XYLOPHONE

Ευαγγελία Αθανασάκη 3076

Ανδρέας Παραβολιάσης 3031

Package MMC

Implements all the different models (free/AtonalRow/AtonalComposition/algorithmic) that can be used to the xylophone, also holds the following classes:

- Composition
- Free
- Atonal
- Algorithmic
- AtonalRow
- AtonalComposition

Abstract class Composition

Contains the basic elements and functions of any model (free/Atonal/algorithmic) of the xylophone.

protected static String *notes*;
protected static String instrument;
protected static String comp_name;
protected static String composer;
notes: *defines the String's note*
instrument: *defines the type of instrument,the default by the Jfugue is PIANO*
comp_name: *defines the title of the composition*
composer: *the name of the composer*

public void SetNotes(String notes); *redirects the String's notes.*

public void SetComp_name(String comp_name); *sets the composition's title.*

public void SetComposer(String composer); *sets the composer's name.*

public void SetInturment(String instrument); *sets the instrument type.*

public String GetNotes(); *returns the String's note.*

public String GetComp_name(); *returns the composition's title.*

public String GetComposer(); *returns the composer's name.*

public String GetInsturmnet(); *returns the instrument type.*

Class Free

This class extends the class Composition.

public Free(); *constructor, constructs a xylophone in free mode.*

public Free(String comp_name); *constructor, constructs a xylophone in free mode and sets the composition's name.*

public Free(String comp_name, String composer); *constructor, constructs a xylophone in free mode and sets the composition's name, and the composer's name.*

public void AddNotes(String notes); *adds a new note to the older one*.

Class Algorithmic

This class extends the class Composition.

private String path;
private Random ran1;
private int x1;
private static Free dd;
path: *is the path of the txt file which contains the String's notes.*
ran1: *defines a random number, used in this mode*
x1: *used in declaration of ran1.*
dd: *an object of free type.*

public Algorithmic(): *constructor, constructs a xylophone in Algorithmic mode.*

public Algorithmic(String path, String comp_name) throws Exception *constructor, constructs a xylophone in Algorithmic mode and sets the composition's name and the path(throws an exception if the given path is not found)*

public Algorithmic(String path, String comp_name, String composer) throws Exception *constructor, constructs a xylophone in Algorithmic mode and sets the composition's name, the composer's name and the path(throws an exception if the given path is not found)*

public void SetPath(String path) throws Exception; *sets the path(throws an exception if the given path is not found).*

public String GetPath(); *returns the path.*

public void generator(String a); *generates random notes depending the given string of notes*

Abstract class Atonal

This class extends the class Composition

public boolean checknotes() throws Exception *checks if the String's note is acceptable(throws exception if not acceptable).*

public void AddNotes(String notes); *adds a new note to the older one*.

Class AtonalRow

This class extends the class Atonal also implements the interface Symmetry
private int j;
private static Free *dd*
j: *used in transpose(…);*
dd: *an object of free type.*

public AtonalRow(); *constructor, constructs a xylophone in AtonalRow mode.*

public AtonalRow(String comp_name); *constructor, constructs a xylophone in AtonalRow mode and sets the composition's name.*

public AtonalRow(String comp_name,String composer); *constructor, constructs a xylophone in AtonalRow mode and sets the composition's name, and the composer's name.*

public int returnNumOfNote(String a); *returns the number of the note .*

public String returnNote(int i); *returns a string with the note depending the number.*

private String antistrofh(String pin); *this function is a helpfull function for the retrograde and reverse the string of notes*

public void doNothing(String a); *lets the existing String as it is.*

public void reflect(String a,int x) throws SymmetryActionOnNonValidAtonalRow; *thinks of the notes as a polygon and reflects them according to a given integer that points to the top of the axis, throws SymmetryActionOnNonValidAtonalRow*

public void transpose(String a,int x); thinks of the notes as a polygon and transposes it clockwise according to a given integer.

public void retrograde(String a); retrogrades the String's notes.

Class AtonalComposition

This class extends the class Atonal also implements the interface Symmetry
private int j;
private static Free *dd*
j: *used in transpose(…);*
dd: *an object of free type.*

public AtonalComposition(); *constructor, constructs a xylophone in AtonalRow mode.*

public AtonalComposition(String comp_name); *constructor, constructs a xylophone in AtonalRow mode and sets the composition's name.*

public AtonalComposition(String comp_name, String composer); *constructor, constructs a xylophone in AtonalRow mode and sets the composition's name, and the composer's name.*

public int returnNumOfNote(String a); *returns the number of the note .*

public String returnNote(int i); *returns a string with the note depending the number.*

private String antistrofh(String pin); *this function is a helpfull function for the retrograde and reverse the string of notes*

public void doNothing(String a); adds to the existing String the new one which is the same as before.

public void reflect(String a,int x) throws SymmetryActionOnNonValidAtonalRow; *thinks of the notes as a polygon and reflects them according to a given integer that points to the top of the axis and adds this to the existing String's notes throws SymmetryActionOnNonValidAtonalRow.*

public void transpose(String a,int x); thinks of the notes as a polygon and transposes it clockwise according to a given integer and adds this to the existing String's notes

public void retrograde(String a); retrogrades the String's notes and adds this to the existing one.

