Αντικειμενοστρεφής Προγραμματισμός 2
Εργασία χειμερινού εξαμήνου 2022-2023
Μέρος 2

Δημόπουλος Μιχαήλ Άγγελος it21822
Βασίλης Καραουλάνης it217114

Το δεύτερο μέρος της εργασίας, στοχεύει στην δημιουργία παραθυρικής εφαρμογής η οποία χρησιμοποιεί την υλοποίηση του πρώτου μέρους. Ο κώδικας του πρώτου παραδοτέου δεν μας βοήθησε στην συνέχεια της εργασίας, για αυτό κάναμε κάποιες αλλαγές ώστε κάθε κλάση να υλοποιεί συγκεκριμένους στόχους που θα βοηθήσουν στην συνολική δημιουργία της εφαρμογής.

Διάρθρωση των κλάσεων:

Αρχικά η βασική κλάση, που παρουσιάζει την εφαρμογή ονομάζεται GUI και είναι αυτή που δημιουργεί και εμφανίζει το Frame. Κάνει implement το ActionListener interface καθώς και τον PlayerListener της musicplayer βιβλιοθήκης. Διαθέτει την μέθοδο createAndShowGUI και την addListeners. Στην main κλάση δημιουργούμε ένα αντικείμενο GUI και καλούμε τις παραπάνω δύο μεθόδους.

Εκτός από την παραπάνω κλάση δημιουργούμε επίσης την κλάση Song, η οποία διαθέτει 2 βασικά πεδία το name και το path, καθώς και διάφορα άλλα που χρειαζόμαστε για το προαιρετικό μέρος.

Στην συνέχεια έχουμε την κλάση Playlist, η οποία διαθέτει μια βασική μέθοδο την createPlaylist. Η μέθοδος αυτή καλείται με ένα File και δημιουργεί μια λίστα με αντικείμενα Song. Επίσης έχει και αρκετές βοηθητικές μεθόδους που θα χρειαστούμε μέσα από την κλάση GUI της εφαρμογής για να διαχειριστούμε την λίστα.

Για την στρατηγική δημιουργούμε την κλάση Strategy με την μέθοδο nextSong η οποία δέχεται ένα αντικείμενο Playlist και έναν ακέραιο με την θέση του τρέχοντος τραγουδιού και επιστρέφει το επόμενο τραγούδι σύμφωνα με την στρατηγική που έχει ορίσει ο χρήστης.

Επιπλέον διαθέτουμε την κλάση MetaData, για το προαιρετικό μέρος η οποία χρησιμοποιώντας την βιβλιοθήκη jaudiotagger επιστρέφει με την μέθοδο getMetaData(String path), τα meta-data του τραγουδιού. Ακόμα χρησιμοποιούμε την κλάση Highlighter η οποία κάνει extend τον Mouse Adapter, για να κάνουμε customized τα κουμπιά της εφαρμογής.

Τέλος θέλαμε να έχουμε την κλάση PlayerController, που θα είχε ως στόχο την δημιουργία ενός Player ο οποίος χρησιμοποιώντας την βιβλιοθήκη θα αναπαρήγαγε τα τραγούδια. Αυτό δούλευε κανονικά, αλλά δεν καταφέραμε να την συγχρονίσουμε με τα μηνύματα που εμφανίζονται στο GUI με αποτέλεσμα να τα υλοποιήσουμε στην ίδια κλάση (GUI).

Υλοποίηση:

Κλάση GUI:

createAndShowGUI():
Για την εμφάνιση της εφαρμογής στην κλάση GUI δημιουργούμε ένα βασικό frame με τρία panel. Το filePanel, το οποίο διαθέτει το κουμπί του Open, καθώς και ένα infoLabel το οποίο ενημερώνει για διάφορα γεγονότα κατά την διάρκεια εκτέλεσης της εφαρμογής. Το playlistPanel, είναι αυτό που θα παρουσιάζει τα τραγούδια του επιλεγμένου φακέλου ή της m3u λίστας (με ένα JList). Τέλος το controlPanel περιέχει όλα τα κουμπιά που χρειαζόμαστε καθώς και ένα JComboBox με τις στρατηγικές. Τα τρία αυτά Panel τα στοιχίζουμε σε south, center και north αντίστοιχα στο frame, το οποίο έχει border layout.

