



NORTHWESTERN POLYTECHNIC
UNIVERSITY

JukeBox(Junit TestSuite)

Prepared For:

Mr. Henry Chang
Software Quality Assurance and Test Automation CS522
Fall 2021
Northwestern Polytechnic University

Prepared By:

Ms. Nagalla, Santhi Sree ID:19568

Table of Contents

- *Quality Assurance in Software Testing*
- *Junit*
- *Jukebox*
 - *Execute JukeBox Program using Eclipse*
 - *Run Junit Test cases using Eclipse*
 - *Run Test Suite using Eclipse*
 - *Compile Jukebox Code on Command Line*
 - *Run Jukebox Code on Command Line*
- *References*

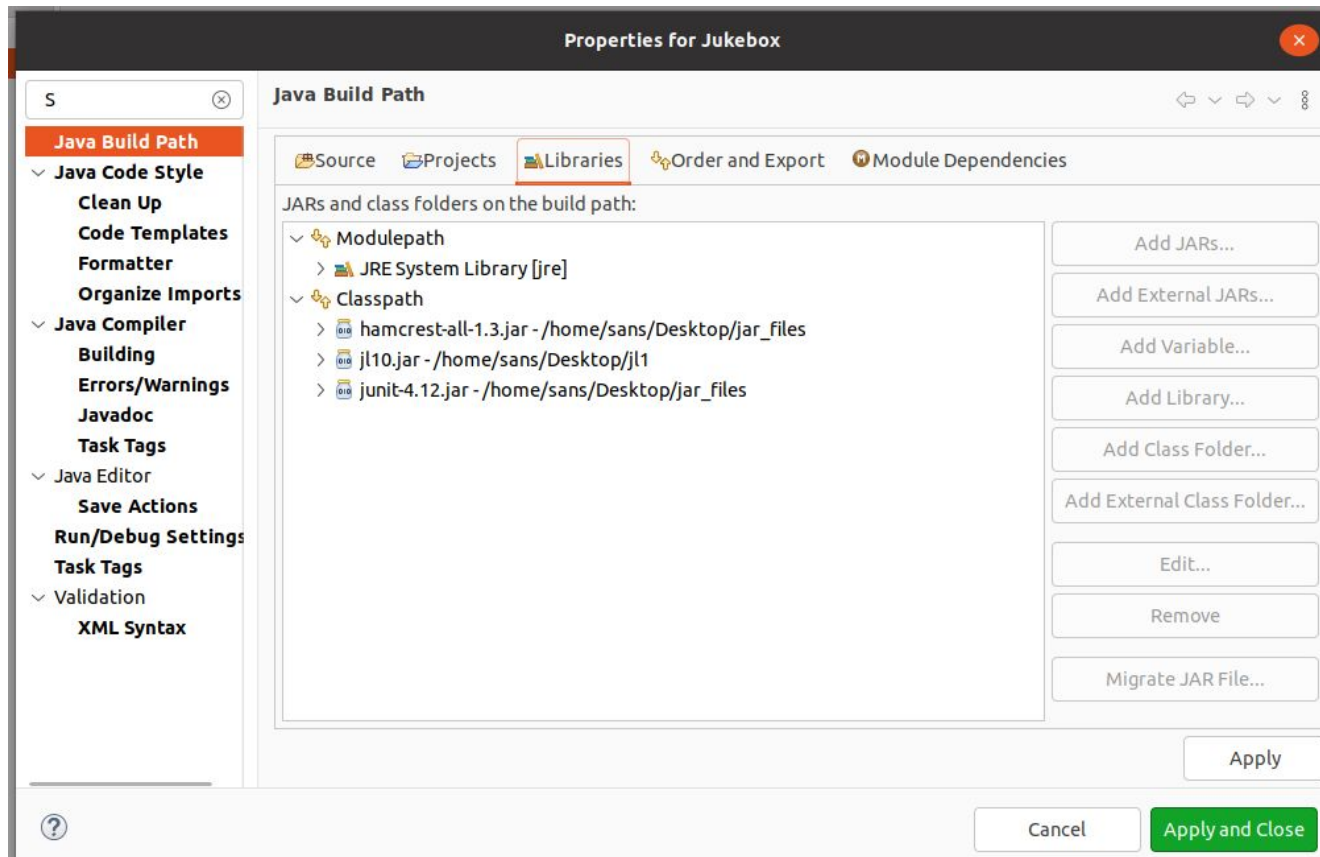
Quality Assurance in Software Testing

- Quality Assurance in Software Testing is defined as a procedure to ensure the quality of software products or services provided to the customers by an organization. Quality assurance focuses on improving the software development process and