Package GRAFICS

Contains the graphics of any mode (Free/Algorithmic/AtonalRow/AtonalComposition), also holds the following classes:

- EFC
- algorithmicMMC
- atonalMMC
- atonalCompMMC
- atonalRowMMC
- freeMMC

Abstract class EFC

Contains the main graphics for the modes

public static JFrame *F*;
protected static String *mode*;
mode: *defines the mode.*
F: *an object JFrame type*

public void setMode(String mode); *sets the mode.*

public String getMode(); *returns the mode which chosen any time .*

public void setWind(JFrame F); *sets a window.*

public JFrame getWind(); *returns the window .*

public EFC(String mode); *contains the graphics that are public for all modes.*

Class freeMMC

This class extends the class EFC

public freeMMC(); *constructor, constructs the mode to "free".*

Class algorithmicMMC

This class extends the class EFC

public algorithmicMMC(); *constructor, constructs the mode to "algorithmic" and also is in charge for the depending garfics.*

Class atonalMMC

This class extends the class EFC

public atonalMMC(); *constructor, constructs the mode to "atonal", also contains the graphics for the butons that are public for atonalRowMMC and atonalCompMMC*

Class atonalRowMMC

This class extends the atonalMMC

public freeMMC(); *constructor, constructs the mode to "AtonalRow" and also is in charge for the depending graphics.*

Class atonalCompMMC

This class extends the atonalMMC

public freeMMC(); *constructor, constructs the mode to "AtonalComposition" and also is in charge for the depending graphics.*

Package Gen_Grafs

Contains the general graphics those are needed throw the implementation of the xylophone.
This class holds the following classes:

- main
- butt_1
- butt_2
- play_pause
- text_area
- menu_bar

Class main

This class is the main one ,which creates a window with a xylophone at the default mode(Free).

Class butt_1

This class implements the interface ActionListener

private JButton donothing;
private JButton retrograde,
private JButton transpose;
private JButton reflect;
private JButton generate;

creates buttuns by using the JButton library

private JRadioButton box1…box11, box0a… box11a;
private static Free *dd*
box1…box11: *radio buttons used in transpose choice.*
box0a… box11a: *radio buttons used in reflect choice.*
dd: *an object of free type.*

public JButton Getdonothing(); *returns the "donithing" button.*

public JButton Getretrograde(); *returns the "retrograde" button.*

public JButton Gettranspose(); *returns the "transpose" button.*

public JButton Getreflect(); *returns the "reflect" button.*

public butt_1(); *constructor, constructs the buttons.*

public void setX(int nx); *sets the number of x where is called .*

public int getX(); *gets the number of x where is needed.*

public void actionPerformed(ActionEvent E); *actions of the buttons if get pressed.*

Class butt_2

This class implements the interface ActionListener. As the previous class and this onw uses JButton to create buttons needed for the xylophone, especially creates the notes buttons (C,C#,D,D# …).

public butt_2(); *constructor, constructs the buttons.*

public JButton GetC1()
public JButton GetC2();
public JButton GetD1();
public JButton GetD2();
public JButton GetE();
public JButton GetF1();
public JButton GetF2();
public JButton GetG1();
public JButton GetG2();
public JButton GetA1();
public JButton GetA2();
public JButton GetB();

returns the respectively *buttons;*

public void actionPerformed(ActionEvent arg0); *actions of the buttons if get pressed.*

Class play_pause

This class implements the interface ActionListener

private JButton pl_pa;
private int metrhths;

private String mode;
private Player d;

pl_pa: *using JButoon library to create a button for the action play/pause.*
metrhths: *using a counter in order to show alternately play/pause icon.*
mode: *defines the mode.*
d: *an object of Player type.*

public play_pause (String mode); *this constructs the button and sets the background , the size and the location.*

public JButton getPl_Pa(); *returns the Pl_Pa button.*

public play_pause (); *constructor, constructs a button.*

public void actionPerformed(ActionEvent e); *actions of the button if get pressed.*

public void run(); *contains the function of the thread.*

Class text_area

This class is responsible for the text area which is needed to every mode.

private JTextArea stoixeia;
private JScrollPane a;
stoixeia: *using JTextArea library to create a text Area for strings .*
a: *using JScrollPane to make the text area scrollable.*

public text_area(String mode); *this constructs the size , the location in the frame and the background of the text area*

public text_area(); *constructor, constructs a text area.*

public JScrollPane GetJEditorPane(); *returns the panel.*

public JTextArea GetJTextArea(); *returns the text area.*

Class menu_bar

This class extends JFrame library and implemens ActionListener interface. Using the JMenuItem libray we create the fields we like to the menu bar.

