

MP3Java

Generated by Doxygen 1.12.0

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 com.example.App Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Member Function Documentation	8
4.1.2.1 main()	8
4.1.2.2 start()	8
4.2 com.example.AudioHandler Class Reference	9
4.2.1 Detailed Description	10
4.2.2 Constructor & Destructor Documentation	10
4.2.2.1 AudioHandler()	10
4.2.3 Member Function Documentation	10
4.2.3.1 getMediaPlayer()	10
4.2.3.2 pause()	11
4.2.3.3 play()	11
4.3 com.example.FileHandler Class Reference	11
4.3.1 Detailed Description	12
4.3.2 Member Function Documentation	12
4.3.2.1 createTrackFromPath()	12
4.3.2.2 readDir()	13
4.3.2.3 searchTracks()	14
4.3.3 Member Data Documentation	15
4.3.3.1 trackList	15
4.4 com.example.GuiActions Class Reference	16
4.4.1 Detailed Description	17
4.4.2 Member Function Documentation	17
4.4.2.1 fillAlbum()	17
4.4.2.2 fillArtist()	17
4.4.2.3 fillProgress()	17
4.4.2.4 fillStatus()	18
4.4.2.5 fillTitle()	18
4.4.2.6 playPressed()	18
4.4.2.7 searchTrack()	18
4.4.2.8 selectTrack()	18
4.4.3 Member Data Documentation	19

4.4.3.1 pressed	19
4.4.3.2 selected	20
4.5 com.example.GuiHandler Class Reference	20
4.5.1 Detailed Description	21
4.5.2 Member Function Documentation	21
4.5.2.1 init()	21
4.5.3 Member Data Documentation	21
4.5.3.1 actions	21
4.5.3.2 albumField	21
4.5.3.3 artistField	21
4.5.3.4 fileSelectButton	22
4.5.3.5 frame	22
4.5.3.6 playButton	22
4.5.3.7 progressField	22
4.5.3.8 progressTimer	22
4.5.3.9 searchField	22
4.5.3.10 titleField	22
4.5.3.11 track	23
4.6 com.example.MainTest Class Reference	23
4.6.1 Detailed Description	23
4.7 com.example.Track Class Reference	24
4.7.1 Detailed Description	25
4.7.2 Constructor & Destructor Documentation	25
4.7.2.1 Track() [1/2]	25
4.7.2.2 Track() [2/2]	25
4.7.3 Member Function Documentation	25
4.7.3.1 getAlbum()	25
4.7.3.2 getArtist()	26
4.7.3.3 getLength()	26
4.7.3.4 getPath()	26
4.7.3.5 getTitle()	26
4.7.3.6 toString()	27
4.7.4 Member Data Documentation	27
4.7.4.1 album	27
4.7.4.2 artis	27
4.7.4.3 length	27
4.7.4.4 path	27
4.7.4.5 title	27
5 File Documentation	29
5.1 App.java	29
5.2 AudioHandler.java	29

5.3 FileHandler.java	30
5.4 GuiActions.java	31
5.5 GuiHandler.java	32
5.6 Track.java	34
5.7 module-info.java	35
5.8 MainTest.java	35
Index	37

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

com.example.AudioHandler	9
com.example.FileHandler	11
com.example.GuiActions	16
com.example.GuiHandler	20
com.example.MainTest	23
com.example.Track	24
Application	
com.example.App	7

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

com.example.App	7
com.example.AudioHandler	9
com.example.FileHandler	11
com.example.GuiActions	16
com.example.GuiHandler	20
com.example.MainTest	23
com.example.Track	24

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

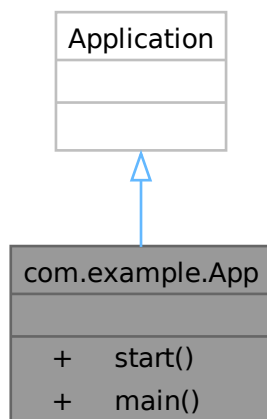
src/main/java/ module-info.java	35
src/main/java/com/example/ App.java	29
src/main/java/com/example/ AudioHandler.java	29
src/main/java/com/example/ FileHandler.java	30
src/main/java/com/example/ GuiActions.java	31
src/main/java/com/example/ GuiHandler.java	32
src/main/java/com/example/ Track.java	34
src/test/java/com/example/ MainTest.java	35

Chapter 4

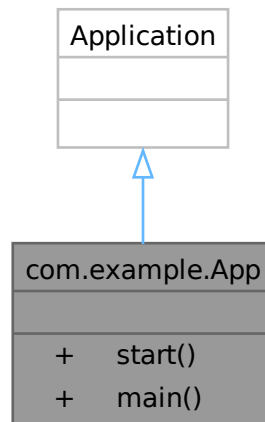
Class Documentation

4.1 com.example.App Class Reference

Inheritance diagram for com.example.App:



Collaboration diagram for com.example.App:



Public Member Functions

- void [start](#) (Stage stage) throws UnsupportedOperationException, InvalidDataException, IOException

Static Public Member Functions

- static void [main](#) (String[] args)

4.1.1 Detailed Description

This class contains the main method

Definition at line 13 of file [App.java](#).

4.1.2 Member Function Documentation

4.1.2.1 main()

```
static void com.example.App.main (  
    String[] args) [static]
```

Definition at line 39 of file [App.java](#).

```
00039                                     {  
00040     launch(args);  
00041 }
```

4.1.2.2 start()

```
void com.example.App.start (  
    Stage stage) throws UnsupportedOperationException, InvalidDataException, IOException
```

javaFx requires this [start\(\)](#) method to be overwritten This [start\(\)](#) called in main.

Parameters

stage	No idea what is this and how it is works
-------	--

Definition at line 21 of file [App.java](#).

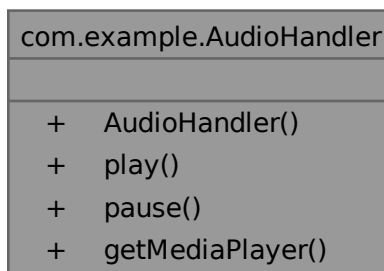
```
00021                                     {
00022         // INICIALIZÁLÁS
00023         // FileHandler fileHandler = new FileHandler();
00024
00025         // // Beolvassa a megadott dir össze `.mp3` fájlját és kollekcióba rakja őket.
00026         // fileHandler.readDir("/home/i3honor/Suli/Prog3/nagyHF/Fasz/mp3java/src/main/resources");
00027         // // A kiválasztott lejátszandó track
00028         // Track track = fileHandler.trackList.get(2);
00029
00030         // AudioHandler audioHandler = new AudioHandler(track);
00031         // Itt hozom létre a tui handlert !!
00032         GuiHandler tuiHandler = new GuiHandler();
00033
00034         // TUI handler kezeli az inputut és onnan hívja meg a megfelelő play/pause metódusokat
00035         // Creates a new thread that wll run the tuiHandler
00036         new Thread(tuiHandler::init).start();
00037     }
```

The documentation for this class was generated from the following file:

- src/main/java/com/example/App.java

4.2 com.example.AudioHandler Class Reference

Collaboration diagram for com.example.AudioHandler:



Public Member Functions

- [AudioHandler](#) ([Track](#) track)
- void [play](#) ()
- void [pause](#) ()
- [MediaPlayer](#) [getMediaPlayer](#) ()

4.2.1 Detailed Description

Class that handle every audio related method

Definition at line 11 of file [AudioHandler.java](#).

