Jukebox JUNIT

Unit Tests

Presented by
Sai Harshinee Roopakula
19577

Table of Contents

- Introduction
  - JUNIT
- Problem Statement
- Code
  - Song.java
  - Database.java
  - Jukebox.java
- Environment setup for JUNIT
- JUNIT test cases
- Conclusion
- Bibliography

Introduction - JUNIT

JUnit is a unit testing framework for the Java Programming Language. It is important in the test driven development, and is one of a family of unit testing frameworks collectively known as xUnit.
JUnit promotes the idea of "first testing then coding", which emphasis on setting up the test data for a piece of code which can be tested first and then can be implemented .

JUnit is an open source framework which is used for writing & running tests. It provides

- Annotation to identify the test methods.
- Assertions for testing expected results.
- Test runners for running tests.

JUnit tests allow you to write code faster which increasing quality. JUnit is elegantly simple. It is less complex & takes less time. JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results. JUnit tests can be organized into test suites containing test cases and even other test suites. Junit shows test progress in a bar that is green if test is going fine and it turns red when a test fails.

Unit Test Case:
A formal written unit test case is characterized by a known input and by an expected output.There must be at least two unit test cases for each requirement - One positive test and one negative test.
If a requirement has sub-requirements, each sub-requirement must have at least two test cases as positive and negative.

Problem Statement

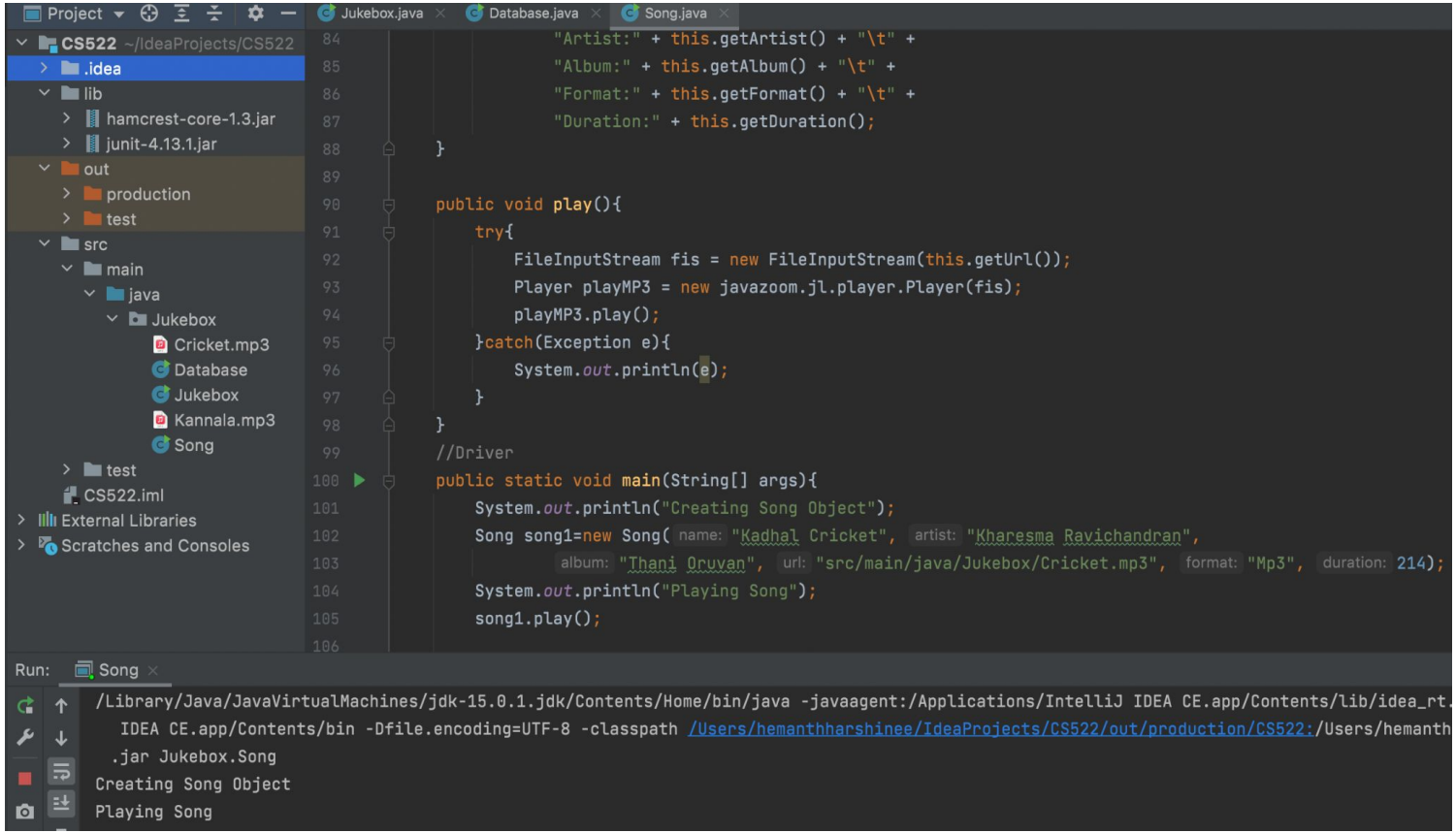
Description

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.