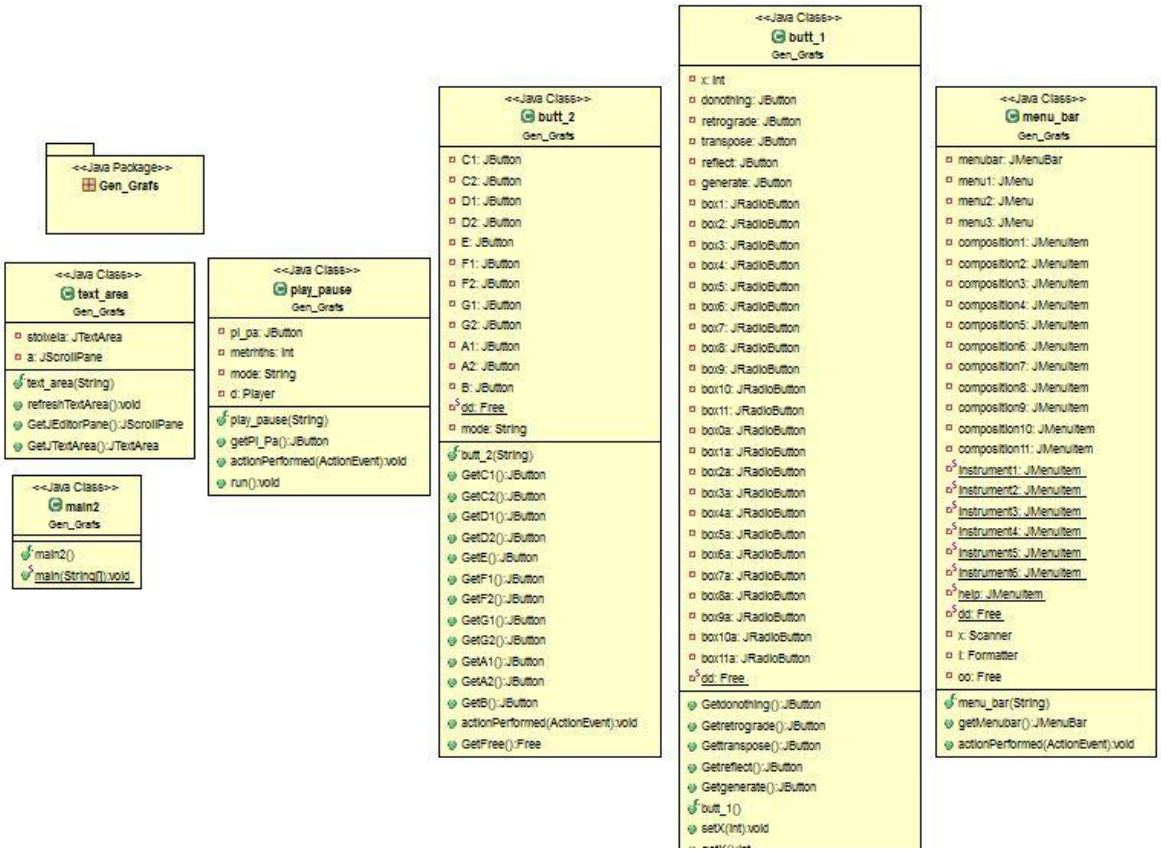
private JMenuBar menubar;
private JMenu menu1,menu2,menu3;
private JMenuItem composition1…composition11, *instrument1…instrument6*,*help*;
private static Free *dd*;
private Scanner x;

