Junit Test

1. **DatabaseTest**
2. **JukeBoxTest**
3. **SongTest**
4. **JukeBoxTestSuite**
5. **JBTestSuiteRunner**

- **DatabaseTest**

```
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertNull;
import static org.junit.Assert.assertSame;
import static org.junit.Assert.assertTrue;

import java.util.ArrayList;
import java.util.List;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Ignore;
import org.junit.Test;

import com.qa.jukebox.Database;
import com.qa.jukebox.Song;

/**
 *
 * @author Mittal
 */
public class DatabaseTest {
	private List<Song> songList;
	private Song testSong1, testSong2;
	private Database testDB;

	public DatabaseTest() {
	}

	@BeforeClass
```

```
	public static void setUpClass() {
		System.out.println("in before class...DatabasePositiveTest.java");
	}

	@AfterClass
	public static void tearDownClass() {
		System.out.println("in after class...DatabasePositiveTest.java");
	}

	@Before
	public void setUp() {
		songList = new ArrayList<>();

		testSong1 = new Song(“Kalank Nahi”, “Arijit  Singh”, “Kalank”,”kalank.mp3”,
“MP3”, 312);
		testSong2 = new Song("Kannala Kannala", "Kaushik Krish", "Thani Oruvan",
"Kannala.mp3", "Mp3", 215);

		songList.add(testSong1);
		songList.add(testSong2);

		testDB = new Database(songList);
	}

	@After
	public void tearDown() {
		songList = null;
		testDB = null;
	}

	@Test
	public void testDefaultDatabaseConstructor() {
		Database db = new Database();
		assertEquals(testDB.getClass(), db.getClass());

	}

	@Test
  public void testDatabaseParameterizedConstructor() {
		Database db = new Database(songList);
		for(int i = 0; i < testDB.getSongList().size(); i++) {
			assertEquals(testDB.getSongList(i), db.getSongList(i));
		}
  }
```

```
	@Test
	public void testGetSongListReturnsList() {
		songList = new ArrayList<>();

		testSong1 = new Song(“Kalank Nahi”, “Arijit  Singh”, “Kalank”,”kalank.mp3”,
“MP3”, 312);
		testSong2 = new Song("Kannala Kannala", "Kaushik Krish", "Thani Oruvan",
"Kannala.mp3", "Mp3", 215);

		songList.add(testSong1);
		songList.add(testSong2);

		List<Song> result = testDB.getSongList();
		assertEquals(songList, result);

	}
	@Test
	public void testGetSongListReturnsSong() {
		assertSame(testSong1, testDB.getSongList(0));
	}

	@Test
	public void testIsEmpty() {
		assertFalse("List is not empty", testDB.isEmpty());
	}

	@Test
	public void testSetSongList() {
		songList = new java.util.ArrayList();
		Song song1 = new Song(“Kalank Nahi”, “Arijit  Singh”, “Kalank”,”kalank.mp3”,
“MP3”, 312);
		Song song2 = new Song("Kannala Kannala", "Kaushik Krish", "Thani Oruvan",
"Kannala.mp3", "Mp3", 215);

		songList.add(song1);
		songList.add(song2);

		Database testSetterDB = new Database();
		testSetterDB.setSongList(songList);

		List<Song> result = new ArrayList<Song>(testSetterDB.getSongList());
		assertEquals(songList, result);
	}
```

```
	@Test
	public void testAddSong() {
		Song newSong = new Song("hai", "rehman", "jeans", "jeans.mp3", "Mp3", 300);
		testDB.addSong(newSong);

		assertSame("addSong() method tested", newSong,testDB.getSongList(2));
	}


	@Test
	public void testRemoveSongObjSong() {
		Song newSong = new Song("hai", "rehman", "jeans", "jeans.mp3", "Mp3", 300);
		testDB.removeSong(newSong);

		List<Song> result = new ArrayList<Song>(testDB.getSongList());
		assertEquals(songList, result);// assertEquals(expResult, result);

	}

	@Test
	public void testRemoveSongIndex() {
		Song newSong = new Song("hai", "rehman", "jeans", "jeans.mp3", "Mp3", 300);
		testDB.addSong(newSong);
		testDB.removeSong(2);

		List<Song> result = new ArrayList<Song>(testDB.getSongList());
		assertEquals(songList, result);// assertEquals(expResult, result);

	}


	@Ignore

	@Test(expected =  Exception.class)
public void testExceptionPLAYIndex() {
		 Song songnull = new Song(null,null,null,null,null,0);
		 testDB.addSong(songnull);
		 testDB.play(2);
}

	@Ignore

	@Test(expected =  Exception.class)
```

```
    public void testExceptionPLAY() {
                    Song songnull = new Song(null,null,null,null,null,0);
                    testDB.addSong(songnull);
                    testDB.play(2);
    }

            @Test
            public void testMtoString() {
                    Database testDB2 = new Database(songList);
             assertEquals(testDB.toString(), testDB2.toString());
    }



}
```