addListeners():
Η void συνάρτηση addListeners προσθέτει Action Listeners στα κουμπιά και στο combo box της εφαρμογής (με το this λόγω του implement). Αυτό μας επιτρέπει να κάνουμε Override την actionPerformed ώστε να διαχειριστούμε τα action event των components. Η addListener ακόμα προσθέτει και έναν PlayerListener στον Player p, με σκοπό την διαχείριση του σε κάθε αλλαγή του status του. Τέλος προσθέτουμε και έναν WindowListener στο frame, με έναν WindowAdapter, δουλειά του οποίου είναι να κλείνει τον Player (p.close) όταν κλείνει την εφαρμογή.

actionPerformed(ActionEvent actionEvent):
Ένα action Event διαθέτει την συνάρτηση getSource, η οποία επιστρέφει τον component που την κάλεσε. Θα έχουμε λοιπόν ένα if με 6 συνθήκες, μια για κάθε κουμπί και μια για το combo box.
Πρώτα ελέγχουμε την περίπτωση που έχουμε actionEvent στο open JButton. Στην περίπτωση αυτή δημιουργούμε ένα JFileChooser με SelectionMode αρχεία (m3u ή mp3) και καταλόγους. Αν η showOpenDialog επιστρέψει αποδεκτή τιμή, σημαίνει ότι ο χρήστης επέλεξε αρχείο ή κατάλογο τον οποίο περνάμε στην συνάρτηση showList.

showList():
Σκοπός της showList είναι η εμφάνιση της λίστα στο playlistPanel. Για να το πετύχουμε αυτό αρχικά ορίζουμε ένα νέο αντικείμενο private Playlist. Αρχικά ελέγχουμε αν το playlist έχει στοιχεία (από προηγούμενο open), ώστε να τα κάνουμε clear (χρησιμοποιώντας την βοηθητική συνάρτηση clear της κλάσης Playlist) αλλά και να τα αφαιρέσουμε από το panel. Δεν αρκεί μόνο η διαγραφή τους, καθώς χρειάζεται και η μέθοδος repaint του component η οποία κάνει erase και redraw (ασύγχρονα) το panel σε μικρό χρονικό διάστημα. Στην συνέχεια δημιουργούμε ένα αντικείμενο Playlist (το playlist) και καλούμε την μέθοδο του, createPlaylist (την οποία εξηγούμε αναλυτικά παρακάτω) ώστε να δημιουργήσει την λίστα. Στην περίπτωση που η λίστα είναι κενή τότε κάνουμε set το text του infoLabel για να ενημερώσουμε τον χρήστη ότι ο κατάλογος ή το m3u δεν περιέχει κάτι που μπορούμε να αναπαράγουμε. Στην άλλη περίπτωση δημιουργούμε ένα νέο JList το songList. Αυτό που θέλουμε να πετύχουμε είναι η εμφάνιση των ονομάτων των τραγουδιών και όχι ολόκληρου του path. Για αυτό κάνοντας set τα data στο JList, δεν εισάγουμε το ArrayList με τα τραγούδια αλλά, χρησιμοποιούμε την βοηθητική συνάρτηση getSongNames της Playlist η οποία θα γεμίσει την λίστα μόνο με τα ονόματα. Αν η λίστα είναι έτοιμη, τότε μόνο προσθέτουμε στο controlPanel το cb, για επιλογή στρατηγικής και ένα JScrollPane. Η default τιμή της στρατηγικής είναι η Normal. Μετά ορίζουμε τον δείκτη της JList (songList) στο 0 και προσθέτουμε ένα ListSelectionListener, δουλειά του οποίου είναι να εμφανίζει σε κάθε αλλαγή τιμής τον τίτλο του τραγουδιού (και τα meta-data αν υπάρχουν). Τέλος προσθέτουμε και έναν mouseListener με έναν Mouse Adapter, ο οποίος αναπαράγει το τραγούδι με double click (με την playSong η οποία εξηγείται παρακάτω).

