# MP3Java

Generated by Doxygen 1.12.0

# 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

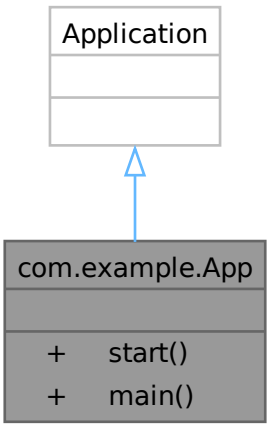
Here is a list of all documented files with brief descriptions:

# Chapter 4

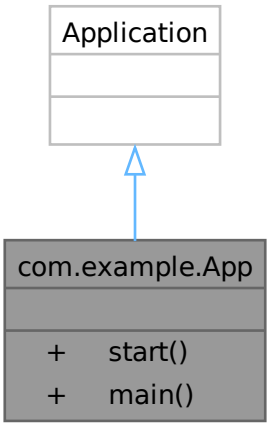
# Class Documentation

## 4.1 com.example.App Class Reference

Inheritance diagram for com.example.App:



Collaboration diagram for com.example.App:




**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:


### Public Member Functions

- AudioHandler (Track track)
- void play ()

- void pause ()
- MediaPlayer getMediaPlayer ()
- Track getTrackFromAH ()

**Package Attributes**

- Track track

## 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

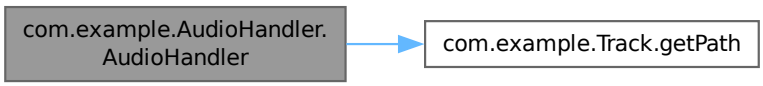
**Parameters**

| | |
|---|---|
| *track* | dal amit le akarok játszani |

Definition at line 19 of file AudioHandler.java.

```
00019                                       {
00020         String path = track.getPath();
00021         this.track = track;
00022         Media media = new Media(new File(path).toURI().toString());
00023         mediaPlayer = new MediaPlayer(media);
00024     }
```

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 43 of file AudioHandler.java.

```
00043                                       {
00044         return mediaPlayer;
00045     }
```

#### 4.2.3.2 getTrackFromAH()

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

Definition at line 47 of file AudioHandler.java.

```
00047                                   {
00048         return track;
00049     }
```

#### 4.2.3.3 pause()

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

Calls pause() methond on the mediaPlayer

Definition at line 38 of file AudioHandler.java.

```
00038                           {
00039         mediaPlayer.pause();
00040         System.out.println("Paused.");
00041     }
```

#### 4.2.3.4 play()

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

Calls the play() method on the initialized media player Works as a resume() too

Definition at line 30 of file AudioHandler.java.

```
00030                           {
00031         mediaPlayer.play();
00032         System.out.println("Playing...");
00033     }
```

### 4.2.4 Member Data Documentation

#### 4.2.4.1 track

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

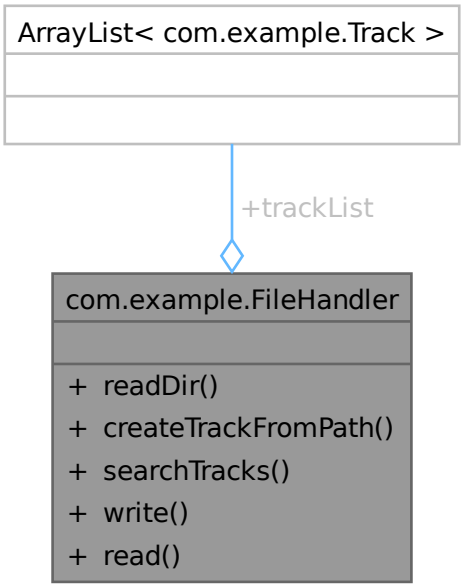
Definition at line 13 of file AudioHandler.java.

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
- void write (AudioHandler ah, String path)
- ArrayList< Track > read (String path)

### Public Attributes

- ArrayList< Track > trackList = new ArrayList<>()

### 4.3.1 Detailed Description

FileHandler - Minden aminek fileokhoz van köze itt van kezelve

Definition at line 22 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

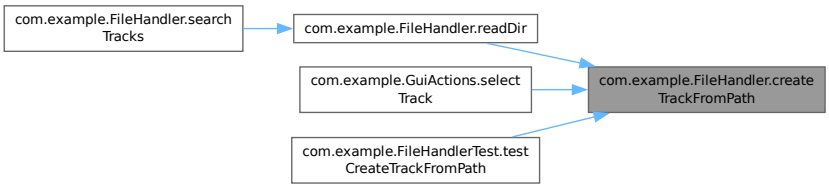
**Parameters**

| | |
|---|---|
| *path* | - a dal elérési útvonala |

Definition at line 63 of file FileHandler.java.

```
00064                           {
00065         // Inicializálás
00066         Mp3File mp3File = new Mp3File(path);
00067         long length = 0;
00068         String artist = null;
00069         String title = null;
00070         String album = null;
00071 
00072         // Megnézem, hogy milyen típusú tagjei vannak
00073         if(mp3File.hasId3v1Tag()){
00074             ID3v1 v1Tag = mp3File.getId3v1Tag();
00075             length = mp3File.getLengthInSeconds();
00076             artist = v1Tag.getArtist();
00077             title = v1Tag.getTitle();
00078             album = v1Tag.getAlbum();
00079         }
00080         else if(mp3File.hasId3v2Tag()){
00081             ID3v2 v2Tag = mp3File.getId3v2Tag();
00082             length = mp3File.getLengthInSeconds();
00083             artist = v2Tag.getArtist();
00084             title = v2Tag.getTitle();
00085             album = v2Tag.getAlbum();
00086         }
00087         else System.err.println("Se v1 se v2 tagjei nincsenek a fájlnak.");
00088 
00089         // Trakc konstruálás, lényegi rész
00090         return new Track(path, title, artist, album, length);
00091     } // end of createTrackFromPath()
```

Here is the caller graph for this function:




#### 4.3.2.2 read()

```
ArrayList< Track > com.example.FileHandler.read (
            String path)
```

Definition at line 155 of file FileHandler.java.

```
00155                                           {
00156         Gson gson = new Gson();
00157         ArrayList<Track> tracks = new ArrayList<>();
00158 
00159         try (FileReader reader = new FileReader(path)) {
00160             Type listType = new TypeToken<ArrayList<Track>>() {}.getType();
00161             tracks = gson.fromJson(reader, listType);
00162         } catch (IOException e) {
00163             e.printStackTrace();
00164         }
00165 
00166         // tracks.forEach(track -> System.out.println(track.getTitle()));
00167         return tracks;
00168     }
```