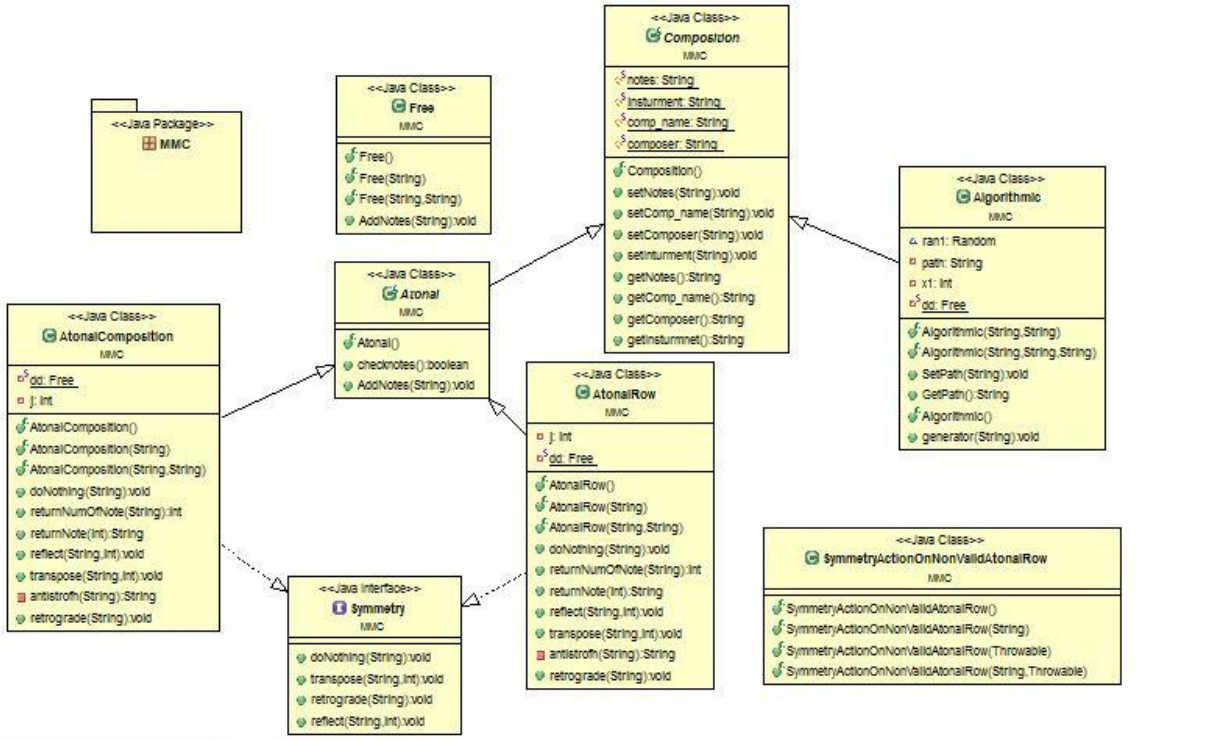
private Formatter l;
menubar: *using JMenuBar library to create a tmenu bar .*
menu1,menu2,menu3: *are the opoions of the menu bar.*
composition1…composition11, *instrument1…instrument6,help:* *using JScrollPane to make the text area scrollable.*
Free: *an object* of free type.
x: *a variable of Scanner type*.
l: *a variable of Formatter type.* .

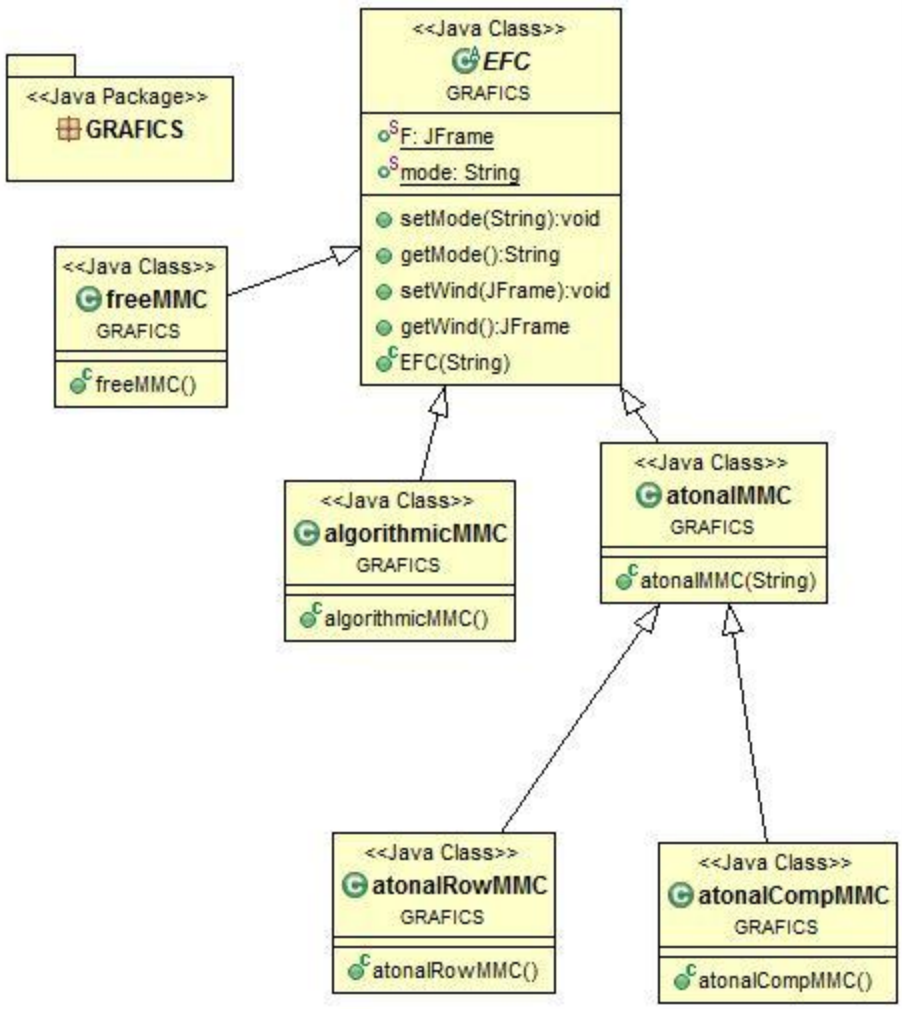
public menu_bar(); *constructor, constructs a menu bar.*

