Project: JukeBox

Long Nguyen Hoang
CS522

Subjects

- Description
- Juke Box spec
- UML class diagram
- Use case diagram
- Sequence diagram
- Jukebox Junit Test Cases
- References

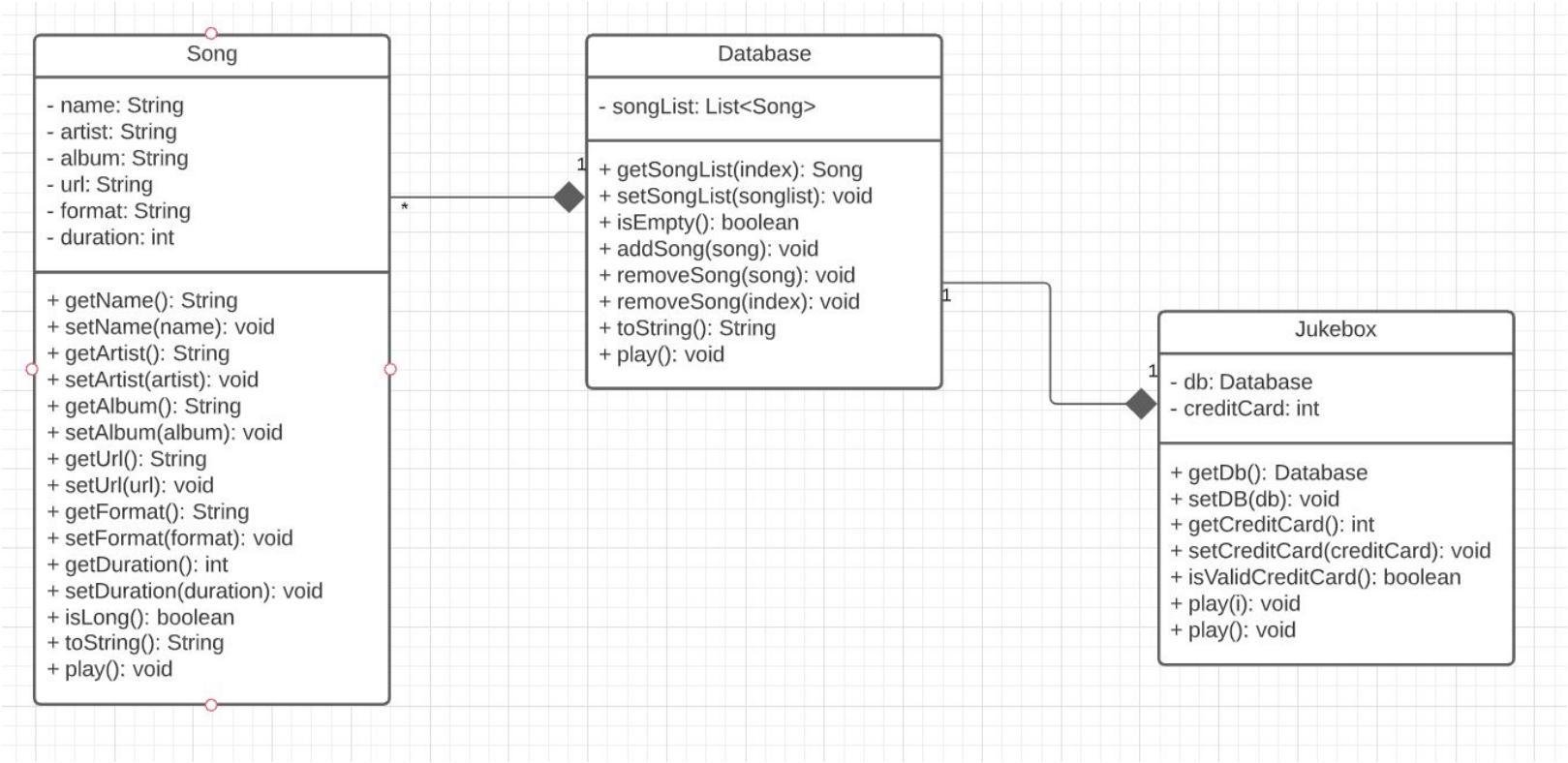
Description

- 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 Juke Box, 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.