making it efficient and effective as per the quality standards defined for software products. Quality Assurance is popularly known as QA Testing.

JUnit

JUnit as a tool to write simple tests against small bits of logical behavior in your code. The tests you write involve:

- Setting up some context (put the system into a certain state, by perhaps populating elements or invoking some methods).
- Executing whatever it is you want to verify.
- Asserting against conditions you expect to hold true at this point. If an assertion does not hold true, the test fails.



- Create Project and add .Java files in project.
- Add Jar files(Junit ,hamcrest, jl10) in “Java Build path” under Libraries section.

Execute JukeBox Program using Eclipse

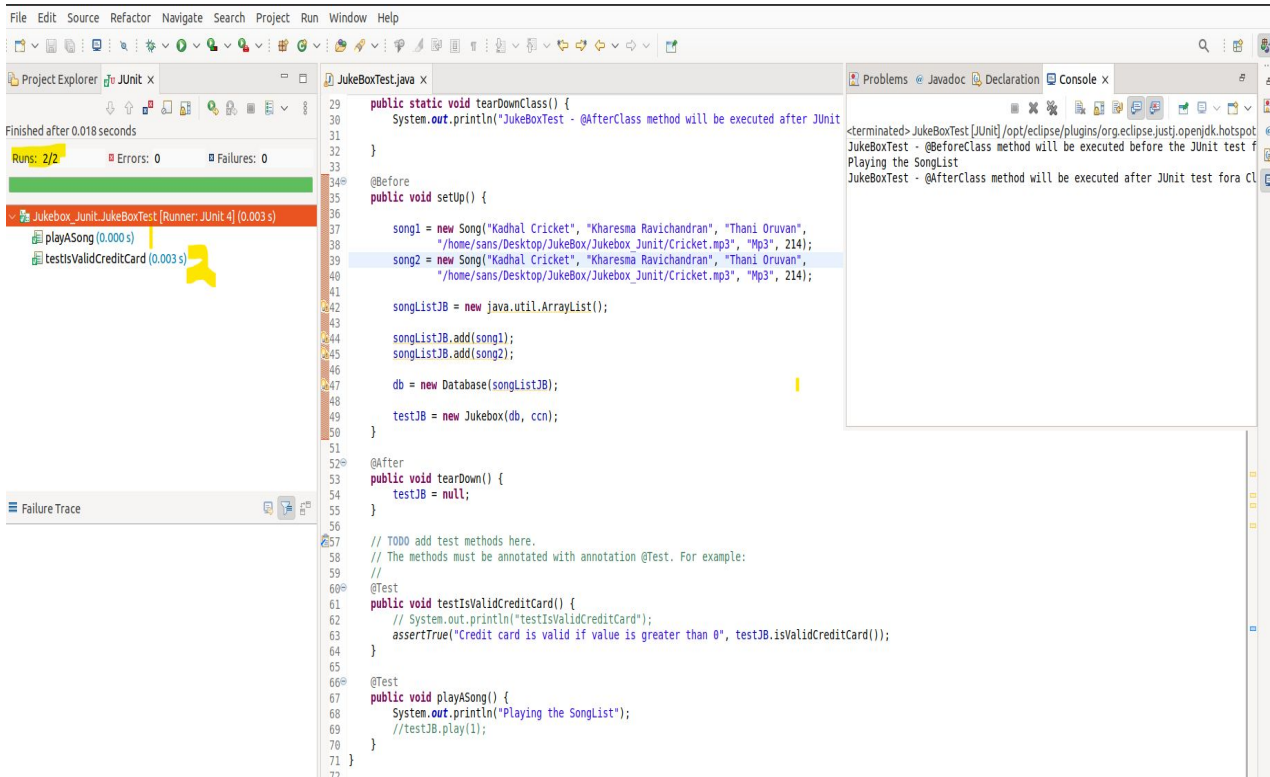
The screenshot displays the Eclipse IDE interface with the following components:

- Package Explorer (Left):** Shows the project structure. The 'Jukebox' package is expanded, revealing 'src' and 'Jukebox_Unit'. The 'Jukebox_Unit' package is further expanded, showing 'Database.java', 'Jukebox.java', and 'Song.java' (which is selected and highlighted in orange).
- Editor (Center):** Displays the source code of 'Song.java'. The code defines a 'Song' class with attributes (name, artist, album, url, format, duration) and methods (constructor, getName, setName, getArtist, setArtist, getAlbum). The code is as follows:

```
1 package Jukebox_Package;
2
3 import java.io.FileInputStream;
4
5
6
7
8
9 public class Song {
10     String name;
11     String artist;
12     String album;
13     String url;
14     String format;
15     // Duration of Song in seconds
16     int duration;
17     public Song(){
18
19     }
20     public Song(String name, String artist, String album, String url,
21                 String format, int duration) {
22         super();
23         this.name = name;
24         this.artist = artist;
25         this.album = album;
26         this.url = url;
27         this.format = format;
28         this.duration = duration;
29     }
30
31     public String getName() {
32         return name;
33     }
34
35     public void setName(String name) {
36         this.name = name;
37     }
38
39     public String getArtist() {
40         return artist;
41     }
42
43     public void setArtist(String artist) {
44         this.artist = artist;
45     }
46
47     public String getAlbum() {
48         return album;
```
- Console (Right):** Shows the output of the program execution. The output is:

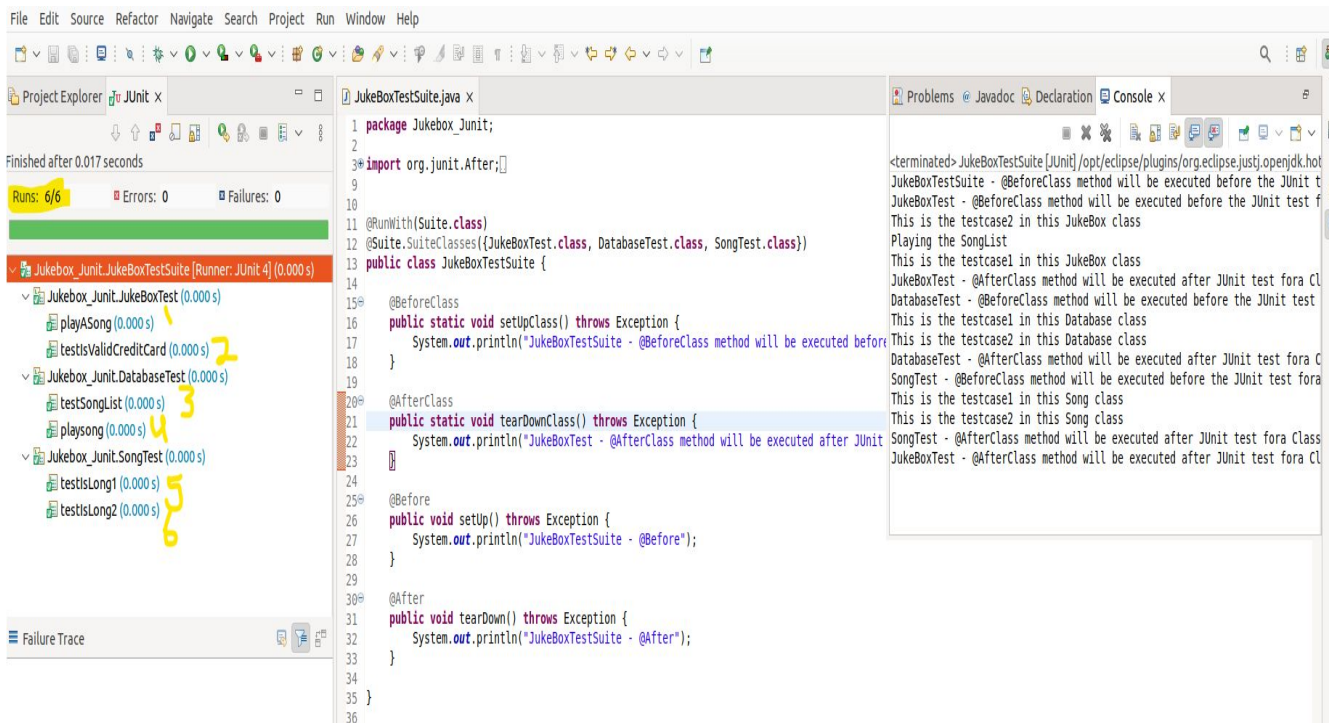
```
Song [Java Application] /opt/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.lin
Creating Song Object
Playing Song
```

