int id) {
      melos = fm.getMemberById(id);
      return melos;
}
```

Επιστρέφει το μέλος με συγκεκριμένο id

```
public ArrayList<Member> search(String text) {
            return fm.searchMembers(text);
      }
```

Δινει ένα string επιστρέφει ένα ArrayList με τα μέλη που περιέχουν συτό το String

```
public ArrayList<String> print(Member item){
            ArrayList<String> result = new ArrayList<String>();

            result.add("" + item.getId());
            result.add(item.getName());
            result.add(item.getBirthdate());
            result.add(item.getAddress());
            result.add(item.getPhone());
            result.add(item.getMobile());
            result.add(item.getPosition());
            result.add(item.getNotes());

            String temp = "";
            for(int i=0; i<item.getSuits().size(); i++){
                  temp += item.getSuits().get(i);
            }
```

```
            result.add(temp);

            if(item.isActive()){
                  result.add("Ναι");
            }else{
                  result.add("Οχι");
            }
            result.add(item.getRegistrationDate().toString());

            return result;
      }
```

Η print(Member item)παίρνει σαν ορισμα ένα μέλος.Απο το μέλος αυτό δημιουργεί έναν ArrayList οπου εκεί αποθηκεύονται σε μορφή String τα στοιχεία του μέλους.

```
public void printList(ArrayList<Member> list){
            try {
                  PrintStream           writer           =           new
PrintStream("members_list.txt");
                  ArrayList<String> output;

                  output = getLabels();
                  for(int i=0; i<output.size(); i++){
                        writer.printf("|%-20s", "" + output.get(i));
                  }
                  writer.println();
                  writer.println("--------------------");

                  for(int j=0; j<list.size(); j++){
                        output = print(list.get(j));
                        for(int i=0; i<output.size(); i++){
                              writer.printf("|%-20s",         ""         +
output.get(i));
                        }
                        writer.println();
                        writer.println("--------------------");
                  }

                  writer.close();
            } catch (IOException e) {
                  e.printStackTrace();
            }
            }
```

Η `printList(ArrayList<Member> list)` παίρνει μια λίστα μελών και την εκτυπώνει σε ένα αρχείο txt. Η εργασία γίνεται με τη βοήθεια της print μεθόδου. Η μορφή της εκτύπωσης είναι είναι εγγραφές μια κάτω από την άλλη χωρισμένες με τον χαρακτήρα «|».Επίσης εκτυπώνονται και οι λεζάντες-ονομα στηλών.

```
public void printSelected(Member selected){
            try {
```

```
                  PrintStream writer = new PrintStream("member.txt");
                  ArrayList<String> labels = getLabels();
                  ArrayList<String> output = print(selected);

                  for(int i=0; i<output.size(); i++){
                        writer.printf("%s:      %s",      labels.get(i),
output.get(i));
                        writer.println();
                  }


                  writer.close();
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
```

Η printSelected(Member selected) από ένα μέλος που παίρνει εκτυπώνει τα στοιχεία του σε ένα αρχείο txt.Η εκτύπωση μοιάζει σαν ετικέτα του μέλους.
Αντίστοιχες μέθοδοι υπάρχουν και για τις υπόλοιπες κλάσεις.

8.3 Views
Διαχείριση μέλους

Save Button

```
JButton saveButton = new JButton("Αποθήκευση");
        saveButton.addActionListener(new ActionListener() {

                  public void actionPerformed(ActionEvent e) {
                  ArrayList<String> suits = new ArrayList<String>();
                  Object[]                  arr                  =
memberSuitsList.getSelectedValues();

                  //apo8hkeyoume ola ta epilegmena koustoumia se ena
arraylist
                  for(int i=0; i<arr.length; i++){
                        suits.add((String)arr[i]);
                  }
                  //kaloume   thn   katallhlh   synarthsh   apo   ton
controller kai afhnoume olous tous elegxous gia ekei
                        memberController.saveMember(memberId,
                                    memberName.getText(),
                                    memberBirthDate.getText(),
                                    memberAddress.getText(),
                                    memberPhone.getText(),
                                    memberMobile.getText(),
```

```
      (String)memberPositionComboBox.getSelectedItem(),
                                    memberNotes.getText(),
                                    suits,
                                    memberActive.isSelected(),
                                    memberRegDate
                              );
                        //ananewnoume th lista wste na fanoun oi
allages se pragmatiko xrono
                        refreshMemberList();
                  }
            });
```

Με το κουμπί αποθήκευσης αποθηκεύουμε ένα νέο μέλος που δεν υπήρχε καταχωρημένο στο σύστημα μας ή την τροποποίηση ενός υπάρχοντος .
Κατ αρχήν δημιουργούμε ένα jButton με όνομα Αποθήκευση. Προσθέτουμε έναν ActionListener οπου του λέμε τι θα στείλει στον controller για αποθήκευση. Κατ αρχήν επειδή τα έχουμε να κάνουμε με μια λίστα από κουστούμια και ο χρήστης από αυτή τη λίστα έχει επιλέξει κάποια κουστούμια θα πρέπει αυτά τα κουστούμια να τα αποθηκεύσουμε σε ένα array list και την ονομάζουμε suit.
Στη συνέχεια παίρνουμε τις τιμές που έχει καταχωρίσει ο χρήστης στα πεδία και τις εισάγουμε για να τις περάσουμε στον Controller με τη χρήση της μεθόδου saveMember.Τέλος ανανεώνουμε τη λίστα με τα μέλη.

Button Διαγραφής

```
JButton deleteButton = new JButton("Διαγραφή");
        deleteButton.addActionListener(new ActionListener() {
                  @Override
                  public void actionPerformed(ActionEvent e) {
                        //kaloume thn katallhlh methodo apo ton
controller
                        memberController.deleteMember(memberId);
                        //kai ananewnoume th lista mas
                        refreshMemberList();
                  }
            });
```

Με το κουμπί διαγραφής «διαγράφουμε» ένα μέλος από τη λίστα μελών.Αφού έχου επιλέξει το μέλος που μας ενδιαφέρει πατάμε το κουμπί διαγραφής .Τοτε παίρνουμε το id του και με τη μέθοδο deleteMember το στέλνουμε στον Controller για διαγραφή.

Button καθαρισμού

```
//dhmiourgoume to koumpi ka8arismos pou leitourgei san "new"
        JButton resetButton = new JButton("Καθαρισμός");
        resetButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                      //adeiazoume ola ta pedia apo thn forma
                      memberName.setText("");
                      memberBirthDate.setText("");
                      memberAddress.setText("");
                      memberPhone.setText("");
                      memberMobile.setText("");;
                      memberRegDate = new Date();

     memberRegistrationDate.setText(memberRegDate.toString());
//bazoume th shmerinh hmeromhnia ws hmeromhnia eggrafhs
                      memberNotes.setText("");
                      memberActive.setSelected(true);
                      memberPositionComboBox.setSelectedIndex(0);
                      memberSuitsList.clearSelection();

                      memberId = 0; //kai to shmanthkotero, mhdenizoume
to id, wste na einai etoimh h forma na dextei ena neo melos
                }
            });
```

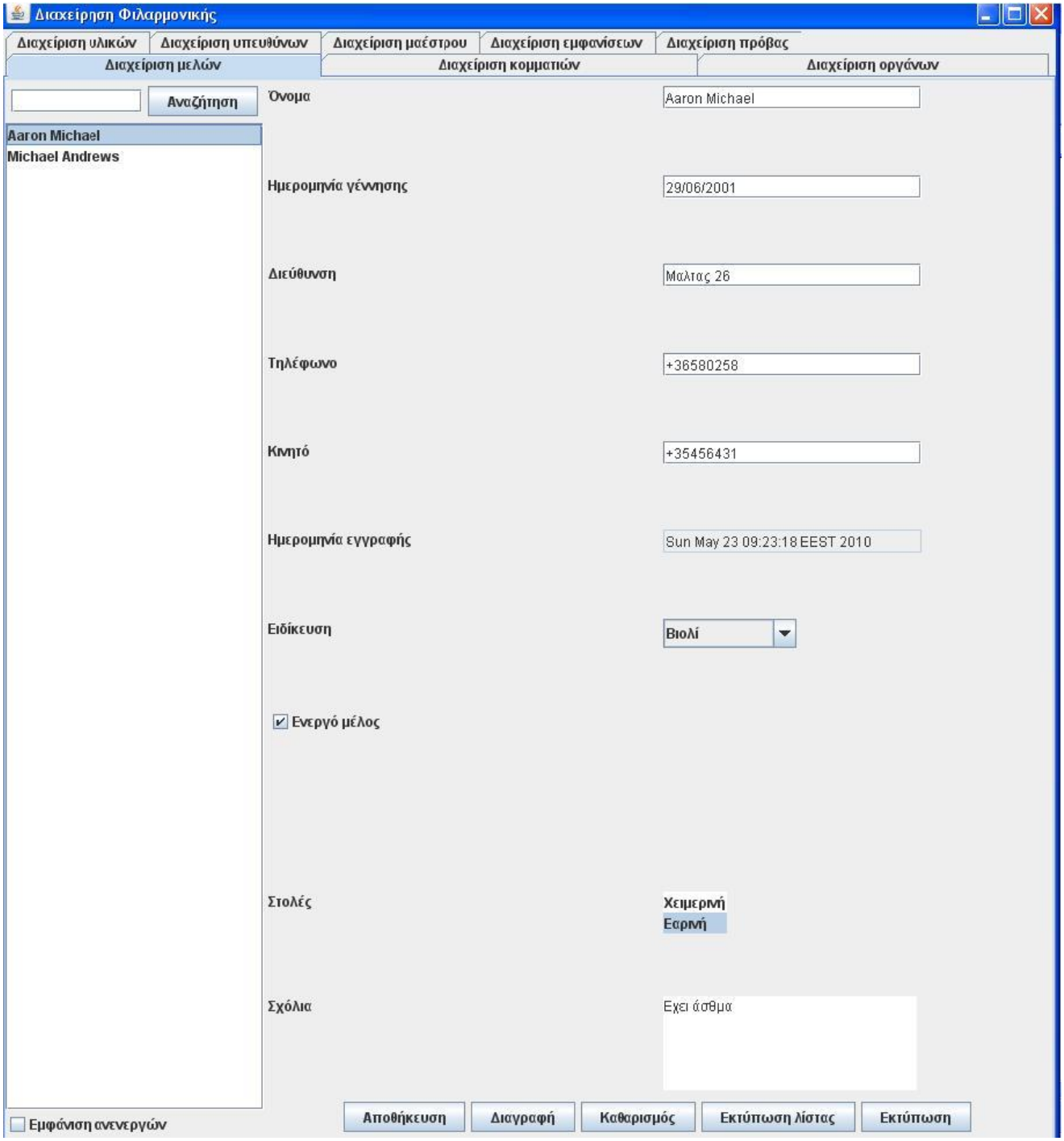
Με το κουμπί καθαρισμού το μηδενίζουμε τα πεδία που εχουμε στη φόρμα μας.επιπλέον αλλάζουμε την ημερομηνία στη σημερινή και το memberid το κάνουμε 0.Αυτό γίνεται ώστε να είναι έτοιμη η φόρμα για την εισαγωγή ενός νέου μέλους.

Επιλογή μέλους από λίστα

```
public void valueChanged(ListSelectionEvent arg0) {
                      Member                  selected                  =
(Member)memberList.getSelectedValue();

                      if(selected == null)
                            return;
                      //Μεταφέρουμε τα επιλεγμένα στοιχεία του μέλους
στα αντίστοιχα πεδία της φόρμας
                      memberId = selected.getId(); //Το id δεν είαν
ορατό στη φόρμα αλλά είναι αυτό που ξεχωρίσει τις εγγρφές(μοναδικό
χαρακτηριστικό)                memberName.setText(selected.getName());
memberBirthDate.setText(selected.getBirthdate());
                memberAddress.setText(selected.getAddress());
                memberPhone.setText(selected.getPhone());
                memberMobile.setText(selected.getMobile());
     memberRegistrationDate.setText(selected.getRegistrationDate().toStr
ing());                       memberNotes.setText(selected.getNotes());
```

```
      memberActive.setSelected(selected.isActive());
      memberPositionComboBox.setSelectedItem(selected.getPosition());
      memberSuitsList.clearSelection();                    //Συγκρίνουμε
κάθε στοιχείο της λίστας κουστουμιών  (που είναι αποθηκευμενα ως Sting)
με κάθε τιμή των στοιχείων της λίστας των στολών ετσι δημιουργούμε τη
λίστα των στολών που πρέπει να εμφανιστούν ως επιλεγμένα
                        ArrayList<Integer>      indexList      =      new
ArrayList<Integer>();
                        for(int i=0; i<selected.getSuits().size(); i++){

                              for(int                                j=0;
j<memberSuitsList.getModel().getSize(); j++){

      if(memberSuitsList.getModel().getElementAt(j).equals(selected.getSu
its().get(i))){
                                          indexList.add(j);
                                    }
                              }
                        }

                        //Μετατρέπουμε  τη  λίστα  που  καταληξαμε  σε
επιλογές του Combo box
                        int       selectedSuitsArray[]       =       new
int[indexList.size()];
                        for(int i=0; i<indexList.size(); i++){
                              selectedSuitsArray[i] = indexList.get(i);
                        }

      memberSuitsList.setSelectedIndices(selectedSuitsArray);

                  }
            });
```

Με τη valueChanged ο χρήστης όταν επιλέγει μια εγγραφή από τη jList τότε τα στοιχεία της εγγραφής αυτής εμφανίζονται στα textbox της φόρμας. Επίσης ανακτά τις στολές από το arrayList και τις μετατρέπει σε επιλογές comboBox.



8.1 View Διαχείριση μέλους

Διαχείριση Μουσικών Κομματιών

Save Button

```
JButton saveButton = new JButton("Αποθήκευση");
        saveButton.addActionListener(new ActionListener() {
            @Override
                  public void actionPerformed(ActionEvent e) {
                  pieceController.savePiece(pieceId,
                              pieceNumber.getText(),
                              pieceTitle.getText(),
                              pieceType.getText(),
                              pieceStatus.getText(),
                              pieceNotes.getText()
                        );
                        //ananewnoume th lista wste na fanoun oi allages
se pragmatiko xrono
                  refreshPieceList();

                  }
            });
        buttonPanel.add(saveButton);
```

Το Button Save υλικού είναι υπεύθυνο για την αποθήκευση ή την τροποποίηση ενός μουσικού κομματιου. Παίρνουμε τις τιμές από τα αντίστοιχα textbox και τα προωθούμε στον Controller Piece .Αφού στείλουμε τα στοιχεία του νέου – τροποποιημένου μουσικού κομματιου στον controller θα πρέπει να ανανεώσουμε τη λίστα των κομματιών ώστε να συμπεριλαμβάνεται και η νέα εγγραφή. Τέλος θα πρέπει να προσθέσουμε και το κουμπί στο panel μας.

Εύρεση μουσικού κομματιού

```
pieceSearchField = new JTextField(10);
            JButton searchButton = new JButton("Αναζήτηση");
            searchButton.addActionListener(new ActionListener() {

                  @Override
                  public void actionPerformed(ActionEvent arg0) {
                        ArrayList<Piece>              list              =
pieceController.search(pieceSearchField.getText());
                        Piece[]     memberArray     =     list.toArray(new
Piece[]{});