#### 4.3.2.3 readDir()

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

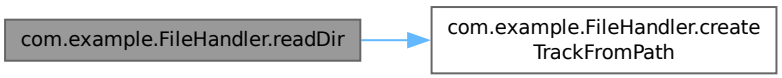
**Parameters**

| | |
|---|---|
| *path* | dal elérési útvonala A megadott dir végigpásztázása mp3 fájlok után kutatva A megtalált mp3 fájlokból egyesével lértehoz egy T //rack objektumot |

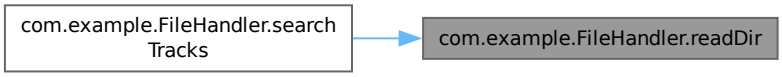
Definition at line 35 of file FileHandler.java.

```
00036                           {
00037         File dir = new File(path);
00038         File[] listOfFiles = dir.listFiles();
00039         String name;
00040 
00041         if(listOfFiles != null){
00042             for(int i = 0; i < listOfFiles.length; i++){
00043                 name = listOfFiles[i].getName();
00044 
00045                 if(listOfFiles[i].isFile() && name.endsWith(".mp3")){
00046                     Track track = createTrackFromPath(listOfFiles[i].getAbsolutePath());
00047                     trackList.add(track);
00048                 }
00049             } // end of for loop
00050         }
00051         // Error handling
00052         else System.out.println("Hiba a mappa pásztázásakor");
00053     } // end of readDir()
```

Here is the call graph for this function:




Here is the caller graph for this function:



#### 4.3.2.4 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 117 of file FileHandler.java.

```
00117     {
00118         trackList.clear();
00119         readDir(path);
00120 
00121         String lowerCasepattern = pattern.toLowerCase();
00122 
00123         ArrayList<Track> matchingTracks = new ArrayList<>();
00124 
00125         for (Track t : trackList) {
00126             if (t.getTitle().toLowerCase().contains(lowerCasepattern) ||
00127                 t.getArtist().toLowerCase().contains(lowerCasepattern)) {
00128                 matchingTracks.add(t);
00129             }
00130         }
00131 
00132         return matchingTracks;
00133     }
```

Here is the call graph for this function:

#### 4.3.2.5 write()

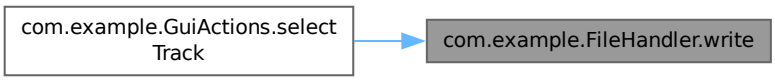
```
void com.example.FileHandler.write (
            AudioHandler ah,
            String path)
```

Szerializáció Minden lejátszott dal (pontosabban audioplayerbe helyezett dal) belekerül a listába és ki lesz írva a playedTracks file-ba

Definition at line 140 of file FileHandler.java.

```
00140                                           {
00141         toBeSerialized.add(ah.getTrackFromAH());
00142
00143         Gson gson = new GsonBuilder().setPrettyPrinting().create(); // For formatted JSON
00144
00145         try (FileWriter writer = new FileWriter(path)) {
00146             gson.toJson(toBeSerialized, writer);
00147             System.out.println("Written out");
00148         } catch (IOException e) {
00149         e.printStackTrace();
00150         }
00151     }
```

Here is the caller graph for this function:




### 4.3.3 Member Data Documentation

#### 4.3.3.1 trackList

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

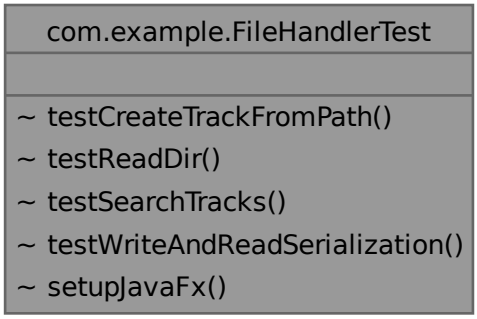
Definition at line 25 of file FileHandler.java.

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

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

## 4.4 com.example.FileHandlerTest Class Reference

Collaboration diagram for com.example.FileHandlerTest:




### Package Functions

- void testCreateTrackFromPath () throws Exception
- void testReadDir () throws UnsupportedTagException, InvalidDataException, IOException
- void testSearchTracks () throws UnsupportedTagException, InvalidDataException, IOException
- void testWriteAndReadSerialization () throws UnsupportedTagException, InvalidDataException, IOException

### Static Package Functions

- static void setupJavaFx ()

### 4.4.1 Detailed Description

Definition at line 18 of file FileHandlerTest.java.

### 4.4.2 Member Function Documentation

#### 4.4.2.1 setupJavaFx()

```
static void com.example.FileHandlerTest.setupJavaFx ()  [static], [package]
```

Definition at line 21 of file FileHandlerTest.java.

```
00021                           {
00022     if (!Platform.isImplicitExit()) {
00023         Platform.startup(() -> {});
00024     }
00025 }
```

#### 4.4.2.2 testCreateTrackFromPath()

```
void com.example.FileHandlerTest.testCreateTrackFromPath () throws Exception  [package]
```

Azért van előbb mert a readdir ezt a fv-t futtatja és a pásztázott fileokat trackké varátsolja

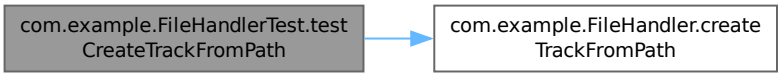
**Exceptions**

| *Exception* | |
|---|---|

Definition at line 34 of file FileHandlerTest.java.

```
00034                                                {
00035         String song = "src/test/resources/mockData/Linkin-1.mp3";
00036         FileHandler fh = new FileHandler();
00037         Track track = fh.createTrackFromPath(song);
00038         String artist = track.getArtist();
00039 
00040         assertNotNull(track);
00041         assertEquals("Linkin Park", artist);
00042     }
```

Here is the call graph for this function:




#### 4.4.2.3 testReadDir()

```
void com.example.FileHandlerTest.testReadDir () throws UnsupportedTagException, InvalidData↩
Exception, IOException  [package]
```

Definition at line 45 of file FileHandlerTest.java.

```
00045                                                                                {
00046         String dir = "src/test/resources/mockData/";
00047         FileHandler fh = new FileHandler();
00048         fh.readDir(dir);
00049 
00050         assertEquals(2, fh.trackList.size());
00051         assertTrue(fh.trackList.stream().anyMatch(track -> track.getPath().endsWith("Linkin-1.mp3")));
00052         assertTrue(fh.trackList.stream().anyMatch(track -> track.getPath().endsWith("Linkin-2.mp3")));
00053     }
```