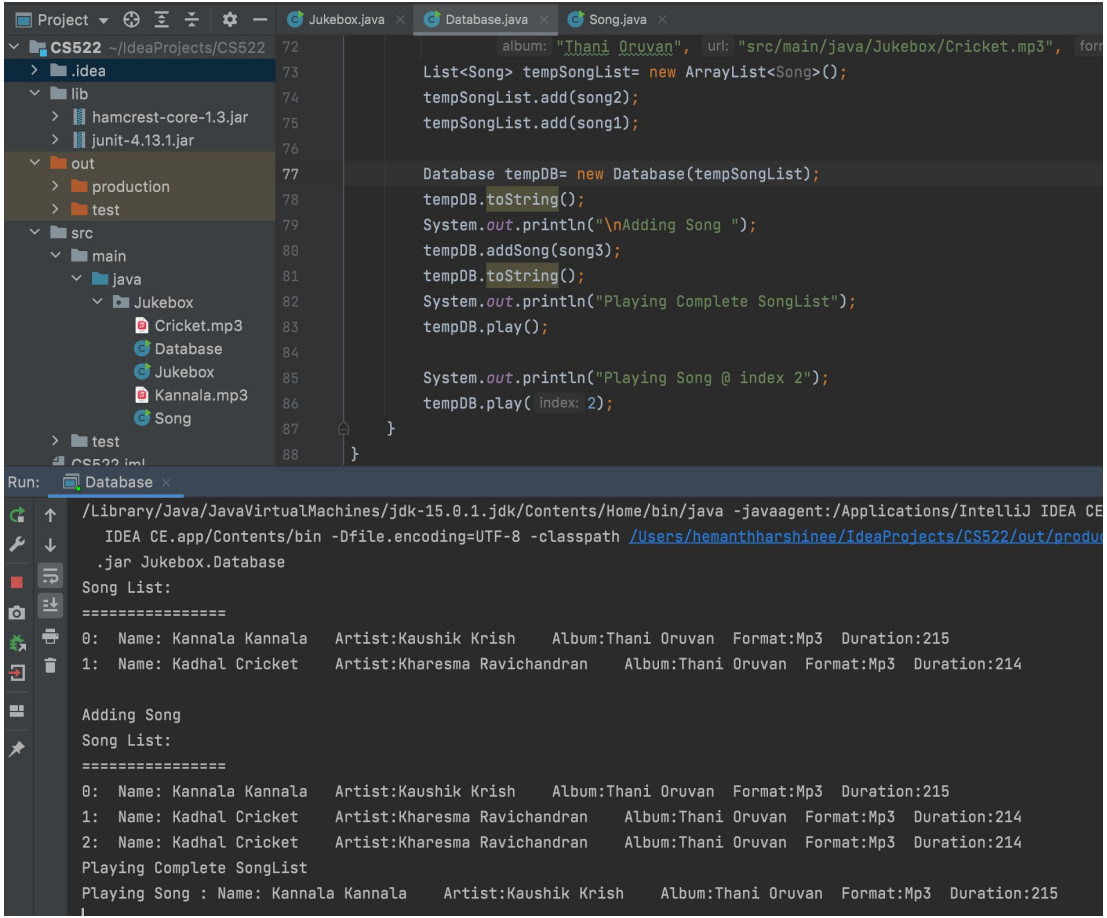
Juke Box spec

- 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.

