MP3Java

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 com.example.App Class Reference

Inheritance diagram for com.example.App:

Collaboration diagram for com.example.App:

```
          ┌──────────────┐
          │ Application  │
          ├──────────────┤
          │              │
          ├──────────────┤
          │              │
          └──────────────┘
                  △
                  │
          ┌──────────────┐
          │com.example.App│
          ├──────────────┤
          │              │
          ├──────────────┤
          │  +   start() │
          │  +   main()  │
          └──────────────┘
```
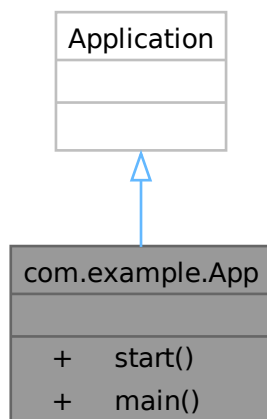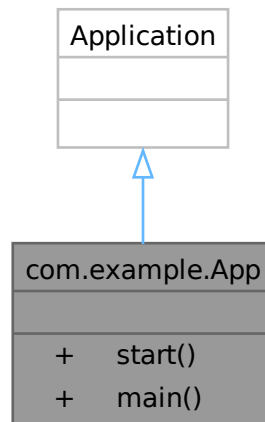
Public Member Functions

- void start (Stage stage) throws UnsupportedTagException, InvalidDataException, IOException

Static Public Member Functions

- static void main (String[ ] args)

4.1.1 Detailed Description

This class conatins 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 UnsupportedTagException, InvalidDataException, IOException
```

javaFx requeres 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/i3hunor/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:

com.example.AudioHandler
+   AudioHandler()
+   play()
+   pause()
+   getMediaPlayer()
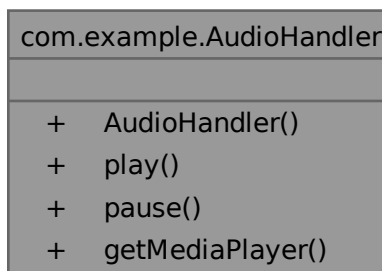
Public Member Functions

- 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

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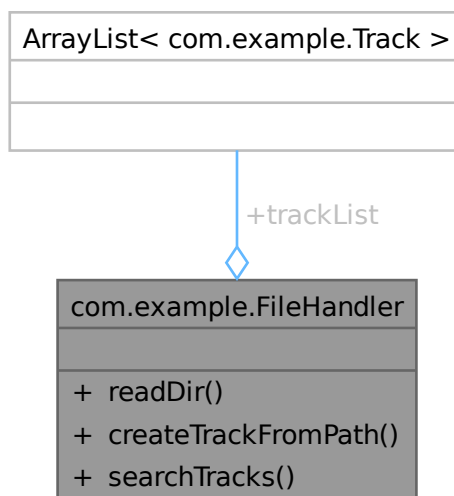
```
00028                          {
00029          mediaPlayer.play();
00030          System.out.println("Playing...");
00031     }
```

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, InvalidData↩
  Exception
- ArrayList< Track > searchTracks (String pattern, String path) throws IOException, UnsupportedTag↩
  Exception, 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

path	- 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 fájlnak.");
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:
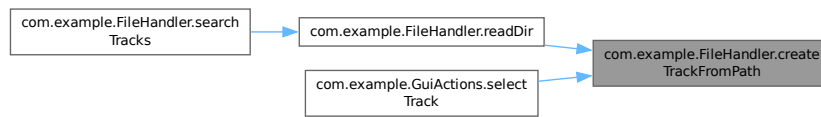
4.3.2.2 readDir()

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

Parameters

path	dal elérési útvonala A megadott dir végigpásztázása mp3 fájlok után kutatva A megtalált mp3 fájlokból egyesével lértehoz 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:
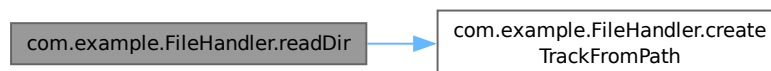
Here is the caller graph for this function:
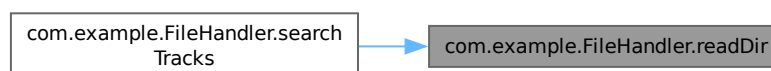
4.3.2.3 searchTracks()

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

Parameters

pattern	keresett cím stringben megadva

Returns

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

Parameters

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

Returns

matching trackek

Exceptions

IOException	
UnsupportedTagException	
InvalidDataException	

Definition at line 111 of file FileHandler.java.