#### 4.4.2.4 testSearchTracks()

```
void com.example.FileHandlerTest.testSearchTracks () throws UnsupportedTagException, Invalid↩
DataException, IOException  [package]
```

Definition at line 56 of file FileHandlerTest.java.

```
00056                                                                                    {
00057         String pattern = "Linkin";
00058         String path = "src/test/resources/mockData/";
00059         FileHandler fh = new FileHandler();
00060         ArrayList<Track> list;
00061         list = fh.searchTracks(pattern, path);
00062 
00063         assertNotNull(list);
00064         assertEquals(2, list.size());
00065         assertTrue(list.stream().anyMatch(track -> track.getArtist().contains("Linkin")));
00066     }
```

#### 4.4.2.5 testWriteAndReadSerialization()

```
void com.example.FileHandlerTest.testWriteAndReadSerialization () throws UnsupportedTag↩
Exception, InvalidDataException, IOException  [package]
```

Definition at line 69 of file FileHandlerTest.java.

```
00069
     {
00070         FileHandler fileHandler = new FileHandler();
00071         String testFile = "serialTest.json";
00072
00073         String songPath = "src/test/resources/mockData/Linkin-1.mp3";
00074         FileHandler fh = new FileHandler();
00075         Track track = fh.createTrackFromPath(songPath);
00076         AudioHandler ah = new AudioHandler(track);
00077
00078         fileHandler.write(ah, testFile);
00079
00080         ArrayList<Track> readTracks = fileHandler.read(testFile);
00081
00082         assertNotNull(readTracks);
00083         assertEquals(1, readTracks.size());
00084         assertEquals("Linkin Park", readTracks.get(0).getArtist());
00085     }
```

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

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

## 4.5 com.example.GuiActions Class Reference

Collaboration diagram for com.example.GuiActions:


### Public Member Functions

- Track selectTrack (JFrame frame) throws UnsupportedTagException, InvalidDataException, IOException
- void selectedFromSearch (Track track)

- 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

### 4.5.1 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.

### 4.5.2 Member Function Documentation

#### 4.5.2.1 fillAlbum()

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

Definition at line 94 of file GuiActions.java.

```
00094                                           {
00095         if(selected != null){
00096             String album = selected.getAlbum();
00097             field.setText(album);
00098             System.out.println("Album filed filled");
00099         }
00100         else field.setText("NA");
00101     }
```

#### 4.5.2.2 fillArtist()

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

Definition at line 76 of file GuiActions.java.

```
00076                                            {
00077         if(selected != null){
00078             String artist = selected.getArtist();
00079             field.setText(artist);
00080             System.out.println("Artist filed filled");
00081         }
00082         else field.setText("NA");
00083     }
```

#### 4.5.2.3 fillProgress()

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

Definition at line 103 of file GuiActions.java.

```
00103                                                {
00104         if(selected != null){
00105             long len = selected.getLength();
00106             int curr = (int) audioHandler.getMediaPlayer().getCurrentTime().toSeconds();
00107             field.setText(curr + "/" + len);
00108             // System.out.println("Length of the track: " + len);
00109         }
00110         else field.setText("NA");
00111     }
```

#### 4.5.2.4 fillStatus()

```
String com.example.GuiActions.fillStatus ()
```

Definition at line 113 of file GuiActions.java.

```
00113                                {
00114         if(selected != null){
00115              String status = audioHandler.getMediaPlayer().getStatus().toString();
00116              System.out.println("Status filled");
00117              return status;
00118         }
00119         return "Not selected";
00120     }
```

#### 4.5.2.5 fillTitle()

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

Definition at line 85 of file GuiActions.java.

```
00085                                           {
00086         if(selected != null){
00087              String artist = selected.getTitle();
00088              field.setText(artist);
00089              System.out.println("Titke filed filled");
00090         }
00091         else field.setText("NA");
00092     }
```

#### 4.5.2.6 playPressed()

```
void com.example.GuiActions.playPressed ()
```

Definition at line 65 of file GuiActions.java.

```
00065                                {
00066         if (!pressed) {
00067             audioHandler.play();
00068             pressed = true;
00069         }
00070         else {
00071             audioHandler.pause();
00072             pressed = false;
00073         }
00074     }
```

#### 4.5.2.7 searchTrack()

```
ArrayList< Track > com.example.GuiActions.searchTrack (
            String pattern) throws UnsupportedTagException, InvalidDataException, IOException
```

Definition at line 122 of file GuiActions.java.

```
00122
      {
00123         return fileHandler.searchTracks(pattern,
      "/home/i3hunor/Suli/Prog3/nagyHF/Fasz/mp3java/src/main/resources");
00124     }
```

#### 4.5.2.8 selectedFromSearch()

```
void com.example.GuiActions.selectedFromSearch (
            Track track)
```

Definition at line 59 of file GuiActions.java.

```
00059                                           {
00060         selected = track;
00061         audioHandler = new AudioHandler(track);
00062         fileHandler.write(audioHandler, "playedTracks.json"); // szerializáció
00063     }
```

#### 4.5.2.9 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

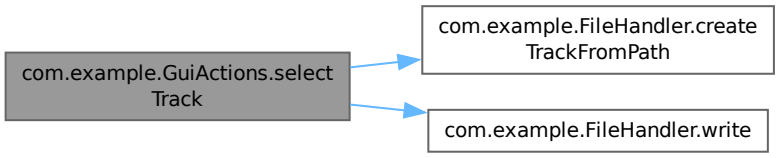
csak mp3 fájlokat látsz

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             fileHandler.write(audioHandler, "playedTracks.json"); // szerializáció
00052
00053             return selected;
00054         }
00055
00056         return null;
00057     }
```

Here is the call graph for this function:




## 4.5.3 Member Data Documentation

### 4.5.3.1 pressed

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

Definition at line 22 of file GuiActions.java.

### 4.5.3.2 selected

