

Course: CS522 - Software Quality Assurance and Test Automation
Faculty: Prof. Henry Chang

Project: Three UML diagrams for JukeBox

Name: Haripriya A
ID: 19579



NORTHWESTERN POLYTECHNIC
UNIVERSITY

Table of Contents

1. Introduction
2. About Software Development Life Cycle (SDLC)
3. UML
4. Implementing JukeBox
 - 4.1. Use Case diagram
 - 4.2. Class diagram
 - 4.3. Sequence diagram
5. Executions
6. Conclusion
7. References



Introduction

- Software Testing can be started from the Requirements Gathering phase and continued till the deployment of the software.
- For Software Quality Assurance and Test Automation one should have basic understanding of below
 - Software Development Life Cycle (SDLC).
 - Software programming using any programming language.



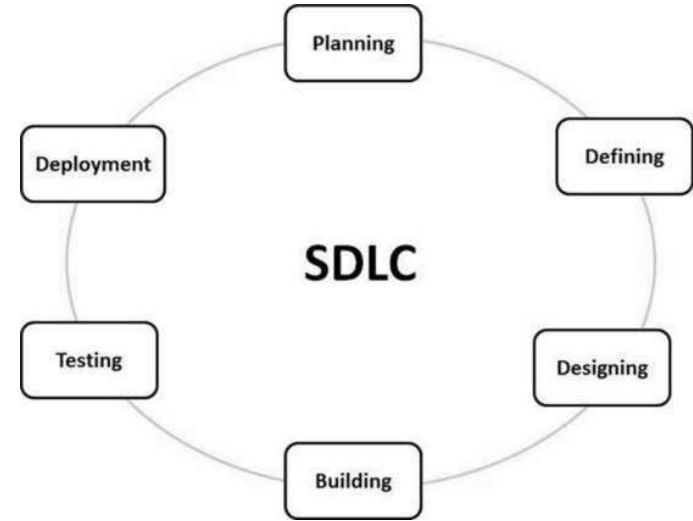
About Software Development Life Cycle

- Software Development Life Cycle depends on the development model that is being used.
 - In the Waterfall model, formal testing is conducted in the testing phase
 - In the incremental model, testing is performed at the end of every increment/iteration and the whole application is tested at the end.
- Testing is done in different forms at every phase of SDLC:
 - During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
 - Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
 - Testing performed by a developer on completion of the code is also categorized as testing.



About Software Development Life Cycle

- The life cycle defines a methodology for improving the quality of software and the overall development process. Below is the SDLC Cycle stages
 - Stage 1: Planning and Requirement Analysis
 - Stage 2: Defining Requirements
 - Stage 3: Designing the product architecture
 - Stage 4: Building or Developing the Product
 - Stage 5: Testing the Product
 - Stage 6: Deployment in the Market and Maintenance



UML

- A good policy and process for the software development effort is necessary
- Mechanisms for capturing design are needed
 - Unified Modeling Language (UML).
- The UML is the collective brainchild of Grady Booch, Ivar Jacobson, and Jim Rumbaugh; three world renown object-oriented technologists who combined their efforts to create a standard method for capturing and visualizing object-oriented designs.
- UML has been one of those buzzwords in the software community that has found itself on the same hot list as Java, XML, and .NET. UML stands as a powerful and capable method for designing software. Unfortunately, it is vastly underutilized within the software development community.



Implementation of JukeBox

- Planning: Our goal is to design a JukeBox that allows customers to select songs they want played or to submit a playlist that they have already created previously.
- Requirements: JukeBox spec
 - Allow customers to
 - select songs they want to play.
 - submit a playlist that they have already created previously.
 - To provide a mechanism for owners, record companies and artists to earn a profit. The Juke Box contains
 - A card swipe mechanism
- Design: UML Diagrams