```
00111
     {
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     }
```

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:
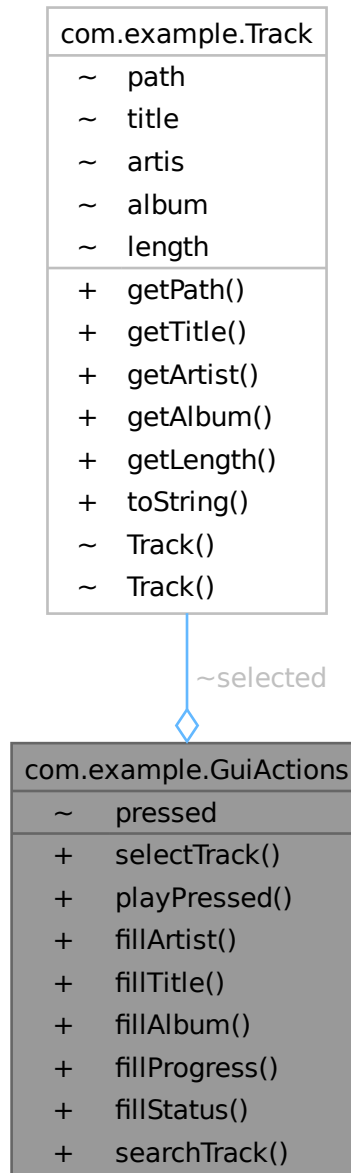
Public Member Functions

- 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 UnsupportedTagException, 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 UnsupportedTagException, InvalidDataException, IOException
```

Definition at line 114 of file GuiActions.java.
```
00114
      {
00115          return fileHandler.searchTracks(pattern,
      "/home/i3hunor/Suli/Prog3/nagyHF/Fasz/mp3java/src/main/resources");
00116     }
```

4.4.2.8 selectTrack()

```
Track com.example.GuiActions.selectTrack (
              JFrame frame) throws UnsupportedTagException, InvalidDataException, IOException
```

A dal/file kiválasztásáért felel

Parameters

frame	

Returns

  selected track

Exceptions

UnsupportedTagException	
InvalidDataException	
IOException	

curr directoryt nyitja meg

csak mp3 fájlokat látsz

Definition at line 33 of file GuiActions.java.

```
00033    {
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    }
```

Here is the call graph for this function:
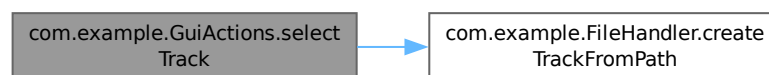
4.4.3 Member Data Documentation

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]
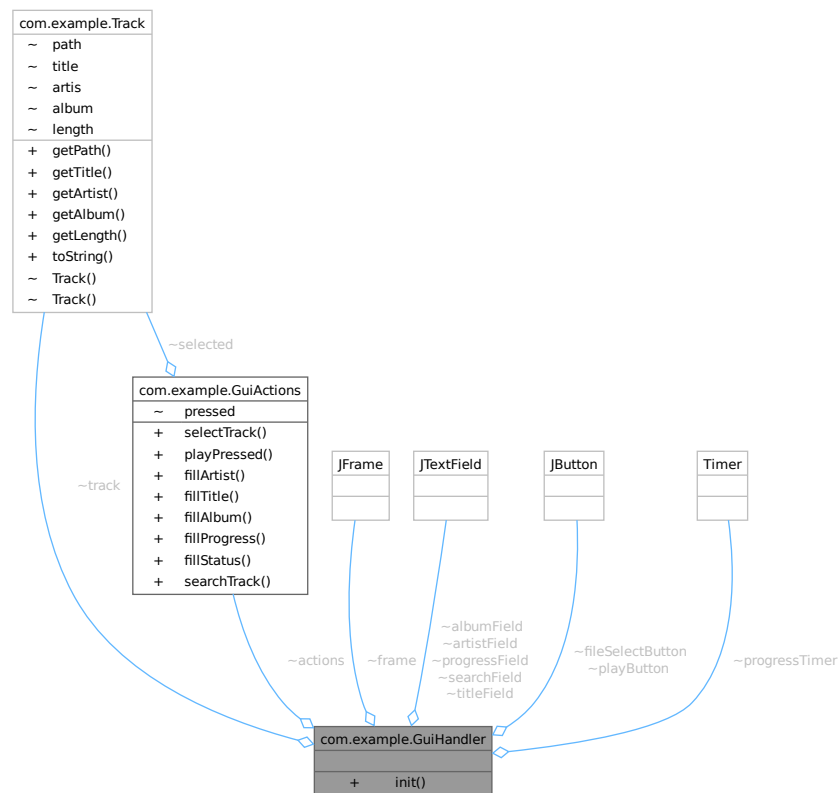
Definition at line 23 of file GuiActions.java.