                        pieceList.setListData(memberArray);
                  }
            });
```

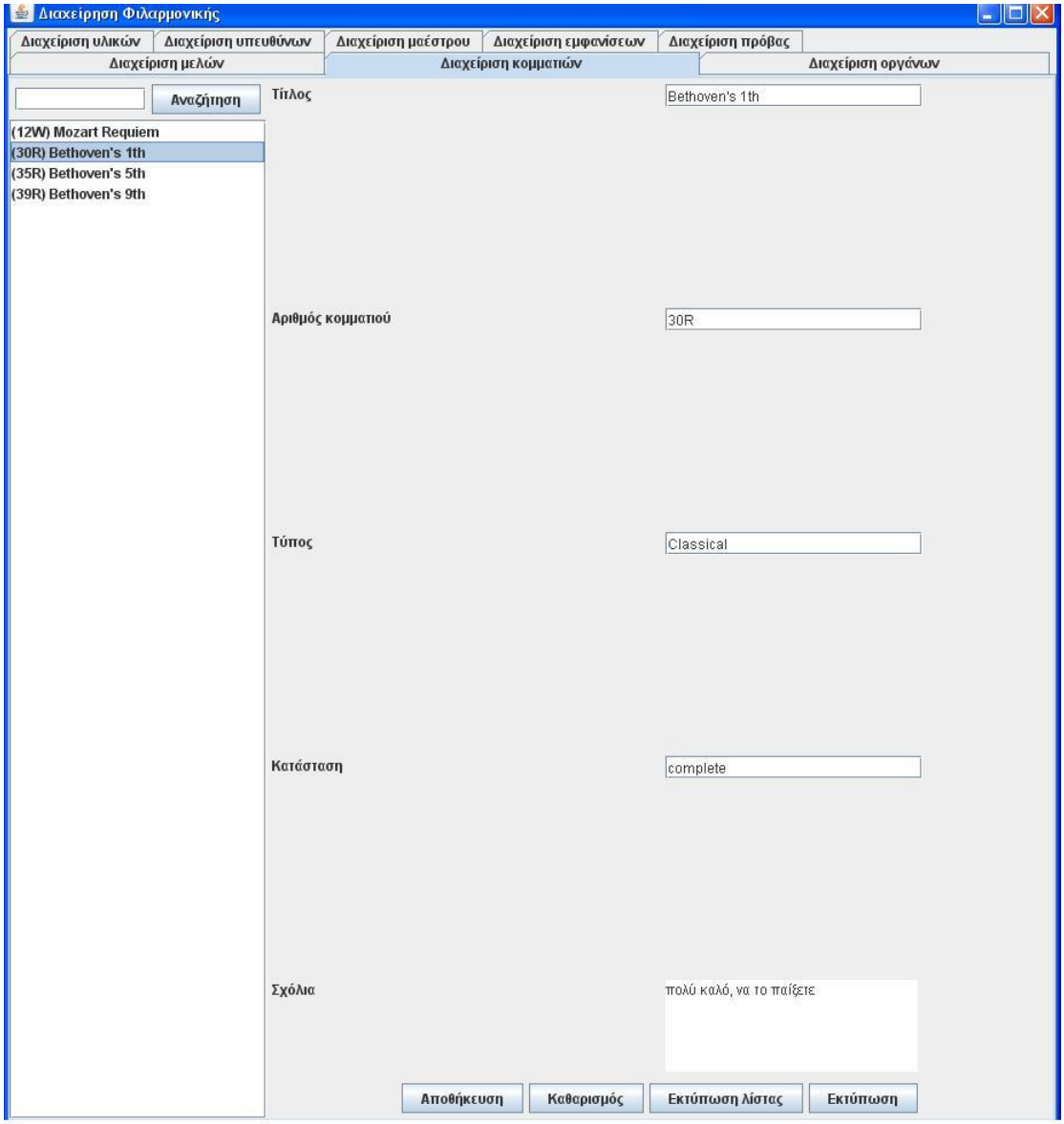
Με την εύρεση μπορούμε να ψάξουμε από το σύνολο των κομματιών λέξεις κλειδιά οπου θα μας τις εμφανίσει στη λίστα μουσικών κομματιών. Αρα στέλνουμε

στην μέθοδο search(String text) το κλειδί(πεδίο αναζήτησης) της αναζήτησης ώστε να γίνει εύρεση στο σύνολο των εγγραφών. Οταν βρεθούν οι εγγραφές θα επιστραφούν ως ένας πίνακας και θα αντικαταστήσουνε τις υπάρχουσες εγγραφές της λίστας μας.

Δημιουργία Λίστας Κομματιών

```
private JPanel createPieceList(){
            ArrayList<Piece> list = pieceController.getAllPieces(); //to
antigrafo ths listas pou 8a emfanisoume
            Piece[]  pieceArray  =  list.toArray(new  Piece[]{});  //to
apo8hkeyoume ws pinaka gia na pai3ei h JList

            //h lista twn melwn einai mia JList me periexomena ton pinaka
twn melwn
            pieceList = new JList(pieceArray);

            //o listener twn clicks panw sta components ths listas.
            pieceList.addListSelectionListener(new
ListSelectionListener() {

                  @Override
                  public void valueChanged(ListSelectionEvent arg0) {
                        Piece                  selected                =
(Piece)pieceList.getSelectedValue();


                        if(selected == null)
                              return;
                        //metaferoume ta stoixeia tou epilegmenou melous
sta antistoixa pedia ths formas
                        pieceId  =  selected.getId();  //to  id  den  einai
orato sth lista alla einai poly shmantiko
                        pieceTitle.setText(selected.getTitle());
                        pieceNumber.setText(selected.getNumber());
                        pieceType.setText(selected.getType());
                        pieceStatus.setText(selected.getStatus());
                        pieceNotes.setText(selected.getNotes());

                  }
            });
```

Δημιουργούμε τη λίστα με τα μουσικά κομμάτια Παίρνουμε το σύνολο των μουσικών κομματιών της φιλαρμονικής και τα εμφανίζουμε στη λίστα μας. Απο αυτή τη λίστα μπορούμε να επιλέξουμε την εγγραφή πού θέλουμε. Από την επιλεγμένη εγγραφή θα πάρουν τιμές τα αντίστοιχα text box.



8.2 Διαχείριση Μουσικών Κομματιών

Διαχείριση Μουσικών Οργάνων

Αποθήκευση

```
  JButton saveButton = new JButton("Αποθήκευση");
        saveButton.addActionListener(new ActionListener() {
           @Override
                 public void actionPerformed(ActionEvent e) {
                 controller.saveInstrument(id,
                           serialTf.getText(),

     instrumentCombo.getSelectedItem().toString(),
                           typeTf.getText(),
                           conditionTf.getText()
                      );
                      //ananewnoume th lista wste na fanoun oi
allages se pragmatiko xrono
                 refreshList();
                 }

           });
        buttonPanel.add(saveButton);
```

Το Button Save υλικού είναι υπεύθυνο για την αποθήκευση ή την τροποποίηση ενός υλικού. Παίρνουμε τις τιμές από τα αντίστοιχα textbox και τα προωθούμε στον Controller Instrument .Αφού στείλουμε τα στοιχεία του νέου – τροποποιημένου υλικού στον controller θα πρέπει να ανανεώσουμε τη λίστα των υλικών ώστε να συμπεριλαμβάνεται και η νέα εγγραφή. Τέλος θα πρέπει να προσθέσουμε και το κουμπί στο panel μας.

Εμφάνιση Λίστας

```
public JPanel createList(){
            ArrayList<Instrument> list = controller.getInstruments();
            Instrument[]      instrumentArray      =      list.toArray(new
Instrument[]{});

            itemList = new JList(instrumentArray);
            itemList.addListSelectionListener(new ListSelectionListener()
{
                 @Override
                 public void valueChanged(ListSelectionEvent arg0) {
                      Instrument               selected               =
(Instrument)itemList.getSelectedValue();

                      if(selected == null)
                           return;
```

```
                        id = selected.getId();

                        serialTf.setText(selected.getSerial());
                        typeTf.setText(selected.getType());
                        conditionTf.setText(selected.getCondition());

      instrumentCombo.setSelectedItem(selected.getInstrument());
                  }
            });

            searchTf = new JTextField(10);
            JButton searchButton = new JButton("Αναζήτηση");
            searchButton.addActionListener(new ActionListener() {
                  @Override
                  public void actionPerformed(ActionEvent e) {
                        ArrayList<Instrument>           list           =
controller.search(searchTf.getText());
                        Instrument[]  instrumentArray  =  list.toArray(new
Instrument[]{});

                        itemList.setListData(instrumentArray);
                  }
            });
```

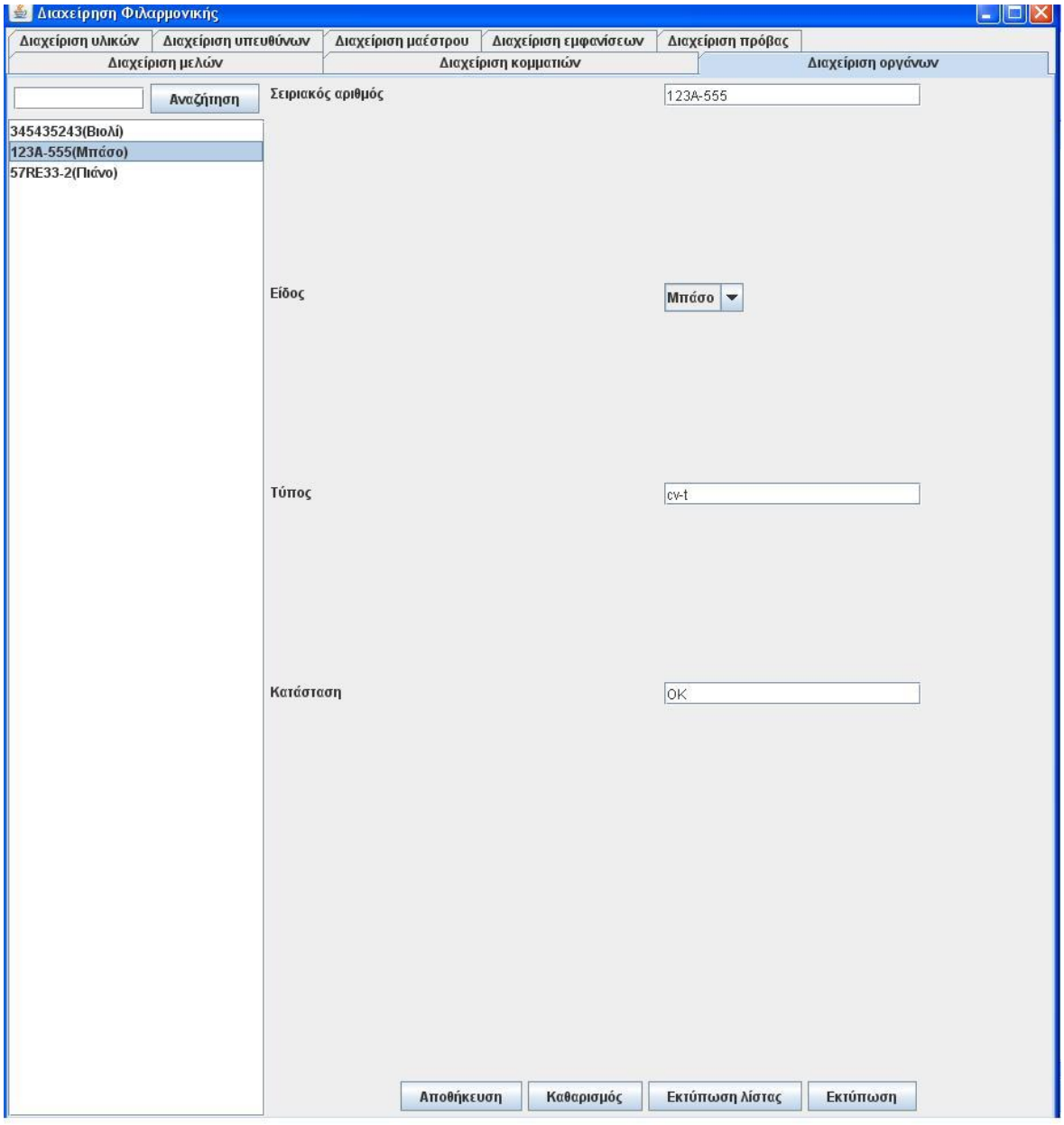
Δημιουργούμε τη λίστα με τα όργανα. Παίρνουμε το σύνολο των οργάνων της φιλαρμονικής και το εμφανίζουμε στη λίστα μας. Απο αυτή τη λίστα μπορούμε να επιλέξουμε την εγγραφή πού θέλουμε. Από την επιλεγμένη εγγραφή θα πάρουμε τα στοιχεία και τα αντίστοιχα textbox θα πάρουν τις τιμές των χαρακτηριστικών του εξοπλισμού που έχει επιλεγεί.

Εύρεση μουσικού οργάνου

```
searchTf = new JTextField(10);
            JButton searchButton = new JButton("Αναζήτηση");
            searchButton.addActionListener(new ActionListener() {
                  @Override
                  public void actionPerformed(ActionEvent e) {
                        ArrayList<Instrument>          list          =
controller.search(searchTf.getText());
                        Instrument[]         instrumentArray         =
list.toArray(new Instrument[]{});

                        itemList.setListData(instrumentArray);
                  }
            });

            JPanel searchPanel = new JPanel(new FlowLayout());
            searchPanel.add(searchTf);
            searchPanel.add(searchButton);
```

```
            //topo8etoume lista sto panel kai to epistrefoume
            JPanel panel = new JPanel(new BorderLayout());
            panel.add(searchPanel, BorderLayout.NORTH);
            panel.add(new                    JScrollPane(itemList),
BorderLayout.CENTER);

            return panel;
      }
```

Με την εύρεση μπορούμε να ψάξουμε από το σύνολο των οργάνων λέξεις κλειδιά οπου θα μας τις εμφανίσει στη λίστα οργάνων. Αρα στέλνουμε στην μέθοδο search(String text) το κλειδί της αναζήτησης ώστε να γίνει η εύρεση στο σύνολο των εγγραφών. Οταν βρεθούν οι εγγραφές θα επιστραφούν ως ένας πίνακας και θα αντικαταστήσουνε τις υπάρχουσες εγγραφές της λίστας μας.



8.3 Διαχείριση Μουσικών Οργάνων

Διαχείριση Υλικού-Εξοπλισμού

Save Button

```
JButton saveButton = new JButton("Αποθήκευση");
        saveButton.addActionListener(new ActionListener() {
           @Override
                 public void actionPerformed(ActionEvent e) {
                 materialController.saveMaterial(pieceId,
                           materialItem.getText(),
                           materialType.getText(),
                           materialStatus.getText(),
                           materialNotes.getText()
                      );
                      //ananewnoume th lista wste na fanoun oi
allages se pragmatiko xrono
                 refreshMaterialList();

                 }
           });
        buttonPanel.add(saveButton);
```

Το Button Save υλικού είναι υπεύθυνο για την αποθήκευση ή την τροποποίηση ενός υλικού. Παίρνουμε τις τιμές από τα αντίστοιχα textbox και τα προωθούμε στον Controller Material .Αφού στείλουμε τα στοιχεία του νέου – τροποποιημένου υλικού στον controller θα πρέπει να ανανεώσουμε τη λίστα των υλικών ώστε να συμπεριλαμβάνεται και η νέα εγγραφή. Τέλος θα πρέπει να προσθέσουμε και το κουμπί στο panel μας.