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AudioHandler()

```
com.example.AudioHandler.AudioHandler (  
    Track track)
```

Contructor that initializes the mediaPlayer with a track

Parameters

<i>track</i>	dal amit le akarok játszani
--------------	-----------------------------

Definition at line 18 of file [AudioHandler.java](#).

```
00018  
00019     String path = track.getPath();  
00020     Media media = new Media(new File(path).toURI().toString());  
00021     mediaPlayer = new MediaPlayer(media);  
00022 }
```

Here is the call graph for this function:



4.2.3 Member Function Documentation

4.2.3.1 getMediaPlayer()

```
MediaPlayer com.example.AudioHandler.getMediaPlayer ()
```

Definition at line 41 of file [AudioHandler.java](#).

```
00041  
00042     return mediaPlayer;  
00043 }
```


4.2.3.2 pause()

```
void com.example.AudioHandler.pause ()
```

Calls [pause\(\)](#) method on the mediaPlayer

Definition at line 36 of file [AudioHandler.java](#).

```
00036      {  
00037          mediaPlayer.pause();  
00038          System.out.println("Paused.");  
00039      }
```

4.2.3.3 play()

```
void com.example.AudioHandler.play ()
```

Calls the [play\(\)](#) method on the initialized media player Works as a resume() too

Definition at line 28 of file [AudioHandler.java](#).

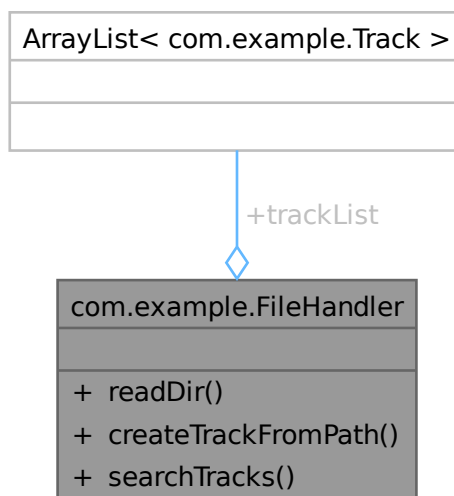
```
00028      {  
00029          mediaPlayer.play();  
00030          System.out.println("Playing...");  
00031      }
```

The documentation for this class was generated from the following file:

- src/main/java/com/example/AudioHandler.java

4.3 com.example.FileHandler Class Reference

Collaboration diagram for com.example.FileHandler:



Public Member Functions

- void [readDir](#) (String path) throws IOException, UnsupportedTagException, InvalidDataException
- [Track](#) [createTrackFromPath](#) (String path) throws IOException, UnsupportedTagException, InvalidDataException
- ArrayList< [Track](#) > [searchTracks](#) (String pattern, String path) throws IOException, UnsupportedTagException, InvalidDataException

Public Attributes

- ArrayList< [Track](#) > [trackList](#) = new ArrayList<>()

4.3.1 Detailed Description

[FileHandler](#) - Minden aminek fileokhoz van köze itt van kezelve

Definition at line 19 of file [FileHandler.java](#).

4.3.2 Member Function Documentation

4.3.2.1 createTrackFromPath()

[Track](#) [com.example.FileHandler.createTrackFromPath](#) (String path) throws IOException, UnsupportedTagException, InvalidDataException

Megadott útvonalból létrehozza a [Track](#) objektumot amit utána el tudok tárolni egy kollekcióban. Csak azért kell, hogy a [readDir\(\)](#)-ben konstruálni tudjak metadatából Tracket amit pedig kollekcióba tudok elhelyezni

Parameters

<i>path</i>	- a dal elérési útvonala
-------------	--------------------------

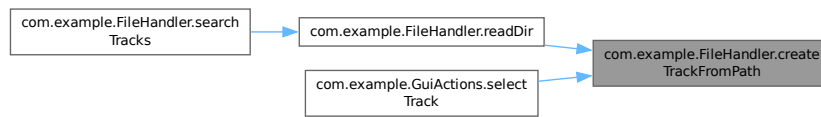
Definition at line 57 of file [FileHandler.java](#).

```

00058     {
00059         // Inicializálás
00060         Mp3File mp3File = new Mp3File(path);
00061         long length = 0;
00062         String artist = null;
00063         String title = null;
00064         String album = null;
00065
00066         // Megnézem, hogy milyen típusú tagjei vannak
00067         if (mp3File.hasId3v1Tag()) {
00068             ID3v1 v1Tag = mp3File.getId3v1Tag();
00069             length = mp3File.getLengthInSeconds();
00070             artist = v1Tag.getArtist();
00071             title = v1Tag.getTitle();
00072             album = v1Tag.getAlbum();
00073         }
00074         else if (mp3File.hasId3v2Tag()) {
00075             ID3v2 v2Tag = mp3File.getId3v2Tag();
00076             length = mp3File.getLengthInSeconds();
00077             artist = v2Tag.getArtist();
00078             title = v2Tag.getTitle();
00079             album = v2Tag.getAlbum();
00080         }
00081         else System.err.println("Se v1 se v2 tagjei nincsenek a fájlnek.");
00082
00083         // Trakc konstruálás, lényegi rész
00084         return new Track(path, title, artist, album, length);
00085     } // end of createTrackFromPath()

```

Here is the caller graph for this function:



4.3.2.2 readDir()

```
void com.example.FileHandler.readDir (
    String path) throws IOException, UnsupportedTagException, InvalidDataException
```

Parameters

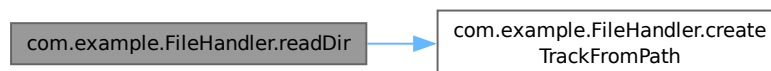
<i>path</i>	dal elérési útvonala A megadott dir végigpásztázása mp3 fájlok után kutatva A megtalált mp3 fájlokból egyesével létrehoz egy T //rack objektumot
-------------	--

Definition at line 29 of file [FileHandler.java](#).

