D20 Project January 2021

My-music-player.

Content

Learning Goals and objectives in the project

During this project, the learning is focused on:
Applying key techniques and tools specific for this discipline for modelling of IT systems at the level of analysis and design. Applying principles and techniques of relevance to the profession to design user interfaces. Using an appropriate software architecture. Managing development focused situations using systems development methods and relevant techniques like version control. Participating in a competent manner in technical and multidisciplinary systems development projects. Using key facilities in the programming language to realize algorithms, design patterns, abstract data types, data structures, design models and user interfaces.

Project Basis

Duration: From Monday January 4 at 9.00 to Friday January 22 at 14.00, where you hand-in on moodle.
Groups: You must work together in groups of 3-4 students.
Any other group sizes must be approved by a teacher. You decide the groups yourself.
There are no normal classes during the project period. Tutors and teachers will be available during the project period – a schedule for when you can get help will be available via moodle.

The Assignment

This assignment is about designing and constructing a JavaFX application providing administration of songs and playlists and playing songs. From the end-user's point of view, the application contains songs and playlists.

Technical Requirements

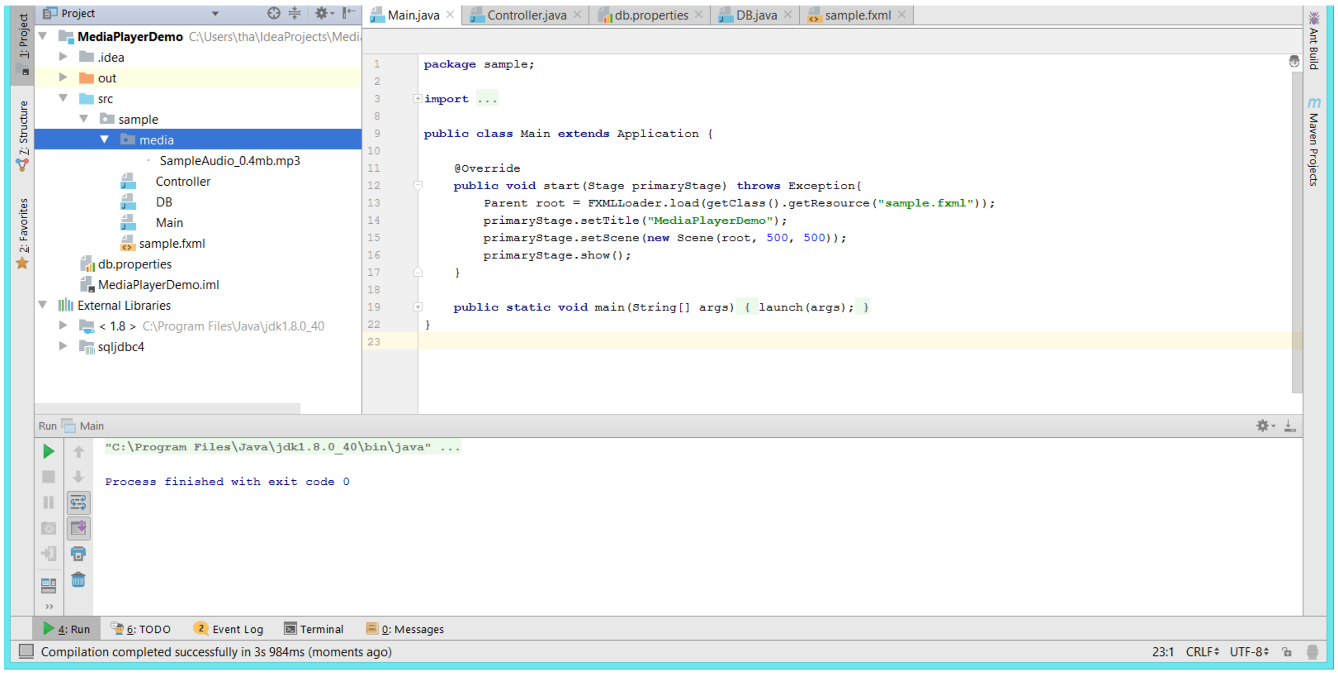
The application must be a desktop application with a GUI

- The whole project is developed in IntelliJ as a JavaFX project.
- The GUI designer is SceneBuilder (via the .fxml file in the project).
- The database used is SQL Server (also using the class DB.java)
- Git and GitHub is used for version control.

