

# 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 emphasizes 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 increases 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

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "CS522". It contains a ".idea" folder, "lib" with "hamcrest-core-1.3.jar" and "junit-4.13.1.jar", an "out" directory with "production" and "test", and a "src" directory with "main" containing "java" and "Jukebox" (which contains "Cricket.mp3", "Database", "Jukebox", "Kannala.mp3", and "Song"). There is also a "test" directory and an "External Libraries" section.
- Code Editor:** The "Song.java" file is open. The code defines a "Song" class with fields for name, artist, album, url, format, and duration. It includes a constructor, a "play()" method that uses a "Player" to play the MP3 file, and a "main" driver method that creates a "Song" object for "Kadhal Cricket" by "Kharesma Ravichandran" from the album "Thani Oruvan" and plays it.
- Run Output:** The "Run" tab shows the command run: "/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea\_rt.jar -Dfile.encoding=UTF-8 -classpath /Users/hemanthharshinee/IdeaProjects/CS522/out/production/CS522:/Users/hemanthharshinee/IdeaProjects/CS522/src/main/java/Jukebox Song". The output shows the creation of the song object and the playing of the song.

```
84         "Artist:" + this.getArtist() + "\t" +
85         "Album:" + this.getAlbum() + "\t" +
86         "Format:" + this.getFormat() + "\t" +
87         "Duration:" + this.getDuration();
88     }
89
90     public void play(){
91         try{
92             FileInputStream fis = new FileInputStream(this.getUrl());
93             Player playMP3 = new javazoom.jl.player.Player(fis);
94             playMP3.play();
95         }catch(Exception e){
96             System.out.println(e);
97         }
98     }
99 //Driver
100    public static void main(String[] args){
101        System.out.println("Creating Song Object");
102        Song song1=new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
103                             album: "Thani Oruvan", url: "src/main/java/Jukebox/Cricket.mp3", format: "Mp3", duration: 214);
104        System.out.println("Playing Song");
105        song1.play();
106    }

```

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

# Code - Database.java

The screenshot shows the IntelliJ IDEA interface with the Database.java file open in the center editor pane. The code implements a Database class that adds songs to a list and plays them. The project structure on the left shows files like Jukebox.java, Song.java, and various resources like mp3 files and jar files in the lib directory.

```
72     album: "Thani Oruvan", url: "src/main/java/Jukebox/Cricket.mp3", format: "Mp3", duration: 214
73     List<Song> tempSongList= new ArrayList<Song>();
74     tempSongList.add(song2);
75     tempSongList.add(song1);
76
77     Database tempDB= new Database(tempSongList);
78     tempDB.toString();
79     System.out.println("\nAdding Song ");
80     tempDB.addSong(song3);
81     tempDB.toString();
82     System.out.println("Playing Complete SongList");
83     tempDB.play();
84
85     System.out.println("Playing Song @ index 2");
86     tempDB.play( index: 2);
87 }
88 }
```

The bottom run tab shows the command to run the application and the resulting console output. The output shows the song list, the addition of a new song ('Kadhal Cricket'), the updated song list, and the playing of the complete list.

```
/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/production/Jukebox
.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
```

# Code - Jukebox.java

The screenshot shows the IntelliJ IDEA interface with the project structure and code editor for the Jukebox.java file.

**Project Structure:**

- 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

**Code Editor (Jukebox.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);
}
```

**Run Output:**

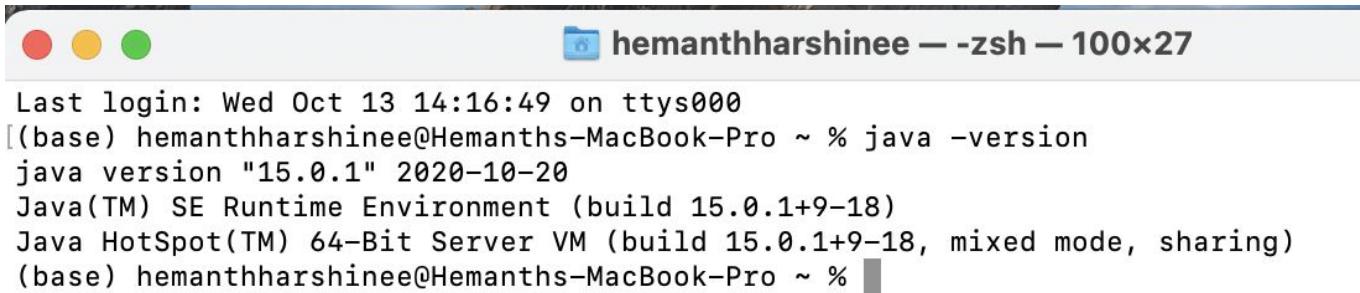
```
/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/hemanthharshinee/IdeaProjects/CS522/out/production/Jukebox Jukebox
Validity of CC : false
Song List:
=====
0: Name: Kadhal Cricket    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
```

# Environmental Setup (MacOS)

## Step 1: Verify Java installation

Type java -version to verify Java installation