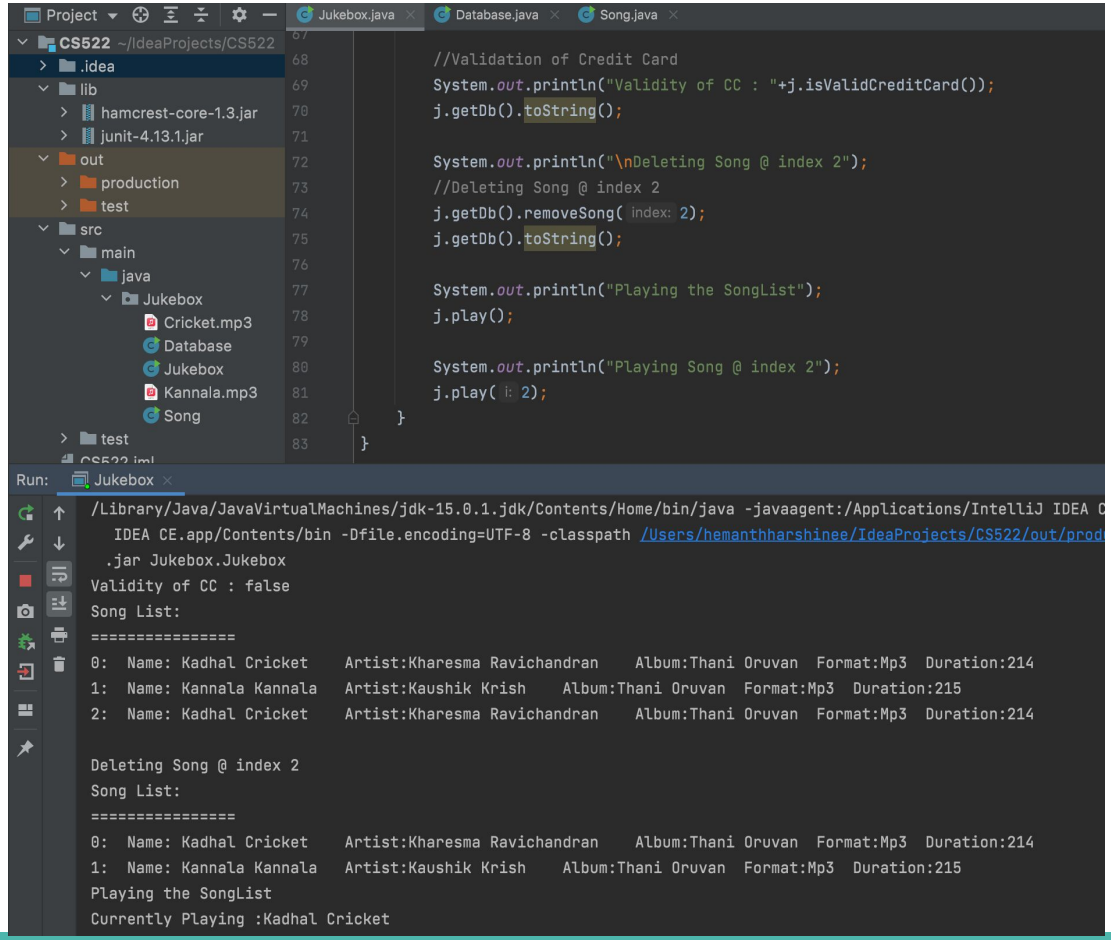
Code - Song.java



Code - Database.java



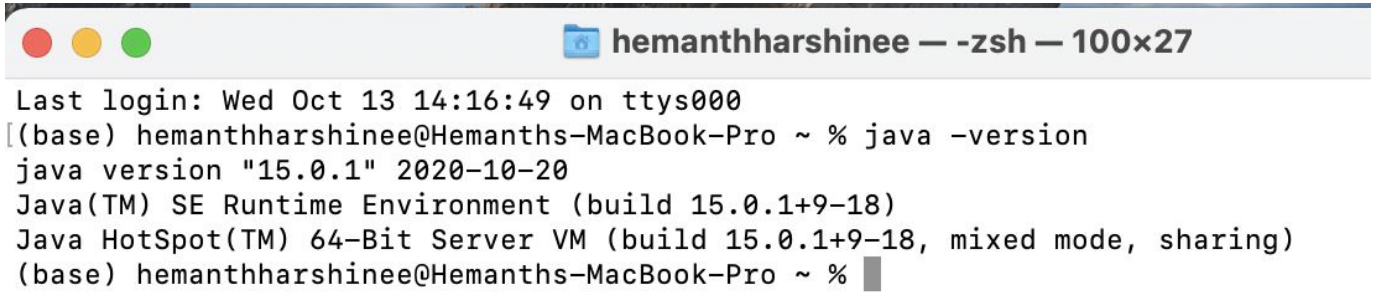
Code - Jukebox.java



Environmental Setup (MacOS)

Step 1: Verify Java installation

Type java -version to verify Java installation



Environmental Setup (MacOS)

Step 2: Set JAVA environment

Set the **JAVA_HOME** environment variable to point to the base directory location where Java is installed on your machine, In my case it is

export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home

Edit the .zshrc file by adding the path mentioned above.

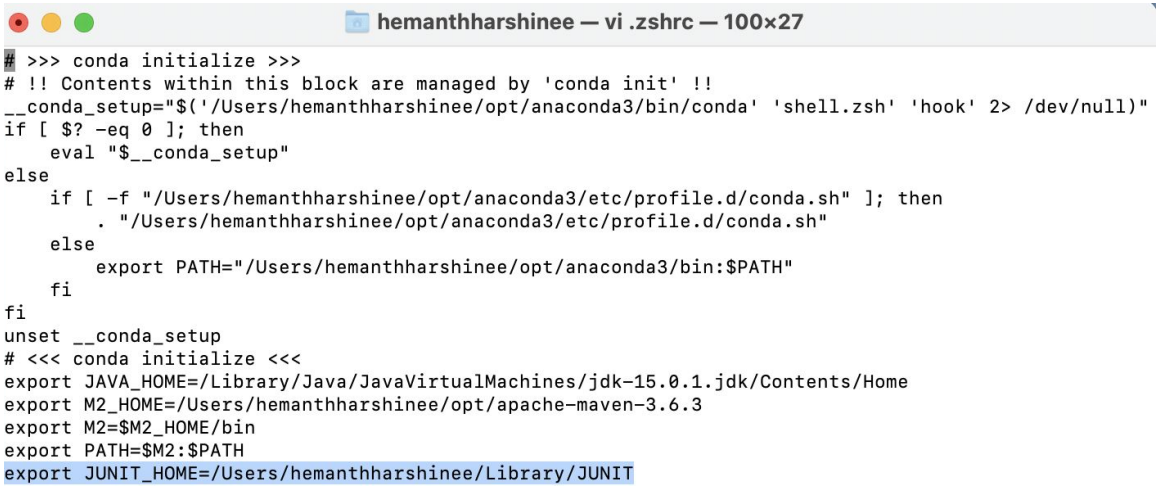
Environmental Setup (MacOS)

Step 3: Download Junit archive
Download latest version of JUnit jar file from http://www.junit.org.

Step 4: Set Junit environment

Set the **JUNIT_HOME** environment variable to point to the base directory location where JUNIT jar is stored on your machine. Assuming, we've stored junit-4.10.jar in JUNIT folder

export JUNIT_HOME=/Library/JUNIT

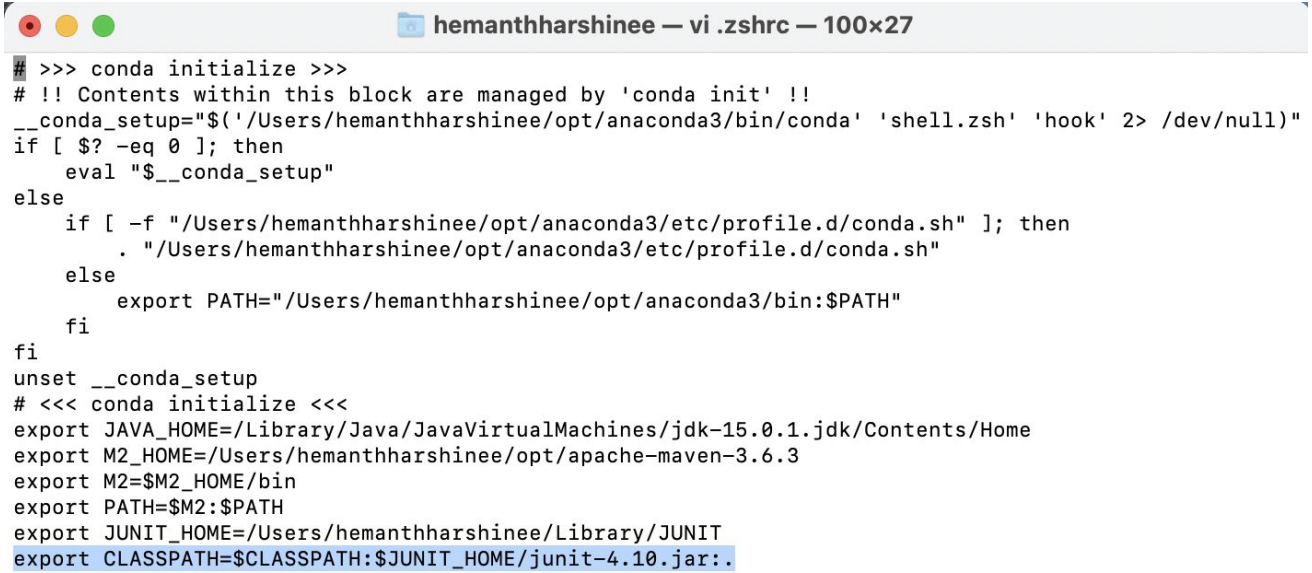


Environmental Setup (MacOS)

Step 5: Set CLASSPATH variable

Set the **CLASSPATH** environment variable to point to the JUNIT jar location. Assuming, we've stored junit-4.10.jar in JUNIT folder

export CLASSPATH=$CLASSPATH:$JUNIT_HOME/junit-4.10.jar:.