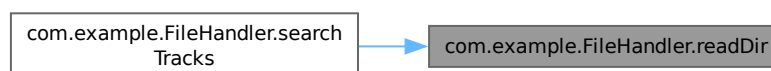
```

00030         {
00031             File dir = new File(path);
00032             File[] listOfFiles = dir.listFiles();
00033             String name;
00034
00035             if(listOfFiles != null){
00036                 for(int i = 0; i < listOfFiles.length; i++){
00037                     name = listOfFiles[i].getName();
00038
00039                     if(listOfFiles[i].isFile() && name.endsWith(".mp3")){
00040                         Track track = createTrackFromPath(listOfFiles[i].getAbsolutePath());
00041                         trackList.add(track);
00042                     }
00043                 } // end of for loop
00044             }
00045             // Error handling
00046             else System.out.println("Hiba a mappa pásztázásakor");
00047         } // end of readDir()
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.2.3 searchTracks()

```
ArrayList< Track > com.example.FileHandler.searchTracks (
    String pattern,
    String path) throws IOException, UnsupportedOperationException, InvalidDataException
```

Parameters

<i>pattern</i>	keresett cím stringben megadva
----------------	--------------------------------

Returns

megtalált dal/dalok kollekciója TODO toLowerCase() Nem teljesen értem mi a retek ez a kód, de lambda Search method

Parameters

<i>pattern</i>	amit keresek, guiban megadtam
<i>path</i>	az absolute path-ja a vizsgált dir-nek.

Returns

matching trackek

Exceptions

<i>IOException</i>	
<i>UnsupportedTagException</i>	
<i>InvalidDataException</i>	

Definition at line 111 of file [FileHandler.java](#).

```
00111
00112 {
00113     trackList.clear();
00114     readDir(path);
00115     String lowerCasepattern = pattern.toLowerCase();
00116     ArrayList<Track> matchingTracks = new ArrayList<>();
00117     for (Track t : trackList) {
00118         if (t.getTitle().toLowerCase().contains(lowerCasepattern) ||
00119             t.getArtist().toLowerCase().contains(lowerCasepattern)) {
00120             matchingTracks.add(t);
00121         }
00122     }
00123     return matchingTracks;
00124 }
00125
00126 }
00127 }
```

Here is the call graph for this function:



4.3.3 Member Data Documentation

4.3.3.1 trackList

```
ArrayList<Track> com.example.FileHandler.trackList = new ArrayList<>()
```

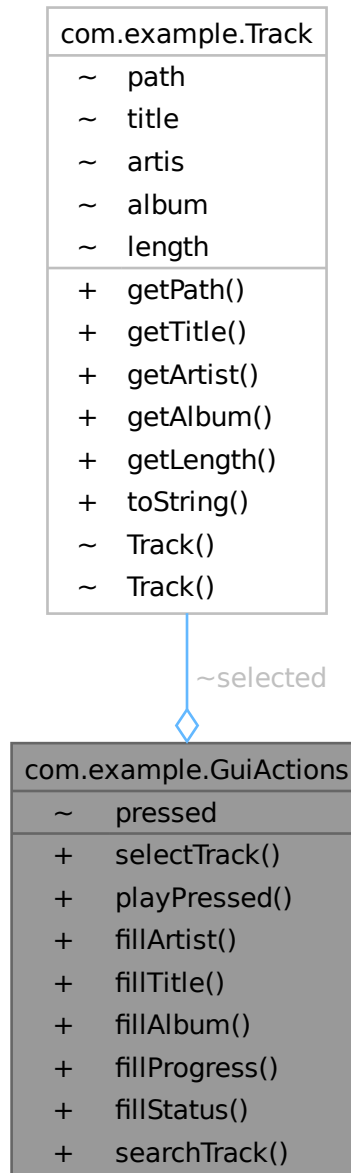
Definition at line 22 of file [FileHandler.java](#).

The documentation for this class was generated from the following file:

- src/main/java/com/example/FileHandler.java

4.4 com.example.GuiActions Class Reference

Collaboration diagram for com.example.GuiActions:



Public Member Functions

- [Track selectTrack](#) (JFrame frame) throws `UnsupportedTagException`, `InvalidDataException`, `IOException`
- void [playPressed](#) ()
- void [fillArtist](#) (JTextField field)
- void [fillTitle](#) (JTextField field)
- void [fillAlbum](#) (JTextField field)

- void [fillProgress](#) (JTextField field)
- String [fillStatus](#) ()
- ArrayList< [Track](#) > [searchTrack](#) (String pattern) throws UnsupportedOperationException, InvalidDataException, IOException

Package Attributes

- boolean [pressed](#) = false
- [Track](#) [selected](#)

4.4.1 Detailed Description

Minden guiHandlerben helyet foglaló gombnak és fieldnek itt vannak definiálva a listenerjei és azok metódusai

Definition at line 19 of file [GuiActions.java](#).

4.4.2 Member Function Documentation

4.4.2.1 fillAlbum()

```
void com.example.GuiActions.fillAlbum (  
    JTextField field)
```

Definition at line 86 of file [GuiActions.java](#).

```
00086                                     {  
00087     if(selected != null){  
00088         String album = selected.getAlbum();  
00089         field.setText(album);  
00090         System.out.println("Album filed filled");  
00091     }  
00092     else field.setText("NA");  
00093 }
```

4.4.2.2 fillArtist()

```
void com.example.GuiActions.fillArtist (  
    JTextField field)
```

Definition at line 68 of file [GuiActions.java](#).

```
00068                                     {  
00069     if(selected != null){  
00070         String artist = selected.getArtist();  
00071         field.setText(artist);  
00072         System.out.println("Artist filed filled");  
00073     }  
00074     else field.setText("NA");  
00075 }
```

4.4.2.3 fillProgress()

```
void com.example.GuiActions.fillProgress (  
    JTextField field)
```

Definition at line 95 of file [GuiActions.java](#).

```
00095                                     {  
00096     if(selected != null){  
00097         long len = selected.getLength();  
00098         int curr = (int) audioHandler.getMediaPlayer().getCurrentTime().toSeconds();  
00099         field.setText(curr + "/" + len);  
00100         // System.out.println("Length of the track: " + len);  
00101     }  
00102     else field.setText("NA");  
00103 }
```

4.4.2.4 fillStatus()

String com.example.GuiActions.fillStatus ()

Definition at line 105 of file [GuiActions.java](#).

```
00105         {
00106             if(selected != null){
00107                 String status = audioHandler.getMediaPlayer().getStatus().toString();
00108                 System.out.println("Status filled");
00109                 return status;
00110             }
00111             return "Not selected";
00112         }
```

4.4.2.5 fillTitle()

void com.example.GuiActions.fillTitle (
 JTextField field)

Definition at line 77 of file [GuiActions.java](#).

```
00077         {
00078             if(selected != null){
00079                 String artist = selected.getTitle();
00080                 field.setText(artist);
00081                 System.out.println("Titke filed filled");
00082             }
00083             else field.setText("NA");
00084         }
```

4.4.2.6 playPressed()

void com.example.GuiActions.playPressed ()

Definition at line 57 of file [GuiActions.java](#).