```
Last login: Wed Oct 13 14:16:49 on ttys000
(base) hemanthharshinee@Hemanths-MacBook-Pro ~ % java -version
java version "15.0.1" 2020-10-20
Java(TM) SE Runtime Environment (build 15.0.1+9-18)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.1+9-18, mixed mode, sharing)
(base) hemanthharshinee@Hemanths-MacBook-Pro ~ %
```

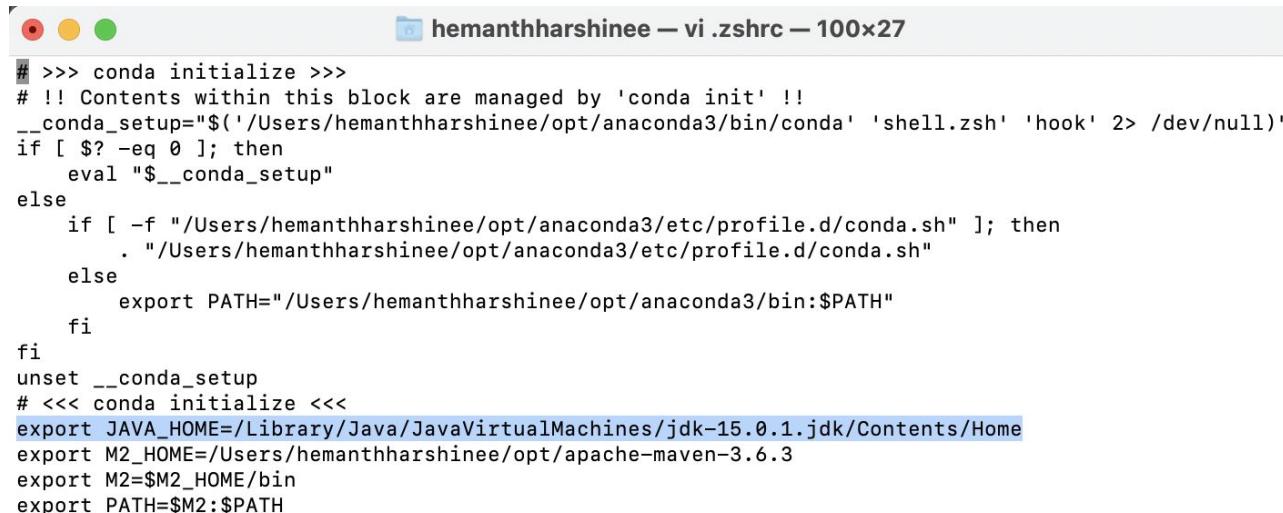
# Environmental Setup (MacOS)

## 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.