Run Junit Test cases using Eclipse



- Write Junit Test Cases(SongTest,DatabaseTest,JukeBoxTest) and RunAs - > JunitTest.
- Follow below link to create Junit Test cases-
<https://www.softwaretestinghelp.com/junit-tests-examples/>

Run Test Suite using Eclipse

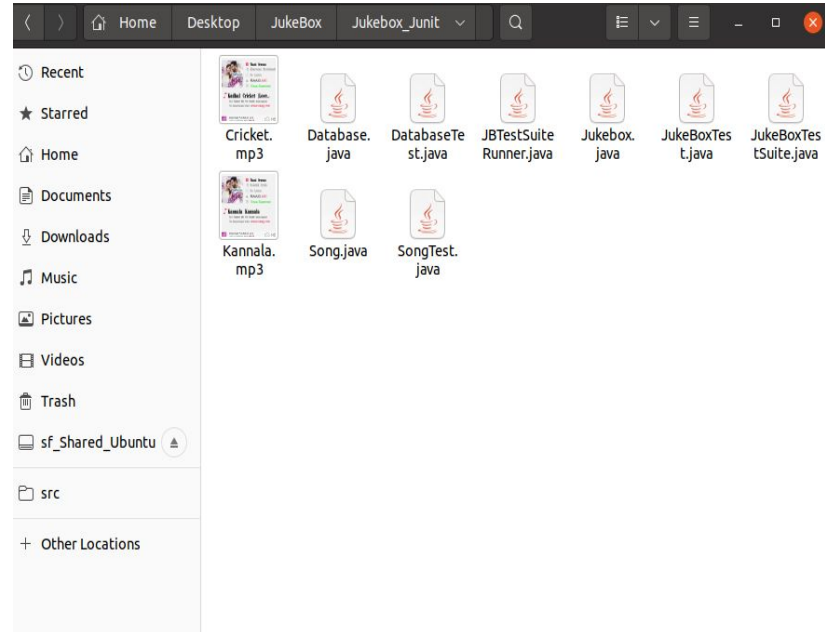


- Test suite is used to bundle a few test cases and run them together. In JUnit, both `@RunWith` and `@Suite` annotations are used to run the suite tests. This example having three test classes, `JukeBoxTest` & `DatabaseTest`, `SongTest` that run together using `Test Suite`.

Compile Jukebox Code on Command Line

Place all the .java classes in Folder
compile the code with .Jar files(Add .jar
files with path) using below command.