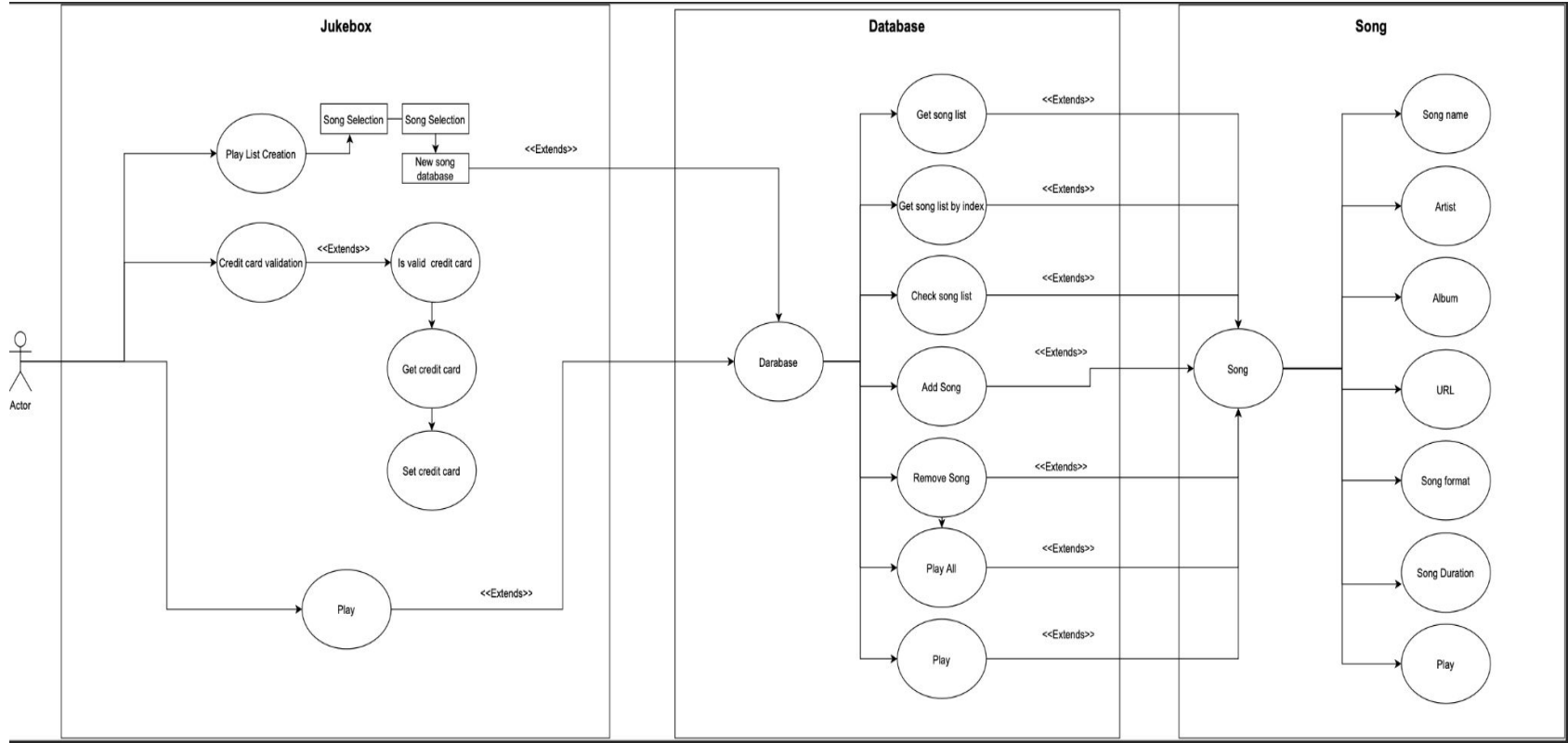


Jukebox SDLC

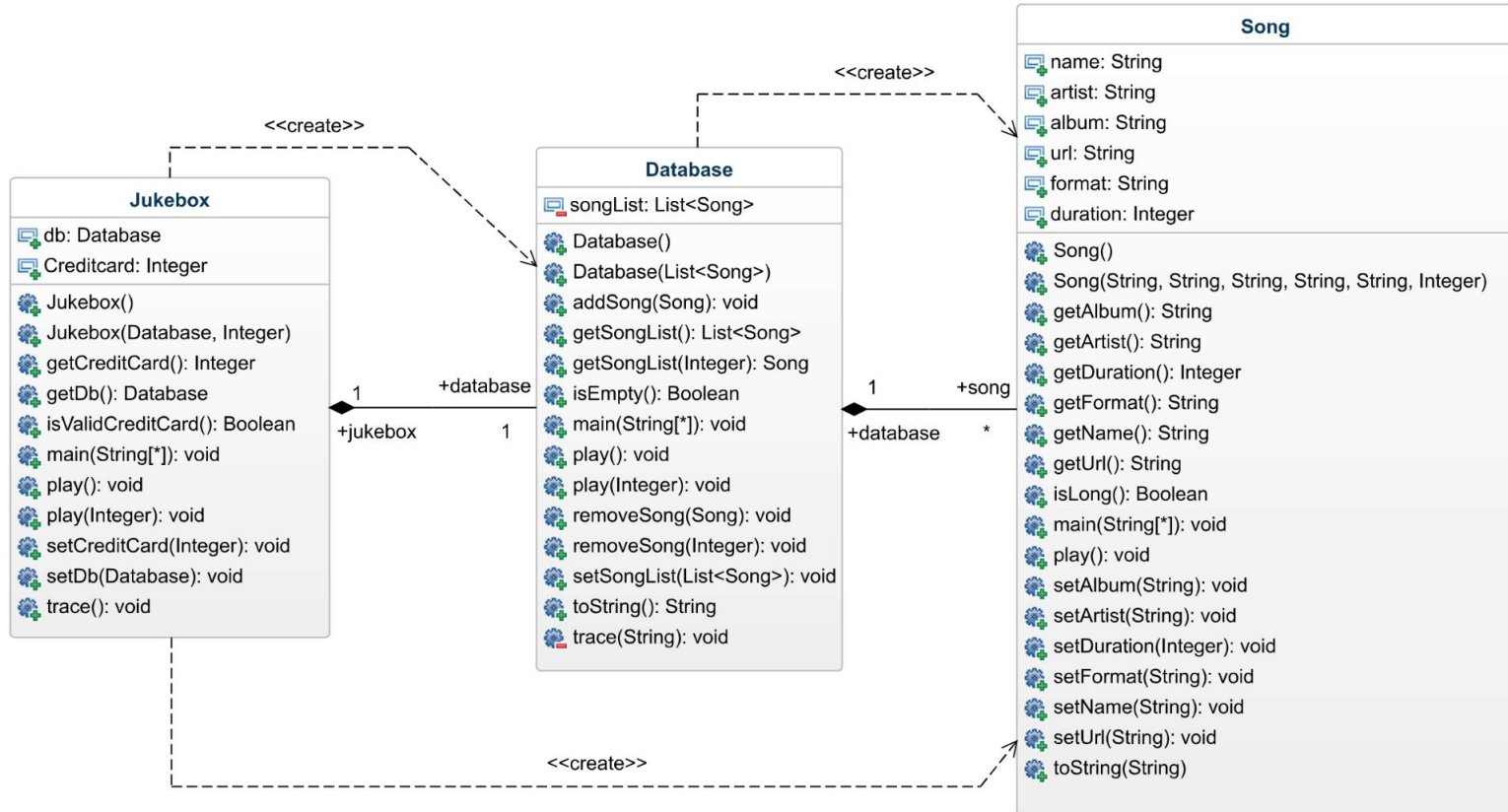
- Planning: Our goal is to design a JukeBox that allows customers to select songs they want played or to submit a playlist that they have already created previously.
- Requirements: JukeBox spec
 - Allow customers to
 - select songs they want to play.
 - submit a playlist that they have already created previously.
 - To provide a mechanism for owners, record companies and artists to earn a profit. The Juke Box contains
 - A card swipe mechanism
- Design: JukeBox UML