Κλάση Playlist:
Η κλάση αυτή έχει διάφορες μεθόδους για την δημιουργία, την επεξεργασία και την επιστροφή της λίστας.

createPlaylist(File file):
Έχει ίδια λογική με την διαδικασία που ακολουθούμε στο πρώτο μέρος, με κάποιες extra λειτουργίες. Δέχεται σαν όρισμα ένα File, που στο

βασικό if ελέγχεται για το αν είναι αρχείο ή κατάλογος. Στην περίπτωση καταλόγου, δεν μας αρκεί να τον κάνουμε προσπέλαση και να κρατήσουμε μόνο το mp3 αρχεία καθώς θέλουμε να προσθέσουμε στην λίστα και όλα τα mp3 αρχεία των υποκαταλόγων. Οπότε αν το file είναι directory καλούμε την συνάρτηση findAllMp3 με όρισμα το file. Εκεί έχουμε ένα for με το οποίο κάνουμε προσπέλαση όλα τα στοιχεία του καταλόγου χρησιμοποιώντας την listFiles. Αν το αρχείο είναι υπαρκτό mp3 αρχείο, τότε δημιουργούμε ένα νέο Song με το όνομα και το path του και το προσθέτουμε στο playlist (ArrayList<Song>). Αν το αρχείο είναι κατάλογος, καλούμε αναδρομικά την findAllMp3. Τέλος αν το αρχείο δεν είναι τίποτα από αυτά, εμφανίζουμε στον logger, με level info ότι το αρχείο δεν είναι αποδεκτό.
Στην άλλη περίπτωση που το αρχείο είναι m3u λίστα, ακολουθούμε τον κώδικα του πρώτου μέρους, με κάποιες τροποποιήσεις καθώς καταλάβαμε ότι δεν καλύπτουμε όλες τις περιπτώσεις. Η βασική λογική είναι η ίδια, έχουμε έναν Buffered Reader με τον οποίο διαβάζουμε μια μια τις γραμμές της λίστας ξεπερνώντας αυτές που είναι σχόλιο και αυτές που είναι κενές. Μια m3u λίστα είναι στην ουσία ένα text file με τα paths των audio αρχείων. Για να καλύψουμε κάθε ενδεχόμενο (relative ή absolute path), δημιουργήσαμε την μεταβλητή absolutePath. Έπειτα έχουμε τις εξής επιλογές: Η γραμμή να ξεκινάει με ‘/’ οπότε το absolute path θα είναι όλη η γραμμή. Αλλιώς το path είναι relative και θα πρέπει να το μετατρέψουμε σε absolute. Για να το πετύχουμε αυτό βρήκαμε έναν τρόπο προσθέτοντας το path του πατέρα (file.getParent) με το ‘/’ και το όνομα του αρχείου. Οπότε έτσι δημιουργούμε ένα νέο Song με path (absolute path) και name, το οποίο παίρνουμε χρησιμοποιώντας την συνάντηση getNameFromPath. Ακόμα εκτός από m3u η εφαρμογή μπορεί να ανοίγει και ένα mp3 αρχείο οπότε ελέγχουμε και αυτήν την περίπτωση.
Τέλος η Playlist περιέχει της μεθόδους getSongNames, getSong, clear, getSize, hasNext, isEmpty οι οποίες βοηθούν να διαχειριστούμε το αντικείμενο στην GUI κλάση.