```
00057         {
00058             if (!pressed) {
00059                 audioHandler.play();
00060                 pressed = true;
00061             }
00062             else {
00063                 audioHandler.pause();
00064                 pressed = false;
00065             }
00066         }
```

4.4.2.7 searchTrack()

ArrayList< [Track](#) > com.example.GuiActions.searchTrack (
 String pattern) throws UnsupportedOperationException, InvalidDataException, IOException

Definition at line 114 of file [GuiActions.java](#).

```
00114         {
00115             return fileHandler.searchTracks(pattern,
00116                 "/home/i3honor/Suli/Prog3/nagyHF/Fasz/mp3java/src/main/resources");
00116         }
```

4.4.2.8 selectTrack()

[Track](#) com.example.GuiActions.selectTrack (
 JFrame frame) throws UnsupportedOperationException, InvalidDataException, IOException

A dal/file kiválasztásáért felel

Parameters

<i>frame</i>	
--------------	--

Returns

selected track

Exceptions

<i>UnsupportedTagException</i>	
<i>InvalidDataException</i>	
<i>IOException</i>	

curr directoryt nyitja meg

csak mp3 fájlokat látsz

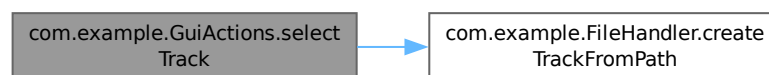
Definition at line 33 of file [GuiActions.java](#).

```

00033
00034 {
00035     JFileChooser fileChooser = new JFileChooser();
00036     String currentDirectory = System.getProperty("user.dir");
00037     fileChooser.setCurrentDirectory(new File(currentDirectory));
00038
00040     fileChooser.setFileFilter(new javax.swing.filechooser.FileNameExtensionFilter("MP3 Files",
00041 "mp3"));
00042     int result = fileChooser.showOpenDialog(frame);
00043
00044     if (result == JFileChooser.APPROVE_OPTION) {
00045         File selectedFile = fileChooser.getSelectedFile();
00046
00047         selected = fileHandler.createTrackFromPath(selectedFile.getAbsolutePath());
00048
00049         audioHandler = new AudioHandler(selected);
00050
00051         return selected;
00052     }
00053
00054     return null;
00055 }

```

Here is the call graph for this function:



4.4.3 Member Data Documentation

4.4.3.1 pressed

boolean com.example.GuiActions.pressed = false [package]

Definition at line 22 of file [GuiActions.java](#).

4.4.3.2 selected

[Track](#) com.example.GuiActions.selected [package]

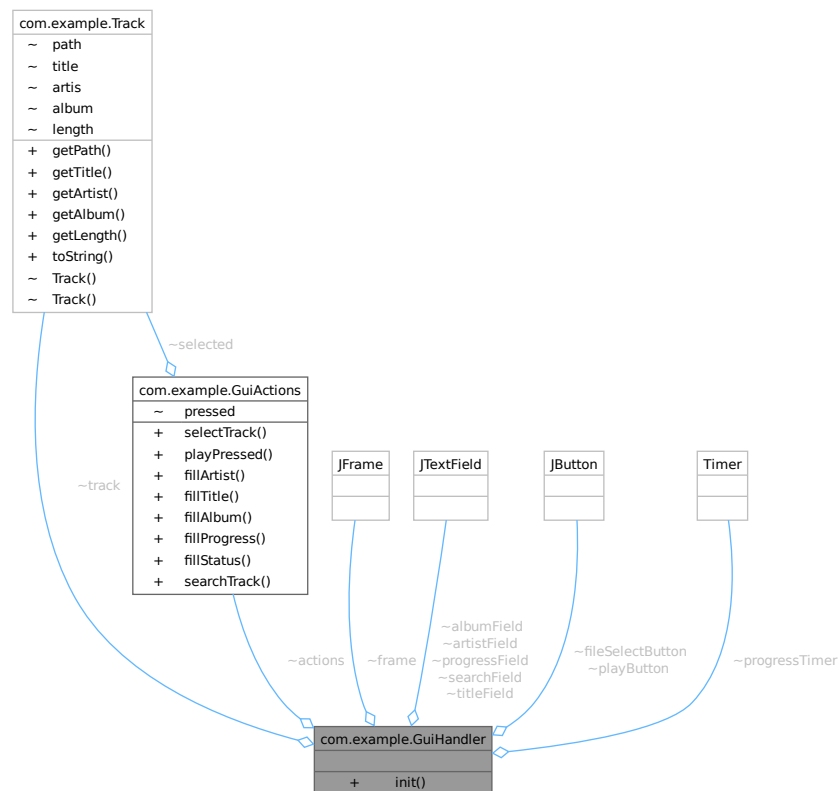
Definition at line 23 of file [GuiActions.java](#).

The documentation for this class was generated from the following file:

- src/main/java/com/example/GuiActions.java

4.5 com.example.GuiHandler Class Reference

Collaboration diagram for com.example.GuiHandler:



Public Member Functions

- void [init](#) ()

Package Attributes

- [GuiActions](#) actions = new [GuiActions](#)()
- [Track](#) track
- [JFrame](#) frame
- [JTextField](#) searchField
- [JTextField](#) artistField
- [JTextField](#) titleField
- [JTextField](#) albumField
- [JTextField](#) progressField
- [JButton](#) playButton
- [JButton](#) fileSelectButton
- [Timer](#) progressTimer

4.5.1 Detailed Description

Definition at line 17 of file [GuiHandler.java](#).

4.5.2 Member Function Documentation

4.5.2.1 init()

```
void com.example.GuiHandler.init ()
```

Definition at line 25 of file [GuiHandler.java](#).

```
00025         {
00026             initFrame();
00027             initComponents();
00028             configureProgressTimer();
00029     }
```

4.5.3 Member Data Documentation

4.5.3.1 actions

```
GuiActions com.example.GuiHandler.actions = new GuiActions() [package]
```

Definition at line 18 of file [GuiHandler.java](#).

4.5.3.2 albumField

```
JTextField com.example.GuiHandler.albumField [package]
```

Definition at line 21 of file [GuiHandler.java](#).

4.5.3.3 artistField