- **JukeBoxTest**

```
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Ignore;
import org.junit.Test;

import com.qa.jukebox.Database;
import com.qa.jukebox.Jukebox;
import com.qa.jukebox.Song;

import static org.junit.Assert.*;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Mittal
 */
public class JukeBoxTest {
	private Jukebox testJB;
	private Database db;
	private java.util.List songListJB;
	private Song song1, song2;
	private int ccn = 123;

	public JukeBoxTest() {
	}

	@BeforeClass
	public static void setUpClass() {
		System.out.println("in before class...JukeBoxPositiveTest.java");
	}

	@AfterClass
	public static void tearDownClass() {
		System.out.println("in before class...JukeBoxPositiveTest.java");
	}

	@Before
	public void setUp() {
```

```
		song1 = new Song(“Kalank Nahi”, “Arijit  Singh”, “Kalank”,”kalank.mp3”, “MP3”, 312);
		song2 = new Song("Kannala Kannala", "Kaushik Krish", "Thani Oruvan", "Kannala.mp3", "Mp3", 215);

		songListJB = new java.util.ArrayList();

		songListJB.add(song1);
		songListJB.add(song2);

		db = new Database(songListJB);

		testJB = new Jukebox(db, ccn);
	}

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

	@Test
	public void testDefaultJukeboxConstructor() {
		Jukebox jk = new Jukebox();
		assertEquals(testJB.getClass(), jk.getClass());

	}


	@Test
  public void testJukeboxParameterizedConstructor() {
		Jukebox jk = new Jukebox(db, ccn);
		for(int i = 0; i < jk.getDb().getSongList().size(); i++) {
			assertEquals(jk.getDb().getSongList(0), testJB.getDb().getSongList(0));
		}
		assertEquals(jk.getCreditCard(), testJB.getCreditCard());
  }


	@Test
	public void testGetDB() {
		assertSame(db, testJB.getDb());
	}
```

```
	@Test
	public void testSetDB() {
		Song testSong1 = new Song“Kalank Nahi”, “Arijit  Singh”, “Kalank”,”kalank.mp3”,
“MP3”, 312);
		List songListTest = new ArrayList<Song>();
		songListTest.add(testSong1);
		Database testDBSetter = new Database();
		testDBSetter.setSongList(songListTest);
		Jukebox testJBsetter = new Jukebox(testDBSetter, 444);

		assertSame(testDBSetter, testJBsetter.getDb());

	}


  @Test
  public void testSetCreditCard() {
	testJB.setCreditCard(234);
    assertEquals(234, testJB.getCreditCard());
  }


  @Test
  public void testGetCreditcard() {
	assertEquals(123, testJB.getCreditCard());
  }

	@Test
	public void testIsValidCreditCard() {
		assertTrue("creditcard has value > 0",testJB.isValidCreditCard());
		//assertEquals("Credit card is valid if value is greater than 0", true,
testJB.isValidCreditCard());
	}


	@Ignore

	@Test(expected =  Exception.class)
  public void testExceptionPLAYIndex() {
		Song song=db.getSongList(0);
    song.play();
  }

	@Ignore
```

```
        @Test(expected =  Exception.class)
    public void testExceptionPLAY() {
                    for (int index=0;index<db.getSongList().size();index++)
                {
                  Song song=db.getSongList(index);
                  song.play();
                }
    }


}
```