Use Case Diagram

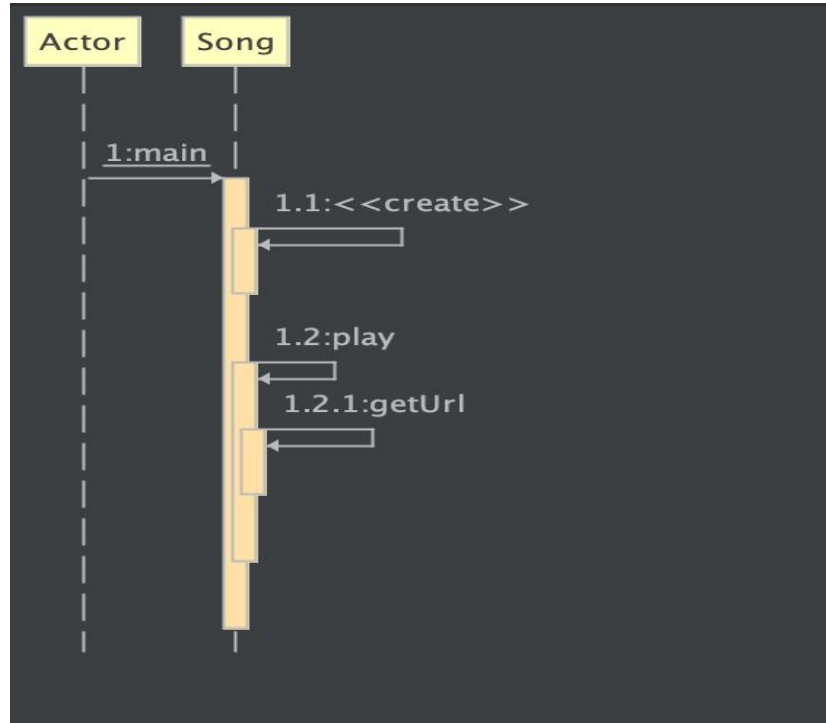


Class Diagram



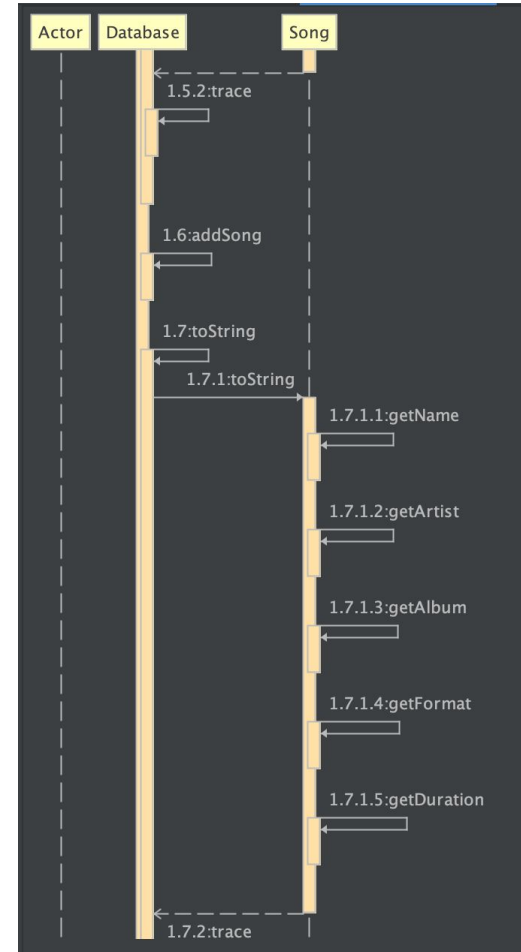
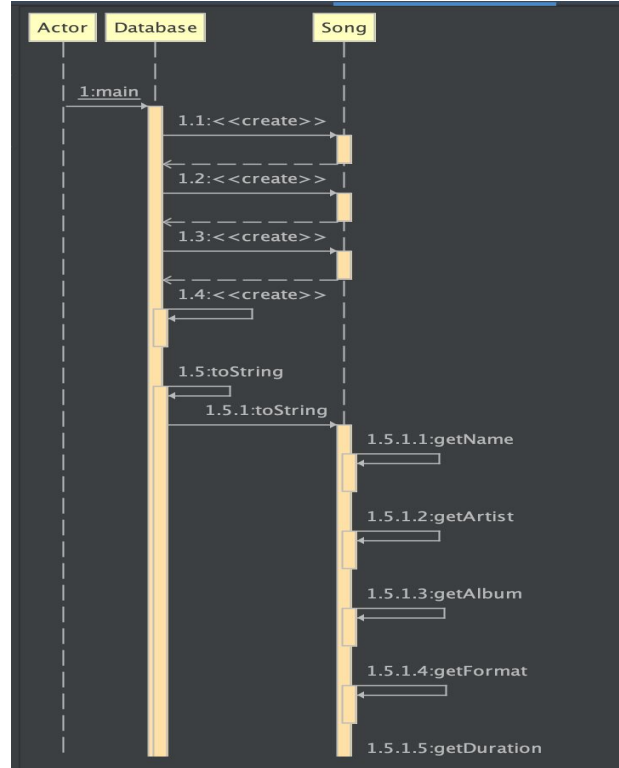
Sequence Diagram

- A Sequence Diagram is a powerful visual aid for developers in understanding the interaction of objects within a system
- Song.java