```
JTextField com.example.GuiHandler.artistField [package]
```

Definition at line 21 of file [GuiHandler.java](#).

4.5.3.4 fileSelectButton

`JButton com.example.GuiHandler.fileSelectButton [package]`

Definition at line 22 of file [GuiHandler.java](#).

4.5.3.5 frame

`JFrame com.example.GuiHandler.frame [package]`

Definition at line 20 of file [GuiHandler.java](#).

4.5.3.6 playButton

`JButton com.example.GuiHandler.playButton [package]`

Definition at line 22 of file [GuiHandler.java](#).

4.5.3.7 progressField

`JTextField com.example.GuiHandler.progressField [package]`

Definition at line 21 of file [GuiHandler.java](#).

4.5.3.8 progressTimer

`Timer com.example.GuiHandler.progressTimer [package]`

Definition at line 23 of file [GuiHandler.java](#).

4.5.3.9 searchField

`JTextField com.example.GuiHandler.searchField [package]`

Definition at line 21 of file [GuiHandler.java](#).

4.5.3.10 titleField

`JTextField com.example.GuiHandler.titleField [package]`

Definition at line 21 of file [GuiHandler.java](#).

4.5.3.11 track

[Track](#) com.example.GuiHandler.track [package]

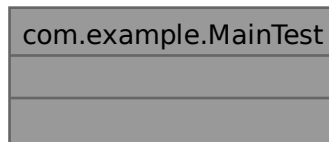
Definition at line 19 of file [GuiHandler.java](#).

The documentation for this class was generated from the following file:

- src/main/java/com/example/GuiHandler.java

4.6 com.example.MainTest Class Reference

Collaboration diagram for com.example.MainTest:



4.6.1 Detailed Description

Definition at line 7 of file [MainTest.java](#).

The documentation for this class was generated from the following file:

- src/test/java/com/example/MainTest.java

4.7 com.example.Track Class Reference

Collaboration diagram for com.example.Track:

com.example.Track
~ path
~ title
~ artis
~ album
~ length
+ getPath()
+ getTitle()
+ getArtist()
+ getAlbum()
+ getLength()
+ toString()
~ Track()
~ Track()

Public Member Functions

- String [getPath](#) ()
- String [getTitle](#) ()
- String [getArtist](#) ()
- String [getAlbum](#) ()
- long [getLength](#) ()
- String [toString](#) ()

Package Functions

- [Track](#) (String path, String [title](#), String artist, String album, long length)
Sec-ben megadott dal hossz.
- [Track](#) ()
Paraméter nélküli konstruktor.

Package Attributes

- String [path](#)
- String [title](#)
Absolute path that is set by the [FileHandler.readDir\(\)](#)
- String [artis](#)
- String [album](#)
- long [length](#)

4.7.1 Detailed Description

Definition at line 3 of file [Track.java](#).

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Track() [1/2]

```
com.example.Track.Track (  
    String path,  
    String title,  
    String artist,  
    String album,  
    long length) [package]
```

Sec-ben megadott dal hossz.

Konstruktor

Definition at line 11 of file [Track.java](#).

```
00011                                     {  
00012         this.path = path;  
00013         this.title = title;  
00014         this.artis = artist;  
00015         this.album = album;  
00016         this.length = length;  
00017     }
```

4.7.2.2 Track() [2/2]

```
com.example.Track.Track () [package]
```

Paraméter nélküli konstruktor.

Definition at line 20 of file [Track.java](#).

```
00020     {  
00021         this.path = null;  
00022         this.title = null;  
00023         this.artis = null;  
00024         this.album = null;  
00025         this.length = 0;  
00026     }
```

4.7.3 Member Function Documentation

4.7.3.1 getAlbum()

```
String com.example.Track.getAlbum ()
```

Definition at line 46 of file [Track.java](#).

```
00046                                     {  
00047         return album;  
00048     }
```

4.7.3.2 getArtist()

```
String com.example.Track.getArtist ()
```

Definition at line 42 of file [Track.java](#).

```
00042     {
00043         return artis;
00044     }
```

4.7.3.3 getLength()

```
long com.example.Track.getLength ()
```

Definition at line 50 of file [Track.java](#).

```
00050     {
00051         return length;
00052     }
```

4.7.3.4 getPath()

```
String com.example.Track.getPath ()
```

The fileHandler class Readdi method sets this attribute

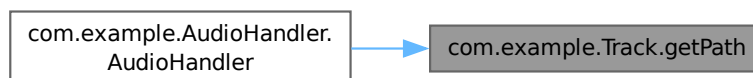
Returns

path absolute path to the song

Definition at line 34 of file [Track.java](#).

```
00034     {
00035         return path;
00036     }
```

Here is the caller graph for this function:



4.7.3.5 getTitle()

```
String com.example.Track.getTitle ()
```

Definition at line 38 of file [Track.java](#).

```
00038     {
00039         return title;
00040     }
```


4.7.3.6 toString()

String com.example.Track.toString ()

Definition at line 55 of file [Track.java](#).

```
00055     {  
00056         return "title=" + title + ", artist='" + artis + " ";  
00057     }
```

4.7.4 Member Data Documentation

4.7.4.1 album

String com.example.Track.album [package]

Definition at line 7 of file [Track.java](#).

4.7.4.2 artis

String com.example.Track.artis [package]

Definition at line 6 of file [Track.java](#).

4.7.4.3 length

long com.example.Track.length [package]

Definition at line 8 of file [Track.java](#).

4.7.4.4 path

String com.example.Track.path [package]

Definition at line 4 of file [Track.java](#).

4.7.4.5 title

String com.example.Track.title [package]

Absolute path that is set by the [FileHandler.readDir\(\)](#)

Definition at line 5 of file [Track.java](#).

The documentation for this class was generated from the following file:

- src/main/java/com/example/Track.java

Chapter 5

File Documentation

5.1 App.java

```
00001 package com.example;
00002
00003 import java.io.IOException;
00004
00005 import com.mpatric.mp3agic.InvalidDataException;
00006 import com.mpatric.mp3agic.UnsupportedTagException;
00007
00008 import javafx.application.Application;
00009 import javafx.stage.Stage;
00013 public class App extends Application {
00014
00020     @Override
00021     public void start(Stage stage) throws UnsupportedTagException, InvalidDataException, IOException {
00022         // INICIALIZÁLÁS
00023         // FileHandler fileHandler = new FileHandler();
00024
00025         // // Beolvassa a megadott dir össze `.mp3` fájlját és kollekcióba rakja őket.
00026         // fileHandler.readDir("/home/i3honor/Suli/Prog3/nagyHF/Fasz/mp3java/src/main/resources");
00027         // // A kiválasztott lejátszandó track
00028         // Track track = fileHandler.trackList.get(2);
00029
00030         // AudioHandler audioHandler = new AudioHandler(track);
00031         // Itt hozom létre a tui handlert !!
00032         GuiHandler tuiHandler = new GuiHandler();
00033
00034         // TUI handler kezeli az inputut és onnan hívja meg a megfelelő play/pause metódusokat
00035         // Creates a new thread that will run the tuiHandler
00036         new Thread(tuiHandler::init).start();
00037     }
00038
00039     public static void main(String[] args) {
00040         launch(args);
00041     }
00042 }
```

5.2 AudioHandler.java