public JMenuBar getMenubar(); *returns the menu bar.*

public void actionPerformed(ActionEvent E); *actions if an option get pressed.*


<<Java Package>>
Gen_Grafs
<<Java Class>>
text_area
Gen_Grafs
stolvela: JTextArea
a: JScrollPane
text_area(String)
refreshTextArea():void
GetJEditorPane():JScrollPane
GetJTextArea():JTextArea
<<Java Class>>
main2
Gen_Grafs
main2()
main(String[]):void
<<Java Class>>
play_pause
Gen_Grafs
pl_pa: JButton
metrinths: Int
mode: String
d: Player
play_pause(String)
getPl_Pa():JButton
actionPerformed(ActionEvent):void
run():void
<<Java Class>>
butt_2
Gen_Grafs
C1: JButton
C2: JButton
D1: JButton
D2: JButton
E: JButton
F1: JButton
F2: JButton
G1: JButton
G2: JButton
A1: JButton
A2: JButton
B: JButton
dd: Free
mode: String
butt_2(String)
GetC1():JButton
GetC2():JButton
GetD1():JButton
GetD2():JButton
GetE():JButton
GetF1():JButton
GetF2():JButton
GetG1():JButton
GetG2():JButton
GetA1():JButton
GetA2():JButton
GetB():JButton
actionPerformed(ActionEvent):void
GetFree():Free
<<Java Class>>
butt_1
Gen_Grafs
x: Int
donothing: JButton
retrograde: JButton
transpose: JButton
reflect: JButton
generate: JButton
box1: JRadioButton
box2: JRadioButton
box3: JRadioButton
box4: JRadioButton
box5: JRadioButton
box6: JRadioButton
box7: JRadioButton
box8: JRadioButton
box9: JRadioButton
box10: JRadioButton
box11: JRadioButton
box0a: JRadioButton
box1a: JRadioButton
box2a: JRadioButton
box3a: JRadioButton
box4a: JRadioButton
box5a: JRadioButton
box6a: JRadioButton
box7a: JRadioButton
box8a: JRadioButton
box9a: JRadioButton
box10a: JRadioButton
box11a: JRadioButton
dd: Free
Getdonothing():JButton
Getretrograde():JButton
Gettranspose():JButton
Getreflect():JButton
Getgenerate():JButton
butt_1()
setX(Int):void
<<Java Class>>
menu_bar
Gen_Grafs
menubar: JMenuBar
menu1: JMenu
menu2: JMenu
menu3: JMenu
composition1: JMenuItem
composition2: JMenuItem
composition3: JMenuItem
composition4: JMenuItem
composition5: JMenuItem
composition6: JMenuItem
composition7: JMenuItem
composition8: JMenuItem
composition9: JMenuItem
composition10: JMenuItem
composition11: JMenuItem
Instrument1: JMenuItem
Instrument2: JMenuItem
Instrument3: JMenuItem
Instrument4: JMenuItem
Instrument5: JMenuItem
Instrument6: JMenuItem
help: JMenuItem
dd: Free
x: Scanner
t: Formatter
oo: Free
menu_bar(String)
getMenubar():JMenuBar
actionPerformed(ActionEvent):void