Πίσω στην GUI κλάση ελέγχουμε τα action events των controlPanel. Αν ο χρήστης επιλέξει το play, καλούμε την συνάρτηση playSong η οποία παίρνει σαν όρισμα την θέση του τραγουδιού που έχουμε επιλέξει. (songList.getSelectedIndex).

playSong(int index):
Η playSong, είναι παρόμοια με την συνάρτηση του πρώτου παραδοτέου. Αρχικά ελέγχεται ότι η κατάσταση του Player είναι IDLE, καθώς σε κάθε άλλη περίπτωση κάνουμε stop. Δημιουργούμε ένα νέο FileInputStream, στο οποίο για να περάσουμε το path του τραγουδιού καλούμε την getSong (βοηθητική συνάρτηση της κλάσης Playlist) με την θέση index. Έχοντας το τραγούδι, με την getPath, έχουμε έτοιμο το αρχείο για αναπαραγωγή. Ακόμα ανανεώνουμε το text του infoLabel για να ενημερώσει ποιο τραγούδι παίζει, ενώ παράλληλα κρατάμε το όνομα του και το position του στην λίστα, καθώς οι μεταβλητές αυτές θα μας φανούν χρήσιμες στα υπόλοιπα κουμπιά του controlPanel. Τέλος κάνουμε catch και τα exception που μπορούν να εμφανιστούν και στον logger εμφανίζουμε μήνυμα Warning με το level SEVERE.

Ο κώδικας για τα actions του pause, resume, stop είναι παρόμοιος. Κάθε φορά ελέγχουμε το status. Pause κάνουμε μόνο όταν το status είναι PLAYING, resume μόνο όταν είναι PAUSE και stop μόνο όταν δεν είναι IDLE. Στις παραπάνω περιπτώσεις η τρέχων selected τιμή επιστρέφει στο τραγούδι που άλλαξε το status του (έχοντας το position από την playSong). Για το next, χρειάζεται αρχικά να αναλύσουμε πως υλοποιούμε την στρατηγική.
Δημιουργούμε ένα αντικείμενο Strategy και το αρχικοποιούμε στην showList με την τιμή ‘Normal’. Κάθε φορά που γίνεται κάποιο action στο cb (strategy check box), τότε κάνουμε set την στρατηγική με το νέο item. Επομένως κάθε φορά που γίνεται αλλαγή η κλάση Strategy, θα έχει το σωστό value.

Κλάση Strategy:
Η κλάση αυτή διαθέτει ένα πεδίο strategy, με getter και setter.

nextSong(Playlist playlist, int position):
Η μέθοδος αυτή καλείται μέσα από τον PlayerListener και την statusUpdate ενώ καλείται και υπό προϋποθέσεις μέσα από το next και το play. Δέχεται την λίστα με τα τραγούδια και την θέση του επιλεγμένου τραγουδιού. Αρχικά ελέγχει την τρέχουσα τιμή της στρατηγικής. Αν είναι Normal, τότε εξετάζει αν υπάρχει επόμενο τραγούδι στην λίστα (βοηθητική συνάρτηση hasNext της Playlist) επιστρέφοντας position+1.

Αλλιώς επιλέξαμε στην Normal στρατηγική να πηγαίνει στο πρώτο τραγούδι. Αν η στρατηγική είναι Order, τότε με την ίδια λογική επιστρέφουμε είτε position + 1 , είναι -1 (για να δείξουμε ότι η λίστα τελείωσε). Στην στρατηγική Loop επιστρέφουμε πολύ απλά το position. Τέλος στην Random στρατηγική ακολουθούμε την εξής λογική: Την πρώτη φορά που θα εκτελεστεί θα δημιουργήσουμε μια random λίστα. Την λίστα αυτή γεμίζουμε με ένα for με αριθμούς από το 0 μέχρι το size-1 της λίστας και την κάνουμε shuffle. Κάθε φορά που καλείται λοιπόν θα εξετάζεται αν η λίστα είναι null ώστε να την γεμίσει. Στην άλλη περίπτωση θα επιστρέφει το στοιχείο 0 της λίστας αφου πρώτα το διαγράψει. Με τον τρόπο αυτό σε κάθε nextSong, η random λίστα θα μικραίνει κατά ένα. Στο τελευταίο next πριν γίνει η επιστροφή του τελευταίου στοιχείου προσθέτουμε στην λίστα το -1. Οπότε μπαίνοντας πάλι στην nextSong υπάρχει μόνο το -1 το οποίο σημαίνει ότι η λίστα τελείωσε.