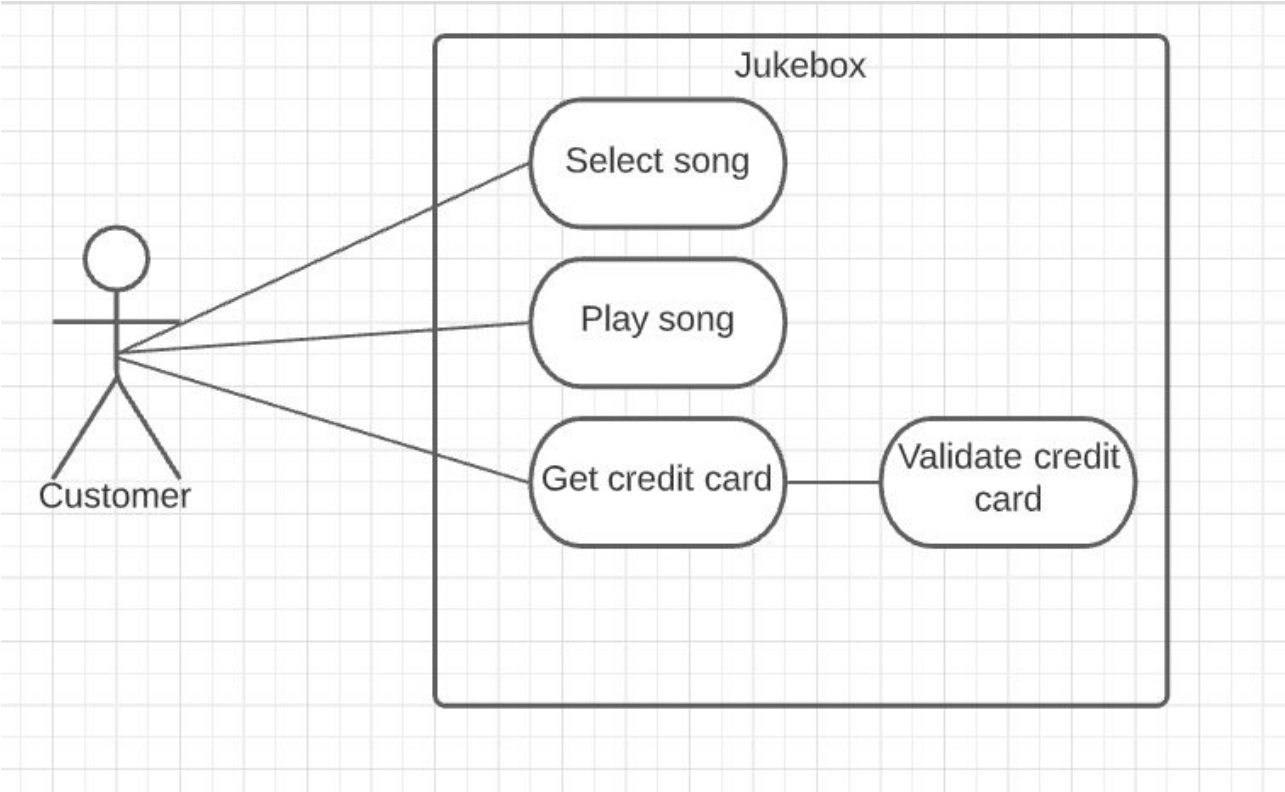
Juke Box spec

- 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 Juke Box.
- 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.

