Jukebox UML diagrams

Use Case diagram, Class Diagram and Sequence Diagram

Presented by
Sai Harshinee Roopakula
19577

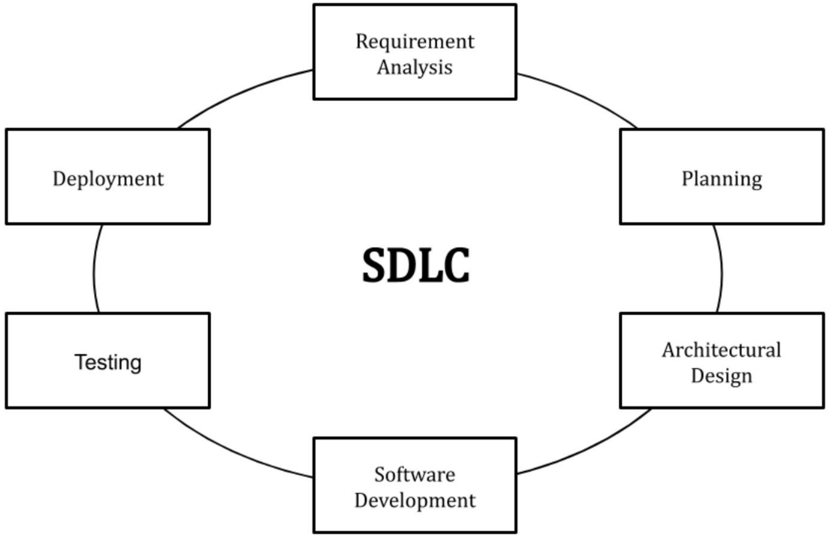
Table of Contents

Introduction - SDLC

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use.

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. in detail, the SDLC methodology focuses on the following phases of software development:

- Requirement analysis
- Planning
- Software design such as architectural design
- Software development
- Testing
- Deployment



Introduction - SDLC

The most common SDLC examples or SDLC models are listed below.
Waterfall Model - This SDLC model is the oldest and most straightforward. With this methodology, we finish one phase and then start the next. Each phase has its own mini-plan and each phase "waterfalls" into the next. The biggest drawback of this model is that small details left incomplete can hold up the entire process.
Agile Model
The Agile SDLC model separates the product into cycles and delivers a working product very quickly. This methodology produces a succession of releases. Testing of each release feeds back info that's incorporated into the next version.
Iterative Model
This SDLC model emphasizes repetition. Developers create a version very quickly and for relatively little cost, then test and improve it through rapid and successive versions.
V-Shaped Model
An extension of the waterfall model, this SDLC methodology tests at each stage of development. As with waterfall, this process can run into roadblocks.
Big Bang Model
This high-risk SDLC model throws most of its resources at development and works best for small projects. It lacks the thorough requirements definition stage of the other methods.
Spiral Model
The most flexible of the SDLC models, the spiral model is similar to the iterative model in its emphasis on repetition. The spiral model goes through the planning, design, build and test phases over and over, with gradual improvements at each pass.

Introduction - UML diagram

Unified Modeling Language (UML) - It is a way to visually represent the architecture, design, and implementation of complex software systems. When you're writing code, there are thousands of lines in an application, and it's difficult to keep track of the relationships and hierarchies within a software system. UML diagrams divide that software system into components and subcomponents.

UML diagrams can help engineering teams:

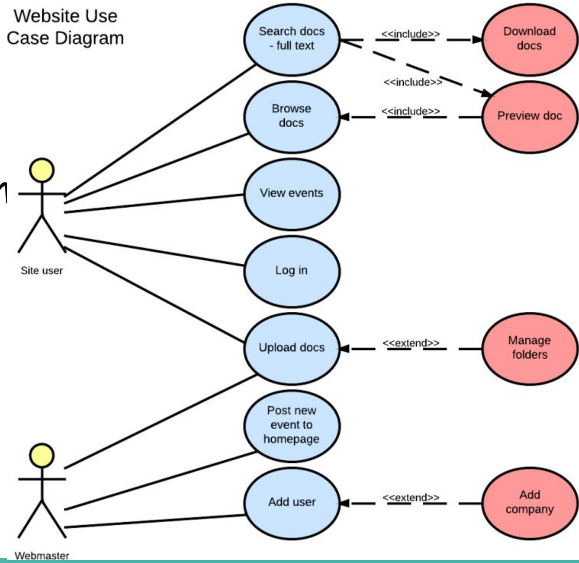
- Bring new team members or developers switching teams up to speed quickly.
- Navigate source code.
- Plan out new features before any programming takes place.
- Communicate with technical and non-technical audiences more easily.

Introduction - UML - Use Case diagram

Use case diagrams model how users, displayed as stick figures called "actors," interact with the system. This type of UML diagram should be a high-level overview of the relationships between actors and systems, so it can be a great tool for explaining your system to a non-technical audience.

Use case diagrams describe what a system does from the standpoint of an external observer. The emphasis is on what a system does rather than how. A use case diagram is a collection of actors, use cases, and their communications.

- A use case is a summary of scenarios for a single task or goal.
- An actor is who or what initiates the events involved in that task.
    - Actors are simply roles that people or objects play.
    - Actors can be composed of users, external processes, or ren



Use case diagrams are **helpful** in three areas.

- Determining features (requirements)
- Communicating with clients
- Generating test cases

Introduction - UML - Class diagram

Class diagrams show the **static** structure of a system, including classes, their attributes and behaviors, and the relationships between each class.

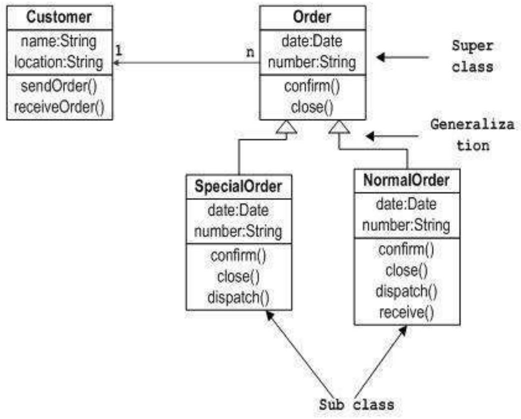
A class is represented by a rectangle that contains three compartments stacked vertically—the top compartment contains the class's name and is mandatory, but the bottom two compartments give details about the class attributes and class operations or behaviors.
Relationships between classes are the connecting links.

- ***Association*** - There is an association between two classes if an instance of one class must know about the other in order to perform its work.
- ***Aggregation*** - An association in which one class belongs to a collection. An aggregation has a diamond end pointing to the part containing the whole.
- ***Generalization*** - A inheritance link indicating one class is a superclass of the other. A generalization has a triangle pointing to the superclass.

A navigability arrow on an association shows which direction the association can be traversed or queried.

The **multiplicity** of an association end is the number of possible instances of the class associated with a single instance of the other end.

  - Multiplicities are single numbers or ranges of numbers.



Introduction - UML - Sequence Diagram

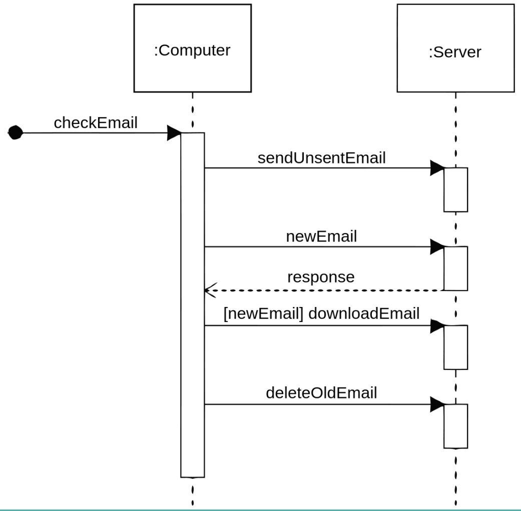
A **sequence** or Interaction diagrams are **dynamic**, shows the order in which objects interact. It details how operations are carried out - what messages are sent and when.