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 24 of file GuiHandler.java.

4.5.2 Member Function Documentation

4.5.2.1 init()

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

Definition at line 32 of file GuiHandler.java.
```
00032                    {
00033          initFrame();
00034          initComponents();
00035          configureProgressTimer();
00036    }
```

4.5.3 Member Data Documentation

4.5.3.1 actions

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

Definition at line 25 of file GuiHandler.java.

4.5.3.2 albumField

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

Definition at line 28 of file GuiHandler.java.

4.5.3.3 artistField

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

Definition at line 28 of file GuiHandler.java.

4.5.3.4 fileSelectButton

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

Definition at line 29 of file GuiHandler.java.

4.5.3.5 frame

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

Definition at line 27 of file GuiHandler.java.

4.5.3.6 playButton

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

Definition at line 29 of file GuiHandler.java.

4.5.3.7 progressField

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

Definition at line 28 of file GuiHandler.java.

4.5.3.8 progressTimer

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

Definition at line 30 of file GuiHandler.java.

4.5.3.9 searchField

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

Definition at line 28 of file GuiHandler.java.

4.5.3.10 titleField

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

Definition at line 28 of file GuiHandler.java.

4.5.3.11 track

```
Track com.example.GuiHandler.track  [package]
```

Definition at line 26 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
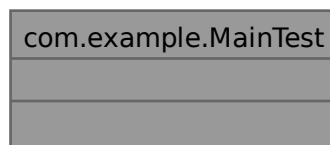
Definition at line 7 of file MainTest.java.

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

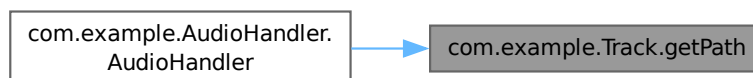**Returns**

　　　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/i3hunor/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     }
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 fájlnak.");
```

```
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 toLoweCase()
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 UnsupportedTagException,
    InvalidDataException, IOException {
00115        return fileHandler.searchTracks(pattern,
    "/home/i3hunor/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 // A felugró ablakhoz kell
00018 import javax.swing.JList;
00019 import javax.swing.ListSelectionModel;
00020 import javax.swing.JScrollPane;
00021
00022
00023
00024 public class GuiHandler {
00025     GuiActions actions = new GuiActions();
00026     Track track;
00027     JFrame frame;
00028     JTextField searchField, artistField, titleField, albumField, progressField;
00029     JButton playButton, fileSelectButton;
00030     Timer progressTimer;
00031
00032     public void init() {
00033         initFrame();
00034         initComponents();
00035         configureProgressTimer();
00036     }
00037
00038     private void initFrame() {
00039         frame = new JFrame("Mp3Java");
00040         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
00041         frame.setSize(400, 250);
00042         frame.setLayout(new GridBagLayout());
00043         setupLayout();
00044         frame.pack();
00045         frame.setLocationRelativeTo(null);
00046         frame.setVisible(true);
00047     }
00048
00052     private void setupLayout() {
00053         GridBagConstraints gbc = new GridBagConstraints();
00054         gbc.fill = GridBagConstraints.HORIZONTAL;
00055         gbc.weightx = 1.0;
00056
00057         // Top row: Search field
00058         addSearchLabelAndField(gbc, 0, "Search:", searchField = new JTextField(20));
00059
00060         // Artist row
00061         addLabelAndField(gbc, 1, "Artist:", artistField = new JTextField(20));
00062
00063         // Title row
00064         addLabelAndField(gbc, 2, "Title:", titleField = new JTextField(20));
00065
00066         // Album row
00067         addLabelAndField(gbc, 3, "Album:", albumField = new JTextField(20));
00068
00069         // Middle row: Progress field
00070         gbc.gridx = 0;
00071         gbc.gridy = 4;
00072         gbc.gridwidth = 3;
00073         progressField = createReadOnlyField();
00074         frame.add(progressField, gbc);
00075
00076         // Bottom row: File Select and Play/Stop buttons
00077         gbc.gridwidth = 1;
00078         gbc.gridy = 5;
00079         fileSelectButton = new JButton("File Select");
00080         gbc.gridx = 0;
00081         frame.add(fileSelectButton, gbc);
00082
00083         playButton = new JButton("Play");
00084         gbc.gridx = 1;
00085         frame.add(playButton, gbc);
00086
00087         addButtonListeners();
00088     }
00089
00097     private void addSearchLabelAndField(GridBagConstraints gbc, int row, String label, JTextField
     field) {
00098         gbc.gridx = 0;
00099         gbc.gridy = row;
00100         frame.add(new JLabel(label), gbc);
00101         field.setEditable(true); // Make searchField editable
00102         gbc.gridx = 1;
00103         gbc.gridwidth = 2;
00104         frame.add(field, gbc);
00105         gbc.gridwidth = 1;
```