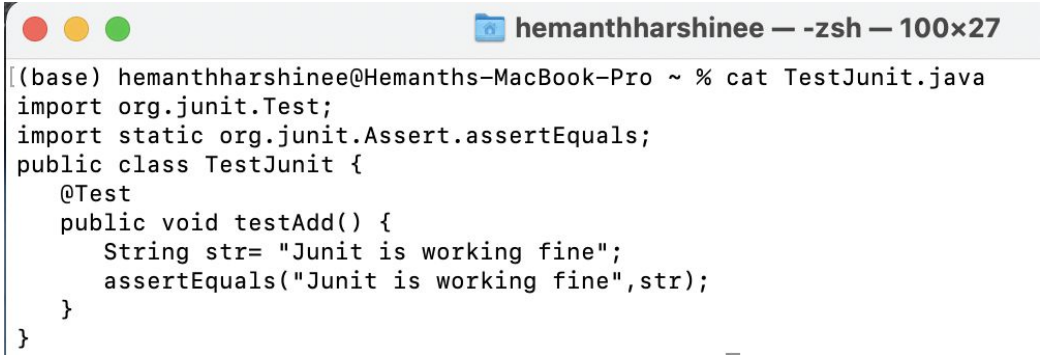


Environmental Setup (MacOS)

Step 6: Test JUnit Setup

Create a java class file name **TestJunit**

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestJunit {
  @Test
  public void testAdd() {
    String str= "Junit is working fine";
    assertEquals("Junit is working fine",str);
  }
}
```

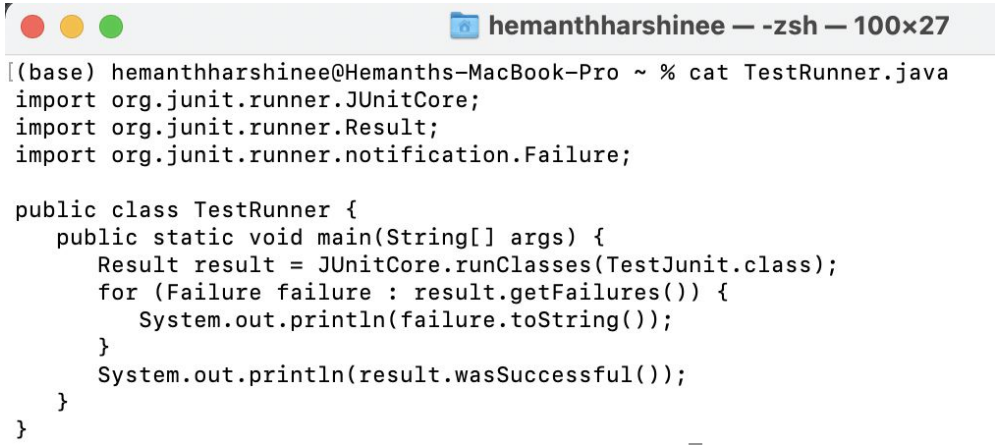


Environmental Setup (MacOS)

Step 6: Test JUnit Setup

Create a java class file name **TestRunner**

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class TestRunner {
  public static void main(String[] args) {
    Result result = JUnitCore.runClasses(TestJunit.class);
    for (Failure failure : result.getFailures()) {
      System.out.println(failure.toString());
    }
    System.out.println(result.wasSuccessful());
  }
}
```



Environmental Setup (MacOS)

Step 7: Verify the result

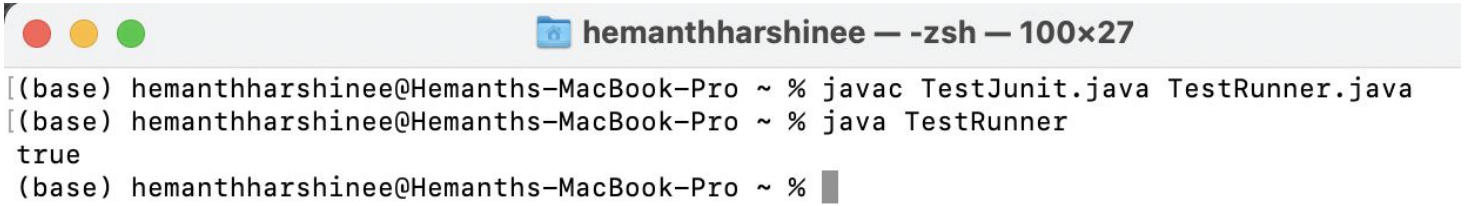
Compile the classes using **javac** compiler :

```
javac TestJunit.java TestRunner.java
```

Now run the Test Runner to see the result:

```
java TestRunner
```

Verify the output, it should be **true**



JUNIT test cases - SongTest.java

SongTest.java

```java
package jukebox;

import Jukebox.Song;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import static org.junit.Assert.*;

/**
 *
 * @author Sai Harshinee
 */

public class SongTest {
    private Song testSong1, testSong2;

    public SongTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }
```

JUNIT test cases - SongTest.java

SongTest.java

```
@Before
public void setUp() {
    testSong1 = new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
            album: "Thani Oruvan", url: "Cricket.mp3", format: "Mp3", duration: 214);
    testSong2 = new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
            album: "Thani Oruvan", url: "Cricket.mp3", format: "Mp3", duration: 214);
}

@After
public void tearDown() {
    testSong1 = null;
    testSong2 = null;
}

@Test
public void testGetName() {
    System.out.println("Running testGetName method(Song class) ...");
    assertEquals( message: "Song names are equal", expected: "Kadhal Cricket", testSong1.getName());
}

@Test
public void testSetName() {
    System.out.println("Running testSetName method(Song class) ...");
    testSong1.setName("Kannala Kannala");
    assertEquals( message: "Song names are equal", expected: "Kannala Kannala", testSong1.getName());
}

@Test
public void testGetArtist() {
    System.out.println("Running testGetArtist method(Song class)...");
    assertEquals( message: "Song artists are equal", expected: "Kharesma Ravichandran", testSong1.getArtist());
}
```