Sequence diagrams are organized according to time.

- The time progresses as you go down the page.
- The objects involved in the operation are listed from left to right according to when they take part in the message sequence.

Notation

- Each vertical dotted line is a lifeline, representing the time that an object exists.
- Each arrow is a message call.
    - An arrow goes from the sender to the top of the activation bar of the message on the receiver's lifeline.
- The activation bar represents the duration of execution of the message.



Introduction - Jukebox

A jukebox is a partially automated music-playing device, usually a coin-operated machine, that will play a patron's selection from self-contained media.

The classic jukebox has buttons, with letters and numbers on them, which are used to select a specific record. Some may use compact discs instead.
Disc changers are similar devices that are intended for home use, are small enough to fit in a shelf, may hold up to hundreds of discs, and allow discs to be easily removed, replaced, and inserted by the user.

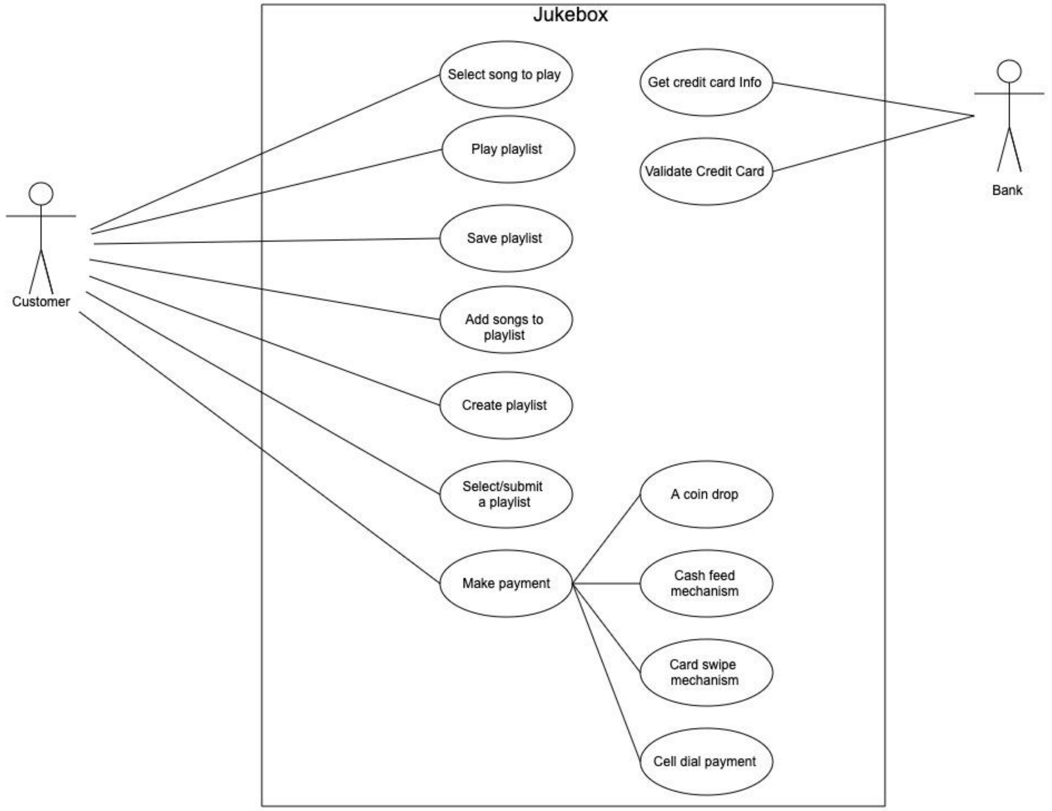
Problem Statement

Description
Our goal is to design a Juke Box that allows customers to select songs they want played or to submit a playlist that they have already created previously. If a request is made for song that is not contained by a local Jukebox, it will query for that song from other Jukeboxes elsewhere in the country - thus they are networked. Although reminiscent of Napster, like the original jukebox, we want to provide a mechanism for owners, record companies and artists to earn a profit. Therefore, for this Jukebox we want to provide not only a coin drop and cash feed mechanism, but also a card swipe mechanism and a cell dial payment capability.