```
private JPanel createMaterialList(){
           ArrayList<Material>               list               =
materialController.getMaterials(); //to antigrafo ths listas pou
8a emfanisoume
           Material[]     materialArray     =     list.toArray(new
Material[]{}); //to apo8hkeyoume ws pinaka gia na pai3ei h JList

           //h lista twn melwn einai mia JList me periexomena ton
pinaka twn melwn
           materialList = new JList(materialArray);

           //o listener twn clicks panw sta components ths listas.
           materialList.addListSelectionListener(new
ListSelectionListener() {

                 @Override
                 public void valueChanged(ListSelectionEvent arg0)
{
```

```
                        Material              selected              =
(Material)materialList.getSelectedValue();


                        if(selected == null)
                              return;
                        //metaferoume ta stoixeia tou epilegmenou
melous sta antistoixa pedia ths formas
                        pieceId = selected.getId(); //to id den
einai orato sth lista alla einai poly shmantiko
                        materialItem.setText(selected.getItem());

                        materialType.setText(selected.getType());

      materialStatus.setText(selected.getCondition());
                        materialNotes.setText(selected.getNotes());

                  }
            });
```

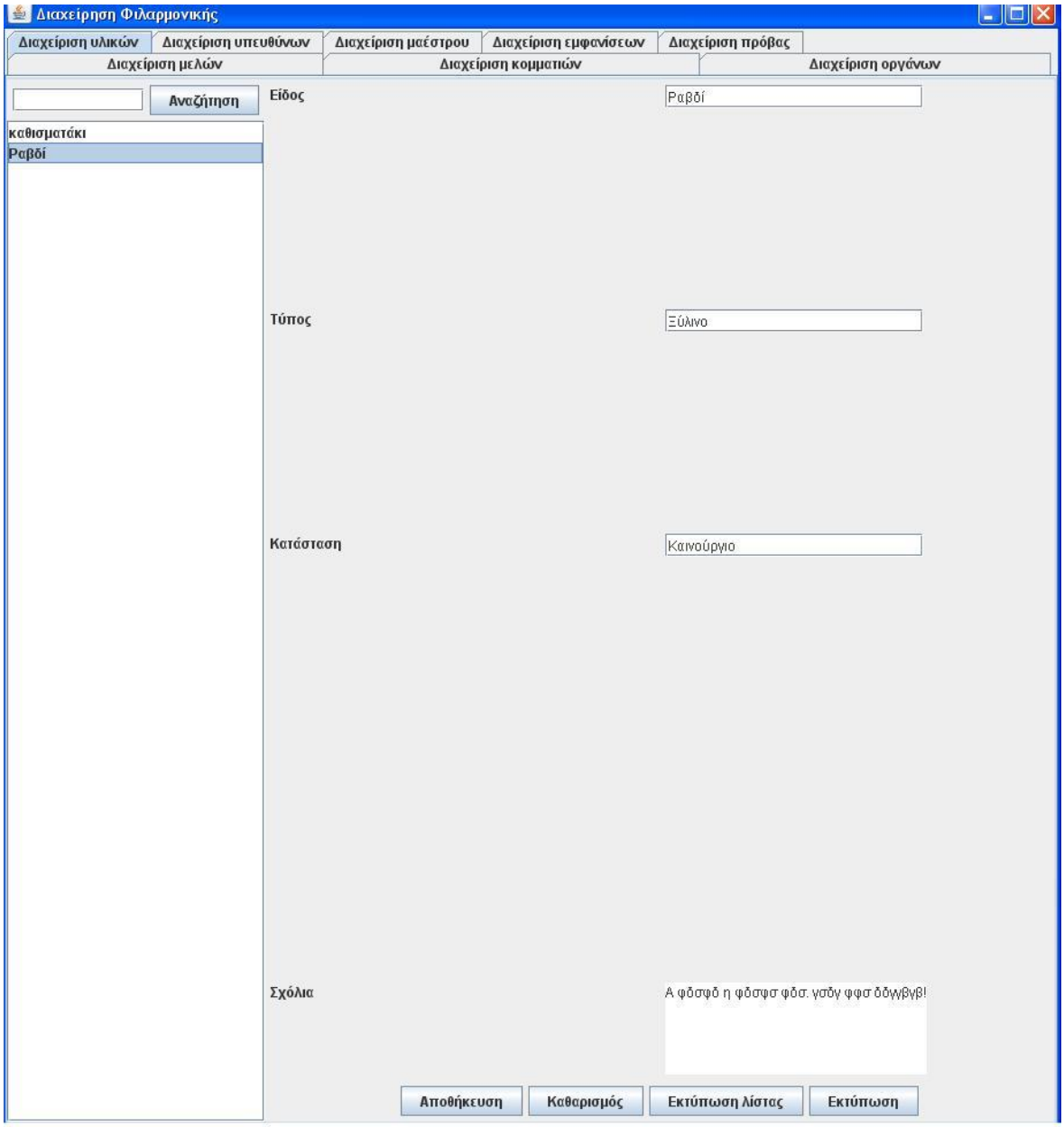
Δημιουργούμε τη λίστα με το εξοπλισμό. Παίρνουμε το σύνολο του υλικού της φιλαρμονικής και το εμφανίζουμε στη λίστα μας. Απο αυτή τη λίστα μπορούμε να επιλέξουμε την εγγραφή πού θέλουμε. Από την επιλεγμένη εγγραφή θα πάρουμε τα στοιχεία και τα αντίστοιχα textbox θα πάρουν τις τιμές των χαρακτηριστικών του εξοπλισμού που έχει επιλεγεί.

Εύρεση Εξοπλισμού

```
materialSearchField = new JTextField(10);
            JButton searchButton = new JButton("Αναζήτηση");
            searchButton.addActionListener(new ActionListener() {

                  @Override
                  public void actionPerformed(ActionEvent arg0) {
                        ArrayList<Material>            list            =
materialController.search(materialSearchField.getText());
                        Material[]   materialArray   =   list.toArray(new
Material[]{});

                        materialList.setListData(materialArray);
                  }
            });

            JPanel searchPanel = new JPanel(new FlowLayout());
            searchPanel.add(materialSearchField);
            searchPanel.add(searchButton);
```

Με την εύρεση μπορούμε να ψάξουμε από το σύνολο του εξοπλισμού λέξεις κλειδιά οπου θα μας τις εμφανίσει στη λίστα εξοπλισμού. Αρα στέλνουμε στην μέθοδο search(String text) το κλειδί(πεδίο αναζήτησης) της αναζήτησης ώστε να γίνει εύρεση στο σύνολο των εγγραφών. Οταν βρεθούν οι εγγραφές θα

επιστραφούν ως ένας πίνακας και θα αντικαταστήσουνε τις υπάρχουσες εγγραφές της λίστας μας.



8.4 Διαχείριση Υλικού

Διαχείριση Αρχιμουσικού

Save Button

```
JButton saveButton = new JButton("Αποθήκευση");
        saveButton.addActionListener(new ActionListener() {
            @Override
                  public void actionPerformed(ActionEvent e) {
                  controller.saveBandMaster(id,
                              nameTf.getText(),
                              phoneTf.getText(),
                              mobileTf.getText(),
                              addressTf.getText(),
                              diplomasTa.getText(),
                              startDateTf.getText(),
                              endDateTf.getText()
                        );
                        //ananewnoume th lista wste na fanoun oi allages
se pragmatiko xrono
                  refreshList();
                  }

            });
        buttonPanel.add(saveButton);
```

Με το κουμπί αποθήκευσης αποθηκεύουμε ή τροποποιούμε μια εγγραφή.Πέρνουμε τις τιμές από τα textbox και τις προωθούμε στον Controller για τη δημιουργία –τροποποίηση ενός αρχιμουσικού. Αφού αποθηκευτεί ο αρχιμουσικός τότε ανανεώνουμε και τη λίστα μας.

Λίστα Αρχιμουσικών

```
public JPanel createList(){
           ArrayList<BandMaster>               list               =
controller.getBandMasters();  //to  antigrafo  ths  listas  pou  8a
emfanisoume
           BandMaster[]     masterArray     =     list.toArray(new
BandMaster[]{}); //to apo8hkeyoume ws pinaka gia na pai3ei h JList

                        itemList = new JList(masterArray);

           //o listener twn clicks panw sta components ths listas.
           itemList.addListSelectionListener(new
ListSelectionListener() {

                  @Override
```

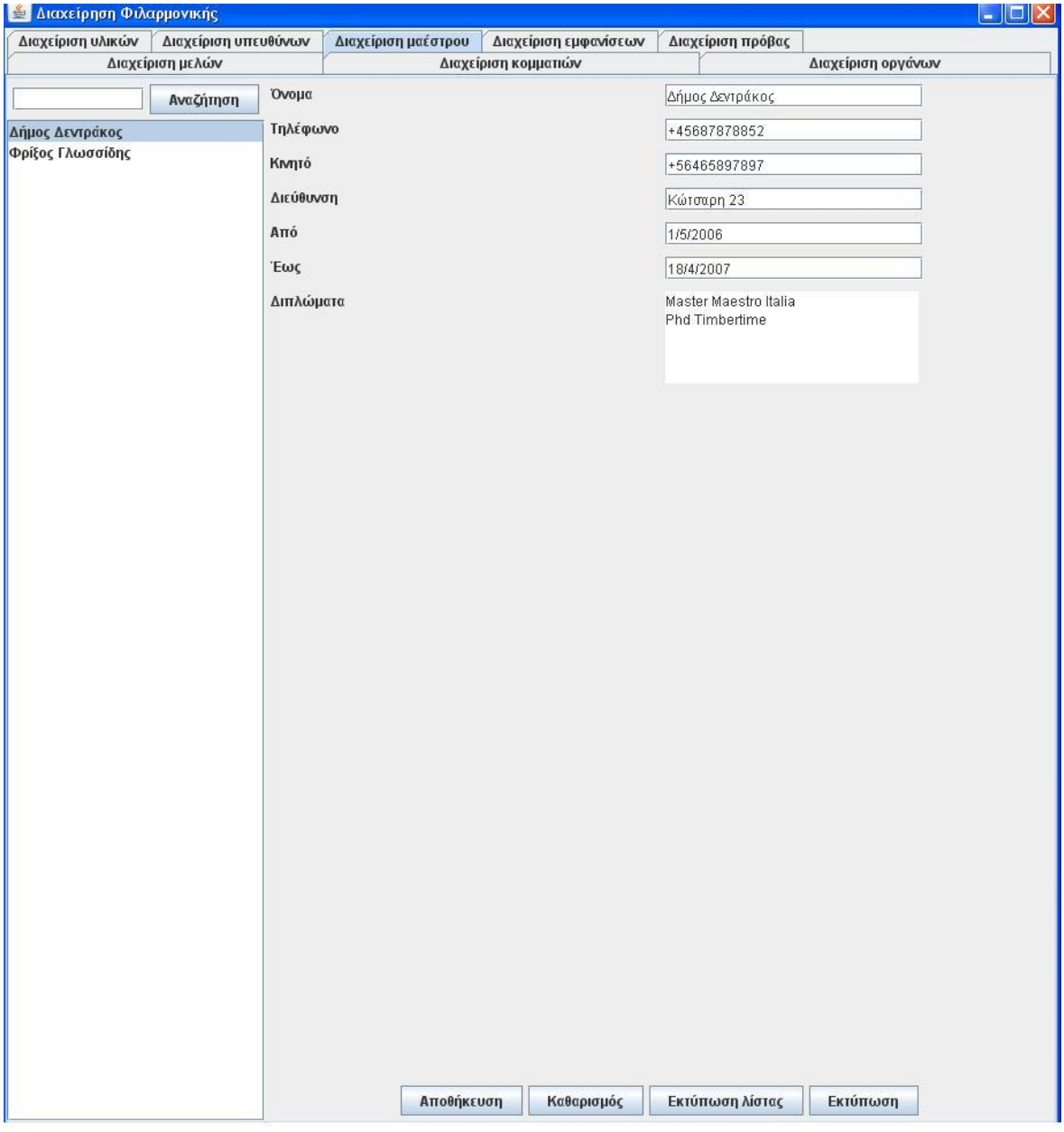
```
                public void valueChanged(ListSelectionEvent arg0)
{
                        BandMaster             selected             =
(BandMaster)itemList.getSelectedValue();


                        if(selected == null)
                              return;
                        //metaferoume ta stoixeia tou epilegmenou
arximousikoy sta antistoixa pedia ths formas
                        id = selected.getId(); //to id den einai
orato sth lista alla einai poly shmantiko
                        nameTf.setText(selected.getName());
                        phoneTf.setText(selected.getTelephone());
                        mobileTf.setText(selected.getMobile());
                        addressTf.setText(selected.getAddress());
                        diplomasTa.setText(selected.getDiplomas());

     startDateTf.setText(selected.getStartDate());
                        endDateTf.setText(selected.getEndDate());

                }
           });
```

Δημιουργούμε ένα αντικείμενο jlist και το εισάγουμε στο πάνελ μας. Παίρνουμε το σύνολο των αρχιμουσικών και το εισάγουμε στη λίστα μας. Επίσης πρέπει να φτιάξουμε και τη μέθοδο οπου σε περίπτωση επιλογής να εισάγει τα στοιχεία του επιλεγμένου αρχιμουσικού στα αντίστοιχα πεδία. Η valueChanged(ListSelectionEvent arg0) παίρνει τον επιλεγμένο αρχιμουσικό και εισάγει τα στοιχεία στα textbox της φόρμας μας.

Αναζήτηση Αρχιμουσικού

```
searchTf = new JTextField(10);
           JButton searchButton = new JButton("Αναζήτηση");
           searchButton.addActionListener(new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent arg0) {
                        ArrayList<BandMaster>          list          =
controller.search(searchTf.getText());
                        BandMaster[] memberArray = list.toArray(new
BandMaster[]{});

                        itemList.setListData(memberArray);
                }
           });
```

Με το κουμπί αναζήτησης βρίσκουμε από το σύνολο των αρχιμουσικών τον-τους αρχιμουσικούς που περιέχουν το λεκτικό που έχουμε δώσει.Την εργασία την κάνει η μέθοδος search.Η search μας επιστρέφει ένα σύνολο από εγγρφές που ταιριάζουν στο λεκτικό που έχουμε δώσει.



8.5 Διαχείριση Αρχιμουσικού

Διαχείριση Υπεύθυνου

Save button

```
JButton saveButton = new JButton("Αποθήκευση");
        saveButton.addActionListener(new ActionListener() {
           @Override
                 public void actionPerformed(ActionEvent e) {
                 controller.saveResponsible(id,
                           nameTf.getText(),
                           phoneTf.getText(),
                           mobileTf.getText(),
                           addressTf.getText(),
                           startDateTf.getText(),
                           endDateTf.getText()
                      );
                      //ananewnoume th lista wste na fanoun oi
allages se pragmatiko xrono
                 refreshList();
                 }

           });
        buttonPanel.add(saveButton);
```

Με το κουμπί αποθήκευσης αποθηκεύουμε ή τροποποιούμε μια εγγραφή. Παίρνουμε τις τιμές από τα textbox και τις προωθούμε στον Controller για τη δημιουργία –τροποποίηση ενός υπέυθυνου. Αφού αποθηκευτεί ο υπεύθυνος τότε ανανεώνουμε και τη λίστα μας. Ώστε η λίστα να περιλαμβάνει και τις νέες εγγραφές.

Δημιουργία Λίστας υπεύθυνων

```
public JPanel createList(){
           ArrayList<Responsible> list = controller.getResponsibles();
//to antigrafo ths listas pou 8a emfanisoume
           Responsible[] pieceArray = list.toArray(new Responsible[]{});
//to apo8hkeyoume ws pinaka gia na pai3ei h JList

           //h lista twn melwn einai mia JList me periexomena ton pinaka
twn melwn
           itemList = new JList(pieceArray);

           //o listener twn clicks panw sta components ths listas.
           itemList.addListSelectionListener(new ListSelectionListener()
{

                 @Override
                 public void valueChanged(ListSelectionEvent arg0) {
```

```
                        Responsible              selected              =
(Responsible)itemList.getSelectedValue();


                        if(selected == null)
                              return;
                        //metaferoume   ta   stoixeia   tou   epilegmenou
upeu8unou sta antistoixa pedia ths formas
                        id = selected.getId(); //to id den einai orato
sth lista alla einai poly shmantiko
                        nameTf.setText(selected.getName());
                        phoneTf.setText(selected.getPhone());
                        mobileTf.setText(selected.getMobile());
                        addressTf.setText(selected.getAddress());
                        startDateTf.setText(selected.getStartDate());
                        endDateTf.setText(selected.getEndDate());