Πίσω στην κλάση GUI το κουμπί next αρχικά ελέγχει την τιμή της στρατηγικής και το status του Player. Μόνο στην περίπτωση που η στρατηγική είναι normal ή το Status είναι IDLE καλεί τη nextSong. Στις άλλες περιπτώσεις απλά κάνει stop τον Player θέτωντας το status σε IDLE ώστε να αφήσει τον Listener να διαχειριστεί το επόμενο τραγούδι.

statusUpdate(PlayerEvent playerEvent):
Κάθε φορά που ένα τραγούδι τελειώνει υπάρχουν τέσσερα διαφορετικά σενάρια, ένα για κάθε στρατηγική. Ο κώδικας του statusUpdate είναι γενικός, δηλαδή σε περίπτωση που εισάγουμε νέα στρατηγική δεν θα χρειαστεί κάποια αλλαγή. καθώς κάθε περίπτωση ελέγχεται στην κλάση Strategy. Το απλό σενάριο είναι όταν η στρατηγική είναι Normal. Τότε όταν η κατάσταση του Player γίνει IDLE απλά δεν κάνουμε τίποτα. Σε κάθε άλλη περίπτωση καλούμε την playSong δίνοντας της το value που επιστρέφει η nextSong της Strategy. Αν η nextSong γυρίσει αρνητικό αριθμό όπως είπαμε, η λίστα έχει τελειώσει οπότε ενημερώνουμε μέσα από το infoLabel τον χρήστη.

Κλάση MetaData:
Η κλάση MetaData, ανήκει στο προαιρετικό κομμάτι της εργασίας και χρησιμοποιεί την βιβλιοθήκη της jaudiotagger, για την αποθήκευση των μεταδεδομένων. Την βιβλιοθήκη αυτή χρησιμοποιώ εισάγοντας το dependency στο pom.xml.

getMetaData(String path):
Έχοντας ως όρισμα το path του τραγουδιού, η getMetaData επιστρέφει μια λίστα με τα δεδομένα του τραγουδιού. Αρχικά εντοπίζει το audio αρχείο (MP3File) και δημιουργεί ένα tag σε αυτό. Έπειτα δημιουργεί ένα ArrayList<String>, στο οποίο αποφάσισα να περάσω τα εξής μεταδεδομένα: artist, album, genre, language, year, rating. Η συνάρτηση επιστρέφει αυτό το ArrayList.

hasMetaData(String path):
Η συνάρτηση αυτή ελέγχει αν υπάρχουν μεταδεδομένα στο τραγούδι. Η βιβλιοθήκη αυτή διαθέτει την κλάση ID32Tag η οποία αποθηκεύει τα δεδομένα του Mp3 αρχείου και την κλάση ID3v1Tag, που είναι ένα παλαιότερο version. Επομένως ελέγχουμε με την hasID3v2Tag και την hasID3v1Tag αν υπάρχουν μεταδεδομένα.

Στην κλάση Playlist, κάθε φορά που δημιουργούμε ένα αντικείμενο Song, καλούμε την setMetaData με το path του Song. Εκεί δημιουργούμε ένα νέο αντικείμενο MetaData, και καλούμε την getMetaData. Τα αποτελέσματα που μας γυρνάει τα αποθηκεύουμε σε μια λίστα και έπειτα τα κάνουμε set στα πεδία της κλάσης Song. Η διαδικασία αυτή γίνεται μόνο αν η hasMetaData μας επιστρέψει true στην αρχή.

Για να επιστρέψουμε αυτά τα μεταδεδομένα στην κλάση Playlist δημιουργούμε την μέθοδο getMetaData που παίρνει σαν όρισμα ένα Song. Η κλάση αυτή επιστρέφει ένα String, το οποίο περιέχει μόνο όσα μεταδεδομένα έχουν οριστεί. Επειδή παρατηρήσαμε ότι μερικές φορές η hasMetaData επιστρέφει true ενώ τα μεταδεδομένα είναι κενά, κάνουμε όλους τους απαραίτητους ελέγχους.