- `javac -classpath`
`./:/home/ssss/Desktop/jl11/jl10.jar:/h`
`ome/ssss/Desktop/jar_files/junit-4.1`
`2.jar:/home/ssss/Desktop/jar_files/h`
`amcrest-all-1.3.jar -d ../bin`
`-sourcepath . *.java`



Run Jukebox Code on Command Line

- .Class files will be generated in the bin folder.
- Go to bin Folder and run the below command.
- `java -classpath`
`./:/home/ssss/Desktop/jl1/jl10.jar:/home/ssss/Desktop/jar_files/hamcrest-all-1.3.jar:/home/ssss/Desktop/jar_files/junit-4.12.jar`
`org.junit.runner.JUnitCore Jukebox_Junit.JukeBoxTestSuite`

```
sans@sans-vb:~/Desktop/JukeBox/bin$ java -classpath ./:/home/sans/Desktop/jl1/jl10.jar:/home/sans/Desktop/jar_files/hamcrest-all-1.3.jar:/home/sans/Desktop/jar_files/junit-4.12.jar org.junit.runner.JUnitCore Jukebox_Junit.JukeBoxTestSuite
JUnit version 4.12
JukeBoxTestSuite - @BeforeClass method will be executed before the JUnit test for a Class starts
JukeBoxTest - @BeforeClass method will be executed before the JUnit test for a Class starts
.This is the testcase2 in this JukeBox class
Playing the SongList
.This is the testcase1 in this JukeBox class
JukeBoxTest - @AfterClass method will be executed after JUnit test for a Class Completed
DatabaseTest - @BeforeClass method will be executed before the JUnit test for a Class starts
.This is the testcase1 in this Database class
.This is the testcase2 in this Database class
DatabaseTest - @AfterClass method will be executed after JUnit test for a Class Completed
SongTest - @BeforeClass method will be executed before the JUnit test for a Class starts
.This is the testcase1 in this Song class
.This is the testcase2 in this Song class
SongTest - @AfterClass method will be executed after JUnit test for a Class Completed
JukeBoxTest - @AfterClass method will be executed after JUnit test for a Class Completed

Time: 0.005

OK (6 tests)
```

```
sans@sans-vb:~/Desktop/JukeBox/bin$ java -classpath ./:/home/sans/Desktop/jl1/jl10.jar:/home/sans/Desktop/jar_files/hamcrest-all-1.3.jar:/home/sans/Desktop/jar_files/junit-4.12.jar org.junit.runner.JUnitC
ore Jukebox_Junit.SongTest
JUnit version 4.12
SongTest - @BeforeClass method will be executed before the JUnit test fora Class starts
.This is the testcase1 in this Song class
.This is the testcase2 in this Song class
SongTest - @AfterClass method will be executed after JUnit test fora Class Completed

Time: 0.002

OK (2 tests)
```

```
sans@sans-vb:~/Desktop/JukeBox/bin$ java -classpath ./:/home/sans/Desktop/jl1/jl10.jar:/home/sans/Desktop/jar_files/hamcrest-all-1.3.jar:/home/sans/Desktop/jar_files/junit-4.12.jar org.junit.runner.JUnitC
ore Jukebox_Junit.DatabaseTest
JUnit version 4.12
DatabaseTest - @BeforeClass method will be executed before the JUnit test fora Class starts
.This is the testcase1 in this Database class
.This is the testcase2 in this Database class
DatabaseTest - @AfterClass method will be executed after JUnit test fora Class Completed

Time: 0.003

OK (2 tests)
```

References

- Github Link -
<https://github.com/santhinagalla/Software-Quality-Assurance-and-Test-Automation/tree/main/QA/Junit/JukeBox>
- <https://www.softwaretestinghelp.com/junit-tests-examples/>
- <https://www.codejava.net/testing/junit-tutorial-for-beginner-with-eclipse>
- <https://courses.cs.washington.edu/courses/cse143/11wi/eclipse-tutorial/junit.shtml>