                  }
            });
```

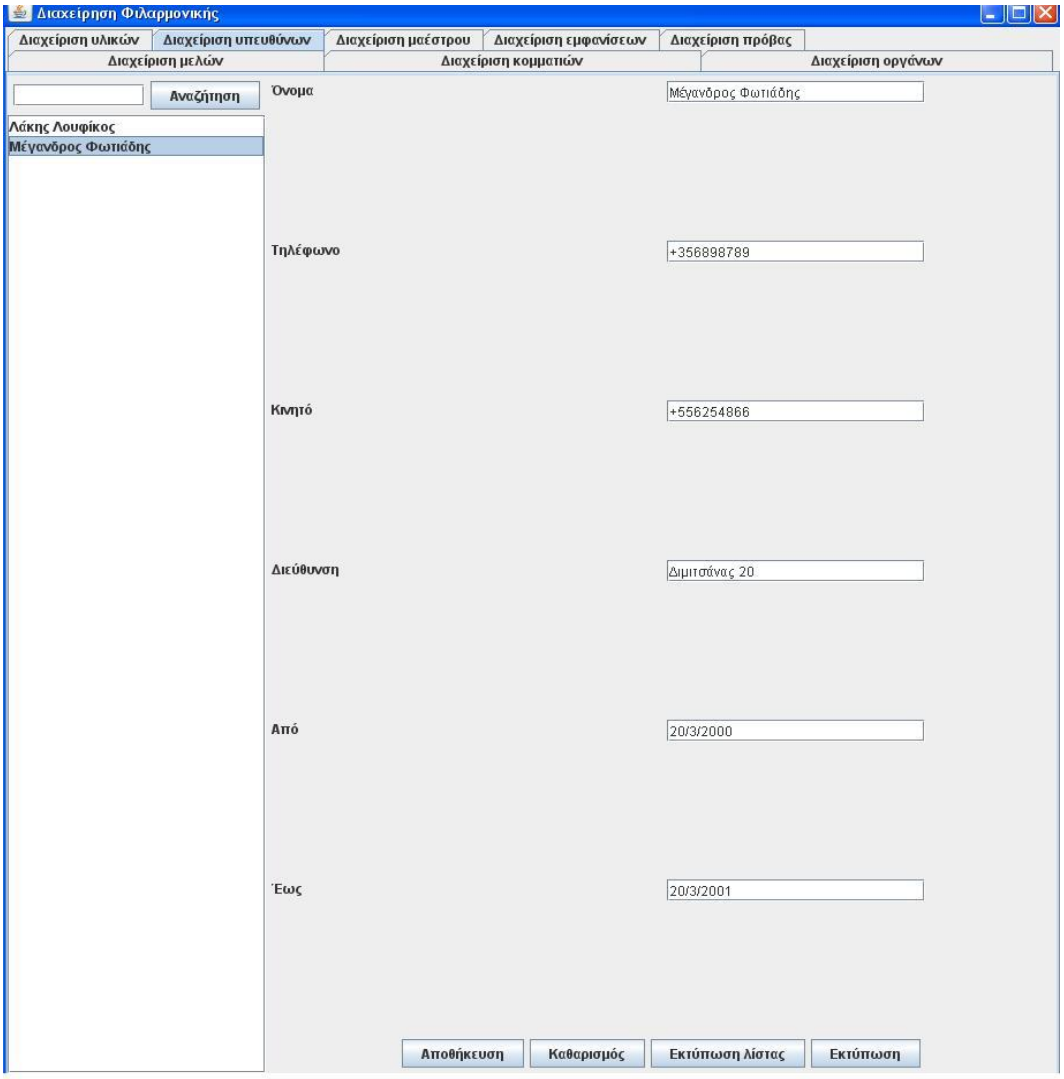
Δημιουργούμε ένα αντικείμενο jlist και το εισάγουμε στο πάνελ μας. Παίρνουμε το σύνολο των υπεύθυνων και το εισάγουμε στη λίστα μας. Επίσης πρέπει να φτιάξουμε και τη μέθοδο οπου σε περίπτωση επιλογής να εισάγει τα στοιχεία του επιλεγμένου υπεύθυνου στα αντίστοιχα πεδία. Η valueChanged(ListSelectionEvent arg0) παίρνει τον επιλεγμένο υπέυθυνο και εισάγει τα στοιχεία στα textbox της φόρμας μας. Ωστε να μπορεί ο χρήστης να τροποποίηση την εγγραφή.

Αναζήτηση Υπεύθυνου

```
searchTf = new JTextField(10);
            JButton searchButton = new JButton("Αναζήτηση");
            searchButton.addActionListener(new ActionListener() {

                  @Override
                  public void actionPerformed(ActionEvent arg0) {
                        ArrayList<Responsible>         list         =
controller.search(searchTf.getText());
                        Responsible[]  itemArray  =  list.toArray(new
Responsible[]{});

                        itemList.setListData(itemArray);
                  }
            });

            JPanel searchPanel = new JPanel(new FlowLayout());
            searchPanel.add(searchTf);
            searchPanel.add(searchButton);
```

```
            //topo8etoume lista sto panel kai to epistrefoume
            JPanel panel = new JPanel(new BorderLayout());
            panel.add(searchPanel, BorderLayout.NORTH);
            panel.add(new                    JScrollPane(itemList),
BorderLayout.CENTER);

            return panel;

      }
```

Με το κουμπί αναζήτησης βρίσκουμε από το σύνολο των υπεύθυνων τον-τους υπεύθυνο-ους που περιέχουν το λεκτικό που έχουμε δώσει. Την εργασία την κάνει η μέθοδος search.Η search μας επιστρέφει ένα σύνολο από εγγραφές που ταιριάζουν στο λεκτικό που έχουμε δώσει.



8.6 Διαχείριση Εμφάνισης

Διαχείριση Εμφάνισης

```
JButton saveButton = new JButton("Αποθήκευση");
        saveButton.addActionListener(new ActionListener() {
            @Override
                  public void actionPerformed(ActionEvent e) {
                  ArrayList<Piece> pieces = new ArrayList<Piece>();
                  Object[] piecesArray = piecesList.getSelectedValues();

                  for(int i=0; i<piecesArray.length; i++){
                        pieces.add( ((Piece)piecesArray[i]) );
                  }

                  ArrayList<Member> members = new ArrayList<Member>();
                  Object[]                membersArray                =
membersList.getSelectedValues();

                  for(int i=0; i<membersArray.length; i++){
                        members.add( ((Member)membersArray[i]) );
                  }

                  controller.saveAppearance(id,
                              occassionTf.getText(),
                              dateTf.getText(),
                              startTimeTf.getText(),
                              placeTf.getText(),

      ((Responsible)responsiblesCombo.getSelectedItem()),

      ((BandMaster)bandmasterCombo.getSelectedItem()),
                              pieces,
                              members,
                              notesTa.getText(),
                              movementsCombo.getSelectedItem().toString()
                        );
                        //ananewnoume th lista wste na fanoun oi allages
se pragmatiko xrono
                  refreshList();
                  }

            });
        buttonPanel.add(saveButton);
```

Η save παίρνει τα δεδομένα από τα αντικείμενα (textbox, compobox) και τα προωθεί στον αντίστοιχο controller για την αποθήκευση τους .Τέλος ανανεώνουμε τη λίστα ώστε να φανούν οι αλλαγές άμεσα.

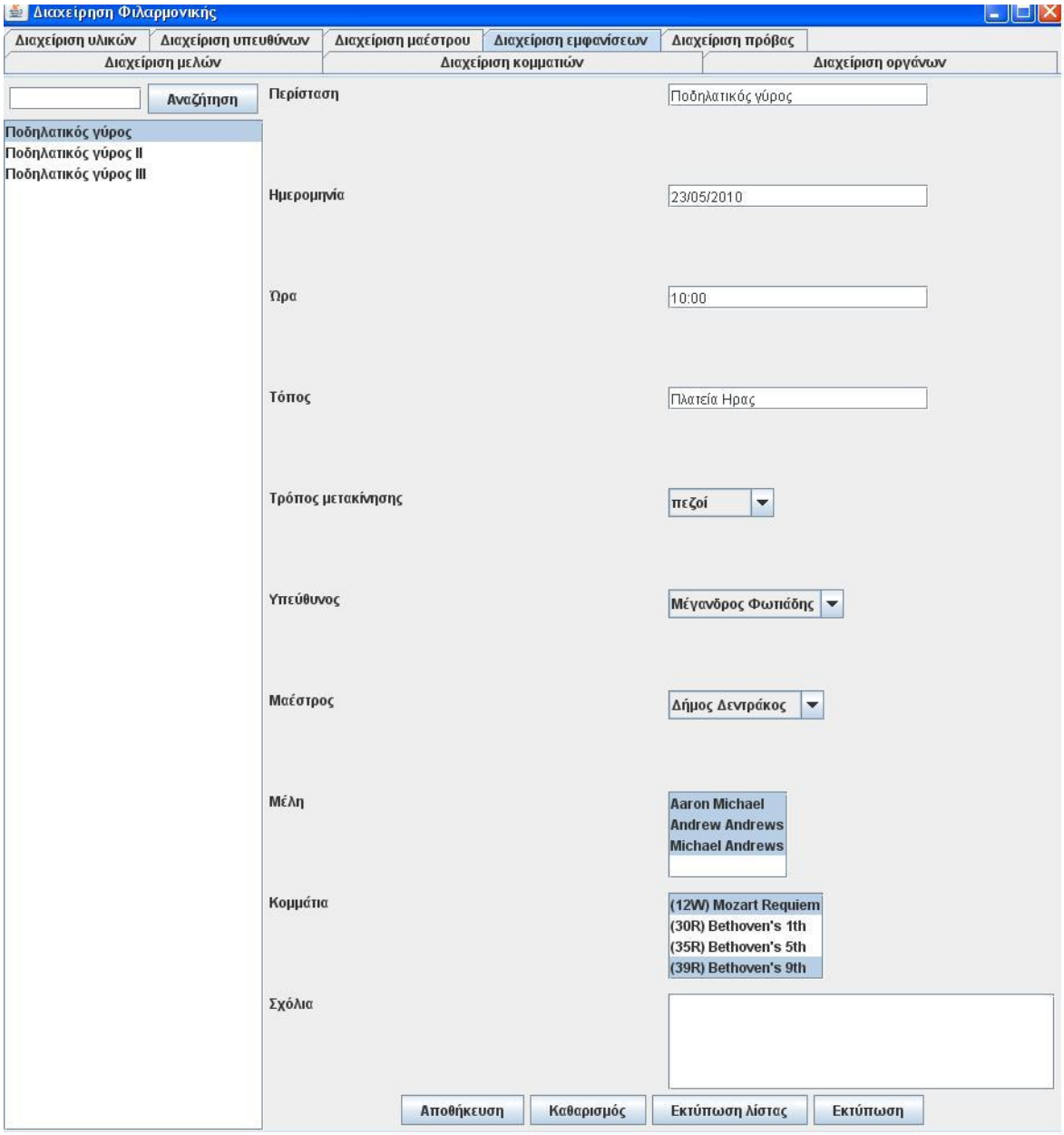
```
public JPanel createList(){
            ArrayList<Appearance> list = controller.getAppearance();
            Appearance[]      appearanceArray      =      list.toArray(new
Appearance[]{});

            itemList = new JList(appearanceArray);
            itemList.addListSelectionListener(new ListSelectionListener()
{
```

```
                  @Override
                  public void valueChanged(ListSelectionEvent arg0) {
                        Appearance               selected               =
(Appearance)itemList.getSelectedValue();

                        if(selected == null)
                              return;

                        id = selected.getId();

                        occassionTf.setText(selected.getOccassion());
                        dateTf.setText(selected.getDate());
                        startTimeTf.setText(selected.getStartTime());
                        placeTf.setText(selected.getPlace());
                        notesTa.setText(selected.getNotes());

                        controller.setResponsibleById(selected,
responsiblesCombo);
                        controller.setBandmasterById(selected,
bandmasterCombo);

      movementsCombo.setSelectedItem(selected.getMovement());

                        controller.setSelectedMembers(selected,
membersList);
                        controller.setSelectedPieces(selected,
piecesList);
                  }
            });
```

Εμφανίζει τις εγγραφές στη φόρμα μας. Τη λίστα των εμφανίσεων τα ονόματα των μελών των κομματιών, τον αρχιμουσικών και των υπεύθυνων

```
public void refreshForm(){
            membersList.setListData(controller.getMembers());
            piecesList.setListData(controller.getPieces());

            bandmasterCombo.removeAllItems();
            BandMaster[] ar = controller.getBandMasters();
            for(int i=0; i<ar.length; i++){
                  bandmasterCombo.addItem(ar[i]);
            }

            responsiblesCombo.removeAllItems();
            Responsible[] ar1 = controller.getResponsibles();
            for(int i=0; i<ar1.length; i++){
                  responsiblesCombo.addItem(ar1[i]);
            }
      }
```

Ανανεώνει τη λίστα των των μελών, των κομματιών, των αρχιμουσικών και των υπεύθυνων ώστε ακόμα και να προσθέσουμε ένα μέλος να εμφανιστέι άμεσα στη λίστα μας.



8.7 Διαχείριση Εμφάνισης

View Πρόβας

```
JButton saveButton = new JButton("Αποθήκευση");
        saveButton.addActionListener(new ActionListener() {
            @Override
                  public void actionPerformed(ActionEvent e) {
                  ArrayList<Piece> pieces = new ArrayList<Piece>();
                  Object[] piecesArray = piecesList.getSelectedValues();

                  for(int i=0; i<piecesArray.length; i++){
                        pieces.add( ((Piece)piecesArray[i]) );
                  }

                  ArrayList<Member> members = new ArrayList<Member>();
                  Object[]                membersArray                =
membersList.getSelectedValues();

                  for(int i=0; i<membersArray.length; i++){
                        members.add( ((Member)membersArray[i]) );
                  }

                  controller.saveTraining(id,
                              dateTf.getText(),
                              startTimeTf.getText(),
                              endTimeTf.getText(),
            ((BandMaster)bandmasterCombo.getSelectedItem()),
                              notesTa.getText(),
                              pieces,
                              members
                        );
                        //ananewnoume th lista wste na fanoun oi allages
se pragmatiko xrono
                  refreshList();
                  }

            });
        buttonPanel.add(saveButton);
```

Η save παίρνει τα δεδομένα από τα αντικείμενα (textbox, compobox) και τα προωθεί στον αντίστοιχο controller για την αποθήκευση τους .Τέλος ανανεώνουμε τη λίστα ώστε να φανούν οι αλλαγές άμεσα.

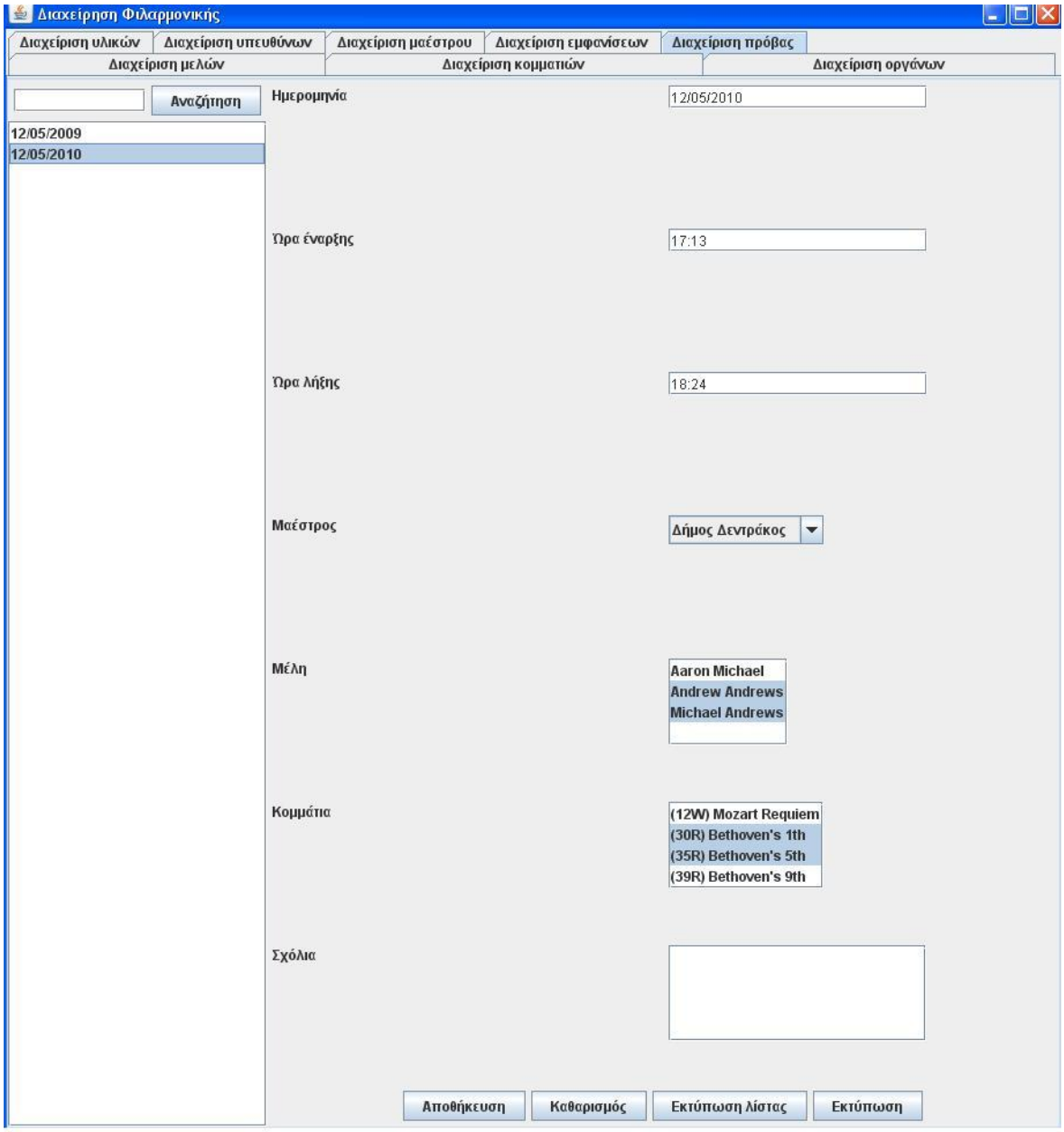
```
public JPanel createList(){
            ArrayList<Trainning> list = controller.getTrainnings();
            Trainning[] trainningArray = list.toArray(new Trainning[]{});

            itemList = new JList(trainningArray);
            itemList.addListSelectionListener(new ListSelectionListener()
{
                  @Override
                  public void valueChanged(ListSelectionEvent arg0) {
                        Trainning               selected               =
(Trainning)itemList.getSelectedValue();

                        if(selected == null)
```

```
                              return;

                        id = selected.getId();

                        dateTf.setText(selected.getDate());
                        startTimeTf.setText(selected.getStartTime());
                        endTimeTf.setText(selected.getEndTime());
                        notesTa.setText(selected.getNotes());

                        controller.setBandmasterById(selected,
bandmasterCombo);
                        controller.setSelectedMembers(selected,
membersList);
                        controller.setSelectedPieces(selected,
piecesList);
                  }
            });
```

Εμφανίζει τις εγγραφές στη φόρμα μας. Τη λίστα των προβών τα ονόματα των μελών των κομματιών και των αρχιμουσικών.