```
Track com.example.GuiActions.selected  [package]
```

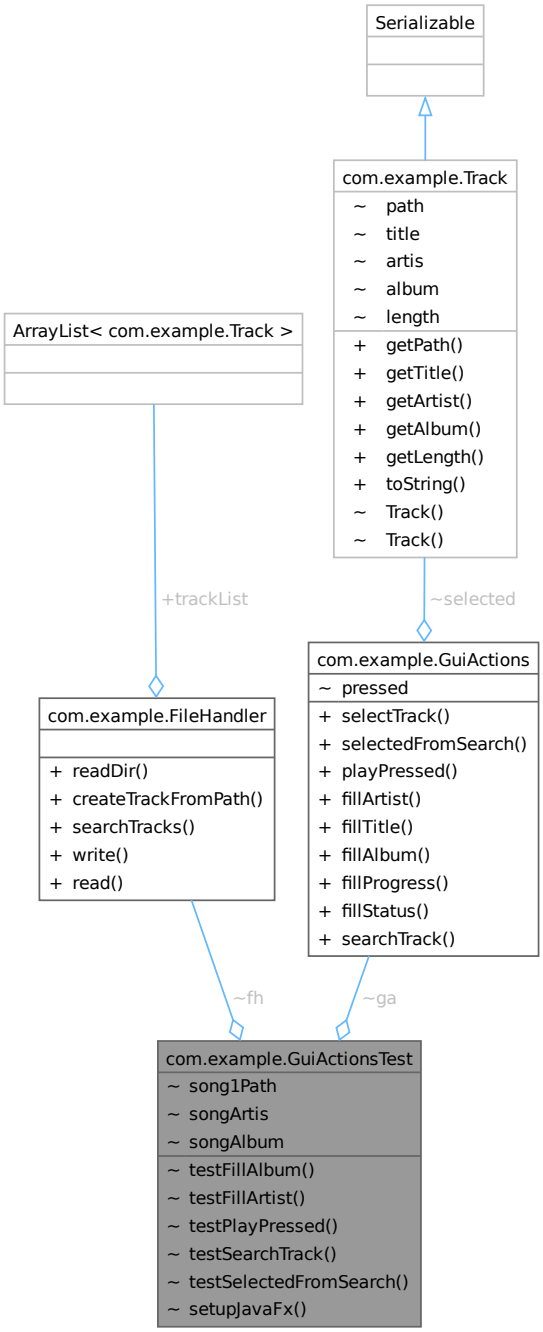
Definition at line 23 of file GuiActions.java.

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

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

## 4.6 com.example.GuiActionsTest Class Reference

Collaboration diagram for com.example.GuiActionsTest:




### Package Functions

- void testFillAlbum () throws UnsupportedTagException, InvalidDataException, IOException
- void testFillArtist () throws UnsupportedTagException, InvalidDataException, IOException

- void testPlayPressed () throws UnsupportedTagException, InvalidDataException, IOException
- void testSearchTrack () throws Exception
- void testSelectedFromSearch () throws UnsupportedTagException, InvalidDataException, IOException

#### Static Package Functions

- static void setupJavaFx ()

#### Package Attributes

- String song1Path = "src/test/resources/mockData/Linkin-1.mp3"
- String songArtis = "Linkin Park"
- String songAlbum = "Meteora"
- FileHandler fh = new FileHandler()
- GuiActions ga = new GuiActions()

### 4.6.1 Detailed Description

Definition at line 20 of file GuiActionsTest.java.

### 4.6.2 Member Function Documentation

#### 4.6.2.1 setupJavaFx()

```
static void com.example.GuiActionsTest.setupJavaFx ()  [static], [package]
```

Definition at line 29 of file GuiActionsTest.java.

```
00029                                {
00030         Platform.startup(() -> {});
00031     }
```

#### 4.6.2.2 testFillAlbum()

```
void com.example.GuiActionsTest.testFillAlbum () throws UnsupportedTagException, InvalidData↩
Exception, IOException  [package]
```

Definition at line 34 of file GuiActionsTest.java.

```
00034                                                                                {
00035         Track track = fh.createTrackFromPath(song1Path);
00036         ga.selectedFromSearch(track);
00037
00038         JTextField albumField = new JTextField();
00039         ga.fillAlbum(albumField);
00040
00041         assertEquals(songAlbum, albumField.getText());
00042
00043     }
```

#### 4.6.2.3 testFillArtist()

```
void com.example.GuiActionsTest.testFillArtist () throws UnsupportedTagException, Invalid↩
DataException, IOException  [package]
```

Definition at line 46 of file GuiActionsTest.java.

```
00046                                                                                {
00047         Track track = fh.createTrackFromPath(song1Path);
00048         ga.selectedFromSearch(track);
00049
00050         JTextField artistField = new JTextField();
00051         ga.fillArtist(artistField);
00052
00053         assertEquals(songArtis, artistField.getText());
00054     }
```

#### 4.6.2.4 testPlayPressed()

```
void com.example.GuiActionsTest.testPlayPressed () throws UnsupportedTagException, Invalid↩
DataException, IOException  [package]
```

Definition at line 57 of file GuiActionsTest.java.

```
00057                                                                                 {
00058         Track track = fh.createTrackFromPath(song1Path);
00059         ga.selectedFromSearch(track);
00060
00061         ga.playPressed();
00062         assertTrue(ga.pressed);
00063
00064         ga.playPressed();
00065         assertFalse(ga.pressed);
00066     }
```

#### 4.6.2.5 testSearchTrack()

```
void com.example.GuiActionsTest.testSearchTrack () throws Exception  [package]
```

Definition at line 70 of file GuiActionsTest.java.

```
00070                                           {
00071         ArrayList<Track> tracks = ga.searchTrack(songArtis);
00072
00073         assertFalse(tracks.isEmpty());
00074         assertTrue(tracks.stream().anyMatch(track -> track.getArtist().equalsIgnoreCase(songArtis)));
00075     }
```

#### 4.6.2.6 testSelectedFromSearch()

```
void com.example.GuiActionsTest.testSelectedFromSearch () throws UnsupportedTagException,
InvalidDataException, IOException  [package]
```

Definition at line 78 of file GuiActionsTest.java.

```
00078                                                                                {
00079         Track track = fh.createTrackFromPath("src/test/resources/mockData/Linkin-1.mp3");
00080
00081         ga.selectedFromSearch(track);
00082
00083         assertEquals(songArtis, track.getArtist());
00084         assertEquals(songAlbum, track.getAlbum());
00085
00086     }
```

### 4.6.3 Member Data Documentation

#### 4.6.3.1 fh

```
FileHandler com.example.GuiActionsTest.fh = new FileHandler()  [package]
```

Definition at line 24 of file GuiActionsTest.java.

#### 4.6.3.2 ga

```
GuiActions com.example.GuiActionsTest.ga = new GuiActions()  [package]
```

Definition at line 25 of file GuiActionsTest.java.

#### 4.6.3.3 song1Path

```
String com.example.GuiActionsTest.song1Path = "src/test/resources/mockData/Linkin-1.mp3"  [package]
```

Definition at line 21 of file GuiActionsTest.java.

#### 4.6.3.4 songAlbum

```
String com.example.GuiActionsTest.songAlbum = "Meteora"  [package]
```

Definition at line 23 of file GuiActionsTest.java.

#### 4.6.3.5 songArtis