<<Java Package>>
MMC
<<Java Class>>
Free
MMC
Free()
Free(String)
Free(String,String)
AddNotes(String):void
<<Java Class>>
Composition
MMC
notes: String
Insturment: String
comp_name: String
composer: String
Composition()
setNotes(String):void
setComp_name(String):void
setComposer(String):void
setinturment(String):void
getNotes():String
getComp_name():String
getComposer():String
getinsturmnet():String
<<Java Class>>
Algorithmic
MMC
ran1: Random
path: String
x1: Int
dd: Free
Algorithmic(String,String)
Algorithmic(String,String,String)
SetPath(String):void
GetPath():String
Algorithmic()
generator(String):void
<<Java Class>>
Atonal
MMC
Atonal()
checknotes():boolean
AddNotes(String):void
<<Java Class>>
AtonalComposition
MMC
dd: Free
j: Int
AtonalComposition()
AtonalComposition(String)
AtonalComposition(String,String)
doNothing(String):void
returnNumOfNote(String):Int
returnNote(Int):String
reflect(String,Int):void
transpose(String,Int):void
antistrofn(String):String
retrograde(String):void
<<Java Class>>
AtonalRow
MMC
j: Int
dd: Free
AtonalRow()
AtonalRow(String)
AtonalRow(String,String)
doNothing(String):void
returnNumOfNote(String):Int
returnNote(Int):String
reflect(String,Int):void
transpose(String,Int):void
antistrofn(String):String
retrograde(String):void
<<Java Interface>>
Symmetry
MMC
doNothing(String):void
transpose(String,Int):void
retrograde(String):void
reflect(String,Int):void
<<Java Class>>
SymmetryActionOnNonValidAtonalRow
MMC
SymmetryActionOnNonValidAtonalRow()
SymmetryActionOnNonValidAtonalRow(String)
SymmetryActionOnNonValidAtonalRow(Throwable)
SymmetryActionOnNonValidAtonalRow(String,Throwable)


<<Java Package>>
GRAFICS
<<Java Class>>
EFC
GRAFICS
F: JFrame
mode: String
setMode(String):void
getMode():String
setWind(JFrame):void
getWind():JFrame
EFC(String)
<<Java Class>>
freeMMC
GRAFICS
freeMMC()
<<Java Class>>
algorithmicMMC
GRAFICS
algorithmicMMC()
<<Java Class>>
atonalMMC
GRAFICS
atonalMMC(String)
<<Java Class>>
atonalRowMMC
GRAFICS
atonalRowMMC()
<<Java Class>>
atonalCompMMC
GRAFICS
atonalCompMMC()

MUSIC PLAYER

Package MCP

Contains everything is needed for the implementation of the MusicColection Player,also holds the following classes:

- AddJList
- Buttons
- Menu
- Collections
- Track
- mainclass

Class AddJList

This class processes the actions that can be made to the add list.

```
private String name;
private String title;
private String[] dok;
private JList<String> all1;
private JScrollPane lala;
private JProgressBar a;
```

name:*declarates the name of the album of every song that is added to the list.*
title: *declarates the name of every song that is added to the list.*
dok: *this is an array that saves the all songs are added in the list.*
all1: *an object of JList type.*
panel: *creates a scrollable panel.*
a: *ceates an object of JScrollPane library.*

public AddJList(String aa) throws FileNotFoundException; *it constructs a list by a given String and throws an exception if the file isn't found.*

public JList<String> getjist(); *returns the JList.*

public JScrollPane getJscrollpane(); *returns the scrollable panel "panel".*

Class Buttons

contains the basic buttons are needed, also his class implements ActionListener interface.

```
private JButton Play ;
private JButton Pause ;
private JButton Stop ;
private JButton NextTrack ;
private JButton PreTrack ;
private JButton vol, vol2;
```

by using the JButton library creates the buttons are needed.

```
private Collection dok1, dok2;
private Player d;
public Thread dk;
private boolean a1,a2,a3,a4;
private   int w,a;
private String aa;
```

dok1,dok2:*variables of Collection type.*
d: *variable of Player type.*
dk: *variable of Thread type.*
a1,a2,a3,a4: *boolean variables.*
w,a: *integers variables.*
aa: *String type variable.*

public JButton GetPlay; *returns the button "PLAY"*

public JButton GetPause(); *returns the button "PAUSE"*

public JButton GetStop(); *returns the button "STOP"*

public JButton GetNextTrack(); *returns the button "NEXT"*

public JButton GetPreTrack(); *returns the button "BACK"*

public JButton Getvol; *returns the button "volume ++"*

public JButton Getvol2; *returns the button "vol2 - -"*

public void run();*contains the function of the thread.*

public Buttons(); *constructor, constructs the buttons.*

public synchronized void actionPerformed(ActionEvent E); *is responsible for the actions of the buttons if they get pressed.*