Test cases for
getName()
setName()
getArtist() methods of
Song.java

JUNIT test cases - SongTest.java

SongTest.java

```
64      @Test
65      public void testSetArtist() {
66          System.out.println("Running testSetArtist method(Song class)...");
67          testSong1.setArtist("Kaushik Krish");
68          assertEquals( message: "Song artists are equal",  expected: "Kaushik Krish", testSong1.getArtist());
69      }
70
71      @Test
72      public void testGetAlbum() {
73          System.out.println("Running testGetAlbum method(Song class)...");
74          assertEquals( message: "Song albums are equal",  expected: "Thani Oruvan", testSong1.getAlbum());
75      }
76      @Test
77      public void testSetAlbum() {
78          System.out.println("Running testSetAlbum method(Song class)...");
79          testSong1.setAlbum("Lovely");
80          assertEquals( message: "Song albums are equal",  expected: "Lovely", testSong1.getAlbum());
81      }
82
83      @Test
84      public void testGetUrl() {
85          System.out.println("Running testGetUrl method(Song class)...");
86          assertEquals( message: "Song urls are equal",  expected: "Cricket.mp3", testSong1.getUrl());
87      }
88
89      @Test
90      public void testSetUrl() {
91          System.out.println("Running testSetUrl method(Song class)...");
92          testSong1.setUrl("Kannala.mp3");
93          assertEquals( message: "Song urls are equal",  expected: "Kannala.mp3", testSong1.getUrl());
94      }
```

Test cases for
setArtist()
getAlbum()
setAlbum()
getUrl()
setUrl() methods of
Song.java

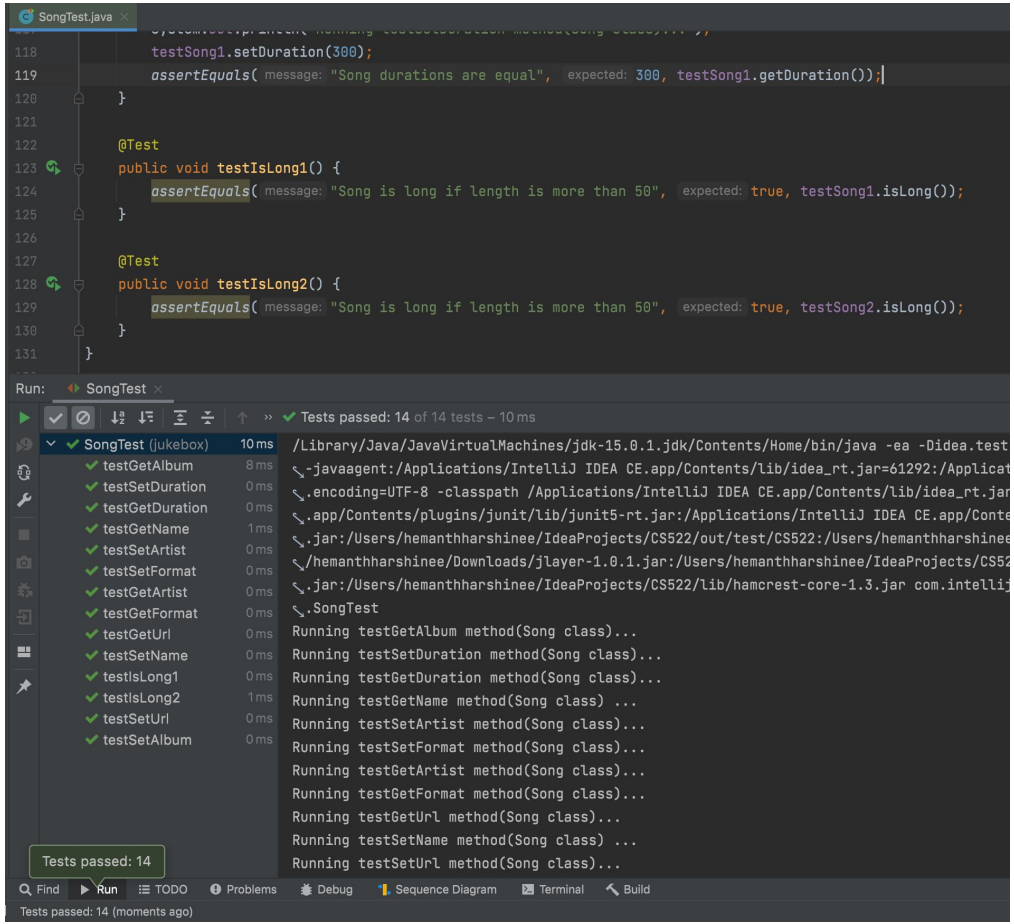
JUNIT test cases - SongTest.java

SongTest.java

```
96      @Test
97      public void testGetFormat() {
98          System.out.println("Running testGetFormat method(Song class)...");
99          assertEquals( message: "Song formats are equal",  expected: "Mp3", testSong1.getFormat());
100     }
101
102     @Test
103     public void testSetFormat() {
104         System.out.println("Running testSetFormat method(Song class)...");
105         testSong1.setFormat("Mp4");
106         assertEquals( message: "Song formats are equal",  expected: "Mp4", testSong1.getFormat());
107     }
108
109     @Test
110     public void testGetDuration() {
111         System.out.println("Running testGetDuration method(Song class)...");
112         assertEquals( message: "Song durations are equal",  expected: 214, testSong1.getDuration());
113     }
114
115     @Test
116     public void testSetDuration() {
117         System.out.println("Running testSetDuration method(Song class)...");
118         testSong1.setDuration(300);
119         assertEquals( message: "Song durations are equal",  expected: 300, testSong1.getDuration());
120     }
121
122     @Test
123     public void testIsLong1() {
124         assertEquals( message: "Song is long if length is more than 50",  expected: true, testSong1.isLong());
125     }
126
127     @Test
128     public void testIsLong2() {
129         assertEquals( message: "Song is long if length is more than 50",  expected: true, testSong2.isLong());
130     }
131 }
```

Test cases for
getFormat()
setFormat()
getDuration()
setDuration()
isLong() methods of
Song.java

JUNIT test cases - SongTest.java Output



All 14 test cases of are **PASSED**.