```
public void refreshForm(){
            membersList.setListData(controller.getMembers());
            piecesList.setListData(controller.getPieces());
            bandmasterCombo.removeAllItems();

            BandMaster[] ar = controller.getBandMasters();
            for(int i=0; i<ar.length; i++){
                  bandmasterCombo.addItem(ar[i]);
            }


      }
```

Ανανεώνει τη λίστα των των μελών των κομματιών, τον αρχιμουσικών ώστε ακόμα και να προσθέσουμε ένα μέλος,κομμάτι ή αρχιμουσικό να εμφανιστεί άμεσα στη λίστα μας.



8.8 Διαχείριση πρόβας

Επίλογος

Μελλοντικές επεκτάσεις

Κάθε εφαρμογή που σχεδιάζεται δεν σχεδιάζεται μόνο για το παρών αλλά σχεδιάζεται και για το μέλλον. Απο την πρώτη μέρα λειτουργίας του προγράμματος δεν μπορούμε να ξέρουμε τι επεκτάσεις ή βελτιώσεις θα χρειαστεί να προσθέσουμε .Δεν μπορούμε να ξέρουμε εξ αρχής τις μελλοντικές ανάγκες των χρηστών του προγράμματος.
Η εφαρμογή καθώς αναπτυσσόταν υπήρχε και ένας παράγοντας που δεν αναφέρθηκε. Αυτός ο παράγοντας είναι η μελλοντική χρήση και επέκταση. Επρεπε να σχεδιαστεί έτσι ώστε οποιαδήποτε αλλαγή να γίνει οσο το δυνατόν πιο εύκολα με τις λιγότερες δυνατές αλλαγές στον κώδικα.(μικρότερο κόστος)

Αν παρατηρήσουμε τον κώδικα μας θα δούμε ότι μπορούμε να χωρίσουμε το πρόγραμμα μας στα εξής μέρη.

Α)Τα μοντέλα(μέλος, όργανο, εμφάνιση)Αυτό το μέρος αφορά τα κύρια μοντέλα-οντότητες της εφαρμογής μας καθώς και την περιγραφή αυτών. Μπορούμε σε κάθε μοντέλο να προσθέσουμε ή και να αλλάξουμε χαρακτηριστικά αφήνοντας το υπόλοιπο σύστημα σχεδόν ανέπαφο.

Β)Διαχείριση αρχείων(FileManager)Στην εφαρμογή μας κάνουμε χρήση αρχείων. Η χρήση τους γίνεται για λόγους που τους αναφέραμε σε προηγούμενη ενότητα. Για τον οποιονδήποτε όμως λόγο μπορεί να χρειαστούμε να αλλάζουμε τον τρόπο ή τη μορφή αποθήκευσης των δεδομένων μας. Επειδή οποιαδήποτε λειτουργία διαχείρισης αρχείου βρίσκεται σε μια κλάση(fileManager) η αλλαγή μπορεί να γίνει εύκολα αλλάζοντας – τροποποιώντας αυτήν την κλάση μόνο. Και το υπόλοιπο πρόγραμμα μας θα παραμείνει ως έχει.

Γ)Views-Interface.Είναι το γραφικό περιβάλλον, η διεπαφή του προγράμματος διεπαφή έχει φτιαχτεί οσον το δυνατόν ανεξάρτητη από το υπόλοιπο πρόγραμμα και αυτό που την συνδέει είναι οι controllers του προγράμματος τη βοήθεια τους μπορούμε να κάνουμε ότι αλλαγές θέλουμε στη δομή του προγράμματος χωρίς να χρειαστεί να τροποποιήσουμε τις διεπαφές ή να κάνουμε αλλαγές στον τρόπο λειτουργίας τους.ή κα το αντίστροφο να κάνουμε αλλαγές στις διεπαφές χωρίς να χρειαστεί να κάνουμε αλλαγές στο πρόγραμμα μας

Δ)Java.Με τη χρήση της java το πρόγραμμα μας μπορεί να λειτουργήσει σε πολλές διαφορετικές πλατφόρμες πχ Windows ή Linux ή ακόμα και σε μέσω διαδικτύου με ελάχιστες τροποποιήσεις έως καθόλου. Σε αυτό συντελεί η τεχνολογία αυτής της γλώσσας προγραμματισμού (JVM).

Ε)Σχεδιασμός. Τέλος ολος αυτός ο διαχωρισμός μας επιτρέπει την μετέπειτα εύκολη προσθήκη λειτουργιών αφήνοντας τα υπόλοιπα κομμάτια ανέπαφα .Για παράδειγμα μπορούμε να προσθέσουμε ένα νέο μοδελ-κλάση πρόσθεση Controller και τέλος την προσθήκη της νέας λειτουργίας στην διεπαφή μας αφήνοντας το υπόλοιπο κομμάτι ανέπαφο.

Βιβλιογραφια

Αλέξανδρος Χατζηγεωργίου Αντιεκειμενοστρεφής σχεδίαση

Martin Fowler Εισαγωγή στη UML

Βώρος νικ & Βώρος Άγγελος Unified Modeling Language

Βασίλης Γερογιάννης, Γιώργος Κακαρόζας, Αχιλλέας Καμέας, Γιάννης Σταμέλος, Πανος Φιτσίλης

Αντικειμενοστρεφής Ανάπτυξη Λογισμικού με τη UML

ΒΑΣΙΛΕΙΟΣ ΒΕΣΚΟΥΚΗΣ Τεχνολογία Λογισμικού ΙΙ

Anya Sotiropoulou Αντικειμενοστραφής Ανάπτυξη Λογισμικού

Craig_Larman - Applying UML and Patterns 2nd Ed (2001)

Εμμανουήλ Στ. Σκορδαλάκη-Λογισμική Μηχανική(2004)

Γιώργος Λιακέας Εισαγωγή στη Java 2

Rogers Cadenheas - Laura Lemay Εγχειρίδιο της java6

Οδηγός Χρήσης Λογισμικού

Διαχείριση Μέλους

Στη διαχείριση μέλους ο χρήστης διαμορφώνει τα μέλη και τα χαρακτηριστικά τους. Κατ αρχήν θα πρέπει να επιλέξει την καρτέλα Διαχείριση μελών.Οταν την επιλέγει εμφανίζεται μπροστά του η οθόνη της διαχείρισης. Στα αριστερά υπάρχει η λίστα με τα ονόματα των μελών που είναι καταχωρημένα στο συστημα μας. Ακριβώς πάνω από τη λίστα υπάρχει το textbox που γράφουμε το κειμενο προς εύρεση και δίπλα του το κουμπί της ευρεσης. Στα δεξιά υπάρχουν τα πεδία εμφάνισης των στοιχείων του μέλους. Τελος στο κάτω μέρος υπάρχουν τα κουμπιά των λειτουργιών (αποθηκευση, διαγραφή, καθαρισμός).

Αποθήκευση Νέου Μέλους

Πηγαίνουμε στην καρτέλα Διαχείριση μέλους. Εισάγουμε στα πεδία τα δεδομένα που γνωρίζουμε από το ατομο που ήρθε να εγγραφεί.

Όνομα: Γράφουμε το όνομα του νέου μέλους. Κατά προτίμηση πρώτα το επώνυμο για να υπάρχει μια ομοιομορφία.

Ημερομηνία Γέννησης: Συμπληρώνουμε την ημερομηνία γέννησης

Διεύθυνση: τη διεύθυνση κατοικίας του μέλους

Τηλέφωνο: Το σταθερό τηλέφωνο επικοινωνίας. Το πεδίο μπορεί να δεχτεί και περισσότερα τηλέφωνα χωρισμένα με κομμα ή παύλα

Κινητό: Το προσωπικό κινητό τηλέφωνο

Ημερομηνία Εγγραφής: Παίρνει αυτόματα τιμές την ημερομηνία εισαγωγής της εγγραφής

Ειδίκευση: Επιλέγει ο αρχιμουσικός το μουσικό όργανο που θα έχει το μέλος(μπορεί να το συμπληρώσει και αργότερα και να το αφήσει κενό μεχρι να αποφασιστεί τι όργανο θα έχει.

Ενεργό. Ορίζει ο χρήστης αν ένα μέλος είναι ενεργό. Μπορεί να αλλάξει την κατάσταση του.

Στολές: επιλέγουμε τι στολές εχει στη διάθεση του το μέλος επιλογή γίνεται με κλικ και σε περίπτωση που θέλουμε να επιλέγουμε παραπάνω από μια εγγραφή τότε πατάμε και το ctrl+click