```
String com.example.GuiActionsTest.songArtis = "Linkin Park"  [package]
```

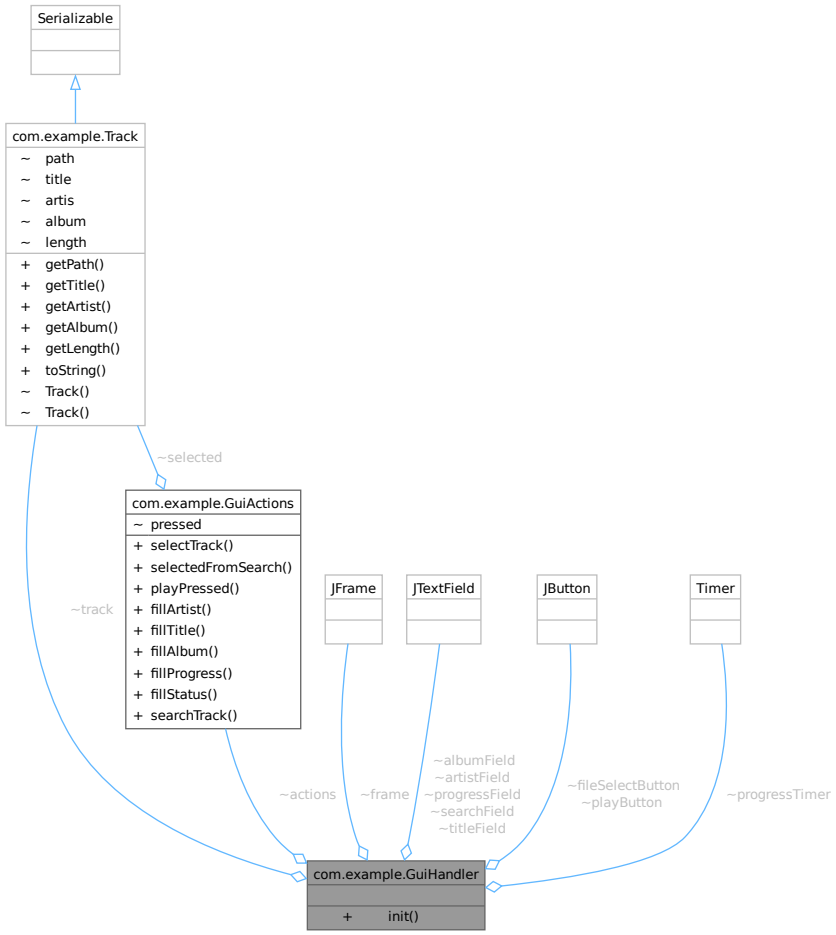
Definition at line 22 of file GuiActionsTest.java.

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

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

## 4.7 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

### 4.7.1 Detailed Description

Definition at line 24 of file GuiHandler.java.

### 4.7.2 Member Function Documentation

#### 4.7.2.1 init()

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

Definition at line 32 of file GuiHandler.java.

```
00032                       {
00033         initFrame();
00034         initComponents();
00035         configureProgressTimer();
00036     }
```

### 4.7.3 Member Data Documentation

#### 4.7.3.1 actions

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

Definition at line 25 of file GuiHandler.java.

#### 4.7.3.2 albumField

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

Definition at line 28 of file GuiHandler.java.

#### 4.7.3.3 artistField

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

Definition at line 28 of file GuiHandler.java.

#### 4.7.3.4 fileSelectButton

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

Definition at line 29 of file GuiHandler.java.

#### 4.7.3.5 frame

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

Definition at line 27 of file GuiHandler.java.

#### 4.7.3.6 playButton

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

Definition at line 29 of file GuiHandler.java.

#### 4.7.3.7 progressField

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

Definition at line 28 of file GuiHandler.java.

#### 4.7.3.8 progressTimer

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

Definition at line 30 of file GuiHandler.java.

#### 4.7.3.9 searchField

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

Definition at line 28 of file GuiHandler.java.

#### 4.7.3.10 titleField

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

Definition at line 28 of file GuiHandler.java.

#### 4.7.3.11 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

## 4.8 com.example.Track Class Reference

Inheritance diagram for com.example.Track:

Collaboration diagram for com.example.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

### 4.8.1 Detailed Description

Definition at line 5 of file Track.java.

### 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 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 13 of file Track.java.

```
00013                                                                   {
00014         this.path = path;
00015         this.title = title;
00016         this.artis = artist;
00017         this.album = album;
00018         this.length = length;
00019     }
```

#### 4.8.2.2 Track() [2/2]

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

Paraméter nélküli konstruktor.

Definition at line 22 of file Track.java.

```
00022           {
00023         this.path = null;
00024         this.title = null;
00025         this.artis = null;
00026         this.album = null;
00027         this.length = 0;
00028     }
```

### 4.8.3 Member Function Documentation

#### 4.8.3.1 getAlbum()

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

Definition at line 48 of file Track.java.

```
00048                               {
00049         return album;
00050     }
```

#### 4.8.3.2 getArtist()

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

Definition at line 44 of file Track.java.

```
00044                                {
00045         return artis;
00046     }
```

#### 4.8.3.3 getLength()

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

Definition at line 52 of file Track.java.

```
00052                               {
00053         return length;
00054     }
```

#### 4.8.3.4 getPath()

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

The fileHandler class Readdi method sets this attribute

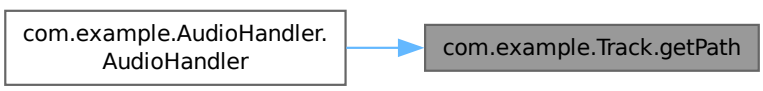
**Returns**

path absolute path to the song

Definition at line 36 of file Track.java.

```
00036                               {
00037         return path;
00038     }
```

Here is the caller graph for this function:



#### 4.8.3.5 getTitle()

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

Definition at line 40 of file Track.java.

```
00040                               {
00041         return title;
00042     }
```

#### 4.8.3.6 toString()

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

Definition at line 57 of file Track.java.