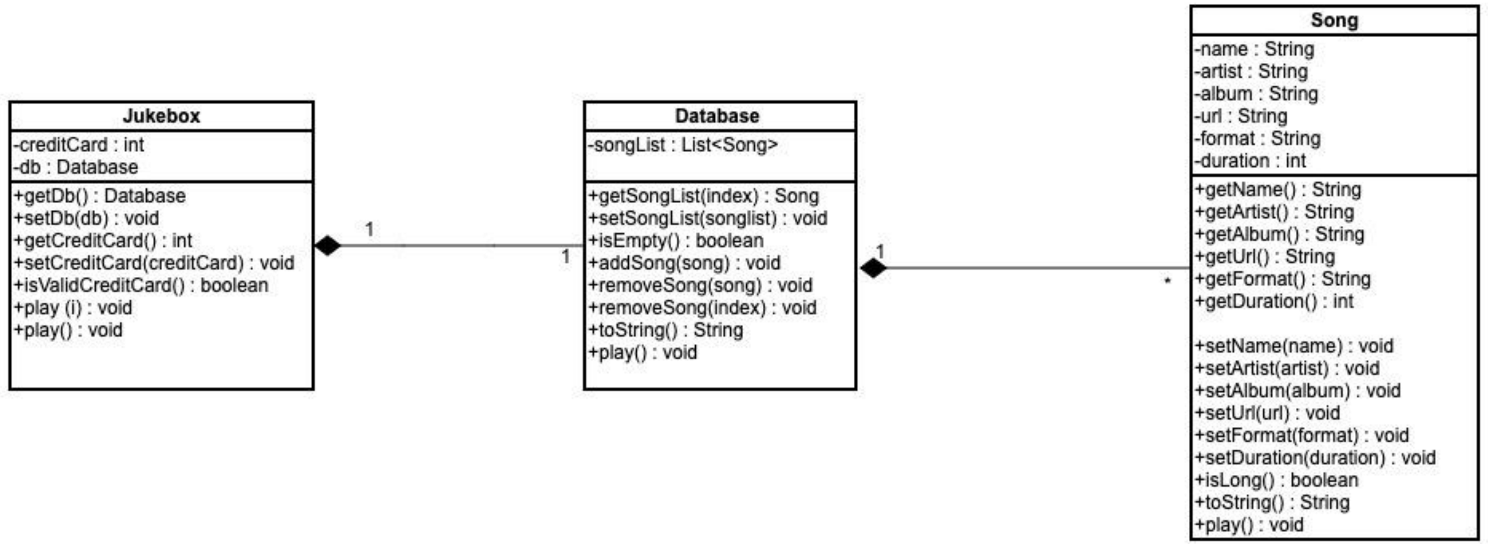
Juke Box spec

- Allow customers to
  - select songs they want to play.
  - submit a playlist that they have already created previously.
- The Juke Box can search other Juke Boxes from Internet for songs that are not contained by a local Jukebox.
- To provide a mechanism for owners, record companies and artists to earn a profit. The Juke Box contains
  - A coin drop
  - Cash feed mechanism
  - A card swipe mechanism
  - A cell dial payment capability.