```
00001 package com.example;
00002
00003 import java.io.File;
00004
00005 import javafx.scene.media.Media;
00006 import javafx.scene.media.MediaPlayer;
00007
00011 public class AudioHandler {
00012     private MediaPlayer mediaPlayer;
00013
00018     public AudioHandler(Track track) {
00019         String path = track.getPath();
00020         Media media = new Media(new File(path).toURI().toString());
00021         mediaPlayer = new MediaPlayer(media);
00022     }
00023
00028     public void play() {
```

```

00029         mediaPlayer.play();
00030         System.out.println("Playing...");
00031     }
00032
00036     public void pause() {
00037         mediaPlayer.pause();
00038         System.out.println("Paused.");
00039     }
00040
00041     public MediaPlayer getMediaPlayer(){
00042         return mediaPlayer;
00043     }
00044
00045 } // END OF AUDIOHANDLER

```

5.3 FileHandler.java

```

00001 package com.example;
00002
00003 import java.io.File;
00004 import java.io.IOException;
00005 import java.util.ArrayList;
00006 import java.util.stream.Collectors;
00007
00008 import javax.swing.plaf.basic.BasicSliderUI.TrackListener;
00009
00010 import com.mpatric.mp3agic.ID3v1;
00011 import com.mpatric.mp3agic.ID3v2;
00012 import com.mpatric.mp3agic.InvalidDataException;
00013 import com.mpatric.mp3agic.Mp3File;
00014 import com.mpatric.mp3agic.UnsupportedTagException;
00015
00019 public class FileHandler {
00020
00021     // Ebben a kollekcióban tárolom a dalokat
00022     public ArrayList<Track> trackList = new ArrayList<>();
00023
00029     public void readDir(String path) throws IOException, UnsupportedTagException,
00030     InvalidDataException{
00031         File dir = new File(path);
00032         File[] listOfFiles = dir.listFiles();
00033         String name;
00034
00035         if(listOfFiles != null){
00036             for(int i = 0; i < listOfFiles.length; i++){
00037                 name = listOfFiles[i].getName();
00038
00039                 if(listOfFiles[i].isFile() && name.endsWith(".mp3")){
00040                     Track track = createTrackFromPath(listOfFiles[i].getAbsolutePath());
00041                     trackList.add(track);
00042                 }
00043             } // end of for loop
00044         }
00045         // Error handling
00046         else System.out.println("Hiba a mappa pásztázásakor");
00047     } // end of readDir()
00048
00049
00057     public Track createTrackFromPath(String path) throws IOException, UnsupportedTagException,
00058     InvalidDataException {
00059         // Inicializálás
00060         Mp3File mp3File = new Mp3File(path);
00061         long length = 0;
00062         String artist = null;
00063         String title = null;
00064         String album = null;
00065
00066         // Megnézem, hogy milyen típusú tagjei vannak
00067         if(mp3File.hasId3v1Tag()){
00068             ID3v1 v1Tag = mp3File.getId3v1Tag();
00069             length = mp3File.getLengthInSeconds();
00070             artist = v1Tag.getArtist();
00071             title = v1Tag.getTitle();
00072             album = v1Tag.getAlbum();
00073         }
00074         else if(mp3File.hasId3v2Tag()){
00075             ID3v2 v2Tag = mp3File.getId3v2Tag();
00076             length = mp3File.getLengthInSeconds();
00077             artist = v2Tag.getArtist();
00078             title = v2Tag.getTitle();
00079             album = v2Tag.getAlbum();
00080         }
00081         else System.err.println("Se v1 se v2 tagjei nincsenek a fájlban.");

```

```

00082
00083         // Trakc konstruálás, lényegi rész
00084         return new Track(path, title, artist, album, length);
00085     } // end of createTrackFromPath()
00086
00087     // /**
00088     //  * @param pattern keresett cím stringben megadva
00089     //  * @return megtalált dal/dalok kollekciója
00090     //  * TODO toLowerCase()
00091     //  * Nem teljesen értem mi a retek ez a kód, de lambda
00092     //  */
00093     // public ArrayList<Track> search(String pattern) {
00094     //     return trackList.stream()
00095     //         .filter(track ->
00096     //             track.getTitle().contains(pattern)
00097     //             || track.getArtist().contains(pattern))
00098     //         .collect(Collectors.toCollection(ArrayList::new));
00099     // } // end of search()
00100
00101
00111     public ArrayList<Track> searchTracks(String pattern, String path) throws IOException,
    UnsupportedTagException, InvalidDataException {
00112         trackList.clear();
00113         readDir(path);
00114
00115         String lowerCasepattern = pattern.toLowerCase();
00116
00117         ArrayList<Track> matchingTracks = new ArrayList<>();
00118
00119         for (Track t : trackList) {
00120             if (t.getTitle().toLowerCase().contains(lowerCasepattern) ||
00121                 t.getArtist().toLowerCase().contains(lowerCasepattern)) {
00122                 matchingTracks.add(t);
00123             }
00124         }
00125
00126         return matchingTracks;
00127     }
00128
00129
00130
00131 } // end of fileHandler class

```

5.4 GuiActions.java

```

00001 package com.example;
00002
00003 import java.io.File;
00004 import java.io.IOException;
00005 import java.util.ArrayList;
00006
00007 import javax.swing.JFileChooser;
00008 import javax.swing.JFrame;
00009 import javax.swing.JTextField;
00010
00011 import com.mpatric.mp3agic.InvalidDataException;
00012 import com.mpatric.mp3agic.UnsupportedTagException;
00013
00019 public class GuiActions {
00020     private FileHandler fileHandler = new FileHandler();
00021     private AudioHandler audioHandler;
00022     boolean pressed = false;
00023     Track selected;
00024
00033     public Track selectTrack(JFrame frame) throws UnsupportedTagException, InvalidDataException,
    IOException {
00034         JFileChooser fileChooser = new JFileChooser();
00035
00036         String currentDirectory = System.getProperty("user.dir");
00037         fileChooser.setCurrentDirectory(new File(currentDirectory));
00038
00040         fileChooser.setFileFilter(new javax.swing.filechooser.FileNameExtensionFilter("MP3 Files",
    "mp3"));
00041
00042         int result = fileChooser.showOpenDialog(frame);
00043
00044         if (result == JFileChooser.APPROVE_OPTION) {
00045             File selectedFile = fileChooser.getSelectedFile();
00046
00047             selected = fileHandler.createTrackFromPath(selectedFile.getAbsolutePath());
00048
00049             audioHandler = new AudioHandler(selected);
00050

```