```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/Users/hemanthharshinee/opt/anaconda3/bin/conda' 'shell.zsh' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/Users/hemanthharshinee/opt/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/Users/hemanthharshinee/opt/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/Users/hemanthharshinee/opt/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home
export M2_HOME=/Users/hemanthharshinee/opt/apache-maven-3.6.3
export M2=$M2_HOME/bin
export PATH=$M2:$PATH
```

# Environmental Setup (MacOS)

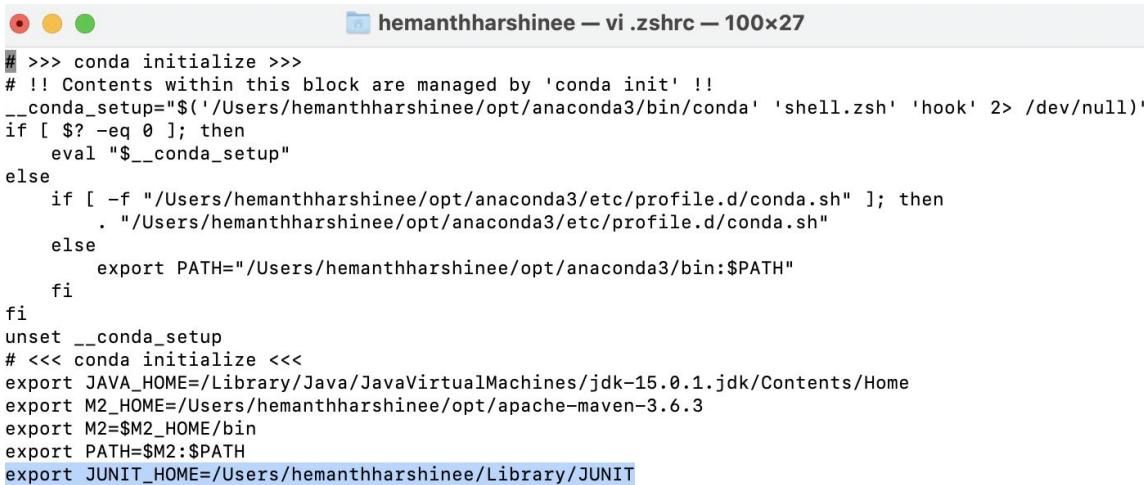
## Step 3: Download Junit archive

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
```



```
# >>> conda initialize >>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/Users/hemanthharshinee/opt/anaconda3/bin/conda' 'shell.zsh' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/Users/hemanthharshinee/opt/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/Users/hemanthharshinee/opt/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/Users/hemanthharshinee/opt/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home
export M2_HOME=/Users/hemanthharshinee/opt/apache-maven-3.6.3
export M2=$M2_HOME/bin
export PATH=$M2:$PATH
export JUNIT_HOME=/Users/hemanthharshinee/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:
```



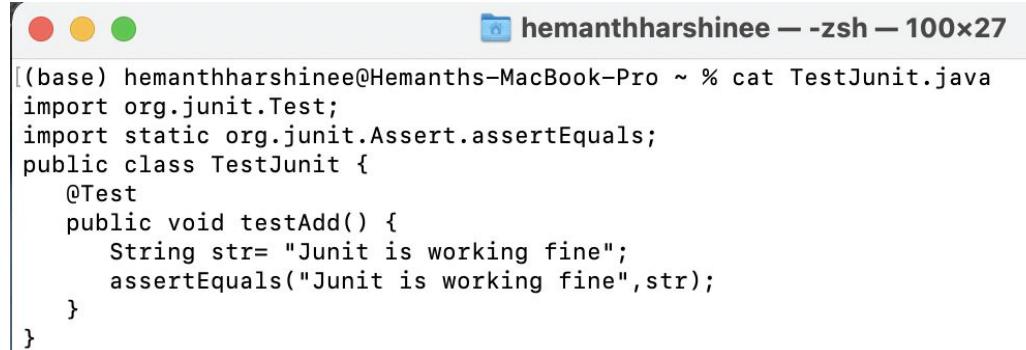
```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/Users/hemanthharshinee/opt/anaconda3/bin/conda' 'shell.zsh' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/Users/hemanthharshinee/opt/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/Users/hemanthharshinee/opt/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/Users/hemanthharshinee/opt/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home
export M2_HOME=/Users/hemanthharshinee/opt/apache-maven-3.6.3
export M2=$M2_HOME/bin
export PATH=$M2:$PATH
export JUNIT_HOME=/Users/hemanthharshinee/Library/JUNIT
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);  
    }  
}
```



The screenshot shows a terminal window with the following details:

- Title bar: hemanthharshinee — zsh — 100x27
- Icon: A small icon with three colored circles (red, yellow, green).
- Content area:
  - Text: [(base) hemanthharshinee@Hemanths-MacBook-Pro ~ % cat TestJUnit.java
  - Java code:

```
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);  
    }  
}
```

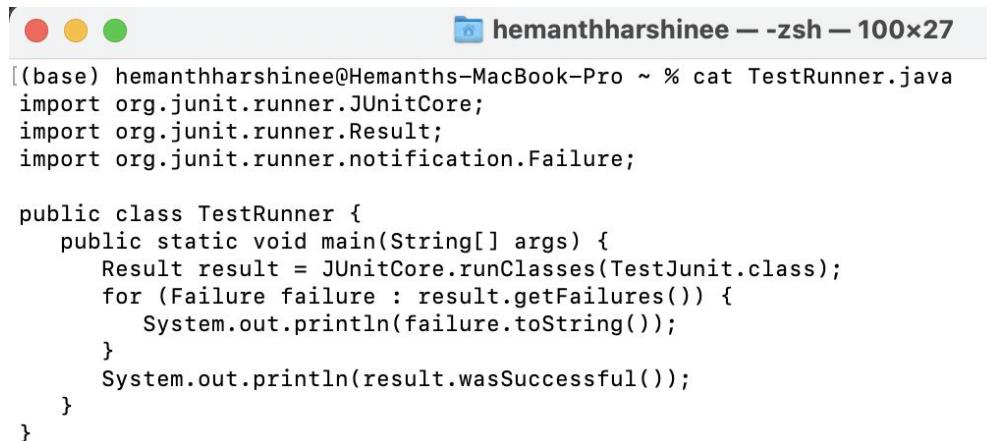
# Environmental Setup (MacOS)