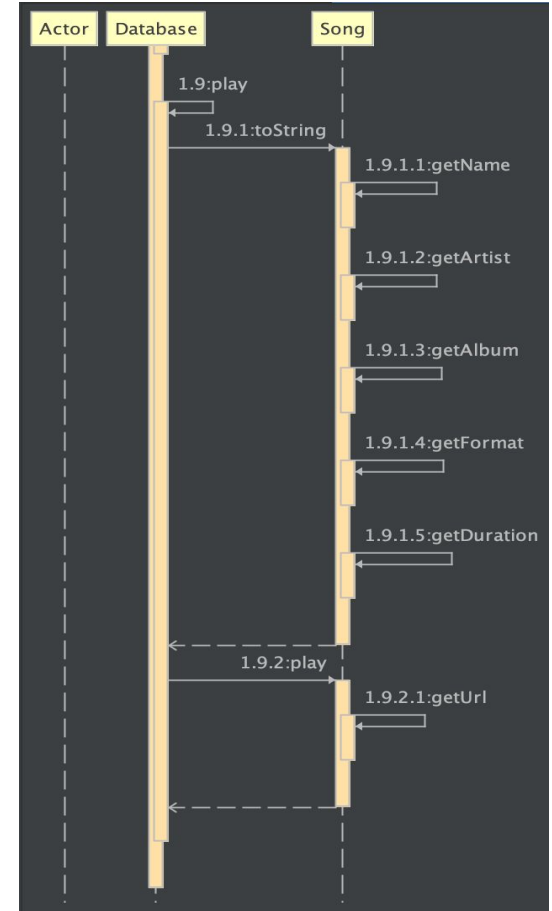
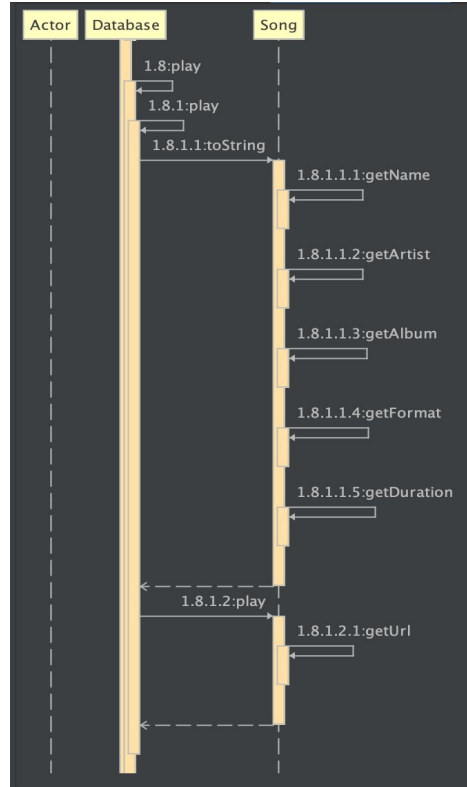
Sequence Diagram