```

00051         return selected;
00052     }
00053
00054     return null;
00055 }
00056
00057 public void playPressed(){
00058     if (!pressed) {
00059         audioHandler.play();
00060         pressed = true;
00061     }
00062     else {
00063         audioHandler.pause();
00064         pressed = false;
00065     }
00066 }
00067
00068 public void fillArtist(JTextField field){
00069     if(selected != null){
00070         String artist = selected.getArtist();
00071         field.setText(artist);
00072         System.out.println("Artist filed filled");
00073     }
00074     else field.setText("NA");
00075 }
00076
00077 public void fillTitle(JTextField field){
00078     if(selected != null){
00079         String artist = selected.getTitle();
00080         field.setText(artist);
00081         System.out.println("Titke filed filled");
00082     }
00083     else field.setText("NA");
00084 }
00085
00086 public void fillAlbum(JTextField field){
00087     if(selected != null){
00088         String album = selected.getAlbum();
00089         field.setText(album);
00090         System.out.println("Album filed filled");
00091     }
00092     else field.setText("NA");
00093 }
00094
00095 public void fillProgress(JTextField field){
00096     if(selected != null){
00097         long len = selected.getLength();
00098         int curr = (int) audioHandler.getMediaPlayer().getCurrentTime().toSeconds();
00099         field.setText(curr + "/" + len);
00100         // System.out.println("Length of the track: " + len);
00101     }
00102     else field.setText("NA");
00103 }
00104
00105 public String fillStatus(){
00106     if(selected != null){
00107         String status = audioHandler.getMediaPlayer().getStatus().toString();
00108         System.out.println("Status filled");
00109         return status;
00110     }
00111     return "Not selected";
00112 }
00113
00114 public ArrayList<Track> searchTrack(String pattern) throws UnsupportedOperationException,
InvalidDataException, IOException {
00115     return fileHandler.searchTracks(pattern,
"/home/i3honor/Suli/Prog3/nagyHF/Fasz/mp3java/src/main/resources");
00116 }
00117
00118
00119 }

```

5.5 GuiHandler.java

```

00001 package com.example;
00002
00003 import java.awt.GridBagConstraints;
00004 import java.awt.GridBagLayout;
00005 import java.io.IOException;
00006 import java.util.ArrayList;
00007
00008 import javax.swing.JButton;
00009 import javax.swing.JFrame;

```

```

00010 import javax.swing.JLabel;
00011 import javax.swing.JTextField;
00012 import javax.swing.Timer;
00013
00014 import com.mpatric.mp3agic.InvalidDataException;
00015 import com.mpatric.mp3agic.UnsupportedTagException;
00016
00017 public class GuiHandler {
00018     GuiActions actions = new GuiActions();
00019     Track track;
00020     JFrame frame;
00021     JTextField searchField, artistField, titleField, albumField, progressField;
00022     JButton playButton, fileSelectButton;
00023     Timer progressTimer;
00024
00025     public void init() {
00026         initFrame();
00027         initComponents();
00028         configureProgressTimer();
00029     }
00030
00031     private void initFrame() {
00032         frame = new JFrame("Mp3Java");
00033         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
00034         frame.setSize(400, 250);
00035         frame.setLayout(new GridBagLayout());
00036         setupLayout();
00037         frame.pack();
00038         frame.setLocationRelativeTo(null);
00039         frame.setVisible(true);
00040     }
00041
00042     /*
00043     * Initeli a layoutot
00044     */
00045     private void setupLayout() {
00046         GridBagConstraints gbc = new GridBagConstraints();
00047         gbc.fill = GridBagConstraints.HORIZONTAL;
00048         gbc.weightx = 1.0;
00049
00050         // Top row: Search field
00051         addSearchLabelAndField(gbc, 0, "Search:", searchField = new JTextField(20));
00052
00053         // Artist row
00054         addLabelAndField(gbc, 1, "Artist:", artistField = new JTextField(20));
00055
00056         // Title row
00057         addLabelAndField(gbc, 2, "Title:", titleField = new JTextField(20));
00058
00059         // Album row
00060         addLabelAndField(gbc, 3, "Album:", albumField = new JTextField(20));
00061
00062         // Middle row: Progress field
00063         gbc.gridx = 0;
00064         gbc.gridy = 4;
00065         gbc.gridwidth = 3;
00066         progressField = createReadOnlyField();
00067         frame.add(progressField, gbc);
00068
00069         // Bottom row: File Select and Play/Stop buttons
00070         gbc.gridwidth = 1;
00071         gbc.gridy = 5;
00072         fileSelectButton = new JButton("File Select");
00073         gbc.gridx = 0;
00074         frame.add(fileSelectButton, gbc);
00075
00076         playButton = new JButton("Play");
00077         gbc.gridx = 1;
00078         frame.add(playButton, gbc);
00079
00080         addButtonListeners();
00081     }
00082
00083     private void addSearchLabelAndField(GridBagConstraints gbc, int row, String label, JTextField
00084     field) {
00085         gbc.gridx = 0;
00086         gbc.gridy = row;
00087         frame.add(new JLabel(label), gbc);
00088         field.setEditable(true); // Make searchField editable
00089         gbc.gridx = 1;
00090         gbc.gridwidth = 2;
00091         frame.add(field, gbc);
00092         gbc.gridwidth = 1;
00093     }
00094
00095     private void addLabelAndField(GridBagConstraints gbc, int row, String label, JTextField field) {
00096         gbc.gridx = 0;

```

```

00111         gbc.gridy = row;
00112         frame.add(new JLabel(label), gbc);
00113         field.setEditable(false);
00114         gbc.gridx = 1;
00115         gbc.gridwidth = 2;
00116         frame.add(field, gbc);
00117         gbc.gridwidth = 1;
00118     }
00119
00120     private JTextField createReadOnlyField() {
00121         JTextField field = new JTextField();
00122         field.setEditable(false);
00123         return field;
00124     }
00125
00129     private void initComponents() {
00130         actions.fillArtist(artistField);
00131         actions.fillTitle(titleField);
00132         actions.fillAlbum(albumField); // Fill the album field if the method is available in
GuiActions
00133     }
00134
00138     private void addButtonListeners() {
00139         fileSelectButton.addActionListener(e -> selectTrack());
00140         playButton.addActionListener(e -> togglePlay());
00141
00142         // Add ActionListener to the search field
00143         searchField.addActionListener(e -> {
00144             try {
00145                 searchTrack(searchField.getText());
00146             } catch (UnsupportedTagException | InvalidDataException | IOException e1) {
00147                 e1.printStackTrace();
00148             }
00149         });
00150     }
00151
00155     private void configureProgressTimer() {
00156         progressTimer = new Timer(1000, e -> actions.fillProgress(progressField));
00157     }
00158
00159     private void selectTrack() {
00160         try {
00161             track = actions.selectTrack(frame);
00162             initComponents();
00163         } catch (UnsupportedTagException | InvalidDataException | IOException e1) {
00164             e1.printStackTrace();
00165         }
00166     }
00167
00173     private void togglePlay() {
00174         actions.playPressed();
00175         updateDynamicFields();
00176
00177         String currentStatus = actions.fillStatus();
00178         playButton.setText(currentStatus.equals("PLAYING") ? "Pause" : "Play");
00179
00180         if (actions.pressed) {
00181             progressTimer.start();
00182         } else {
00183             progressTimer.stop();
00184         }
00185     }
00186
00188     private void updateDynamicFields() {
00189         actions.fillProgress(progressField);
00190     }
00191
00200     private void searchTrack(String pattern) throws UnsupportedTagException, InvalidDataException,
IOException {
00201         System.out.println("Searching for: " + pattern);
00202         ArrayList<Track> tracks = new ArrayList<>();
00203         tracks = actions.searchTrack(pattern);
00204         System.out.println(tracks);
00205     }
00206
00207
00208 }

```

5.6 Track.java