Project Settings

A small demo of some simple functionality and the settings in IntelliJ is located at this Git-repository:
https://github.com/tomhau/DemoMediaPlayer.git

Fig. 1 The project structure in IntelliJ.



```
MediaPlayerDemo  C:\Users\tha\IdeaProjects\Medi
  .idea
  out
  src
    sample
      media
        SampleAudio_0.4mb.mp3
      Controller
      DB
      Main
      sample.fxml
  db.properties
  MediaPlayerDemo.iml
External Libraries
  < 1.8 >  C:\Program Files\Java\jdk1.8.0_40
  sqljdbc4
```

```java
package sample;

import ...

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("MediaPlayerDemo");
        primaryStage.setScene(new Scene(root, 500, 500));
        primaryStage.show();
    }

    public static void main(String[] args) { launch(args); }
}
```

```
"C:\Program Files\Java\jdk1.8.0_40\bin\java" ...

Process finished with exit code 0
```

```
Compilation completed successfully in 3s 984ms (moments ago)
```

Notice: This is a normal JavaFX project but with a ***media folder*** with the audio-file(s)+ properties file, class DB.java and jdbc-driver like when we worked with the WaterWorks project.

The Database

The application must be able to play and store music files in the formats .wav and .mp3 and perhaps others. All songs are placed ***physically*** in the media-folder inside the project - see the project settings above where there is one actual .mp3 file called *SampleAudio_0.4mb.mp3*

The Songs Table

This table contains these fields:

1. The file path to the media folder – inspect the controller code how the path is used to load the file into the MediaPlayer object.
2. The artist/group name.
3. The title of the song.

However, notice: The actual .mp3 or wav file is not stored in the database!

The Playlist as Tables

A playlist is identified by a unique name and contains a sequence of songs in a specific order.
All playlists must also be saved in the database because then a playlist can be loaded when the program opens.
Whenever a playlist changes, the change must be reflected in the database also.

Notice: In order to model a playlist you must create two tables in the database!

Requirements to the functionality of the app

- The user must be able to choose a single song and play/pause/stop the song.
- The user must be able to choose an existing playlist and play/pause/stop this list of songs.
- The user must be able to create a new playlist (containing a number of songs).
- The user must be able to search for songs by title or by artist.
- The user must be able to edit/delete a playlist (but deleting a playlist should not delete the songs)

Design Considerations

You must as a group create your own personal design using layouts/controls of your own choice. Use icons with symbols on the buttons to make it user friendly. In order to display e.g. a list of songs, a ListView in SceneBuilder is a good control to use! Look into the documentation of javaFX on how to use lists.

Initial load of songs to the database.

Early in the process, you should place all the songs your app contains in the media-folder. (You should not pick more than 15-20). Then information about each of these songs filename, artist and title must then be inserted in the database table. This can be done from code using the DB class and some INSERT-Statements!

Non-functional requirements

The non-functional requirements are as follows:

1. The system must be built using "separation of concerns". Code must be placed in well-named methods. SceneBuilder is used for GUI design.
2. All classes and methods must be decorated with documentation.
3. A Git repository must be used and managed through the development process. You can set up collaborators in the repository so all group members can work together.

You must hand-in

One PDF-file containing the report following elements (**The code has to be on GitHub**!):
The report must contain at least the following:

1. **Front page** stating the names of the team members.
2. **State of delivery**
   A short description of the state of your delivery. Which requirements are met?
3. **Data Storage**
   Describe the data structure of your system (E/R diagram), and the folders for media files + images.
4. **Implementation details**
   In case there are issues that caused you troubles or solutions that you find smart, you must describe these. You may insert code snippets whenever you find it beneficial.
5. **Source Control**
   This section describe which system has been used for source control (Git), the name of the repository and how the team has used this through the project.
6. **Source Code**
   Remember to add a link for the teachers to your GitHub repository. Do not change the repository after hand-in.

Evaluation

As a general rule, you will have the assignment approved if all requirements are met, and the user can play songs in the app. You will get an oral feedback on the project.

/The teachers: Frank, Karsten, Tommy