Σχόλια: Γράφουμε ότι σχόλια θέλουμε για ένα άτομο πχ έχει άσθμα

Όταν συμπληρώσουμε τις εγγραφές πατάμε το κουμπί αποθήκευση και η εγγραφή περναει στο σύστημα μας.

Ευρεση μέλους

Στο text box εύρεσης γράφουμε ένα μέρος από αυτό που θέλουμε να βρούμε πχ Παπαδημητρίου, κρουστό,2241092512. Πατάμε το κουμπί εύρεση και μας εμφανίζει τις εγγραφές που ταιριάζουν στη λίστα. Πατώντας σε μια από τις εγγραφές εμφανίζονται τα στοιχεία της στα πεδία στα δεξιά της οθόνης.

Τροποποίηση Μέλους

Αφού βρουμε την εγγραφή που θέλουμε να τροποποιήσουμε τότε την επιλέγουμε. Αλλάζουμε τις τιμές της στα πεδία που θέλουμε. Οταν ολοκληρώσουμε τις αλλαγές πατάμε το κουμπί αποθήκευση και αποθηκεύονται οι αλλαγές μας .

Διαγραφή μέλους.

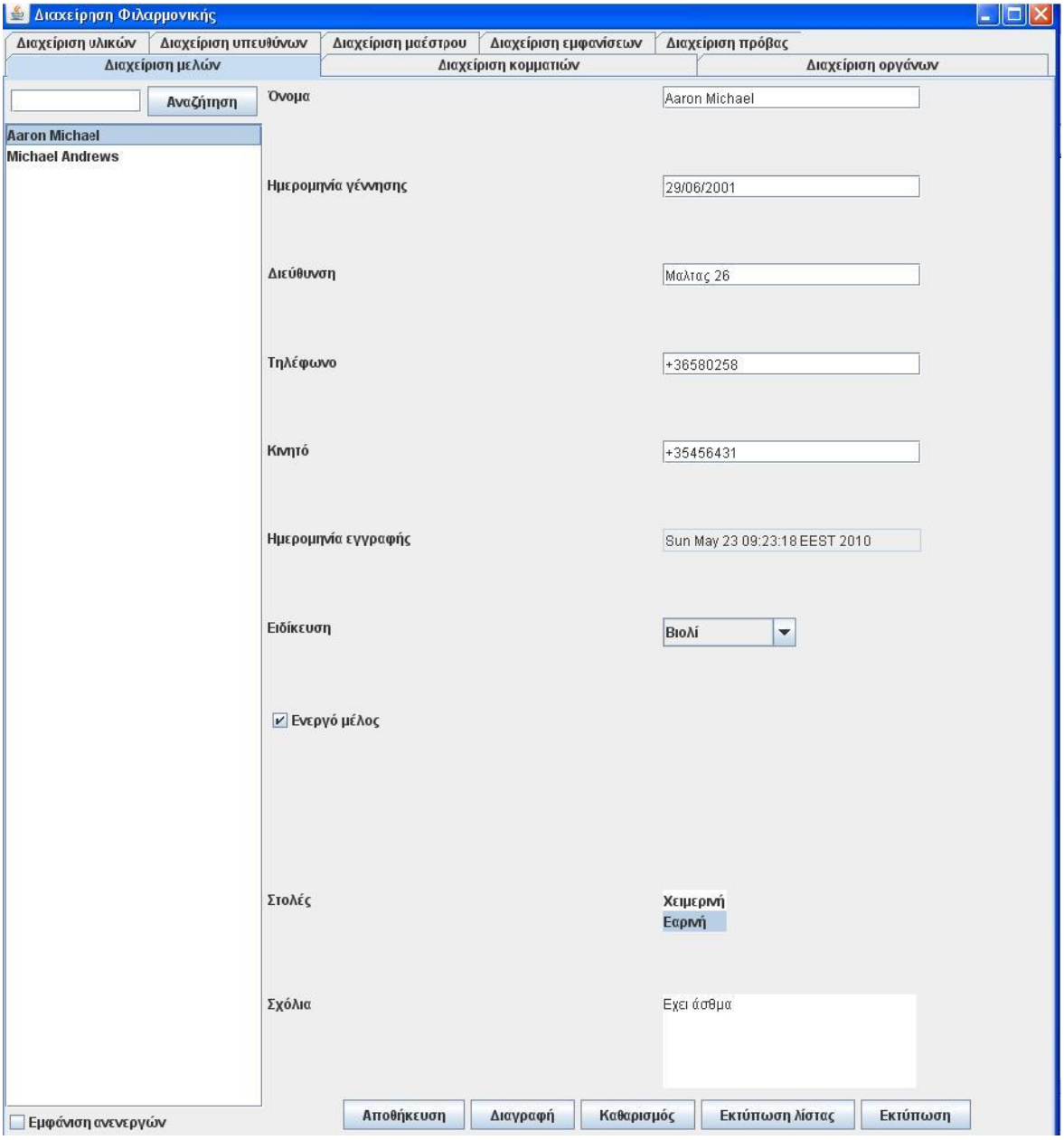
Βρίσκουμε και επιλέγουμε την εγγραφή που θέλουμε να διαγράψουμε.Αφου την επιλέξουμε πατάμε το κουμπι της διαγραφής.

Εκτύπωση Λίστας

Αφού έχουμε εγγραφές στη λίστα μας μπορούμε να επιλέξουμε το κουμπί εκτύπωση λίστας και θα δημιουργήσει ένα αρχείο txt με τα στοιχεία των μελών.

Εκτύπωση

Αφού επιλέξουμε μια εγγραφή πατάμε το κουμπί της εκτύπωσης και τα χαρακτηριστικά του μέλους εκτυπώνονται σε ένα αρχείο txt.



Διαχείριση Μουσικού κομματιού

Στη διαχείριση μουσικού κομματιου ο χρήστης διαμορφώνει τα μουσικά κομματια και τα χαρακτηριστικά τους.
Κατ αρχήν θα πρέπει να επιλέξει την καρτέλα μουσικά κομματια.Οταν την επιλέγει εμφανίζεται μπροστά του η οθόνη της διαχείρισης. Στα αριστερά υπάρχει η λίστα με τους τίτλους των κομματιών που είναι καταχωρημένα στο συστημα μας. Ακριβώς πάνω από τη λίστα υπάρχει το textbox που γράφουμε το κειμενο προς εύρεση και δίπλα του το κουμπί της ευρεσης. Στα δεξιά υπάρχουν τα πεδία εμφάνισης των στοιχείων του κομματιού. Τελος στο κάτω μέρος υπάρχουν τα κουμπιά των λειτουργιών (αποθήκευση, καθαρισμός).

Εισαγωγή Κομματιού

Επιλέγουμε την καρτέλα Διαχείριση Μουσικού Κομματιου.Πατάμε το κουμπί καθαρισμού ώστε να μείνουν κενα τα πεδία. Συμπληρώνουμε τα πεδία που θέλουμε

Τίτλος: Τον τίλο του μουσικού κομματιυ που εισαγουμε.

Αριθμός: Τον αριθμό του φακέλου που είναι αποθηκευμένο το κομμάτι στη βιβλιοθήκη.

Τύπος: Τον τυπο του κομματιού πχ Marcia Religiosa, Συναυλιακό, εθνικός ύμνος

Κατάσταση: την κατασταση του.Μπορεί να είναι ολοκληρωμένο, ελλειπές, δεν εχει το μερος των κρουστών.Χρησημοποιήται για να γνωρίσει ο αρχιμουσικός την καταστη του.

Σχόλια: Διαφορα σχόλια σχετικά με το κομματι.

Ευρεση Μουσικού κομματιού

Στο text box εύρεσης γράφουμε ένα μέρος από αυτό που θέλουμε να βρούμε πχ ελληνικός ύμνος, 34. Πατάμε το κουμπί εύρεση και μας εμφανίζει τις εγγραφές που ταιριάζουν στη λίστα. Πατώντας σε μια από τις εγγραφές εμφανίζονται τα στοιχεία της στα πεδία στα δεξιά της οθόνης.

Τροποποίηση Κομματιού

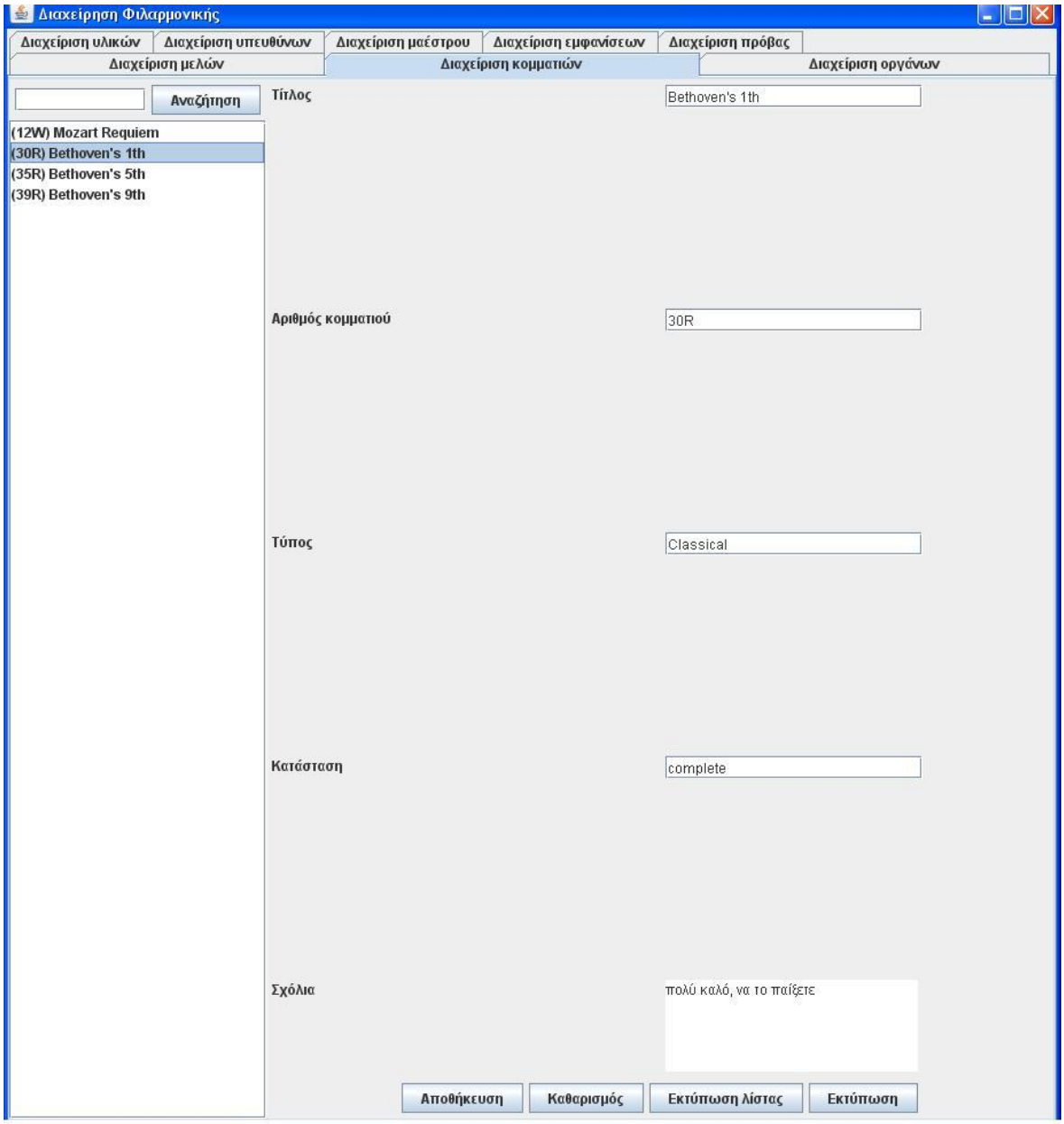
Αφού βρουμε την εγγραφή που θέλουμε να τροποποιήσουμε τότε την επιλέγουμε. Αλλάζουμε τις τιμές της στα πεδία που θέλουμε. Οταν ολοκληρώσουμε τις αλλαγές πατάμε το κουμπί αποθήκευση και αποθηκεύονται οι αλλαγές μας .

Εκτύπωση Λίστας

Αφού έχουμε εγγραφές στη λίστα μας μπορούμε να επιλέξουμε το κουμπί εκτύπωση λίστας και θα δημιουργήσει ένα αρχείο txt με τα στοιχεία του κομματιού.

Εκτύπωση

Αφού επιλέξουμε μια εγγραφή πατάμε το κουμπί της εκτύπωσης και τα χαρακτηριστικά του κομματιού εκτυπώνονται σε ένα αρχείο txt.



Διαχείριση Εξοπλισμού

Στη διαχείριση Εξοπλισμού ο χρήστης διαμορφώνει τον εξοπλισμό που έχει στη διαθεσή του όπως και τα χαρακτηριστικά τους.

Κατ αρχήν θα πρέπει να επιλέξει την καρτέλα Διαχειριση Εξοπλισμού. Οταν την επιλέγει εμφανίζεται μπροστά του η οθόνη της διαχείρισης. Στα αριστερά υπάρχει η λίστα με τα είδη που είναι καταχωρημένα στο συστημα μας. Ακριβώς πάνω από τη λίστα υπάρχει το textbox που γράφουμε το κειμενο προς εύρεση και δίπλα του το κουμπί της ευρεσης. Στα δεξιά υπάρχουν τα πεδία εμφάνισης των στοιχείων

του υλικού. Τελος στο κάτω μέρος υπάρχουν τα κουμπιά των λειτουργιών (αποθήκευση, καθαρισμός, διαγραφή).

Εισαγωγή Υλικού-Εξοπλισμού

Επιλέγουμε την καρτέλα Διαχείριση Εξοπλισμού.Πατάμε το κουμπί καθαρισμού ώστε να μείνουν κενα τα πεδία. Συμπληρώνουμε τα πεδία που θέλουμε

Είδος: Γράφουμε το είδος αναλόγιο, κονσόλα ήχου

Τύπος : Γράφουμε τον τύπο πχ ollivetti kr-450

Κατάσταση: την κατάσταση που βρίσκεται πχ σπασμένη βάση

Σχόλια: διαφορα σχόλια που μπορούμε να γράψουμε.

Όταν συμπληρώσουμε τα πεδία πατάμε το κουμπί της αποθήκευσης

Ευρεση Εξοπλισμού

Στο text box εύρεσης γράφουμε ένα μέρος από αυτό που θέλουμε να βρούμε πχ αναλόγιο. Πατάμε το κουμπί εύρεση και μας εμφανίζει τις εγγραφές που ταιριάζουν στη λίστα. Πατώντας σε μια από τις εγγραφές εμφανίζονται τα στοιχεία της στα πεδία στα δεξιά της οθόνης.

Τροποποίηση Εξοπλισμού

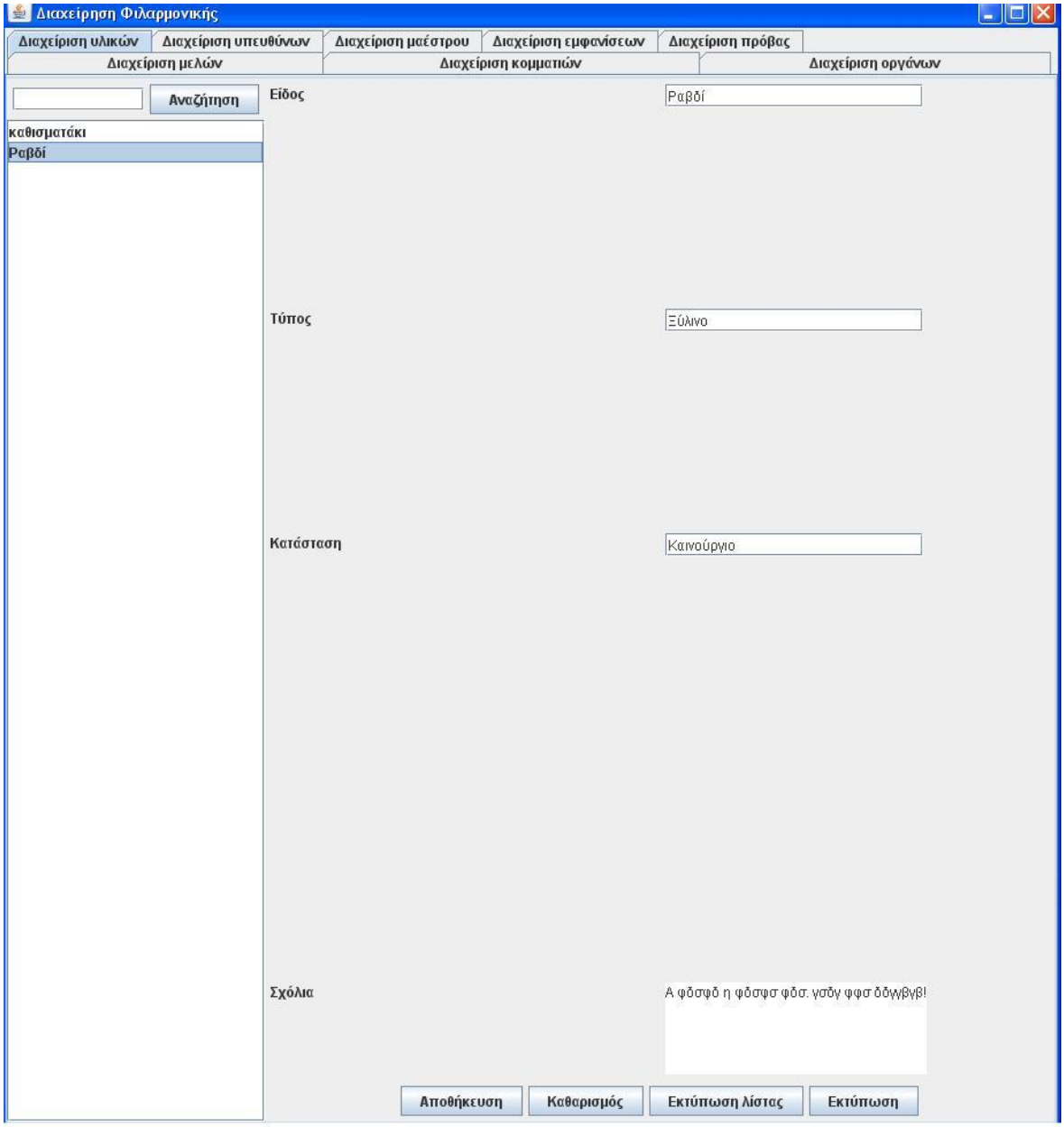
Αφού βρουμε την εγγραφή που θέλουμε να τροποποιήσουμε τότε την επιλέγουμε. Αλλάζουμε τις τιμές της στα πεδία που θέλουμε. Όταν ολοκληρώσουμε τις αλλαγές πατάμε το κουμπί αποθήκευση και αποθηκεύονται οι αλλαγές μας .

Εκτύπωση Λίστας

Αφού έχουμε εγγραφές στη λίστα μας μπορούμε να επιλέξουμε το κουμπί εκτύπωση λίστας και θα δημιουργήσει ένα αρχείο txt με τα στοιχεία του εξοπλισμού.

Εκτύπωση

Αφού επιλέξουμε μια εγγραφή πατάμε το κουμπί της εκτύπωσης και τα χαρακτηριστικά του εξοπλισμού εκτυπώνονται σε ένα αρχείο txt.



Διαχείριση Μουσικού οργάνου

Στη διαχείριση μουσικού οργάνου ο χρήστης διαμορφώνει τα μουσικά οργάνα και τα χαρακτηριστικά τους.