```
00057                                {
00058         return "title=" + title + "', artist='" + artis + " ";
00059     }
```

### 4.8.4 Member Data Documentation

#### 4.8.4.1 album

```
String com.example.Track.album  [package]
```

Definition at line 9 of file Track.java.

#### 4.8.4.2 artis

```
String com.example.Track.artis  [package]
```

Definition at line 8 of file Track.java.

#### 4.8.4.3 length

```
long com.example.Track.length  [package]
```

Definition at line 10 of file Track.java.

#### 4.8.4.4 path

```
String com.example.Track.path  [package]
```

Definition at line 6 of file Track.java.

#### 4.8.4.5 title

```
String com.example.Track.title  [package]
```

Absolute path that is set by the FileHandler.readDir()

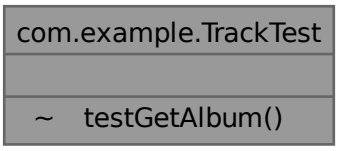
Definition at line 7 of file Track.java.

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

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

## 4.9 com.example.TrackTest Class Reference

Collaboration diagram for com.example.TrackTest:




**Package Functions**

- void testGetAlbum () throws UnsupportedTagException, InvalidDataException, IOException

### 4.9.1 Detailed Description

Definition at line 12 of file TrackTest.java.

### 4.9.2 Member Function Documentation

#### 4.9.2.1 testGetAlbum()

```
void com.example.TrackTest.testGetAlbum () throws UnsupportedTagException, InvalidDataException,
IOException  [package]
```

Definition at line 14 of file TrackTest.java.

```
00014                                                                                {
00015         String song = "src/test/resources/mockData/Linkin-1.mp3";
00016         FileHandler fh = new FileHandler();
00017         Track track = fh.createTrackFromPath(song);
00018         String album = track.getAlbum();
00019
00020         assertEquals("Meteora", album);
00021
00022     }
```

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

- src/test/java/com/example/TrackTest.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     Track track;
00014
00019     public AudioHandler(Track track) {
00020         String path = track.getPath();
00021         this.track = track;
00022         Media media = new Media(new File(path).toURI().toString());
00023         mediaPlayer = new MediaPlayer(media);
00024     }
```

```
00025
00030     public void play() {
00031         mediaPlayer.play();
00032         System.out.println("Playing...");
00033     }
00034
00038     public void pause() {
00039         mediaPlayer.pause();
00040         System.out.println("Paused.");
00041     }
00042
00043     public MediaPlayer getMediaPlayer(){
00044         return mediaPlayer;
00045     }
00046
00047     public Track getTrackFromAH(){
00048         return track;
00049     }
00050 } // END OF AUDIOHANDLER
```

## 5.3 FileHandler.java

```
00001 package com.example;
00002
00003 import java.io.File;
00004 import java.io.FileReader;
00005 import java.io.FileWriter;
00006 import java.io.IOException;
00007 import java.lang.reflect.Type;
00008 import java.util.ArrayList;
00009
00010 import com.google.gson.Gson;
00011 import com.google.gson.GsonBuilder;
00012 import com.google.gson.reflect.TypeToken;
00013 import com.mpatric.mp3agic.ID3v1;
00014 import com.mpatric.mp3agic.ID3v2;
00015 import com.mpatric.mp3agic.InvalidDataException;
00016 import com.mpatric.mp3agic.Mp3File;
00017 import com.mpatric.mp3agic.UnsupportedTagException;
00018
00022 public class FileHandler {
00023
00024     // Ebben a kollekcióban  tárolom a dalokat
00025     public ArrayList<Track> trackList = new ArrayList<>();
00026
00028     private ArrayList<Track> toBeSerialized = new ArrayList<>();
00029
00035     public void readDir(String path) throws IOException, UnsupportedTagException,
00036      InvalidDataException{
00037         File dir = new File(path);
00038         File[] listOfFiles = dir.listFiles();
00039         String name;
00040
00041         if(listOfFiles != null){
00042             for(int i = 0; i < listOfFiles.length; i++){
00043                 name = listOfFiles[i].getName();
00044
00045                 if(listOfFiles[i].isFile() && name.endsWith(".mp3")){
00046                     Track track = createTrackFromPath(listOfFiles[i].getAbsolutePath());
00047                     trackList.add(track);
00048                 }
00049             } // end of for loop
00050         }
00051         // Error handling
00052         else System.out.println("Hiba a mappa pásztázásakor");
00053     } // end of readDir()
00054
00055
00063      public Track createTrackFromPath(String path) throws IOException, UnsupportedTagException,
00064      InvalidDataException {
00065         // Inicializálás
00066         Mp3File mp3File = new Mp3File(path);
00067         long length = 0;
00068         String artist = null;
00069         String title = null;
00070         String album = null;
00071
00072         // Megnézem, hogy milyen típusú tagjei vannak
00073         if(mp3File.hasId3v1Tag()){
00074             ID3v1 v1Tag = mp3File.getId3v1Tag();
00075             length = mp3File.getLengthInSeconds();
00076             artist = v1Tag.getArtist();
00077             title = v1Tag.getTitle();
```

```
00078             album = v1Tag.getAlbum();
00079         }
00080         else if(mp3File.hasId3v2Tag()){
00081             ID3v2 v2Tag = mp3File.getId3v2Tag();
00082             length = mp3File.getLengthInSeconds();
00083             artist = v2Tag.getArtist();
00084             title = v2Tag.getTitle();
00085             album = v2Tag.getAlbum();
00086         }
00087         else System.err.println("Se v1 se v2 tagjei nincsenek a fájlnak.");
00088 
00089         // Trakc konstruálás, lényegi rész
00090         return new Track(path, title, artist, album, length);
00091     } // end of createTrackFromPath()
00092 
00093     // /**
00094     //  * @param pattern keresett cím stringben megadva
00095     //  * @return megtalált dal/dalok kollekciója
00096     //  * TODO toLoweCase()
00097     //  * Nem teljesen értem mi a retek ez a kód, de lambda
00098     //  */
00099     // public ArrayList<Track> search(String pattern) {
00100     //     return trackList.stream()
00101     //             .filter(track ->
00102     //                 track.getTitle().contains(pattern)
00103     //                 || track.getArtist().contains(pattern))
00104     //             .collect(Collectors.toCollection(ArrayList::new));
00105     // } // end of search()
00106 
00107 
00117     public ArrayList<Track> searchTracks(String pattern, String path) throws IOException,
      UnsupportedTagException, InvalidDataException {
00118         trackList.clear();
00119         readDir(path);
00120 
00121         String lowerCasepattern = pattern.toLowerCase();
00122 
00123         ArrayList<Track> matchingTracks = new ArrayList<>();
00124 
00125         for (Track t : trackList) {
00126             if (t.getTitle().toLowerCase().contains(lowerCasepattern) ||
00127                 t.getArtist().toLowerCase().contains(lowerCasepattern)) {
00128                 matchingTracks.add(t);
00129             }
00130         }
00131 
00132         return matchingTracks;
00133     }
00134 
00140     public void write(AudioHandler ah, String path) {
00141         toBeSerialized.add(ah.getTrackFromAH());
00142 
00143         Gson gson = new GsonBuilder().setPrettyPrinting().create(); // For formatted JSON
00144 
00145         try (FileWriter writer = new FileWriter(path)) {
00146             gson.toJson(toBeSerialized, writer);
00147             System.out.println("Written out");
00148         } catch (IOException e) {
00149         e.printStackTrace();
00150         }
00151     }
00152     /*
00153      * DEBUG ONLY
00154      */
00155     public ArrayList<Track> read(String path) {
00156         Gson gson = new Gson();
00157         ArrayList<Track> tracks = new ArrayList<>();
00158 
00159         try (FileReader reader = new FileReader(path)) {
00160             Type listType = new TypeToken<ArrayList<Track>>() {}.getType();
00161             tracks = gson.fromJson(reader, listType);
00162         } catch (IOException e) {
00163             e.printStackTrace();
00164         }
00165 
00166         // tracks.forEach(track -> System.out.println(track.getTitle()));
00167         return tracks;
00168     }
00169 
00170 
00171 } // 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             fileHandler.write(audioHandler, "playedTracks.json"); // szerializáció
00052
00053             return selected;
00054         }
00055
00056         return null;
00057     }
00058
00059     public void selectedFromSearch(Track track){
00060         selected = track;
00061         audioHandler = new AudioHandler(track);
00062         fileHandler.write(audioHandler, "playedTracks.json"); // szerializáció
00063     }
00064
00065     public void playPressed(){
00066         if (!pressed) {
00067             audioHandler.play();
00068             pressed = true;
00069         }
00070         else {
00071             audioHandler.pause();
00072             pressed = false;
00073         }
00074     }
00075
00076     public void fillArtist(JTextField field){
00077         if(selected != null){
00078              String artist = selected.getArtist();
00079              field.setText(artist);
00080              System.out.println("Artist filed filled");
00081         }
00082         else field.setText("NA");
00083    }
00084
00085     public void fillTitle(JTextField field){
00086         if(selected != null){
00087              String artist = selected.getTitle();
00088              field.setText(artist);
00089              System.out.println("Titke filed filled");
00090         }
00091         else field.setText("NA");
00092    }
00093
00094     public void fillAlbum(JTextField field){
00095         if(selected != null){
00096              String album = selected.getAlbum();
00097              field.setText(album);
```

```
00098               System.out.println("Album filed filled");
00099           }
00100           else field.setText("NA");
00101       }
00102
00103       public void fillProgress(JTextField field){
00104           if(selected != null){
00105               long len = selected.getLength();
00106               int curr = (int) audioHandler.getMediaPlayer().getCurrentTime().toSeconds();
00107               field.setText(curr + "/" + len);
00108               // System.out.println("Length of the track: " + len);
00109           }
00110           else field.setText("NA");
00111       }
00112
00113       public String fillStatus(){
00114           if(selected != null){
00115               String status = audioHandler.getMediaPlayer().getStatus().toString();
00116               System.out.println("Status filled");
00117               return status;
00118           }
00119           return "Not selected";
00120       }
00121
00122       public ArrayList<Track> searchTrack(String pattern) throws UnsupportedTagException,
      InvalidDataException, IOException {
00123           return fileHandler.searchTracks(pattern,
      "/home/i3hunor/Suli/Prog3/nagyHF/Fasz/mp3java/src/main/resources");
00124       }
00125
00126
00127 }
```