- Database.java



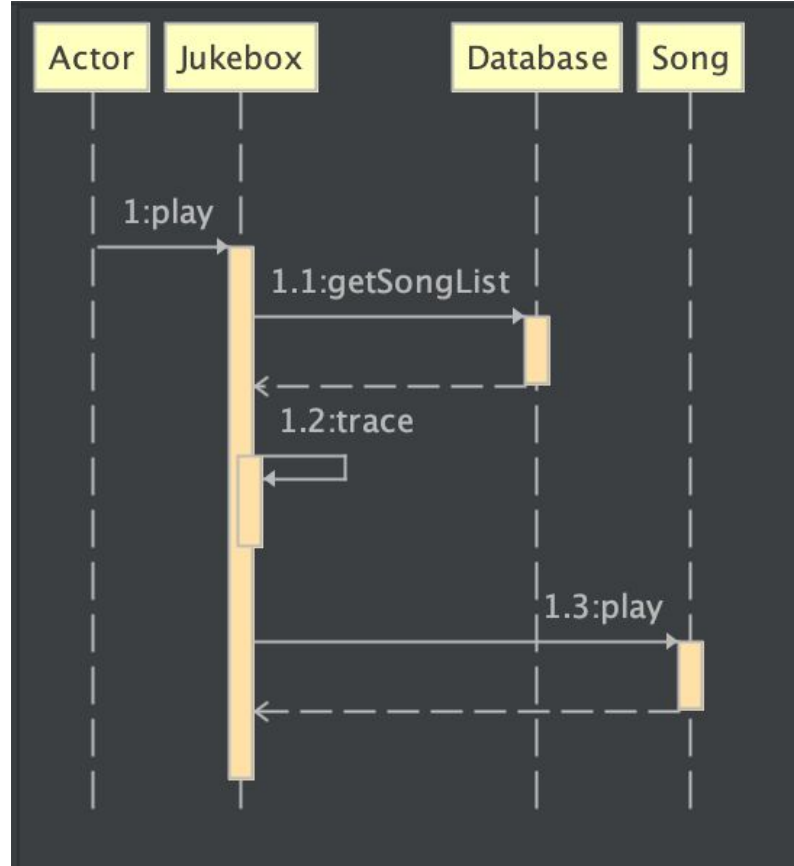
Sequence Diagram

- Database.java



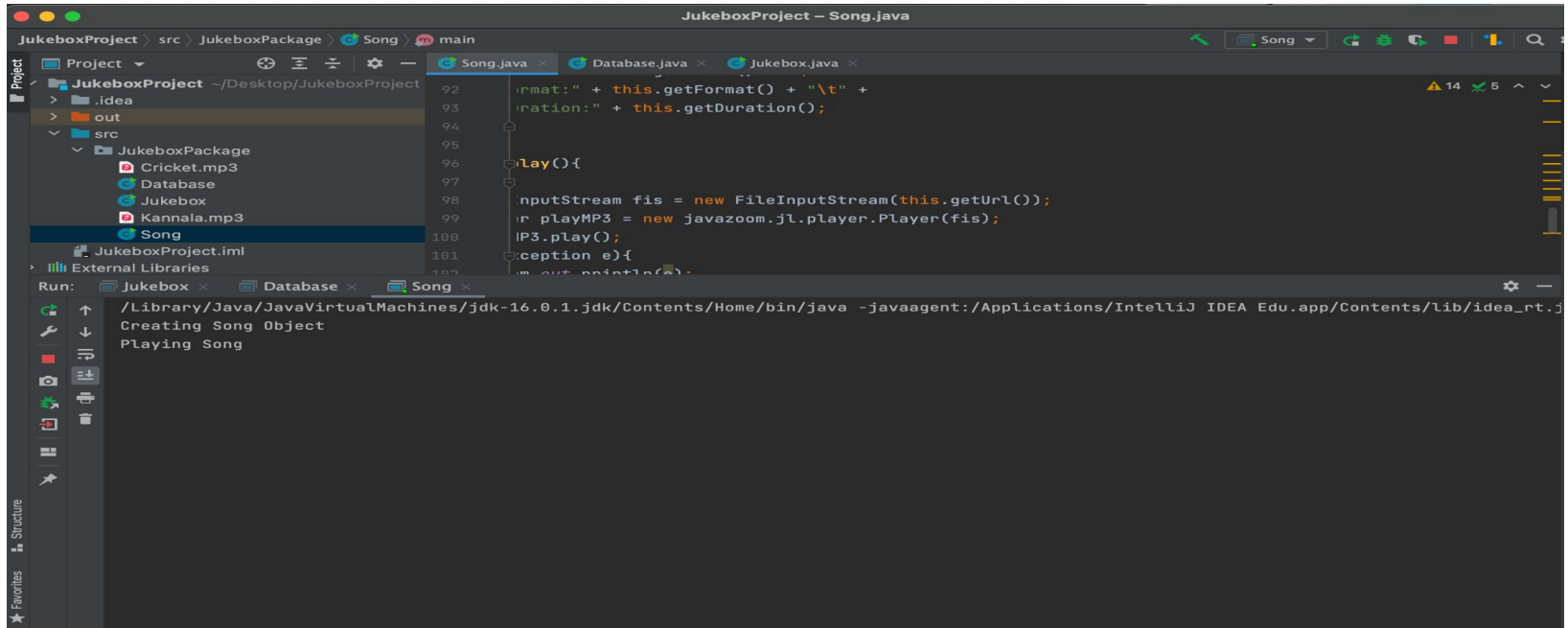
Sequence Diagram

- Jukebox.java



Program Output

- Song.java



The screenshot shows an IDE window titled "JukeboxProject - Song.java". The left sidebar displays the project structure with "JukeboxProject" at the root, containing "src" and "out" folders. Under "src", there is a "JukeboxPackage" folder containing "Cricket.mp3", "Database", "Jukebox", "Kannala.mp3", and "Song". The "Song" file is selected. The main editor shows the code for "Song.java" with the following visible lines:

```
92 format:" + this.getFormat() + "\t" +  
93 iration:" + this.getDuration();  
94  
95  
96  
97  
98  
99  
100  
101  
102
```