## 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());
    }
}
```



The screenshot shows a terminal window on a Mac OS X desktop. The window title is "hemanthharshinee — zsh — 100x27". The terminal prompt is "(base) hemanthharshinee@Hemanths-MacBook-Pro ~ %". The user has typed the Java code for the TestRunner class into the terminal. The code is identical to the one shown in the previous code block. The terminal shows the code being entered line by line.

```
(base) hemanthharshinee@Hemanths-MacBook-Pro ~ % cat TestRunner.java
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**



hemanthharshinee — zsh — 100x27

```
((base) hemanthharshinee@Hemanths-MacBook-Pro ~ % javac TestJUnit.java TestRunner.java
((base) hemanthharshinee@Hemanths-MacBook-Pro ~ % java TestRunner
true
(base) hemanthharshinee@Hemanths-MacBook-Pro ~ %
```

# JUNIT test cases - SongTest.java

```
1 package jukebox;
2
3 import Jukebox.Song;
4 import org.junit.After;
5 import org.junit.AfterClass;
6 import org.junit.Before;
7 import org.junit.BeforeClass;
8 import org.junit.Test;
9
10 import static org.junit.Assert.*;
11
12 /**
13 * @author Sai Harshinee
14 */
15
16
17 public class SongTest {
18     private Song testSong1, testSong2;
19
20     public SongTest() {
21     }
22
23     @BeforeClass
24     public static void setUpClass() {
25     }
26
27     @AfterClass
28     public static void tearDownClass() {
29     }
30 }
```

# JUNIT test cases - SongTest.java

```
SongTest.java ×
30
31     @Before
32     public void setUp() {
33         testSong1 = new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
34                             album: "Thani Oruvan", url: "Cricket.mp3", format: "Mp3", duration: 214);
35         testSong2 = new Song( name: "Kadhal Cricket", artist: "Kharesma Ravichandran",
36                             album: "Thani Oruvan", url: "Cricket.mp3", format: "Mp3", duration: 214);
37     }
38
39     @After
40     public void tearDown() {
41         testSong1 = null;
42         testSong2 = null;
43     }
44
45     @Test
46     public void testGetName() {
47         System.out.println("Running testGetName method(Song class) ...");
48         assertEquals( message: "Song names are equal", expected: "Kadhal Cricket", testSong1.getName());
49     }
50
51     @Test
52     public void testSetName() {
53         System.out.println("Running testSetName method(Song class) ...");
54         testSong1.setName("Kannala Kannala");
55         assertEquals( message: "Song names are equal", expected: "Kannala Kannala", testSong1.getName());
56     }
57
58     @Test
59     public void testGetArtist() {
60         System.out.println("Running testGetArtist method(Song class)...");
61         assertEquals( message: "Song artists are equal", expected: "Kharesma Ravichandran", testSong1.getArtist());
62     }
63
```

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

# JUNIT test cases - SongTest.java

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

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

The screenshot shows the IntelliJ IDEA interface. The top part displays the `SongTest.java` file with several test methods: `testGetAlbum`, `testSetDuration`, `testGetDuration`, `testGetName`, `testSetArtist`, `testGetFormat`, `testGetArtist`, `testGetFormat`, `testGetUrl`, `testSetName`, `testIsLong1`, `testIsLong2`, `testSetUrl`, and `testSetAlbum`. The bottom part shows the `Run` tool window with the `SongTest` configuration selected. The output pane shows the results of 14 tests, all of which passed in 10 ms. The log pane at the bottom shows the individual test executions.

```
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 }
```