Τέλος πίσω στην κλάση του GUI καλούμε την getMetaData κάθε φορά που ο χρήστης επιλέγει ένα Song. Τα μεταδεδομένα εμφανίζονται μόνο στην περίπτωση που υπάρχουν, αλλιώς εμφανίζεται μόνο ο τίτλος.

Χρήσιμες Πληροφορίες:

1) Κάθε φορά που καλείται η songPlay και παίζει το κομμάτι αναφέρει και την στρατηγική. Αν ο χρήστης αλλάξει την στρατηγική δυναμικά, αυτή θα αλλάξει στο text όταν ξανακαλεστεί η songPlay.
2) Κάθε φορά που γίνεται ένα file open φροντίζουμε να κάνουμε stop τον player. Αυτό μας δημιουργούσε ένα bug στο δεύτερο open, όταν η κατάσταση δεν ήταν IDLE, καθώς εκτελούταν ο κώδικας της statusUpdate και μας άλλαζε το text του infoLabel. Για αυτό θα δείτε έναν επιπλέον έλεγχο για το αν το returnVal είναι 0.
3) Αποφασίσαμε να εμφανίζουμε το combo box για την επιλογή στρατηγικής μόνο όταν υπάρχει διαθέσιμη λίστα.
4) Υπάρχουν μερικές περιπτώσεις που κάνουμε set την στρατηγική σε Normal, για να αποφύγουμε κάποια bugs. Όπως κάθε φορά που γίνεται stop ο Player ή όταν γίνει double click για την αναπαραγωγή του τραγουδιού.
5) Η λογική του next είναι ότι όταν η στρατηγική είναι normal, θα παίζει το επόμενο τραγούδι. Σε κάθε άλλη περίπτωση θα σταματάει τον Player για να εκτελέσει την nextSong ο κώδικας του Listener. Κατά την δημιουργία της λίστας αν ο χρήστης πάταγε το Next με μια άλλη στρατηγική τότε θα γινόταν stop ο Player ενώ βρίσκεται ήδη σε IDLE. Το αντιμετωπίσαμε με έναν επιπλέον έλεγχο.
6) Έχουμε χρησιμοποιήσει κάποια style στο infoLabel χρησιμοποιώντας κάποιες εντολές με html που ανακαλύψαμε.
7) Στην περίπτωση που ο χρήστης όριζε την στρατηγική σε Random και πατούσε Play έπαιζε πρώτα το επιλεγμένο τραγούδι και μετά ανακάτευε την λίστα. Για αυτό καλούμε την nextSong από το Play υπό συνθήκη. Έτσι η λίστα ανακατεύεται από την αρχή και παίζει κατευθείαν random song.
8) Σε φακέλους με πολλά αρχεία αργεί η Open, καθώς ψάχνει παντού για mp3 αρχεία (και σε υποκαταλόγους) εμφανίζοντας στον logger καθένα από τα αρχεία που δεν επιλέχτηκαν.

Πηγές:

Χρησιμοποιήσαμε αρκετό δικό σας κώδικα, αλλά λύσαμε και κάποια προβλήματα από το stackoverflow και γενικότερα:

1) https://stackoverflow.com/questions/3333623/how-to-get-rid-of-the-border-with-a-jtable-jscrollpane (Αφαίρεση του border).
2) https://stackoverflow.com/questions/2966334/how-do-i-set-the-colour-of-a-label-coloured-text-in-java (Χρώμα στο text του JLabel).
3) https://www.tabnine.com/code/java/methods/org.jaudiotagger.tag.Tag/getFirst (Διαχείρηση μεταδεδομένων)
4) https://www.programcreek.com/java-api-examples/?api=org.jaudiotagger.tag.Tag
5) https://www.geeksforgeeks.org/file-getparent-method-in-java-with-examples/ (Κατασκευή absolute path).
6) https://www.javatpoint.com/repaint-method-in-java (Repaint JList).