## 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         JButton selectButton = new JButton("Select");
00228         gbc.gridy = 1;
00229         gbc.weighty = 0.0;
00230         gbc.fill = GridBagConstraints.HORIZONTAL;
00231         resultsFrame.add(selectButton, gbc);
00232
00233         // Add functionality to select a track and close the results window
00234         selectButton.addActionListener(e -> {
00235             int selectedIndex = resultsList.getSelectedIndex();
00236             if (selectedIndex >= 0) {
00237                 track = results.get(selectedIndex);
00238                 actions.selectedFromSearch(track);
00239                 initComponents();
00240                 resultsFrame.dispose();
00241             }
00242         });
00243
00244         resultsFrame.setVisible(true);
00245     }
00246
00255     private void searchTrack(String pattern) throws UnsupportedTagException, InvalidDataException,
      IOException {
00256         System.out.println("Searching for: " + pattern);
00257         ArrayList<Track> tracks = actions.searchTrack(pattern);
00258
00259         if (tracks.isEmpty()) {
00260             System.out.println("No results found.");
00261         } else {
00262             showSearchResultsWindow(tracks);
00263         }
```

```
00264     }
00265
00266 } // end of GuiHandler
```

## 5.6 Track.java

```
00001 package com.example;
00002
00003 import java.io.Serializable;
00004
00005 public class Track implements Serializable {
00006     String path;
00007     String title;
00008     String artis;
00009     String album;
00010     long length;
00011
00013     Track(String path, String title, String artist, String album, long length){
00014         this.path = path;
00015         this.title = title;
00016         this.artis = artist;
00017         this.album = album;
00018         this.length = length;
00019     }
00020
00022     Track(){
00023         this.path = null;
00024         this.title = null;
00025         this.artis = null;
00026         this.album = null;
00027         this.length = 0;
00028     }
00029
00030     // Getterek
00031
00036     public String getPath(){
00037         return path;
00038     }
00039
00040     public String getTitle(){
00041         return title;
00042     }
00043
00044     public String getArtist(){
00045         return artis;
00046     }
00047
00048     public String getAlbum(){
00049         return album;
00050     }
00051
00052     public long getLength(){
00053         return length;
00054     }
00055
00056     @Override
00057     public String toString() {
00058         return "title=" + title + "', artist='" + artis + " ";
00059     }
00060
00061
00062
00063 } // end of Track class
00064
00065
```

## 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, com.google.gson; // Allow Gson access to reflection
00006     exports com.example;
00007
00008     // chatgpt
00009     requires javafx.media;
00010     requires mp3agic;
00011
```

```
00012     // Swing + awt
00013     requires java.desktop;
00014
00015     // Gson
00016     requires com.google.gson; // Ensure Gson is explicitly required
00017 }
```