Run: SongTest

Test	Time	Details
testGetAlbum	8 ms	/Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/java -ea -Didea.testName=testGetAlbum -Djavaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=61292:/Appli
testSetDuration	0 ms	\.encoding=UTF-8 -classpath /Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar:/app/Contents/plugins/junit/lib/junit5-rt.jar:/Applications/IntelliJ IDEA CE.app/Content
testGetDuration	0 ms	\.jar:/Users/hemanthharshinee/IdeaProjects/CS522/out/test/CS522:/Users/hemanthharshinee
testGetName	1 ms	\/hemanthharshinee/Downloads/jlayer-1.0.1.jar:/Users/hemanthharshinee/IdeaProjects/CS522:/jar:/Users/hemanthharshinee/IdeaProjects/CS522/lib/hamcrest-core-1.3.jar com.intellij
testSetArtist	0 ms	.SongTest
testGetFormat	0 ms	Running testGetAlbum method(Song class)...
testGetArtist	0 ms	Running testSetDuration method(Song class)...
testGetFormat	0 ms	Running testGetDuration method(Song class)...
testGetUrl	0 ms	Running testGetName method(Song class) ...
testSetName	0 ms	Running testSetArtist method(Song class)...
testIsLong1	0 ms	Running testGetFormat method(Song class)...
testIsLong2	1 ms	Running testGetArtist method(Song class)...
testSetUrl	0 ms	Running testGetFormat method(Song class)...
testSetAlbum	0 ms	Running testGetUrl method(Song class)...

Tests passed: 14

Find Run TODO Problems Debug Sequence Diagram Terminal Build

Tests passed: 14 (moments ago)

All 14 test cases  
of are PASSED.

# JUNIT test cases - DatabaseTest.java



```
DatabaseTest.java x
1 package jukebox;
2
3 import Jukebox.Database;
4 import Jukebox.Song;
5 import org.junit.After;
6 import org.junit.AfterClass;
7 import org.junit.Before;
8 import org.junit.BeforeClass;
9 import org.junit.Test;
10
11 import static org.junit.Assert.*;
12 /**
13 *
14 * @author Sai Harshinee
15 */
16 public class DatabaseTest {
17     private java.util.List songList, songList1;
18     private Song testSong1, testSong2, testSong3;
19     private Database testDB, testDB1;
20
21     public DatabaseTest() {
22     }
23
24     @BeforeClass
25     public static void setUpClass() {
26     }
27
28     @AfterClass
29     public static void tearDownClass() {
30     }
31 }
```

# JUNIT test cases - DatabaseTest.java

```
DatabaseTest.java ×
31
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
53     @After
54     public void tearDown() {
55         songList = null;
56         songList1 = null;
57         testDB = null;
58         testDB1 = null;
59     }
}
```

# JUNIT test cases - DatabaseTest.java

```
DatabaseTest.java x
60
61     @Test
62     public void testGetSongList() {
63         System.out.println("Running testGetSongList method(Database class)...");  

64         assertEquals( message: "get Songlist",songList,testDB.getSongList());
65     }
66     @Test
67     public void testSetSongList() {
68         System.out.println("Running testSetSongList method(Database class)...");  

69         testDB1.setSongList(songList1);
70         assertEquals( message: "get Songlist",songList1,testDB1.getSongList());
71     }
72     @Test
73     public void testAddSong() {
74         System.out.println("Running testAddSong method(Database class)...");  

75         testDB1.addSong(testSong3);
76         assertEquals( message: "get Songlist",songList1,testDB1.getSongList());
77     }
78     @Test
79     public void testRemoveSong() {
80         System.out.println("Running testRemoveSong method(Database class)...");  

81         testDB.removeSong(testSong1);
82         assertEquals( message: "get Songlist",songList,testDB.getSongList());
83     }
84     @Test
85     public void testRemoveSongByIndex() {
86         System.out.println("Running testRemoveSongByIndex method(Database class)...");  

87         testDB.removeSong( index: 1);
88         assertEquals( message: "get Songlist",songList,testDB.getSongList());
89     }
90     @Test
91     public void testIsEmpty() {
92         System.out.println("Running testIsEmpty method(Database class)...");  

93         assertEquals( message: "Database is not empty", expected: false,testDB1.isEmpty());
94     }
```