- **SongTest**

```
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Ignore;
import org.junit.Test;

import com.qa.jukebox.Song;

import static org.junit.Assert.*;
import java.io.IOException;
import java.io.InputStream;

/*
 *
 * @author Mittal
 */
public class SongTest {
   private Song testSong1, testSong2;
   public SongTest() {
   }

   @BeforeClass
   public static void setUpClass() {
         System.out.println("in before class...SongPositiveTest.java");
   }

   @AfterClass
   public static void tearDownClass() {
         System.out.println("in after class...SongPositiveTest.java");
   }

   @Before
   public void setUp() {

      testSong1 = new Song(“Kalank Nahi”, “Arijit  Singh”, “Kalank”,”kalank.mp3”, “MP3”, 322);
      testSong2 = new Song("Kadhal Cricket", "Kharesma Ravichandran",
           "Thani Oruvan", "Cricket.mp3", "Mp3", 214);
   }

   @After
   public void tearDown() {
```

```
   testSong1 = null;
   testSong2 = null;
}


@Test
public void testSongDefaultConstructor() {

       Song song4 = new Song();
   assertEquals(song4.getClass(), new Song().getClass());
}

@Test
public void testSongParameterizedConstructor() {
   assertEquals(testSong2, testSong1);
}



 @Test
 public void testIsLong1() {
  assertTrue("Song is long",testSong1.isLong());
 }



 @Ignore
 @Test(expected =  Exception.class)
 public void testException() {
        try {
                Song song1 = new Song(null,null,null,null,null,0);
                song1.play();
        }catch(Exception e) {
                assertEquals(e.getMessage(),"passed null value or no song in your file system.");
        }
 }


 @Test
 public void testSetSongName() {
    Song song = new Song();
    song.setName("zeans");
    assertTrue(song.getName() == "zeans");
```

```
}


@Test
public void testGetSongName() {
        Song song = new Song();
   song.setName("hai ra hai ra");
   assertTrue(song.getName() == "hai ra hai ra");
}



@Test
public void testSetArtistName() {
   Song song = new Song();
   song.setArtist("rehman");
   assertTrue(song.getArtist() == "rehman");
}

@Test
public void testGetArtistName() {
        Song song = new Song();
   song.setArtist("balu");
   assertTrue(song.getArtist() == "balu");
}


@Test
public void testSetAlbumName() {
   Song song = new Song();
   song.setAlbum("jeans");
   assertTrue(song.getAlbum() == "jeans");
}

@Test
public void testGetAlbumName() {
        Song song = new Song();
   song.setAlbum("roja");
   assertTrue(song.getAlbum() == "roja");
}


@Test
public void testSetUrlName() {
```

```
   Song song = new Song();
   song.setUrl("jeans.mp3");
   assertTrue(song.getUrl() == "jeans.mp3");
}

@Test
public void testGetUrlName() {
        Song song = new Song();
   song.setUrl("roja.mp3");
   assertTrue(song.getUrl() == "roja.mp3");
}


@Test
public void testSetFormatName() {
   Song song = new Song();
   song.setFormat("Mp3");
   assertTrue(song.getFormat() == "Mp3");
}

@Test
public void testGetFormatName() {
        Song song = new Song();---
   song.setFormat("p3");
   assertTrue(song.getFormat() == "p3");
}


@Test
public void testSetDurationName() {
   Song song = new Song();
   song.setDuration(214);
   assertTrue(song.getDuration() == 214);
}


@Test
public void testGetDurationName() {
        Song song = new Song();
   song.setDuration(20);
   assertTrue(song.getDuration() == 20);
}
```

```
    @Test
    public void testMtoString() {
        assertEquals(testSong1.toString(), testSong2.toString());
    }

}
```

- **JukeBoxTestSuite**

```
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 Mittal
 */
@RunWith(Suite.class)
//@Suite.SuiteClasses({JukeBoxTest.class, DatabaseTest.class, SongTest.class})
@Suite.SuiteClasses({JukeBoxTest.class, DatabaseTest.class, SongTest.class,
JukeBoxNegativeTest.class, DatabaseNegativeTest.class, SongNegativeTest.class})
public class JukeBoxTestSuite {

   @BeforeClass
   public static void setUpClass() throws Exception {
         System.out.println("in before Suite class...JukeBoxTestSuite.java");
   }

   @AfterClass
   public static void tearDownClass() throws Exception {
         System.out.println("in after Suite class...JukeBoxTestSuite.java");
   }

   @Before
   public void setUp() throws Exception {
         System.out.println("test case starts...JukeBoxTestSuite.java");
   }

   @After
   public void tearDown() throws Exception {
         System.out.println("test case ends...JukeBoxTestSuite.java");
   }

}
```

- **JBTestSuiteRunner**

```
package com.qa.jukebox.test;

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());
  }
}
```