JUNIT test cases - DatabaseTest.java

DatabaseTest.java

```java
package jukebox;

import Jukebox.Database;
import Jukebox.Song;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import static org.junit.Assert.*;
/**
 *
 * @author Sai Harshinee
 */
public class DatabaseTest {
    private java.util.List songList, songList1;
    private Song testSong1,testSong2, testSong3;
    private Database testDB, testDB1;

    public DatabaseTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

```

JUNIT test cases - DatabaseTest.java

DatabaseTest.java

```
32      @Before
33      public void setUp() {
34          songList =  new java.util.ArrayList();
35          songList1 =  new java.util.ArrayList();
36          testSong1 = new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
37                  album: "Thani Oruvan", url: "Cricket.mp3", format: "Mp3", duration: 214);
38          testSong2 = new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
39                  album: "Thani Oruvan", url: "Cricket.mp3", format: "Mp3", duration: 214);
40
41          testSong3 = new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
42                  album: "Thani Oruvan", url: "Cricket.mp3", format: "Mp3", duration: 214);
43
44          songList.add(testSong1);
45          songList.add(testSong2);
46
47          songList1.add(testSong1);
48
49          testDB = new Database(songList);
50          testDB1 = new Database(songList1);
51      }
52      @After
53      public void tearDown() {
54          songList = null;
55          songList1 = null;
56          testDB = null;
57          testDB1 = null;
58      }
59
```

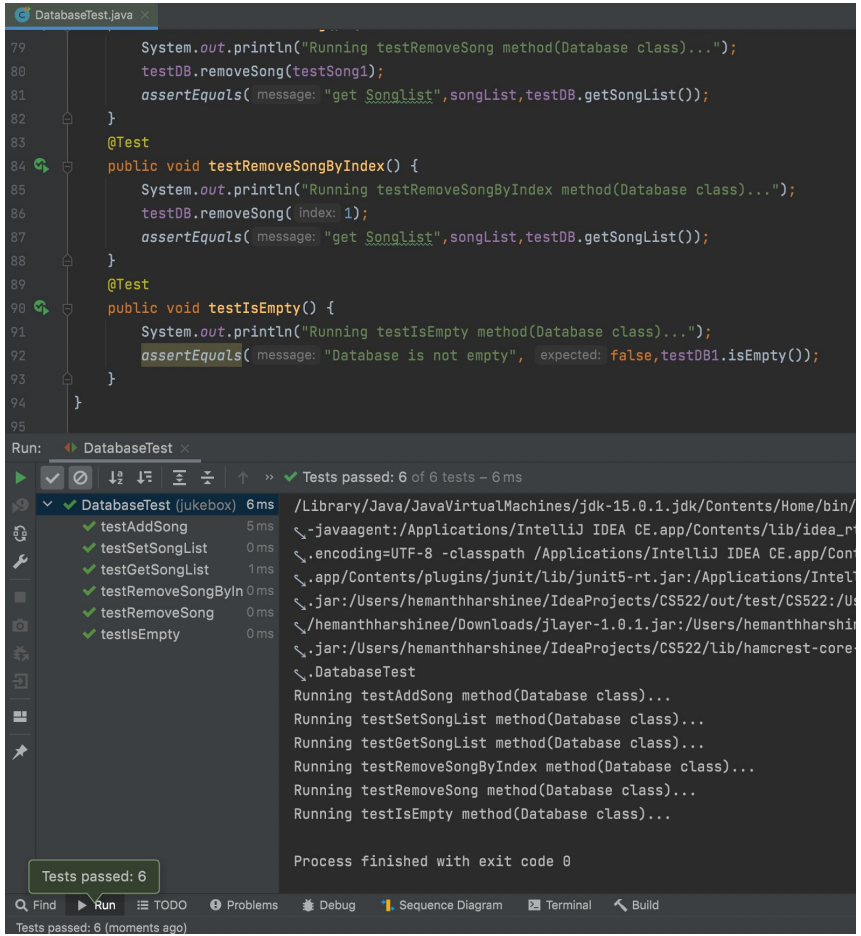
JUNIT test cases - DatabaseTest.java

DatabaseTest.java

```
60      @Test
61      public void testGetSongList() {
62          System.out.println("Running testGetSongList method(Database class)...");
63          assertEquals( message: "get Songlist",songList,testDB.getSongList());
64      }
65      @Test
66      public void testSetSongList() {
67          System.out.println("Running testSetSongList method(Database class)...");
68          testDB1.setSongList(songList1);
69          assertEquals( message: "get Songlist",songList1,testDB1.getSongList());
70      }
71      @Test
72      public void testAddSong() {
73          System.out.println("Running testAddSong method(Database class)...");
74          testDB1.addSong(testSong3);
75          assertEquals( message: "get Songlist",songList1,testDB1.getSongList());
76      }
77      @Test
78      public void testRemoveSong() {
79          System.out.println("Running testRemoveSong method(Database class)...");
80          testDB.removeSong(testSong1);
81          assertEquals( message: "get Songlist",songList,testDB.getSongList());
82      }
83      @Test
84      public void testRemoveSongByIndex() {
85          System.out.println("Running testRemoveSongByIndex method(Database class)...");
86          testDB.removeSong( index: 1);
87          assertEquals( message: "get Songlist",songList,testDB.getSongList());
88      }
89      @Test
90      public void testIsEmpty() {
91          System.out.println("Running testIsEmpty method(Database class)...");
92          assertEquals( message: "Database is not empty",  expected: false,testDB1.isEmpty());
93      }
94  }
```

Test cases for
getSongList()
setSongList()
addSong()
removeSong()
removeSong(index)
isEmpty() methods of
Database.java

JUNIT test cases - DatabaseTest.java Output



All 6 test cases are **PASSED**

JUNIT test cases - JukeboxTest.java

JukeboxTest.java

```java
package jukebox;

import Jukebox.Database;
import Jukebox.Jukebox;
import Jukebox.Song;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import java.util.List;

import static org.junit.Assert.*;

/**
 *
 * @author Sai Harshinee
 */

public class JukeboxTest {
    private Jukebox testJB;
    private Database db1, db2;
    private List songListJB1;
    private List songListJB2;
    private Song song1,song2;
    private int ccn=1234567;

    public JukeboxTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }
```