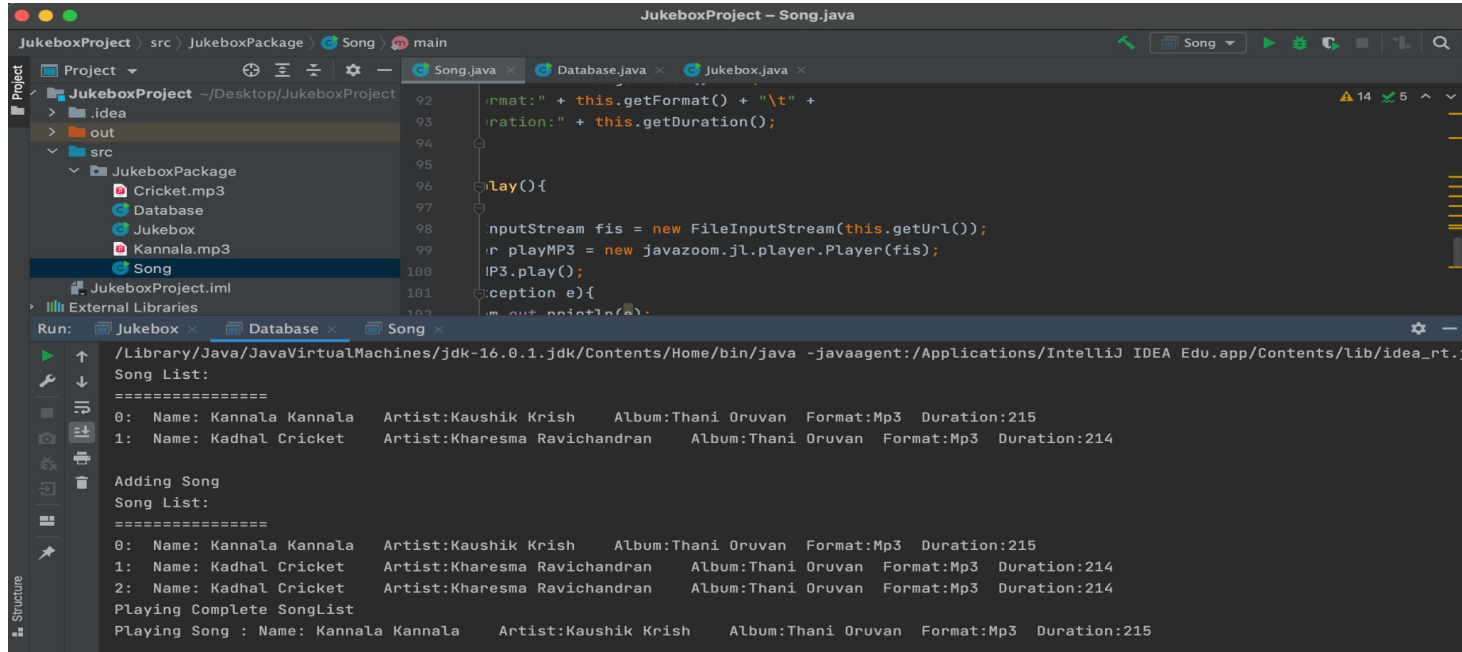
The bottom panel shows the "Run" output for the "Song" class. The output is:

```
/Library/Java/JavaVirtualMachines/jdk-16.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA Edu.app/Contents/lib/idea_rt.j  
Creating Song Object  
Playing Song
```