```

00001 package com.example;
00002
00003 public class Track {

```



```

00004     String path;
00005     String title;
00006     String artis;
00007     String album;
00008     long length;
00009
00011     Track(String path, String title, String artist, String album, long length){
00012         this.path = path;
00013         this.title = title;
00014         this.artis = artist;
00015         this.album = album;
00016         this.length = length;
00017     }
00018
00020     Track(){
00021         this.path = null;
00022         this.title = null;
00023         this.artis = null;
00024         this.album = null;
00025         this.length = 0;
00026     }
00027
00028     // Getterek
00029
00034     public String getPath(){
00035         return path;
00036     }
00037
00038     public String getTitle(){
00039         return title;
00040     }
00041
00042     public String getArtist(){
00043         return artis;
00044     }
00045
00046     public String getAlbum(){
00047         return album;
00048     }
00049
00050     public long getLength(){
00051         return length;
00052     }
00053
00054     @Override
00055     public String toString() {
00056         return "title=" + title + ", artis='" + artis + " ";
00057     }
00058
00059
00060
00061 } // end of Track class
00062
00063

```

5.7 module-info.java

```

00001 module com.example {
00002     requires javafx.controls;
00003     requires javafx.fxml;
00004
00005     opens com.example to javafx.fxml;
00006     exports com.example;
00007
00008     // chatgpt
00009     requires javafx.media;
00010     requires mp3agic;
00011
00012     // Swing + awt
00013     requires java.desktop;
00014 }

```

5.8 MainTest.java

```

00001 package com.example;
00002
00003 import static org.junit.jupiter.api.Assertions.assertEquals;
00004
00005 import org.junit.jupiter.api.Test;

```

```
00006
00007 public class MainTest {
00008
00009     // @Test
00010     // public void testAdd() {
00011     //     Main main = new Main();
00012     //     int result = main.add(1, 2);
00013     //     assertEquals(3, result, "The add method should add two numbers");
00014     // }
00015 }
00016
```

Index

- actions
 - com.example.GuiHandler, 21
- album
 - com.example.Track, 27
- albumField
 - com.example.GuiHandler, 21
- artis
 - com.example.Track, 27
- artistField
 - com.example.GuiHandler, 21
- AudioHandler
 - com.example.AudioHandler, 10
- com.example.App, 7
 - main, 8
 - start, 8
- com.example.AudioHandler, 9
 - AudioHandler, 10
 - getMediaPlayer, 10
 - pause, 10
 - play, 11
- com.example.FileHandler, 11
 - createTrackFromPath, 12
 - readDir, 13
 - searchTracks, 13
 - trackList, 15
- com.example.GuiActions, 16
 - fillAlbum, 17
 - fillArtist, 17
 - fillProgress, 17
 - fillStatus, 17
 - fillTitle, 18
 - playPressed, 18
 - pressed, 19
 - searchTrack, 18
 - selected, 19
 - selectTrack, 18
- com.example.GuiHandler, 20
 - actions, 21
 - albumField, 21
 - artistField, 21
 - fileSelectButton, 21
 - frame, 22
 - init, 21
 - playButton, 22
 - progressField, 22
 - progressTimer, 22
 - searchField, 22
 - titleField, 22
 - track, 22
- com.example.MainTest, 23
- com.example.Track, 24
 - album, 27
 - artis, 27
 - getAlbum, 25
 - getArtist, 25
 - getLength, 26
 - getPath, 26
 - getTitle, 26
 - length, 27
 - path, 27
 - title, 27
 - toString, 26
 - Track, 25
- createTrackFromPath
 - com.example.FileHandler, 12
- fileSelectButton
 - com.example.GuiHandler, 21
- fillAlbum
 - com.example.GuiActions, 17
- fillArtist
 - com.example.GuiActions, 17
- fillProgress
 - com.example.GuiActions, 17
- fillStatus
 - com.example.GuiActions, 17
- fillTitle
 - com.example.GuiActions, 18
- frame
 - com.example.GuiHandler, 22
- getAlbum
 - com.example.Track, 25
- getArtist
 - com.example.Track, 25
- getLength
 - com.example.Track, 26
- getMediaPlayer
 - com.example.AudioHandler, 10
- getPath
 - com.example.Track, 26
- getTitle
 - com.example.Track, 26
- init
 - com.example.GuiHandler, 21
- length
 - com.example.Track, 27

- main
 - com.example.App, [8](#)
- path
 - com.example.Track, [27](#)
- pause
 - com.example.AudioHandler, [10](#)
- play
 - com.example.AudioHandler, [11](#)
- playButton
 - com.example.GuiHandler, [22](#)
- playPressed
 - com.example.GuiActions, [18](#)
- pressed
 - com.example.GuiActions, [19](#)
- progressField
 - com.example.GuiHandler, [22](#)
- progressTimer
 - com.example.GuiHandler, [22](#)
- readDir
 - com.example.FileHandler, [13](#)
- searchField
 - com.example.GuiHandler, [22](#)
- searchTrack
 - com.example.GuiActions, [18](#)
- searchTracks
 - com.example.FileHandler, [13](#)
- selected
 - com.example.GuiActions, [19](#)
- selectTrack
 - com.example.GuiActions, [18](#)
- src/main/java/com/example/App.java, [29](#)
- src/main/java/com/example/AudioHandler.java, [29](#)
- src/main/java/com/example/FileHandler.java, [30](#)
- src/main/java/com/example/GuiActions.java, [31](#)
- src/main/java/com/example/GuiHandler.java, [32](#)
- src/main/java/com/example/Track.java, [34](#)
- src/main/java/module-info.java, [35](#)
- src/test/java/com/example/MainTest.java, [35](#)
- start
 - com.example.App, [8](#)
- title
 - com.example.Track, [27](#)
- titleField
 - com.example.GuiHandler, [22](#)
- toString
 - com.example.Track, [26](#)
- Track
 - com.example.Track, [25](#)
- track
 - com.example.GuiHandler, [22](#)
- trackList
 - com.example.FileHandler, [15](#)