JUNIT test cases - JukeboxTest.java

JukeboxTest.java

```
36      @AfterClass
37      public static void tearDownClass() {
38      }
39
40      @Before
41      public void setUp() {
42          song1 = new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
43                  album: "Thani Oruvan", url: "Cricket.mp3", format: "Mp3", duration: 214);
44          song2 = new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
45                  album: "Thani Oruvan", url: "Cricket.mp3", format: "Mp3", duration: 214);
46
47          songListJB1 = new java.util.ArrayList();
48          songListJB2 = new java.util.ArrayList();
49          songListJB2.add(song1);
50
51          songListJB1.add(song1);
52          songListJB1.add(song2);
53
54          db1 = new Database(songListJB1);
55          db2 = new Database(songListJB2);
56
57          testJB = new Jukebox(db1,ccn);
58      }
59
60      @After
61      public void tearDown() { testJB = null; }
64
```

JUNIT test cases - JukeboxTest.java

JukeboxTest.java

```
65      @Test
66      public void testGetCreditCard() {
67          System.out.println("Running testGetCreditCard method(Jukebox class)...");
68          assertEquals( message: "Get Credit card number", expected: 1234567,testJB.getCreditCard());
69      }
70
71      @Test
72      public void testSetCreditCard() {
73          System.out.println("Running testSetCreditCard method(Jukebox class)...");
74          testJB.setCreditCard(123456789);
75          assertEquals( message: "Get Credit card number", expected: 123456789,testJB.getCreditCard());
76      }
77
78      @Test
79      public void testGetDb() {
80          System.out.println("Running testGetDb method(Jukebox class)...");
81          assertNotNull( message: "Get Database",testJB.getDb());
82          assertTrue( message: "Get Database",testJB.getDb() instanceof Database);
83          assertEquals( message: "Get Credit card number", expected: 1234567,testJB.getCreditCard());
84          assertEquals( message: "Get Song List", expected: 2,testJB.getDb().getSongList().size());
85      }
86
87      @Test
88      public void testSetDb() {
89          System.out.println("Running testSetDb method(Jukebox class)...");
90          testJB.setDb(db2);
91          assertNotNull( message: "Get Database",testJB.getDb());
92          assertTrue( message: "Get Database",testJB.getDb() instanceof Database);
93          assertEquals( message: "Get Credit card number", expected: 1234567,testJB.getCreditCard());
94          assertEquals( message: "Get Song List", expected: 1,testJB.getDb().getSongList().size());
95      }
96
```

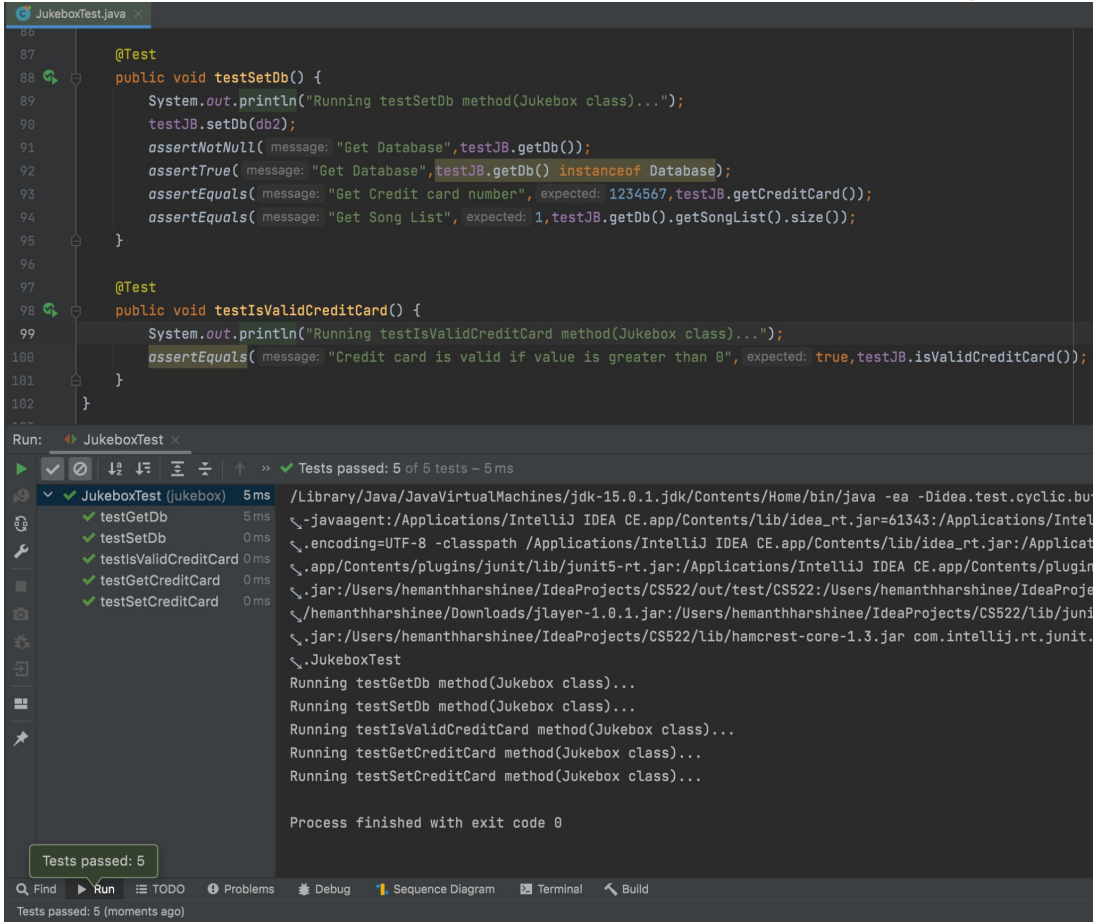
Test cases for
getCreditCard()
setCreditCard()
getDb()
setDb() methods of
Jukebox.java

JUNIT test cases - JukeboxTest.java

```
96
97      @Test
98      public void testIsValidCreditCard() {
99          System.out.println("Running testIsValidCreditCard method(Jukebox class)...");
100         assertEquals( message: "Credit card is valid if value is greater than 0", expected: true,testJB.isValidCreditCard());
101     }
102 }
103
```