Class Menu

This class creates the menu bar, also implements the ActionListener interface

private JMenuBar menubar;
menubar: *creates a menu bar.*

private JMenuItem collection1;
private JMenuItem collection2;
private JMenuItem collection3;
private JMenuItem collection4;
private JMenuItem collection5;
private JMenuItem playback1;
private JMenuItem playback2;
private JMenuItem playback3;
private JMenuItem playback4;
private JMenuItem Help1;

creates the fields of the menu (New Collection/ Add File/Add Folder…

private mainclass a ;
private JFrame neo,helpn ;
private Formatter x;
private Scanner ee;
a: *an object* of mainclass type.
ee: *a variable of Scanner type*.
neo,helpn: *objects of JFame Type;*
x: *a variable of Formatter type.*

public Menu(); *constructor, constructs a menu bar.*

public JMenuBar getMenubar();*returns the menu bar.*

public void actionPerformed(ActionEvent E); *actions if an option get pressed.*

Class Collections

Does all the actions who are related with the collections.

private static String *collName*;
private static String *path*;
private static String *pr_path*;
private static String *playback*
private List<Track> trackList;
private Scanner x;
collName: *defines the name of the any collection.*
playback: *defines the type of playback.*
trackList: *list of objects type track.*
path: *defines the path of the collection.*
x: *a variable of Scanner type*.

public Collection(); *constructor, constructs a collection.*

public Collection(String collName , List<Track> trackList); *constructor, constructs a collection and sets the name and the songs of it.*

public void addTrack(Track a); *adds a song to the collecton.*

public void removeTrack(Track a); *removes a song from the collection.*

public int GetPositionInList(Track a); *gives the posision of the selected song, in the collection.*

public void trackOneUp(Track a); *moves the track one position up.*

public void trackONeDown(Track a); *moves the track one position down.*

public void SettrackList(List<Track> trackList); *sets the tracks of a collection.*

public List<Track> GetTrackList(); return the tracks.

public void SetCollName(String collName); *sets the name of the collection.*

public String GetCollName(); *returns the name of the collection.*

public void Setplayback(String playback); *sets the type of the playback.*

public String GetPlayback(); *returns the type of the playback.*

public String getPrePath(); *returns the previous play back.*

public void setPrePath(String pr_path); *set the previous play back.*

public String GetPath(); *returns the path*;

public int GetNumofTracks(); *returns the number of tracks.*

Class Track

private String title;
private String path;
private String notes;
private int duration;
title: *defines the name of the track.*
path: *defines the path of the track.*
notes: *defines a String's notes.*
duration: *defines the duration of every track;*

public Track(); *constructor, constructs a new track.*

public Track(String path) throws FileNotFoundException; *constructor, constructs a new track by the given path and throws an exception if the path isn't found.*

public Track(String title ,String path , **int** duration); *constructor, constructs a new track also sets the tile, the path and the duration of it.*