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

# JUNIT test cases - DatabaseTest.java Output

The screenshot shows the IntelliJ IDEA interface with the DatabaseTest.java file open in the editor. The code contains six test methods: testAddSong, testSetSongList, testGetSongList, testRemoveSongByIndex, testRemoveSong, and testIsEmpty. The run results show all tests passed in 6 ms.

```
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 }  
95  
Run: DatabaseTest x  
▶ ✓ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ Tests passed: 6 of 6 tests – 6 ms  
DatabaseTest (jukebox) 6 ms /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home/bin/  
└--javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar  
    .encoding=UTF-8 -classpath /Applications/IntelliJ IDEA CE.app/Contents/plugins/junit/junit5-rt.jar:/Applications/IntelliJ IDEA CE.app/Contents/plugins/junit/junit5-engine.jar:/Users/hemanthharshinee/IdeaProjects/CS522/out/test/CS522:/Users/hemanthharshinee/Downloads/jlayer-1.0.1.jar:/Users/hemanthharshinee/IdeaProjects/CS522/lib/hamcrest-core-1.3.jar:/Users/hemanthharshinee/IdeaProjects/CS522/lib/DatabaseTest  
    Running testAddSong method(Database class)...  
    Running testSetSongList method(Database class)...  
    Running testGetSongList method(Database class)...  
    Running testRemoveSongByIndex method(Database class)...  
    Running testRemoveSong method(Database class)...  
    Running testIsEmpty method(Database class)...  
  
Process finished with exit code 0  
Tests passed: 6 (moments ago)
```

All 6 test cases  
are PASSED

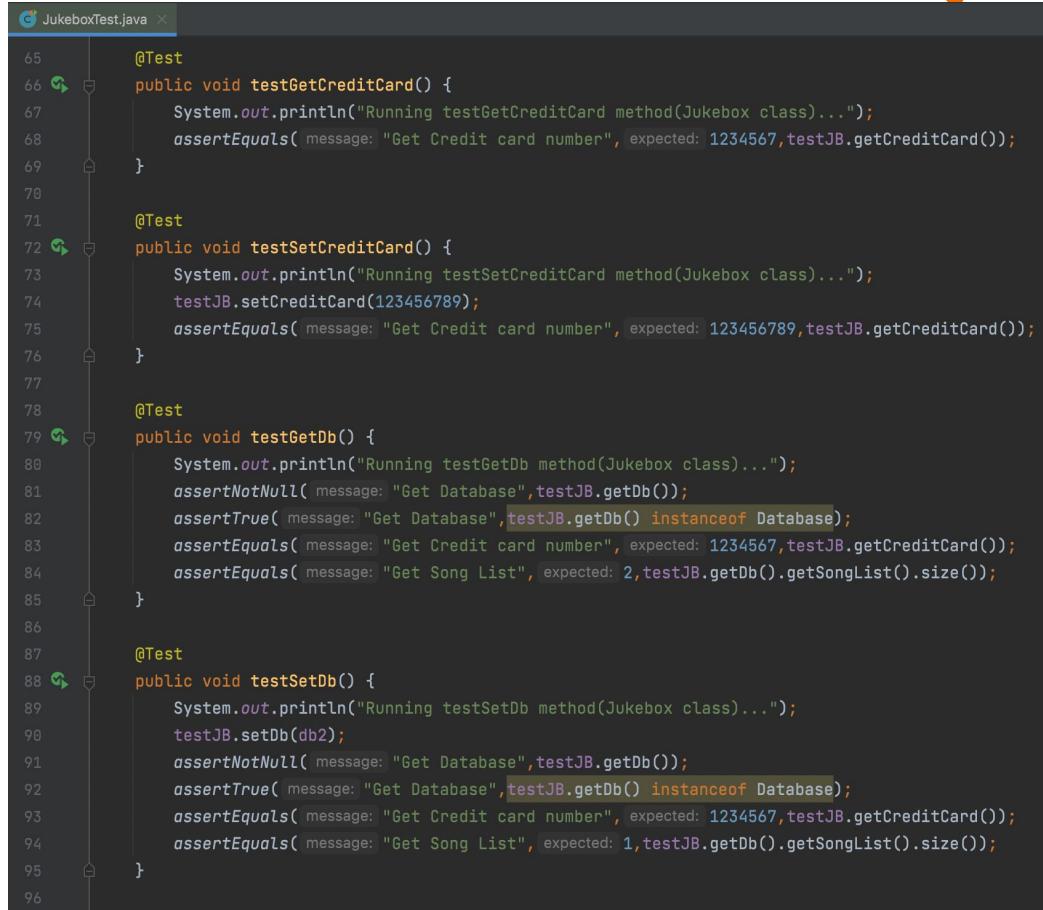
# JUNIT test cases - JukeboxTest.java

```
1  package jukebox;
2
3  import Jukebox.Database;
4  import Jukebox.Jukebox;
5  import Jukebox.Song;
6  import org.junit.After;
7  import org.junit.AfterClass;
8  import org.junit.Before;
9  import org.junit.BeforeClass;
10 import org.junit.Test;
11
12 import java.util.List;
13
14 import static org.junit.Assert.*;
15
16 /**
17 *
18 * @author Sai Harshinee
19 */
20
21 public class JukeboxTest {
22     private Jukebox testJB;
23     private Database db1, db2;
24     private List songlistJB1;
25     private List songListJB2;
26     private Song song1,song2;
27     private int ccn=1234567;
28
29     public JukeboxTest() {
30     }
31
32     @BeforeClass
33     public static void setUpClass() {
34 }
```