Program Output

- Database.java



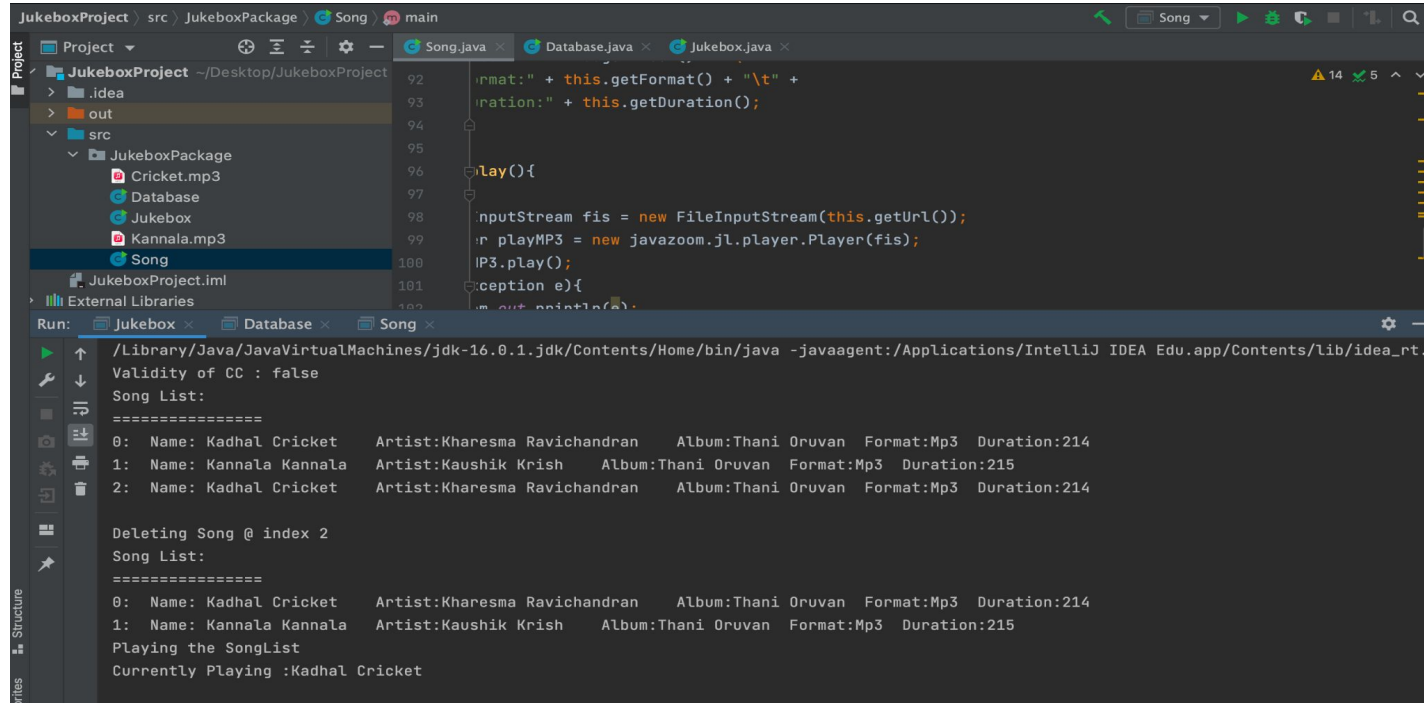
The screenshot shows an IDE window titled "JukeboxProject - Song.java". The project structure on the left includes "JukeboxProject" with subfolders ".idea", "out", and "src". The "src" folder contains "JukeboxPackage" with files "Cricket.mp3", "Database", "Jukebox", "Kannala.mp3", and "Song". The "Database.java" file is open in the editor, showing code for playing a song. The output window at the bottom displays the following text:

```
Run: Jukebox x Database x Song x
/Library/Java/JavaVirtualMachines/jdk-16.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA Edu.app/Contents/lib/idea_rt.jar
Song List:
=====
0: Name: Kannala Kannala Artist:Kaushik Krish Album:Thani Oruvan Format:Mp3 Duration:215
1: Name: Kadhal Cricket Artist:Kharesma Ravichandran Album:Thani Oruvan Format:Mp3 Duration:214
Adding Song
Song List:
=====
0: Name: Kannala Kannala Artist:Kaushik Krish Album:Thani Oruvan Format:Mp3 Duration:215
1: Name: Kadhal Cricket Artist:Kharesma Ravichandran Album:Thani Oruvan Format:Mp3 Duration:214
2: Name: Kadhal Cricket Artist:Kharesma Ravichandran Album:Thani Oruvan Format:Mp3 Duration:214
Playing Complete SongList
Playing Song : Name: Kannala Kannala Artist:Kaushik Krish Album:Thani Oruvan Format:Mp3 Duration:215
```



Program Output

- Jukebox.java



The screenshot displays the IntelliJ IDEA IDE interface. The top toolbar shows the 'Run' button (a green play icon). The 'Project' view on the left shows the 'JukeboxProject' structure, including 'src' and 'out' directories, and a 'JukeboxPackage' containing 'Cricket.mp3', 'Database', 'Jukebox', 'Kannala.mp3', and 'Song'. The 'Song.java' file is open in the editor, showing code for playing a song. The 'Run' console at the bottom shows the output of the program, which includes the song list and the current song being played.

```
Validity of CC : false
Song List:
=====
0: Name: Kadhal Cricket Artist:Kharesma Ravichandran Album:Thani Oruvan Format:Mp3 Duration:214
1: Name: Kannala Kannala Artist:Kaushik Krish Album:Thani Oruvan Format:Mp3 Duration:215
2: Name: Kadhal Cricket Artist:Kharesma Ravichandran Album:Thani Oruvan Format:Mp3 Duration:214

Deleting Song @ index 2
Song List:
=====
0: Name: Kadhal Cricket Artist:Kharesma Ravichandran Album:Thani Oruvan Format:Mp3 Duration:214
1: Name: Kannala Kannala Artist:Kaushik Krish Album:Thani Oruvan Format:Mp3 Duration:215
Playing the SongList
Currently Playing :Kadhal Cricket
```



Conclusion

- Hence the Jukebox UML - use case, class and sequence diagram has been shown in the above slides which is part of the Jukebox design.



References

- https://npu85.npu.edu/~henry/npu/classes/qa/sdlc_tutorialspoint/slide/index_slide.html
- https://npu85.npu.edu/~henry/npu/classes/qa/qa_tutorialspoint/slide/index_slide.html
- https://npu85.npu.edu/~henry/npu/classes/oo/uml_tutorial/slide/index_slide.html

Google slides URL :

<https://docs.google.com/presentation/d/1jqj0RkhkyCa-nfRoOJWqFgxJqdIm9Gx9WZ8I-H7MYls/edit?usp=sharing>