Test cases for isValidCrediCard() methods of Database.java

JUNIT test cases - JukeboxTest.java Output



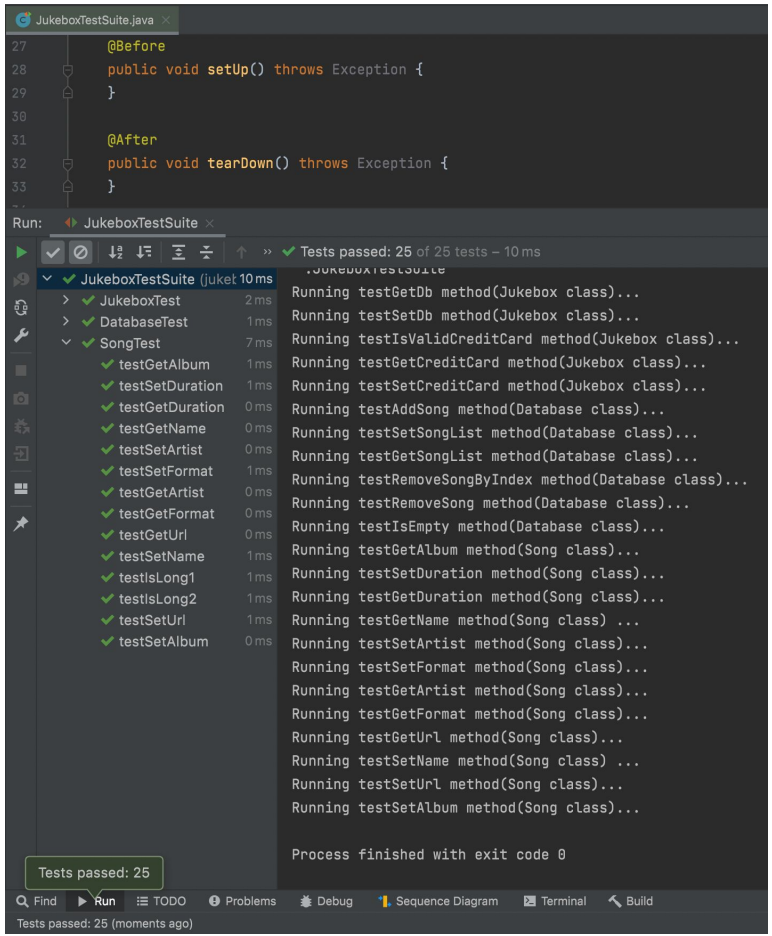
All 5 test cases are **PASSED**

JUNIT test cases - JukeboxTestSuite.java

JukeboxTestSuite.java

```java
package jukebox;


import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

/**
 *
 * @author Sai Harshinee
 */
@RunWith(Suite.class)
@Suite.SuiteClasses({JukeboxTest.class, DatabaseTest.class, SongTest.class})
public class JukeboxTestSuite {

    @BeforeClass
    public static void setUpClass() throws Exception {
    }

    @AfterClass
    public static void tearDownClass() throws Exception {
    }

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

}
```

JUNIT test cases - JukeboxTestSuite.java Output



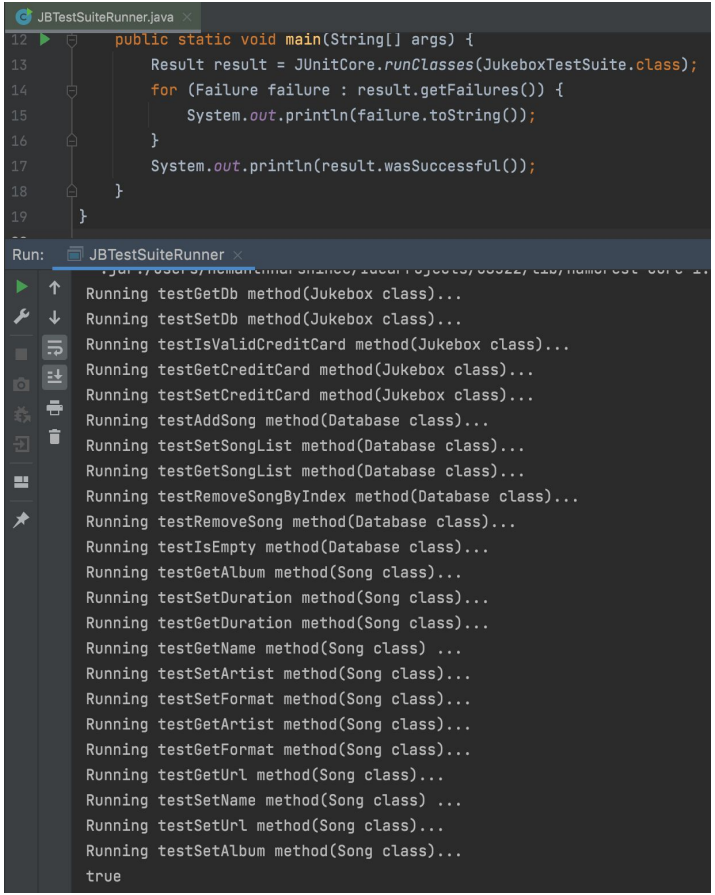
All 25 test cases from SongTest, DatabaseTest, JukeboxTest are **PASSED**

JUNIT test cases - JBTestSuiteRunner.java

JBTestSuiteRunner.java

```java
package jukebox;

/**
 * @author Sai Harshinee
 */
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;


public class JBTestSuiteRunner {
    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(JukeboxTestSuite.class);
        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }
        System.out.println(result.wasSuccessful());
    }
}
```

JUNIT test cases - JBTestSuiteRunner.java Output



Conclusion

JUNIT was used for unit testing of all methods in particular classes.

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/qa/junit_tutorialspoint/hw/Shivani_Dave/CS522A_Wk10_15119_ShivaniDave/CS522A_Wk10_15119_ShivaniDave/Test%20File%20Source%20Codes.txt

Link to view the presentation

https://docs.google.com/presentation/d/1HRi6wW087mgymBmxq5HniWGfDTVkjt7eHuxkcRHTreM/edit?usp=sharing