# 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; }
62
63
64
```

# JUNIT test cases - 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

The screenshot shows the IntelliJ IDEA interface. The top part displays the code for `JukeboxTest.java`, which contains two test methods: `testSetDb()` and `testIsValidCreditCard()`. The bottom part shows the 'Run' tool window with the results of the test execution.

```
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
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}
```

Run: JukeboxTest

Test	Time	Result
testGetDb	5 ms	Passed
testSetDb	0 ms	Passed
testIsValidCreditCard	0 ms	Passed
testGetCreditCard	0 ms	Passed
testSetCreditCard	0 ms	Passed

Tests passed: 5

Process finished with exit code 0

All 5 test cases  
are PASSED

# JUNIT test cases - JukeboxTestSuite.java

```
1 package jukebox;
2
3
4 import org.junit.After;
5 import org.junit.AfterClass;
6 import org.junit.Before;
7 import org.junit.BeforeClass;
8 import org.junit.runner.RunWith;
9 import org.junit.runners.Suite;
10
11 /**
12 * @author Sai Harshinee
13 */
14
15 @RunWith(Suite.class)
16 @Suite.SuiteClasses({JukeboxTest.class, DatabaseTest.class, SongTest.class})
17 public class JukeboxTestSuite {
18
19     @BeforeClass
20     public static void setUpClass() throws Exception {
21     }
22
23     @AfterClass
24     public static void tearDownClass() throws Exception {
25     }
26
27     @Before
28     public void setUp() throws Exception {
29     }
30
31     @After
32     public void tearDown() throws Exception {
33     }
34
35 }
```

# JUNIT test cases - JukeboxTestSuite.java Output

The screenshot shows an IDE interface with two main panes. The top pane displays the code for `JukeboxTestSuite.java`, which includes `@Before` and `@After` annotations. The bottom pane shows the execution results of the suite, with a summary message "Tests passed: 25 of 25 tests – 10 ms" and a detailed log of 25 individual test cases from three classes: `JukeboxTest`, `DatabaseTest`, and `SongTest`. Each test case is marked with a green checkmark and its execution time.