## 5.8 FileHandlerTest.java

```
00001 package com.example;
00002
00003 import static org.junit.jupiter.api.Assertions.assertEquals;
00004 import static org.junit.jupiter.api.Assertions.assertNotNull;
00005 import static org.junit.jupiter.api.Assertions.assertTrue;
00006
00007 import java.io.IOException;
00008 import java.util.ArrayList;
00009
00010 import org.junit.jupiter.api.BeforeAll;
00011 import org.junit.jupiter.api.Test;
00012
00013 import com.mpatric.mp3agic.InvalidDataException;
00014 import com.mpatric.mp3agic.UnsupportedTagException;
00015
00016 import javafx.application.Platform;
00017
00018 public class FileHandlerTest {
00019
00020    @BeforeAll
00021 static void setupJavaFx() {
00022     if (!Platform.isImplicitExit()) {
00023         Platform.startup(() -> {});
00024     }
00025 }
00026
00027
00033     @Test
00034     void testCreateTrackFromPath() throws Exception {
00035         String song = "src/test/resources/mockData/Linkin-1.mp3";
00036         FileHandler fh = new FileHandler();
00037         Track track = fh.createTrackFromPath(song);
00038         String artist = track.getArtist();
00039
00040         assertNotNull(track);
00041         assertEquals("Linkin Park", artist);
00042     }
00043
00044     @Test
00045     void testReadDir() throws UnsupportedTagException, InvalidDataException, IOException {
00046         String dir = "src/test/resources/mockData/";
00047         FileHandler fh = new FileHandler();
00048         fh.readDir(dir);
00049
00050         assertEquals(2, fh.trackList.size());
00051         assertTrue(fh.trackList.stream().anyMatch(track -> track.getPath().endsWith("Linkin-1.mp3")));
00052         assertTrue(fh.trackList.stream().anyMatch(track -> track.getPath().endsWith("Linkin-2.mp3")));
00053     }
00054
00055     @Test
00056     void testSearchTracks() throws UnsupportedTagException, InvalidDataException, IOException {
00057         String pattern = "Linkin";
00058         String path = "src/test/resources/mockData/";
00059         FileHandler fh = new FileHandler();
00060         ArrayList<Track> list;
00061         list = fh.searchTracks(pattern, path);
00062
00063         assertNotNull(list);
00064         assertEquals(2, list.size());
00065         assertTrue(list.stream().anyMatch(track -> track.getArtist().contains("Linkin")));
00066     }
00067
00068     @Test
00069     void testWriteAndReadSerialization() throws UnsupportedTagException, InvalidDataException,
      IOException {
00070         FileHandler fileHandler = new FileHandler();
00071         String testFile = "serialTest.json";
00072
00073         String songPath = "src/test/resources/mockData/Linkin-1.mp3";
00074         FileHandler fh = new FileHandler();
00075         Track track = fh.createTrackFromPath(songPath);
00076         AudioHandler ah = new AudioHandler(track);
00077
00078         fileHandler.write(ah, testFile);
```

```
00079
00080         ArrayList<Track> readTracks = fileHandler.read(testFile);
00081
00082         assertNotNull(readTracks);
00083         assertEquals(1, readTracks.size());
00084         assertEquals("Linkin Park", readTracks.get(0).getArtist());
00085     }
00086
00087 }
```

## 5.9 GuiActionsTest.java

```
00001 package com.example;
00002
00003 import static org.junit.jupiter.api.Assertions.assertEquals;
00004 import static org.junit.jupiter.api.Assertions.assertFalse;
00005 import static org.junit.jupiter.api.Assertions.assertTrue;
00006
00007 import java.io.IOException;
00008 import java.util.ArrayList;
00009
00010 import javax.swing.JTextField;
00011
00012 import org.junit.jupiter.api.BeforeAll;
00013 import org.junit.jupiter.api.Test;
00014
00015 import com.mpatric.mp3agic.InvalidDataException;
00016 import com.mpatric.mp3agic.UnsupportedTagException;
00017
00018 import javafx.application.Platform;
00019
00020 public class GuiActionsTest {
00021     String song1Path = "src/test/resources/mockData/Linkin-1.mp3";
00022     String songArtis = "Linkin Park";
00023     String songAlbum = "Meteora";
00024     FileHandler fh = new FileHandler();
00025     GuiActions ga = new GuiActions();
00026
00027     // Fogalam sincs mi akar ez lenni de működik
00028     @BeforeAll
00029     static void setupJavaFx() {
00030         Platform.startup(() -> {});
00031     }
00032
00033     @Test
00034     void testFillAlbum() throws UnsupportedTagException, InvalidDataException, IOException {
00035         Track track = fh.createTrackFromPath(song1Path);
00036         ga.selectedFromSearch(track);
00037
00038         JTextField albumField = new JTextField();
00039         ga.fillAlbum(albumField);
00040
00041         assertEquals(songAlbum, albumField.getText());
00042
00043     }
00044
00045     @Test
00046     void testFillArtist() throws UnsupportedTagException, InvalidDataException, IOException {
00047         Track track = fh.createTrackFromPath(song1Path);
00048         ga.selectedFromSearch(track);
00049
00050         JTextField artistField = new JTextField();
00051         ga.fillArtist(artistField);
00052
00053         assertEquals(songArtis, artistField.getText());
00054     }
00055
00056     @Test
00057     void testPlayPressed() throws UnsupportedTagException, InvalidDataException, IOException {
00058         Track track = fh.createTrackFromPath(song1Path);
00059         ga.selectedFromSearch(track);
00060
00061         ga.playPressed();
00062         assertTrue(ga.pressed);
00063
00064         ga.playPressed();
00065         assertFalse(ga.pressed);
00066     }
00067
00068
00069     @Test
00070     void testSearchTrack() throws Exception {
00071         ArrayList<Track> tracks = ga.searchTrack(songArtis);
```

```
00072
00073         assertFalse(tracks.isEmpty());
00074         assertTrue(tracks.stream().anyMatch(track -> track.getArtist().equalsIgnoreCase(songArtis)));
00075     }
00076
00077     @Test
00078     void testSelectedFromSearch() throws UnsupportedTagException, InvalidDataException, IOException {
00079         Track track = fh.createTrackFromPath("src/test/resources/mockData/Linkin-1.mp3");
00080
00081         ga.selectedFromSearch(track);
00082
00083         assertEquals(songArtis, track.getArtist());
00084         assertEquals(songAlbum, track.getAlbum());
00085
00086     }
00087 }
```

## 5.10 TrackTest.java

```
00001 package com.example;
00002
00003 import static org.junit.jupiter.api.Assertions.assertEquals;
00004
00005 import java.io.IOException;
00006
00007 import org.junit.jupiter.api.Test;
00008
00009 import com.mpatric.mp3agic.InvalidDataException;
00010 import com.mpatric.mp3agic.UnsupportedTagException;
00011
00012 public class TrackTest {
00013     @Test
00014     void testGetAlbum() throws UnsupportedTagException, InvalidDataException, IOException {
00015         String song = "src/test/resources/mockData/Linkin-1.mp3";
00016         FileHandler fh = new FileHandler();
00017         Track track = fh.createTrackFromPath(song);
00018         String album = track.getAlbum();
00019
00020         assertEquals("Meteora", album);
00021
00022     }
00023 }
```

# Index