public void Settitle(String title); *sets the title of the track.*

public void SetPath(String path); *sets the path of the track,*

public void Setduration(String duration); *sets the durations of the track.*

public String Gettitle(); *returns the title of the track.*

public String GetPath(); *returns the path of the track.*

public int GetDuration(); *returns the duration of the track.*

public void SetNotes(String notes)*;* *sets a string of notes.*

public String GetNotes(); *returns the String's note.*

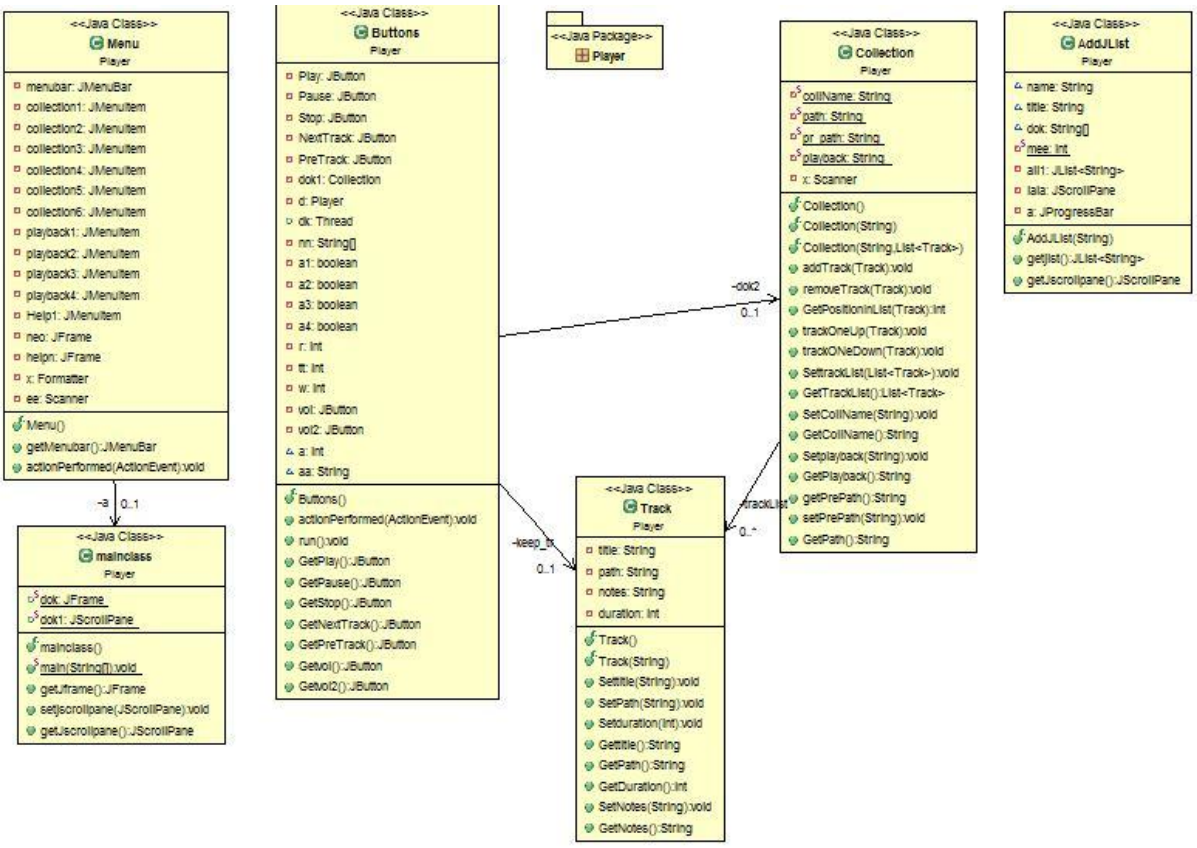
Class mainclass

This class creates a new window ,the player, at the default mode.

public static JFrame *dok*;
public static JScrollPane *dok1*;

dok: *an object of JFrame type.*
dok1: *an object of JScrollPane type.*

public JFrame getJframe(); *returns dok.*
public void setjscrollpane(JScrollPane a); *sets dok1.*
public JScrollPane getJscrollpane(); *returns dok1.*


<<Java Class>>
Menu
Player
menubar: JMenuBar
collection1: JMenuItem
collection2: JMenuItem
collection3: JMenuItem
collection4: JMenuItem
collection5: JMenuItem
collection6: JMenuItem
playback1: JMenuItem
playback2: JMenuItem
playback3: JMenuItem
playback4: JMenuItem
Help1: JMenuItem
neo: JFrame
helpn: JFrame
x: Formatter
ee: Scanner
Menu()
getMenubar():JMenuBar
actionPerformed(ActionEvent):void
-a
0..1
<<Java Class>>
mainclass
Player
dok: JFrame
dok1: JScrollPane
mainclass()
main(String[]):void
getJframe():JFrame
setjscrollpane(JScrollPane):void
getJscrollpane():JScrollPane
<<Java Class>>
Buttons
Player
Play: JButton
Pause: JButton
Stop: JButton
NextTrack: JButton
PreTrack: JButton
dok1: Collection
d: Player
dk: Thread
nn: String[]
a1: boolean
a2: boolean
a3: boolean
a4: boolean
r: int
tt: int
w: int
vol: JButton
vol2: JButton
a: int
aa: String
Buttons()
actionPerformed(ActionEvent):void
run():void
GetPlay():JButton
GetPause():JButton
GetStop():JButton
GetNextTrack():JButton
GetPreTrack():JButton
Getvol():JButton
Getvol2():JButton
<<Java Package>>
Player
-dok2
0..1
-keep_tr
0..1
<<Java Class>>
Track
Player
title: String
path: String
notes: String
duration: int
Track()
Track(String)
Settitle(String):void
SetPath(String):void
Setduration(int):void
Gettitle():String
GetPath():String
GetDuration():int
SetNotes(String):void
GetNotes():String
-trackList
0..*
<<Java Class>>
Collection
Player
collName: String
path: String
pr_path: String
playback: String
x: Scanner
Collection()
Collection(String)
Collection(String,List<Track>)
addTrack(Track):void
removeTrack(Track):void
GetPositioninList(Track):int
trackOneUp(Track):void
trackONeDown(Track):void
SettrackList(List<Track>):void
GetTrackList():List<Track>
SetCollName(String):void
GetCollName():String
Setplayback(String):void
GetPlayback():String
getPrePath():String
setPrePath(String):void
GetPath():String
<<Java Class>>
AddJList
Player
name: String
title: String
dok: String[]
mee: int
all1: JList<String>
lala: JScrollPane
a: JProgressBar
AddJList(String)
getjlist():JList<String>
getJscrollpane():JScrollPane