```
27     @Before
28     public void setUp() throws Exception {
29     }
30
31     @After
32     public void tearDown() throws Exception {
33     }

```

Run: JukeboxTestSuite

Tests passed: 25 of 25 tests – 10 ms

JukeboxTestSuite (juket 10 ms)

- > JukeboxTest 2ms
- > DatabaseTest 1ms
- > SongTest 7ms
  - ✓ testGetAlbum 1ms
  - ✓ testSetDuration 1ms
  - ✓ testGetDuration 0ms
  - ✓ testGetName 0ms
  - ✓ testSetArtist 0ms
  - ✓ testSetFormat 1ms
  - ✓ testGetArtist 0ms
  - ✓ testGetFormat 0ms
  - ✓ testGetUrl 0ms
  - ✓ testSetName 1ms
  - ✓ testIsLong1 1ms
  - ✓ testIsLong2 1ms
  - ✓ testSetUrl 1ms
  - ✓ testSetAlbum 0ms

Running testGetDb method(Jukebox class)...

Running testSetDb method(Jukebox class)...

Running testIsValidCreditCard method(Jukebox class)...

Running testGetCreditCard method(Jukebox class)...

Running testAddSong method(Database class)...

Running testSetSongList method(Database class)...

Running testGetSongList method(Database class)...

Running testRemoveSongByIndex method(Database class)...

Running testRemoveSong method(Database class)...

Running testIsEmpty method(Database class)...

Running testGetAlbum method(Song class)...

Running testSetDuration method(Song class)...

Running testGetDuration method(Song class)...

Running testGetName method(Song class) ...

Running testSetArtist method(Song class)...

Running testSetFormat method(Song class)...

Running testGetArtist method(Song class)...

Running testGetFormat method(Song class)...

Running testGetUrl method(Song class)...

Running testSetName method(Song class) ...

Running testSetUrl method(Song class)...

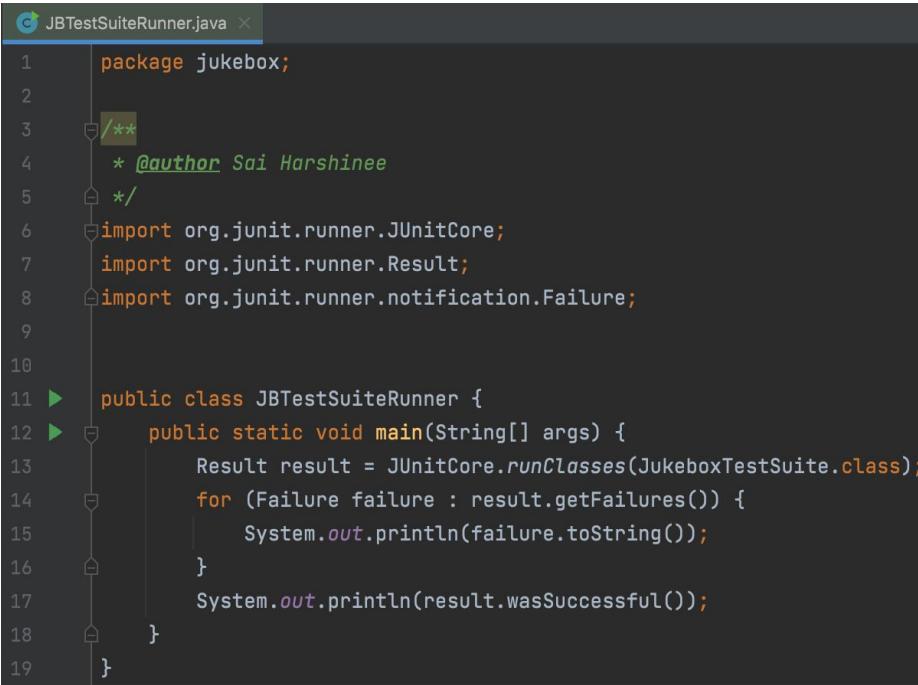
Running testSetAlbum method(Song class)...

Process finished with exit code 0

Tests passed: 25 (moments ago)

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

# JUNIT test cases - JBTestSuiteRunner.java



```
JBTestSuiteRunner.java x
1 package jukebox;
2
3 /**
4 * @author Sai Harshinee
5 */
6 import org.junit.runner.JUnitCore;
7 import org.junit.runner.Result;
8 import org.junit.runner.notification.Failure;
9
10
11 public class JBTestSuiteRunner {
12     public static void main(String[] args) {
13         Result result = JUnitCore.runClasses(JukeboxTestSuite.class);
14         for (Failure failure : result.getFailures()) {
15             System.out.println(failure.toString());
16         }
17         System.out.println(result.wasSuccessful());
18     }
19 }
```

# JUNIT test cases - JBTestSuiteRunner.java Output

The screenshot shows an IDE interface with the following details:

- Code Editor:** The top half displays the `JBTestSuiteRunner.java` file. The code is a simple main method that runs JUnit tests and prints the results to `System.out`.
- Run Output:** The bottom half shows the "Run" tool window with the title "JBTestSuiteRunner". It lists the execution of various test methods across different classes. The output is as follows:
  - Running testGetDb method(Jukebox class)...
  - Running testSetDb method(Jukebox class)...
  - Running testIsValidCreditCard method(Jukebox class)...
  - Running testGetCreditCard method(Jukebox class)...
  - Running testSetCreditCard method(Jukebox class)...
  - Running testAddSong method(Database class)...
  - Running testSetSongList method(Database class)...
  - Running testGetSongList method(Database class)...
  - Running testRemoveSongByIndex method(Database class)...
  - Running testRemoveSong method(Database class)...
  - Running testIsEmpty method(Database class)...
  - Running testGetAlbum method(Song class)...
  - Running testSetDuration method(Song class)...
  - Running testGetDuration method(Song class)...
  - Running testGetName method(Song class) ...
  - Running testSetArtist method(Song class)...
  - Running testSetFormat method(Song class)...
  - Running testGetArtist method(Song class)...
  - Running testGetFormat method(Song class)...
  - Running testGetUrl method(Song class)...
  - Running testSetName method(Song class) ...
  - Running testSetUrl method(Song class)...
  - Running testSetAlbum method(Song class)...

# 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/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](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/1HRi6wW087mgymBmxq5HniWGfDTVkj7eHuxkcRHTreM/edit?usp=sharing>