Κατ αρχήν θα πρέπει να επιλέξει την καρτέλα μουσικά οργάνου.Οταν την επιλέγει εμφανίζεται μπροστά του η οθόνη της διαχείρισης. Στα αριστερά υπάρχει η λίστα με τους ονόματα που είναι καταχωρημένα στο συστημα μας. Ακριβώς πάνω από τη λίστα υπάρχει το textbox που γράφουμε το κειμενο προς εύρεση και δίπλα του το κουμπί της ευρεσης. Στα δεξιά υπάρχουν τα πεδία εμφάνισης των στοιχείων του οργάνου. Τελος στο κάτω μέρος υπάρχουν τα κουμπιά των λειτουργιών (αποθήκευση, καθαρισμός).

Εισαγωγή Υλικού-Εξοπλισμού

Επιλέγουμε την καρτέλα Διαχείριση Μουσικού οργάνου. Πατάμε το κουμπί καθαρισμού ώστε να μείνουν κενα τα πεδία. Συμπληρώνουμε τα πεδία που θέλουμε

Serial: Εισάγουμε το σειριακό αριθμό του μουσικού οργάνου.

Μουσικό όργανο: Επιλέγουμε από τη λίστα το τυπο του μουσικού οργάνου.

Τύπος: Επιλέγουμε τον τύπο

Κατάσταση: Γράφουμε την κατάσταση που βρίσκεται

Αφού συμπληρώσουμε τα πεδία μας πατάμε το κουμπί αποθήκευσης για να εισαχθεί η εγγραφή στο συστημα μας.

Εύρεση Μουσικού οργάνου

Στο text box εύρεσης γράφουμε ένα μέρος από αυτό που θέλουμε να βρούμε πχ τρομπέτα, ΒΑ5469888. Πατάμε το κουμπί εύρεση και μας εμφανίζει τις εγγραφές που ταιριάζουν στη λίστα. Πατώντας σε μια από τις εγγραφές εμφανίζονται τα στοιχεία της στα πεδία στα δεξιά της οθόνης.

Τροποποίηση Μουσικού οργάνου

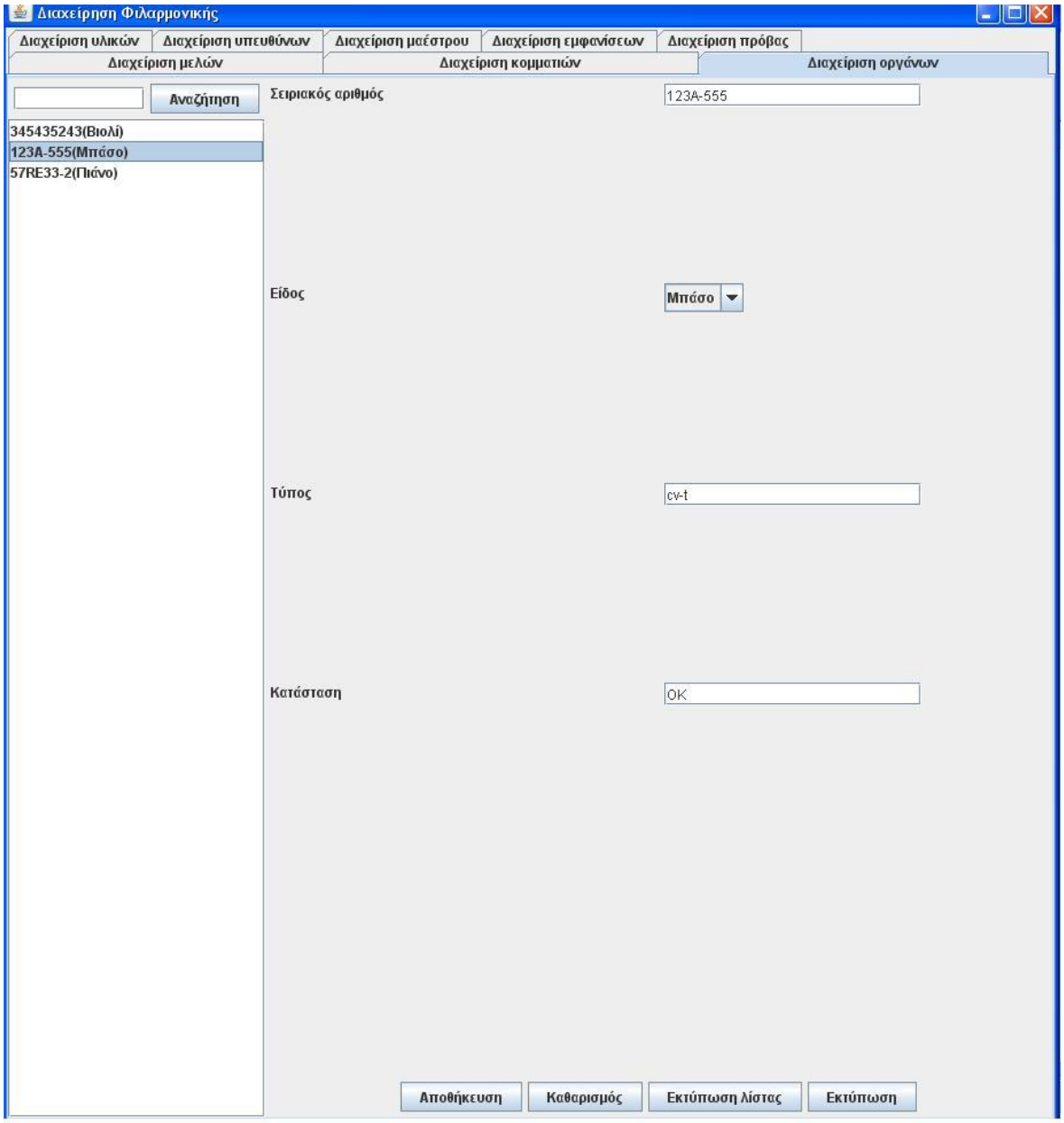
Αφού βρουμε την εγγραφή που θέλουμε να τροποποιήσουμε τότε την επιλέγουμε. Αλλάζουμε τις τιμές της στα πεδία που θέλουμε. Οταν ολοκληρώσουμε τις αλλαγές πατάμε το κουμπί αποθήκευση και αποθηκεύονται οι αλλαγές μας .

Εκτύπωση Λίστας

Αφού έχουμε εγγραφές στη λίστα μας μπορούμε να επιλέξουμε το κουμπί εκτύπωση λίστας και θα δημιουργήσει ένα αρχείο txt με τα στοιχεία των οργάνων.

Εκτύπωση

Αφού επιλέξουμε μια εγγραφή πατάμε το κουμπί της εκτύπωσης και τα χαρακτηριστικά του μουσικού οργάνου εκτυπώνονται σε ένα αρχείο txt.



Διαχείριση Αρχιμουσικού

Στη διαχείριση Αρχιμουσικού ο χρήστης διαμορφώνει τους αρχιμουσικούς και τα στοιχεία τους.

Κατ αρχήν θα πρέπει να επιλέξει την καρτέλα Διαχείριση Αρχιμουσικού .Οταν την επιλέγει εμφανίζεται μπροστά του η οθόνη της διαχείρισης. Στα αριστερά υπάρχει η λίστα με τους ονόματα που είναι καταχωρημένα στο συστημα μας. Ακριβώς

πάνω από τη λίστα υπάρχει το textbox που γράφουμε το κειμενο προς εύρεση και δίπλα του το κουμπί της ευρεσης. Στα δεξιά υπάρχουν τα πεδία εμφάνισης των στοιχείων του οργάνου. Τελος στο κάτω μέρος υπάρχουν τα κουμπιά των λειτουργιών (αποθήκευση, καθαρισμός).

Αποθήκευση Αρχιμουσικού

Πηγαίνουμε στην καρτέλα Διαχείριση αρχιμουσικού. Εισάγουμε στα πεδία τα δεδομένα που γνωρίζουμε.

Όνομα: Γράφουμε το όνομα του αρχιμουσικού. Κατά προτίμηση πρώτα το επώνυμο για να υπάρχει μια ομοιομορφία.

Διεύθυνση: τη διεύθυνση κατοικίας του.

Τηλέφωνο: Το σταθερό τηλέφωνο επικοινωνίας. Το πεδίο μπορεί να δεχτεί και περισσότερα τηλέφωνα χωρισμένα με κομμα ή παύλα

Κινητό: Το προσωπικό κινητό τηλέφωνο

Ημερομηνία Εναρξης: Ημερομηνία που ανέλαβε υπηρεσία.

Ημερομηνια λήξης. Την ημερομηνία λήξης θητείας

Διπλώματα: Εισάγουμε τα διπλώματα-πτυχία που έχει στη διαθεση του.

Όταν συμπληρώσουμε τα στοιχεία πατάμε αποθήκευση.

Εύρεση αρχιμουσικού

Στο text box εύρεσης γράφουμε ένα μέρος από αυτό που θέλουμε να βρούμε πχ Παπαδημητρίου,2241092512. Πατάμε το κουμπί εύρεση και μας εμφανίζει τις εγγραφές που ταιριάζουν στη λίστα. Πατώντας σε μια από τις εγγραφές εμφανίζονται τα στοιχεία της στα πεδία στα δεξιά της οθόνης.

Τροποποίηση αρχιμουσικού

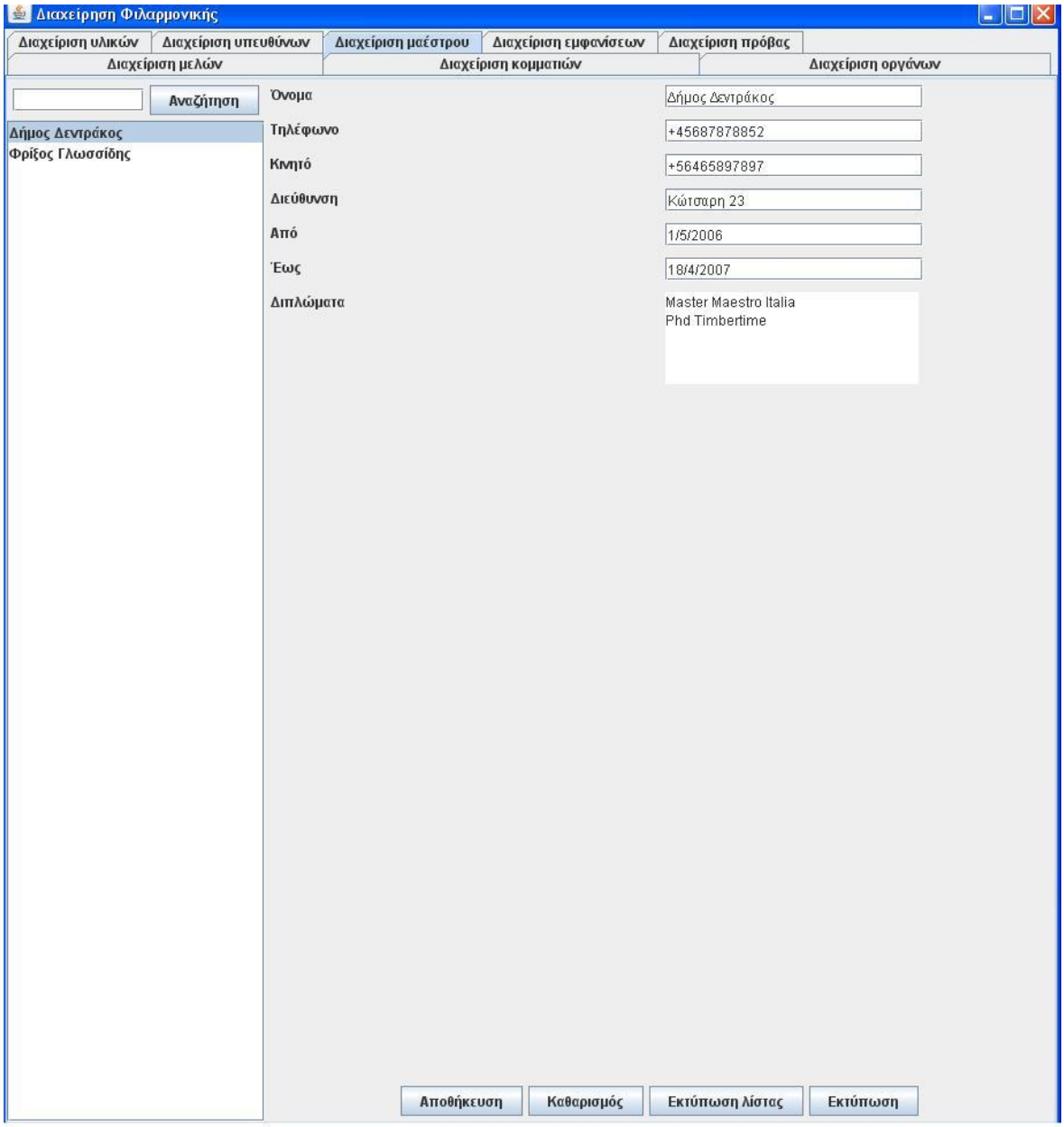
Αφού βρουμε την εγγραφή που θέλουμε να τροποποιήσουμε τότε την επιλέγουμε. Αλλάζουμε τις τιμές της στα πεδία που θέλουμε. Οταν ολοκληρώσουμε τις αλλαγές πατάμε το κουμπί αποθήκευση και αποθηκεύονται οι αλλαγές μας .

Εκτύπωση Λίστας

Αφού έχουμε εγγραφές στη λίστα μας μπορούμε να επιλέξουμε το κουμπί εκτύπωση λίστας και θα δημιουργήσει ένα αρχείο txt με τα στοιχεία των αρχιμουσικών.

Εκτύπωση

Αφού επιλέξουμε μια εγγραφή πατάμε το κουμπί της εκτύπωσης και τα χαρακτηριστικά του αρχιμουσικού εκτυπώνονται σε ένα αρχείο txt



Διαχείριση Υπεύθυνου

Στη διαχείριση Υπεύθυνου ο χρήστης διαμορφώνει τους υπεύθυνους και τα στοιχεία τους.

Κατ αρχήν θα πρέπει να επιλέξει την καρτέλα Διαχείριση Υπεύθυνου .Οταν την επιλέγει εμφανίζεται μπροστά του η οθόνη της διαχείρισης. Στα αριστερά υπάρχει η λίστα με τους ονόματα που είναι καταχωρημένα στο συστημα μας. Ακριβώς πάνω από τη λίστα υπάρχει το textbox που γράφουμε το κειμενο προς εύρεση και

δίπλα του το κουμπί της ευρεσης. Στα δεξιά υπάρχουν τα πεδία εμφάνισης των στοιχείων του οργάνου. Τελος στο κάτω μέρος υπάρχουν τα κουμπιά των λειτουργιών (αποθήκευση, καθαρισμός).

Αποθήκευση Υπεύθυνου

Πηγαίνουμε στην καρτέλα Διαχείριση Υπεύθυνου. Εισάγουμε στα πεδία τα δεδομένα που γνωρίζουμε .

Όνομα: Γράφουμε το όνομα του υπεύθυνου. Κατά προτίμηση πρώτα το επώνυμο για να υπάρχει μια ομοιομορφία.

Διεύθυνση: τη διεύθυνση κατοικίας του.

Τηλέφωνο: Το σταθερό τηλέφωνο επικοινωνίας. Το πεδίο μπορεί να δεχτεί και περισσότερα τηλέφωνα χωρισμένα με κομμα ή παύλα

Κινητό: Το προσωπικό κινητό τηλέφωνο

Ημερομηνία Έναρξης: Ημερομηνία που ανέλαβε υπηρεσία.

Ημερομηνία λήξης. Την ημερομηνία λήξης θητείας

Όταν συμπληρώσουμε τα στοιχεία πατάμε αποθήκευση.

Εύρεση υπεύθυνου

Στο text box εύρεσης γράφουμε ένα μέρος από αυτό που θέλουμε να βρούμε πχ Παπαδημητρίου,2241092512. Πατάμε το κουμπί εύρεση και μας εμφανίζει τις

εγγραφές που ταιριάζουν στη λίστα. Πατώντας σε μια από τις εγγραφές εμφανίζονται τα στοιχεία της στα πεδία στα δεξιά της οθόνης.