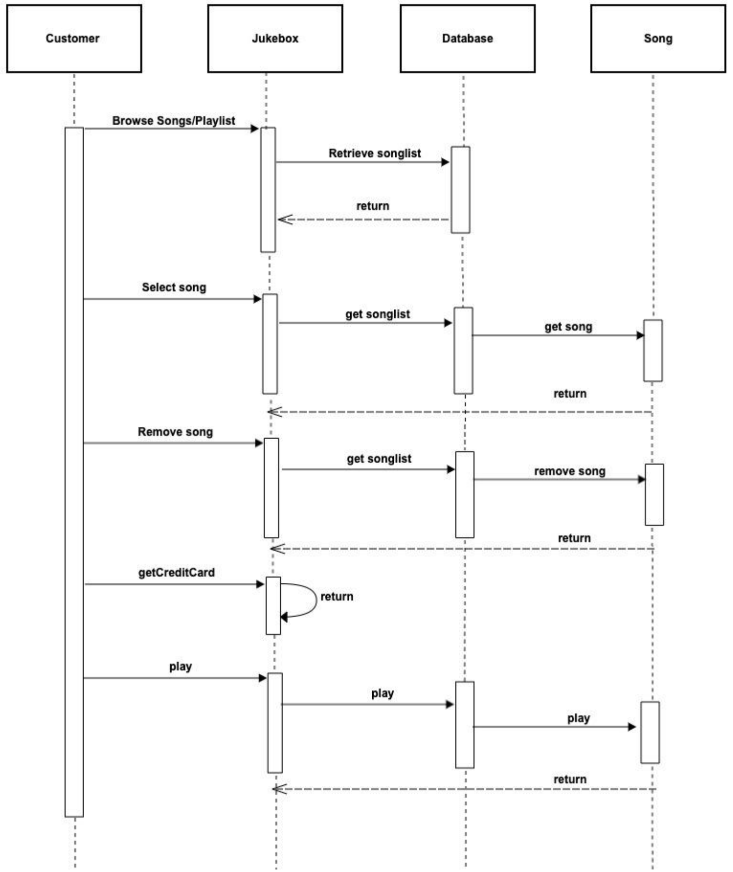
Design - Use Case Diagram



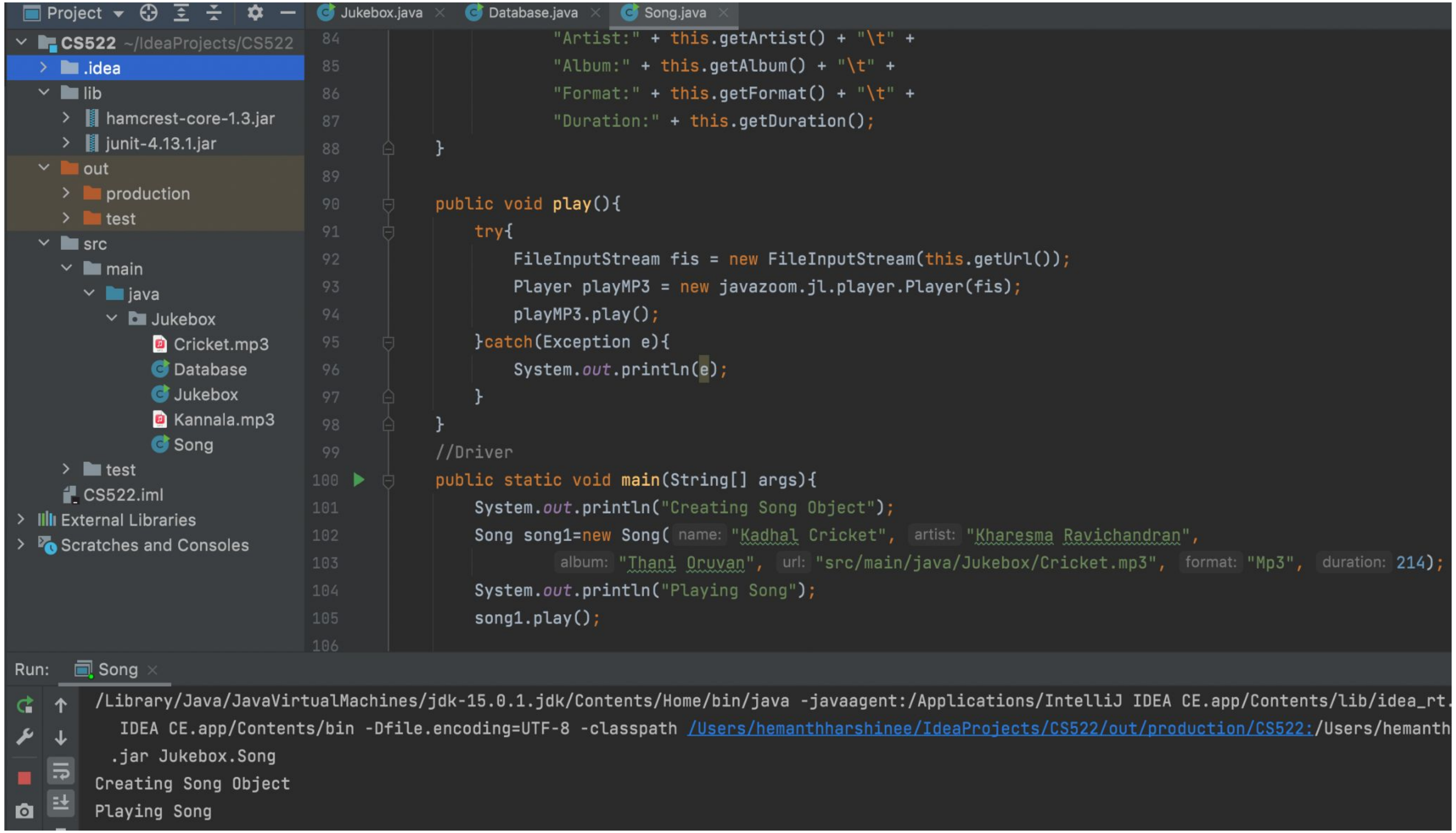
Design - Class diagram



Design - Sequence Diagram



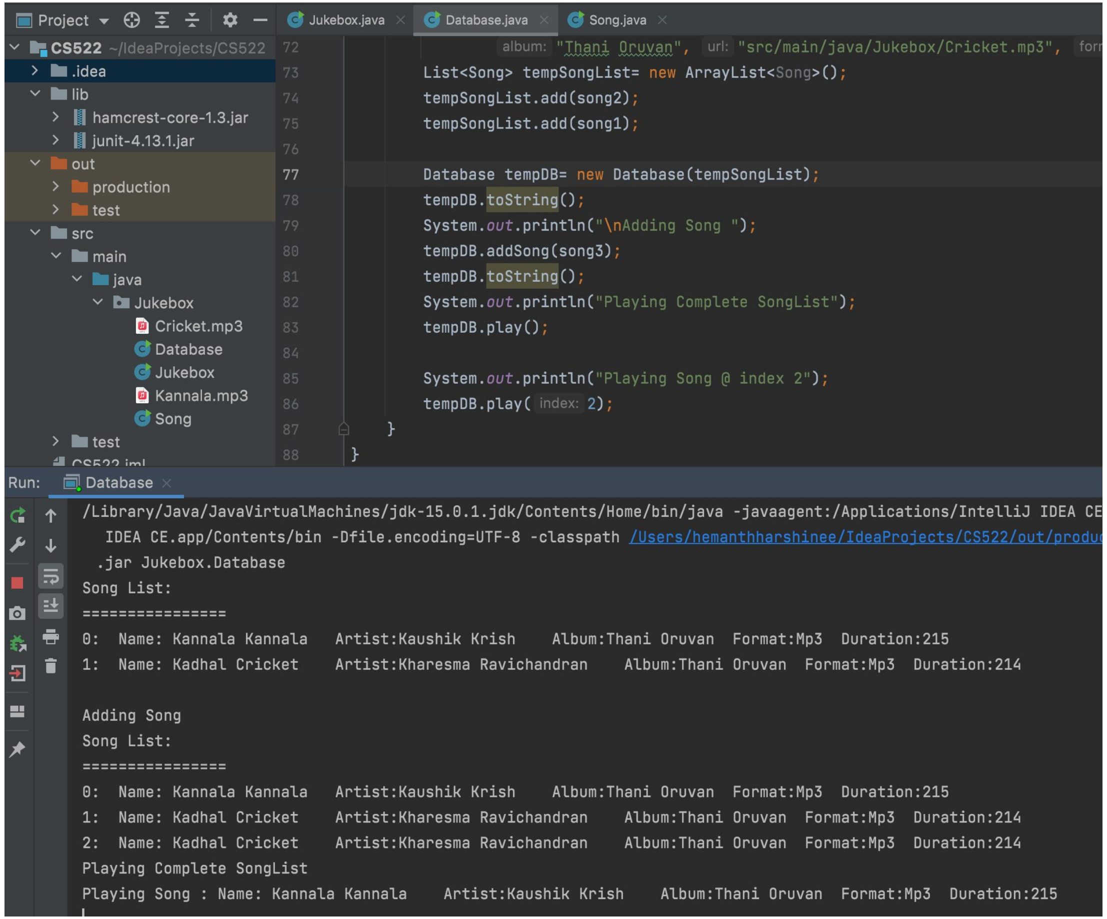
Implementation - Program - Song.java



```java
                    "Artist:" + this.getArtist() + "\t" +
                    "Album:" + this.getAlbum() + "\t" +
                    "Format:" + this.getFormat() + "\t" +
                    "Duration:" + this.getDuration();
    }

    public void play(){
        try{
            FileInputStream fis = new FileInputStream(this.getUrl());
            Player playMP3 = new javazoom.jl.player.Player(fis);
            playMP3.play();
        }catch(Exception e){
            System.out.println(e);
        }
    }
    //Driver
    public static void main(String[] args){
        System.out.println("Creating Song Object");
        Song song1=new Song( name: "Kadhal Cricket",  artist: "Kharesma Ravichandran",
                 album: "Thani Oruvan",  url: "src/main/java/Jukebox/Cricket.mp3",  format: "Mp3",  duration: 214);
        System.out.println("Playing Song");
        song1.play();
```

```
/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.
  IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/hemanthharshinee/IdeaProjects/CS522/out/production/CS522:/Users/hemanth
 .jar Jukebox.Song
Creating Song Object
Playing Song
```

Implementation - Program - Database.java



Project

CS522 ~/IdeaProjects/CS522
.idea
lib
hamcrest-core-1.3.jar
junit-4.13.1.jar
out
production
test
src
main
java
Jukebox
Cricket.mp3
Database
Jukebox
Kannala.mp3
Song
test
CS522.iml