```
00106      }
00107
00116      private void addLabelAndField(GridBagConstraints gbc, int row, String label, JTextField field) {
00117          gbc.gridx = 0;
00118          gbc.gridy = row;
00119          frame.add(new JLabel(label), gbc);
00120          field.setEditable(false);
00121          gbc.gridx = 1;
00122          gbc.gridwidth = 2;
00123          frame.add(field, gbc);
00124          gbc.gridwidth = 1;
00125      }
00126
00127      private JTextField createReadOnlyField() {
00128          JTextField field = new JTextField();
00129          field.setEditable(false);
00130          return field;
00131      }
00132
00136      private void initComponents() {
00137          actions.fillArtist(artistField);
00138          actions.fillTitle(titleField);
00139          actions.fillAlbum(albumField); // Fill the album field if the method is available in
      GuiActions
00140      }
00141
00145      private void addButtonListeners() {
00146          fileSelectButton.addActionListener(e -> selectTrack());
00147          playButton.addActionListener(e -> togglePlay());
00148
00149          // Add ActionListener to the search field
00150          searchField.addActionListener(e -> {
00151              try {
00152                  searchTrack(searchField.getText());
00153              } catch (UnsupportedTagException | InvalidDataException | IOException e1) {
00154                  e1.printStackTrace();
00155              }
00156          });
00157      }
00158
00162      private void configureProgressTimer() {
00163          progressTimer = new Timer(1000, e -> actions.fillProgress(progressField));
00164      }
00165
00166      private void selectTrack() {
00167          try {
00168              track = actions.selectTrack(frame);
00169              initComponents();
00170          } catch (UnsupportedTagException | InvalidDataException | IOException e1) {
00171              e1.printStackTrace();
00172          }
00173      }
00174
00180      private void togglePlay() {
00181          actions.playPressed();
00182          updateDynamicFields();
00183
00184          String currentStatus = actions.fillStatus();
00185          playButton.setText(currentStatus.equals("PLAYING") ? "Pause" : "Play");
00186
00187          if (actions.pressed) {
00188              progressTimer.start();
00189          } else {
00190              progressTimer.stop();
00191          }
00192      }
00193
00195      private void updateDynamicFields() {
00196          actions.fillProgress(progressField);
00197      }
00198
00199
00204      private void showSearchResultsWindow(ArrayList<Track> results) {
00205          JFrame resultsFrame = new JFrame("Search Results");
00206          resultsFrame.setSize(400, 300);
00207          resultsFrame.setLayout(new GridBagLayout());
00208          resultsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
00209
00210          GridBagConstraints gbc = new GridBagConstraints();
00211          gbc.fill = GridBagConstraints.HORIZONTAL;
00212          gbc.weightx = 1.0;
00213          gbc.gridx = 0;
00214          gbc.gridy = 0;
00215
00216          // lista a resultoknak
00217          JList<String> resultsList = new JList<>(
00218                  results.stream().map(track -> track.getTitle() + " - " +
```

```
                track.getArtist()).toArray(String[]::new));
00219           resultsList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
00220           JScrollPane scrollPane = new JScrollPane(resultsList);
00221
00222           gbc.gridy = 0;
00223           gbc.weighty = 1.0;
00224           gbc.fill = GridBagConstraints.BOTH;
00225           resultsFrame.add(scrollPane, gbc);
00226
00227           resultsFrame.setVisible(true);
00228       }
00229
00238       private void searchTrack(String pattern) throws UnsupportedTagException, InvalidDataException,
        IOException {
00239           System.out.println("Searching for: " + pattern);
00240           ArrayList<Track> tracks = actions.searchTrack(pattern);
00241
00242           if (tracks.isEmpty()) {
00243               System.out.println("No results found.");
00244           } else {
00245               showSearchResultsWindow(tracks);
00246           }
00247       }
00248
00249 } // end of GuiHandler
```

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 + "', artist='" + 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