Τροποποίηση υπεύθυνου

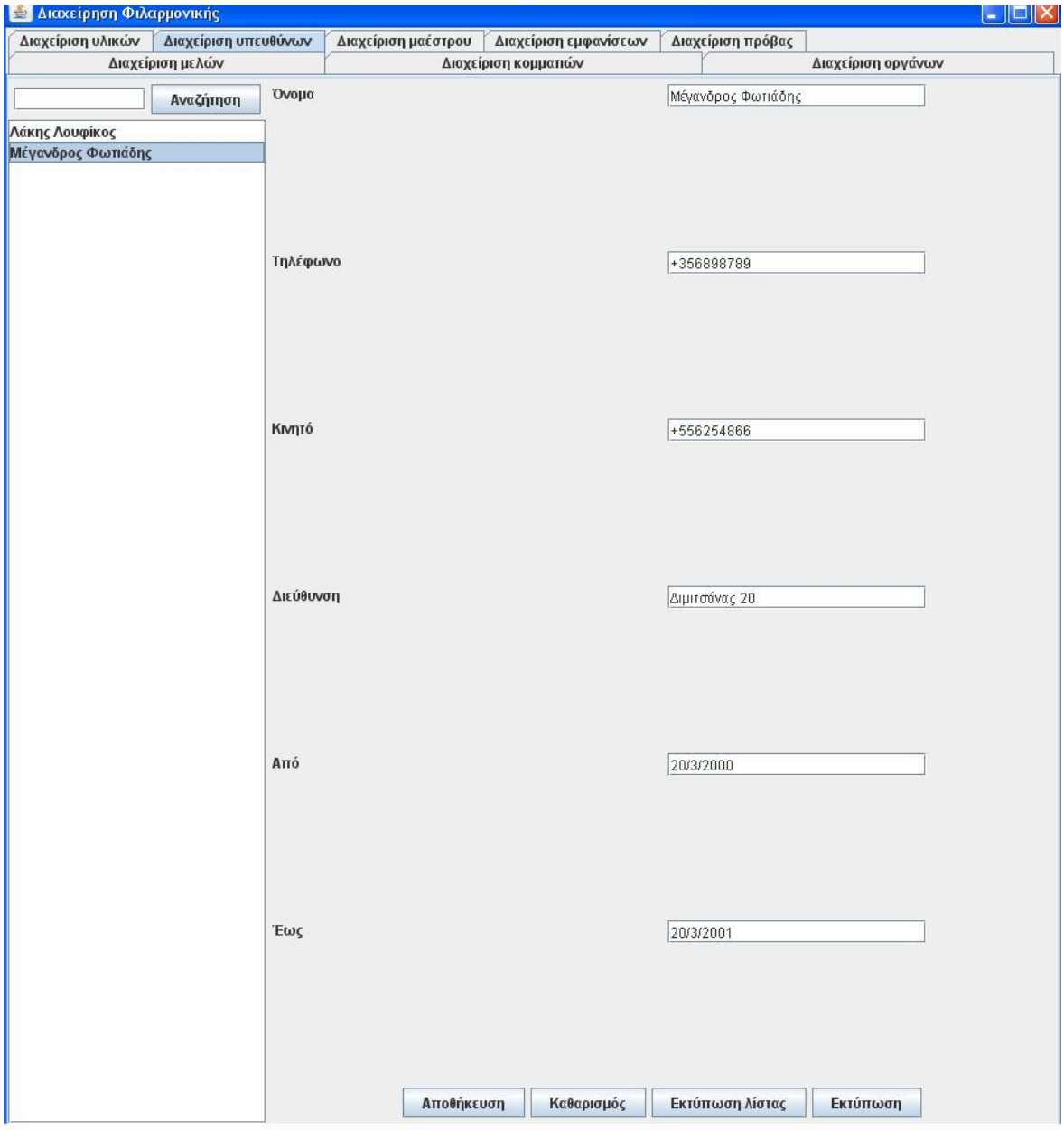
Αφού βρούμε την εγγραφή που θέλουμε να τροποποιήσουμε τότε την επιλέγουμε. Αλλάζουμε τις τιμές της στα πεδία που θέλουμε. Όταν ολοκληρώσουμε τις αλλαγές πατάμε το κουμπί αποθήκευση και αποθηκεύονται οι αλλαγές μας .

Εκτύπωση Λίστας

Αφού έχουμε εγγραφές στη λίστα μας μπορούμε να επιλέξουμε το κουμπί εκτύπωση λίστας και θα δημιουργήσει ένα αρχείο txt με τα στοιχεία της υπέυθυνου.

Εκτύπωση

Αφού επιλέξουμε μια εγγραφή πατάμε το κουμπί της εκτύπωσης και τα χαρακτηριστικά της υπέυθυνου εκτυπώνονται σε ένα αρχείο txt.



Διαχείριση εμφάνισης

Στη διαχείριση Εμφάνισης ο χρήστης διαμορφώνει τις εμφανίσεις και τα στοιχεία τους.

Κατ αρχήν θα πρέπει να επιλέξει την καρτέλα Διαχείριση Εμφάνισης .Οταν την επιλέγει εμφανίζεται μπροστά του η οθόνη της διαχείρισης. Στα αριστερά υπάρχει η λίστα με τις καταχωρημένες εμφανίσεις . Ακριβώς πάνω από τη λίστα υπάρχει το textbox που γράφουμε το κειμενο προς εύρεση και δίπλα του το κουμπί της

ευρεσης. Στα δεξιά υπάρχουν τα πεδία εμφάνισης των στοιχείων της εμφάνισης, η λίστα των μελών,των μουσικών κομματιών και τα combo box των αρχιμουσικών και των υπεύθυνων. Τελος στο κάτω μέρος υπάρχουν τα κουμπιά των λειτουργιών (αποθήκευση, καθαρισμός, εκτύπωση, εκτύπωση λίστας).

Αποθήκευση Εμφάνισης

Πηγαίνουμε στην καρτέλα Διαχείριση Εμφάνισης. Εισάγουμε στα πεδία τα δεδομένα μας

Περίσταση: Γράφουμε το είδος της εμφάνισης

Ημερομηνία: Ημερομηνία διεξαγωγής της εμφάνισης

Ωρα: Ωρα συγκέντρωσης των μελών

Τόπος: Τοπος συγκέντρωσης

Τρόπος μετακίνησης: Επιλέγουμε από τη λίστα των τρόπο μεταφοράς των μελών στην εμφάνιση

Υπεύθυνος: Επιλέγουμε τον υπεύθυνο

Μαέστρος: Επιλέγουμε τον αρχιμουσικό

Μέλη: Επιλέγουμε με το mouse μας τα μέλη που συμμετείχαν(ctrl+click στο μέλος)

Κομμάτια: Επιλέγουμε τα μουσικά κομμάτια που παίχθηκαν

Σχόλια; Σημειώνουμε τυχόν σχόλια ή παρατηρήσεις

Όταν συμπληρώσουμε τα στοιχεία πατάμε αποθήκευση.

Εύρεση Εμφάνισης

Στο text box εύρεσης γράφουμε ένα μέρος από αυτό που θέλουμε να βρούμε πχ Δεκαπεντάυγουστος. Πατάμε το κουμπί εύρεση και μας εμφανίζει τις εγγραφές που ταιριάζουν στη λίστα. Πατώντας σε μια από τις εγγραφές εμφανίζονται τα στοιχεία της στα πεδία στα δεξιά της οθόνης.

Τροποποίηση Εμφάνισης

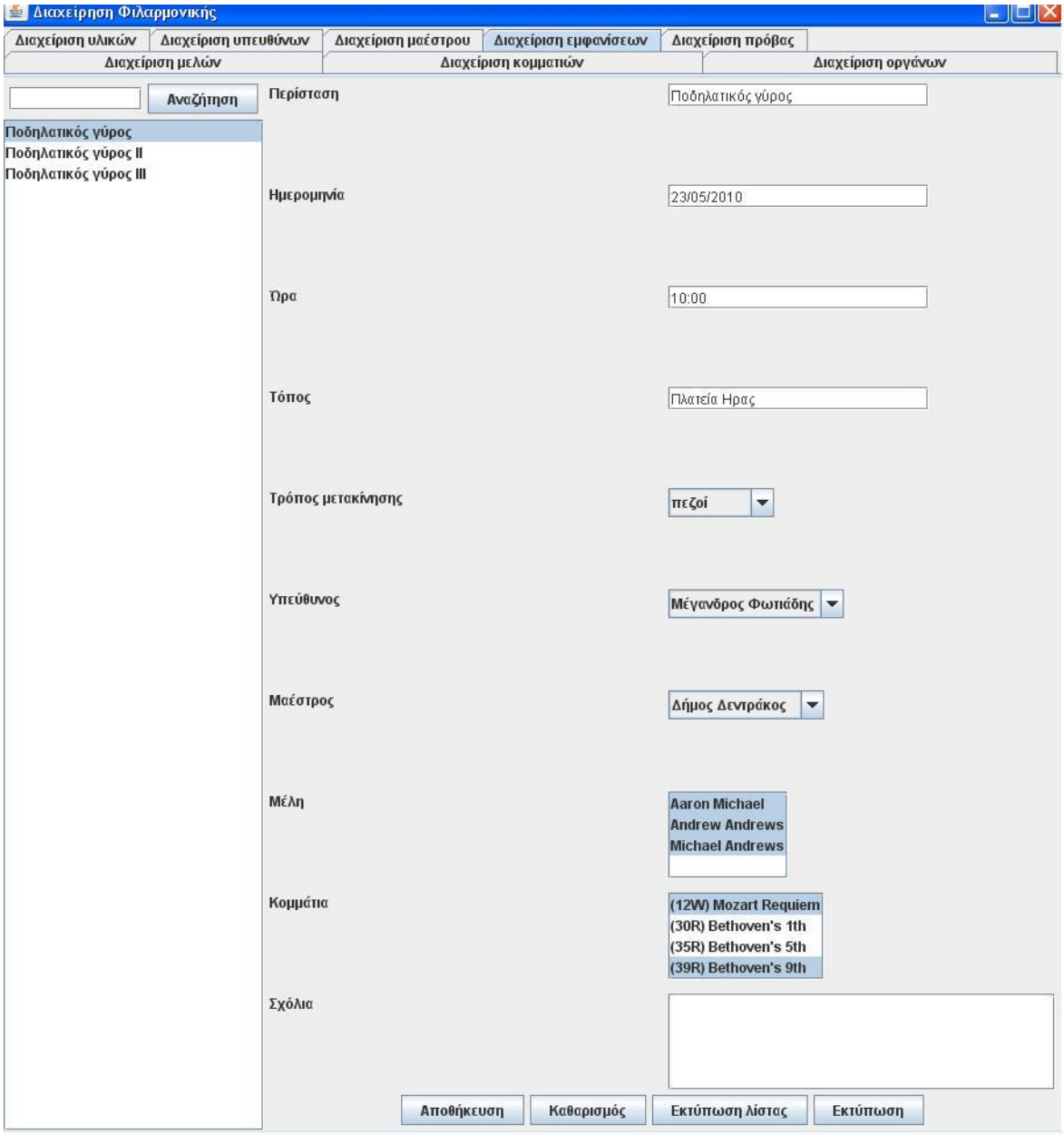
Αφού βρούμε την εγγραφή που θέλουμε να τροποποιήσουμε τότε την επιλέγουμε. Αλλάζουμε τις τιμές της στα πεδία που θέλουμε. Προσθαφαιρέτουμε μέλη κομματια ,αλλάζουμε υπεύθνο ή αρχιμουσικό. Όταν ολοκληρώσουμε τις αλλαγές πατάμε το κουμπί αποθήκευση και αποθηκεύονται οι αλλαγές μας .

Εκτύπωση Λίστας

Αφού έχουμε εγγραφές στη λίστα μας μπορούμε να επιλέξουμε το κουμπί εκτύπωση λίστας και θα δημιουργήσει ένα αρχείο txt με τα στοιχεία της εμφάνισης.

Εκτύπωση

Αφού επιλέξουμε μια εγγραφή πατάμε το κουμπί της εκτύπωσης και τα χαρακτηριστικά της εμφάνισης εκτυπώνονται σε ένα αρχείο txt.



Διαχείριση Πρόβας

Στη διαχείριση πρόβας ο χρήστης διαμορφώνει τις πρόβεςς και τα στοιχεία τους.

Κατ αρχήν θα πρέπει να επιλέξει την καρτέλα Διαχείριση πρόβας .Οταν την επιλέγει εμφανίζεται μπροστά του η οθόνη της διαχείρισης. Στα αριστερά υπάρχει η λίστα με τις καταχωρημένες πρόβες. Ακριβώς πάνω από τη λίστα υπάρχει το textbox που γράφουμε το κειμενο προς εύρεση και δίπλα του το κουμπί της

ευρεσης. Στα δεξιά υπάρχουν τα πεδία των στοιχείων της πρόβας, η λίστα των μελών,των μουσικών κομματιών και το combo box των αρχιμουσικών. Τελος στο κάτω μέρος υπάρχουν τα κουμπιά των λειτουργιών (αποθήκευση, καθαρισμός, εκτύπωση, εκτύπωση λίστας).

Αποθήκευση Πρόβας

Πηγαίνουμε στην καρτέλα Διαχείριση Πρόβας. Εισάγουμε στα πεδία τα δεδομένα μας

Ημερομηνία: Ημερομηνία διεξαγωγής της πρόβας

Ωρα Εναρξης: Ωρα συγκέντρωσης των μελών

Ωρα Λήξης: Ωρα αποχώρησης των μελών

Μαέστρος: Επιλέγουμε τον αρχιμουσικό

Μέλη: Επιλέγουμε με το mouse μας τα μέλη που συμμετείχαν(ctrl+click στο μέλος)

Κομμάτια: Επιλέγουμε τα μουσικά κομμάτια που παίχθηκαν

Σχόλια; Σημειώνουμε τυχόν σχόλια ή παρατηρήσεις

Όταν συμπληρώσουμε τα στοιχεία πατάμε αποθήκευση.

Εύρεση Πρόβας

Στο text box εύρεσης γράφουμε ένα μέρος από αυτό που θέλουμε να βρούμε πχ 22/10/10. Πατάμε το κουμπί εύρεση και μας εμφανίζει τις εγγραφές που

ταιριάζουν στη λίστα. Πατώντας σε μια από τις εγγραφές εμφανίζονται τα στοιχεία της στα πεδία στα δεξιά της οθόνης.

Τροποποίηση Πρόβας

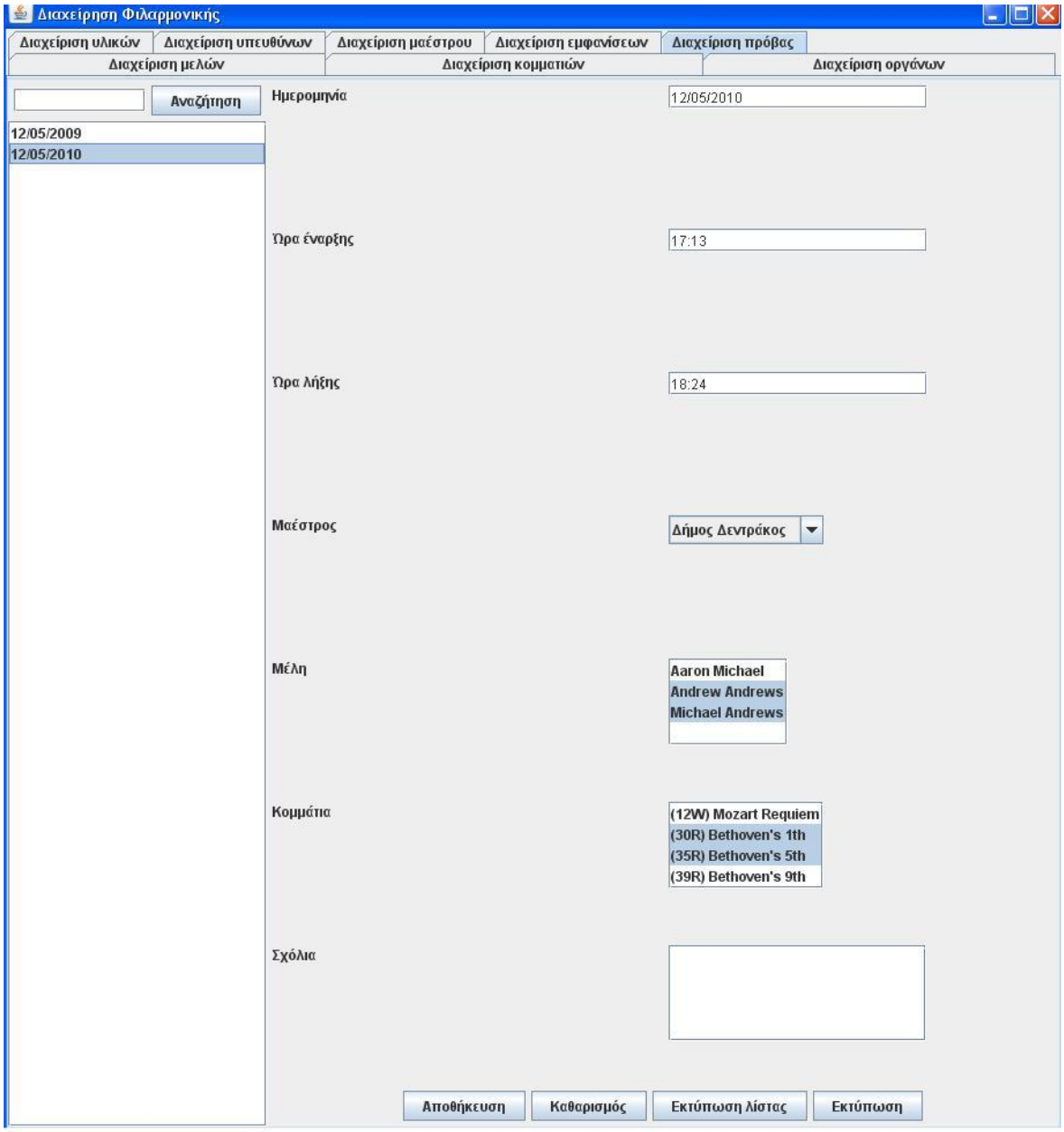
Αφού βρούμε την εγγραφή που θέλουμε να τροποποιήσουμε τότε την επιλέγουμε. Αλλάζουμε τις τιμές της στα πεδία που θέλουμε. Προσθαφαιρούμε μέλη κομμάτια ,αλλάζουμε αρχιμουσικό. Όταν ολοκληρώσουμε τις αλλαγές πατάμε το κουμπί αποθήκευση και αποθηκεύονται οι αλλαγές μας .

Εκτύπωση Λίστας

Αφού έχουμε εγγραφές στη λίστα μας μπορούμε να επιλέξουμε το κουμπί εκτύπωση λίστας και θα δημιουργήσει ένα αρχείο txt με τα στοιχεία της πρόβας.

Εκτύπωση

Αφού επιλέξουμε μια εγγραφή πατάμε το κουμπί της εκτύπωσης και τα χαρακτηριστικά της πρόβας εκτυπώνονται σε ένα αρχείο txt.


Διαχείρηση Φιλαρμονικής
Διαχείριση υλικών
Διαχείριση υπευθύνων
Διαχείριση μαέστρου
Διαχείριση εμφανίσεων
Διαχείριση πρόβας
Διαχείριση μελών
Διαχείριση κομματιών
Διαχείριση οργάνων
Αναζήτηση
12/05/2009
12/05/2010
Ημερομηνία
12/05/2010
Ώρα έναρξης
17:13
Ώρα λήξης
18:24
Μαέστρος
Δήμος Δεντράκος
Μέλη
Aaron Michael
Andrew Andrews
Michael Andrews
Κομμάτια
(12W) Mozart Requiem
(30R) Bethoven's 1th
(35R) Bethoven's 5th
(39R) Bethoven's 9th
Σχόλια
Αποθήκευση
Καθαρισμός
Εκτύπωση λίστας
Εκτύπωση