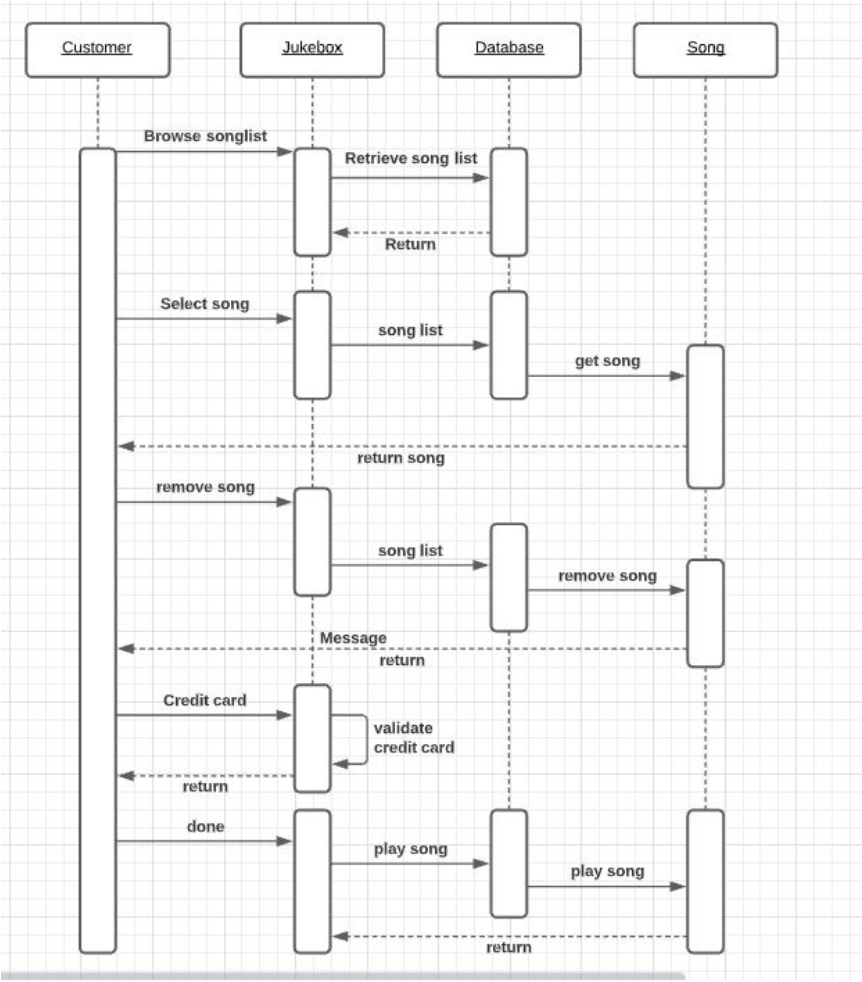
UML Class Diagram



Use case diagram



Sequence diagram



Jukebox Junit Test Case - SongTest

```
@Test
public void testIsLong1() {
  assertEquals("Song is long if length is more than 50",true,testSong1.isLong());
}
@Test
public void testIsLong2() {
  assertEquals("Song is long if length is more than 50",true,testSong2.isLong());
}

@Test
public void testGetName() {
  assertEquals("Test getName method","Kadhal Cricket",testSong2.getName());
}

@Test
public void testSetName() {
    testSong1.setName("Kannala Kannala");
  assertEquals("Test setName method","Kannala Kannala",testSong1.getName());
}

@Test
public void testGetArtist() {
  assertEquals("Test getArtist method","Kharesma Ravichandran",testSong2.getArtist());
}

@Test
public void testSetArtist() {
  testSong2.setArtist("Kaushik Krish");
  assertEquals("Test setArtist method","Kaushik Krish",testSong2.getArtist());
}
```

- Jukebox.SongTest [Runner: JUnit 4] (0.002 s)
  - testGetAlbum (0.001 s)
  - testSetDuration (0.000 s)
  - testGetDuration (0.000 s)
  - testGetName (0.000 s)
  - testSetArtist (0.000 s)
  - testSetFormat (0.000 s)
  - testGetArtist (0.000 s)
  - testGetFormat (0.000 s)
  - testGetUrl (0.000 s)
  - testSetName (0.000 s)
  - testIsLong1 (0.000 s)
  - testIsLong2 (0.000 s)
  - testSetUrl (0.000 s)
  - testSetAlbum (0.000 s)

Jukebox Junit Test Case - DatabaseTest

```
// TODO add test methods here.
// The methods must be annotated with annotation @Test. For example:
//
 @Test
 public void testSongList() {
   assertEquals("SongList if empty should have 0 elements",false,songList.isEmpty());
 }

 @Test
 public void testGetSongList() {
     assertEquals("Test getSongList",songList,testDB.getSongList());
 }

 @Test
 public void testAddSong() {
     testDB.addSong(testSong1);
     assertEquals("Test addSong",songList,testDB.getSongList());
 }
 @Test
 public void testRemoveSong() {
     testDB.removeSong(testSong1);
     assertEquals("Test removeSong",songList,testDB.getSongList());
 }
 @Test
 public void testRemoveSongByIndex() {
     testDB.removeSong(1);
     assertEquals("Test removeSongByIndex method",songList,testDB.getSongList());
 }
```

- Jukebox.DatabaseTest [Runner: JUnit 4] (0.000 s)
  - testSongList (0.000 s)
  - testAddSong (0.000 s)
  - testGetSongList (0.000 s)
  - testRemoveSongByIndex (0.000 s)
  - testRemoveSong (0.000 s)

Jukebox Junit Test Case - JukeboxTest

```java
@Test
public void testIsValidCreditCard() {
    assertEquals("Credit card is valid if value is greater than 0",true,testJB.isValidCredi
}

@Test
public void testGetCreditCard() {
    assertEquals("Test getCreditCard method",123,testJB.getCreditCard());
}

@Test
public void testSetCreditCard() {
    testJB.setCreditCard(0000);
    assertEquals("Test setCreditCard method",0000,testJB.getCreditCard());
}

@Test
public void testGetDb() {
    assertTrue("Test Database exist",testJB.getDb() instanceof Database);
    assertNotNull("Test Database not null",testJB.getDb());
}

@Test
public void testSetDb() {
    Song song3 = new Song("Kadhal Cricket", "Kharesma Ravichandran",
            "Thani Oruvan", "Cricket.mp3", "Mp3", 214);
    songListJB = new java.util.ArrayList();
    songListJB.add(song3);
    Database db1 = new Database(songListJB);
    testJB.setDb(db1);
    assertNotNull("Test Database not null",testJB.getDb());
    assertTrue("Test Database exist",testJB.getDb() instanceof Database);
}
```

Runs: 5/5 Errors: 0 Failures: 0

- Jukebox.JukeBoxTest [Runner: JUnit 4] (0.000 s)
  - testGetDb (0.000 s)
  - testSetDb (0.000 s)
  - testIsValidCreditCard (0.000 s)
  - testGetCreditCard (0.000 s)
  - testSetCreditCard (0.000 s)

References

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