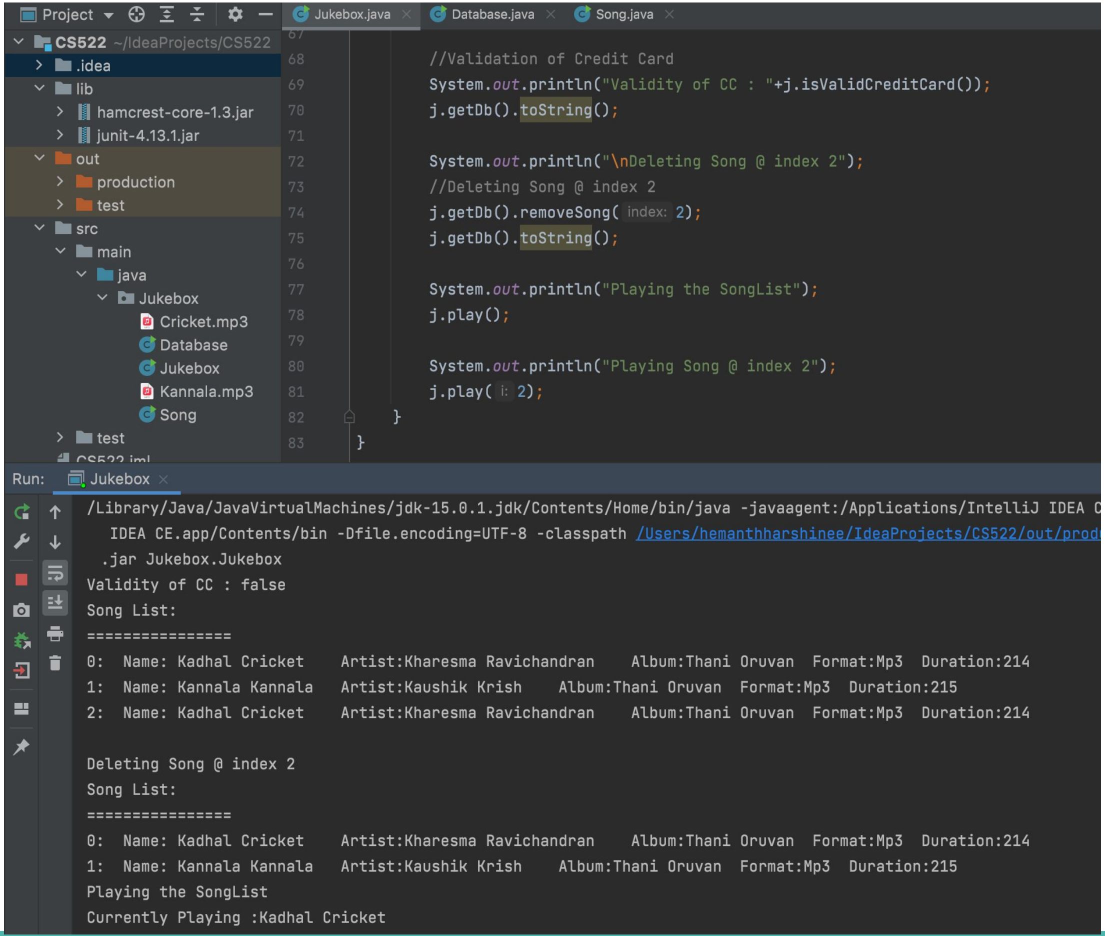
Jukebox.java | Database.java | Song.java

```java
                   album: "Thani Oruvan",  url: "src/main/java/Jukebox/Cricket.mp3",  for
        List<Song> tempSongList= new ArrayList<Song>();
        tempSongList.add(song2);
        tempSongList.add(song1);

        Database tempDB= new Database(tempSongList);
        tempDB.toString();
        System.out.println("\nAdding Song ");
        tempDB.addSong(song3);
        tempDB.toString();
        System.out.println("Playing Complete SongList");
        tempDB.play();

        System.out.println("Playing Song @ index 2");
        tempDB.play( index: 2);
    }
}
```

Run: Database

```
/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE
  IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/hemanthharshinee/IdeaProjects/CS522/out/produ
 .jar Jukebox.Database
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
```

Implementation - Program - Jukebox.java



```java
        //Validation of Credit Card
        System.out.println("Validity of CC : "+j.isValidCreditCard());
        j.getDb().toString();

        System.out.println("\nDeleting Song @ index 2");
        //Deleting Song @ index 2
        j.getDb().removeSong( index: 2);
        j.getDb().toString();

        System.out.println("Playing the SongList");
        j.play();

        System.out.println("Playing Song @ index 2");
        j.play( i: 2);
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA C
  IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/hemanthharshinee/IdeaProjects/CS522/out/prod
  .jar Jukebox.Jukebox
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

Now, that the UML diagrams (Use case diagram, Class diagram and Sequence Diagram) and the code is ready, we need to write test cases

Bibliography

https://npu85.npu.edu/~henry/npu/classes/introjava/java_class/hw/christy_jaeson_augustine/jukebox/index_jukebox.html

https://npu85.npu.edu/~henry/npu/classes/oo/uml/slide/simplified_oo_process.html

Link to view the presentation

https://docs.google.com/presentation/d/1DQqgVtvN_GZi0lfzQq8GPk_a0C0CKjETCjrJZ4ji7MI/edit?usp=sharing