1. DatabaseTest

```
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
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;

public class DatabaseNegativeTest {
	private List<Song> songList;
	private Song testSong1, testSong2;
	private Database testDB;

	public DatabaseNegativeTest(){

	}

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

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

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

```
		testSong1 = new Song("Kadhal Cricket", "Kharesma Ravichandran", "Thani
Oruvan", "Cricket.mp3", "Mp3", 214);
		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;
	}

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

	}

	@Ignore
	@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("Kadhal Cricket", "Kharesma Ravichandran", "Thani
Oruvan", "Cricket.mp3", "Mp3", 214);
		testSong2 = new Song("Kannala Kannala", "Kaushik Krish", "Thani Oruvan",
"Kannala.mp3", "Mp3", 215);
```

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

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

		assertFalse("list is null", result  ==  null);

	}


	@Test
	public void testGetSongListReturnsSong() {
		List<Song> result = testDB.getSongList();
		assertFalse("Song from list is null", result.get(0) == null);
	}


	//testDB.isEmpty()  == false  --->  assert true
	@Test
	public void testIsEmpty() {
		assertTrue("List is not empty", testDB.isEmpty() != true);

	}

	@Test
	public void testSetSongList() {
		songList = new java.util.ArrayList();
		Song song1 = new Song("Kadhal Cricket", "Kharesma Ravichandran", "Thani
Oruvan", "Cricket.mp3", "Mp3", 214);
		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());
		assertFalse("song list has not set to null", testSetterDB.getSongList() == null);
	}

	@Test
```

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

		assertFalse("addSong() method tested", testDB.getSongList(2) == null);//object
expected, object actual
	}


	@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());
		assertFalse("" , testDB.getSongList().size() == 3);// 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());
		assertFalse("" , testDB.getSongList().size() == 3);

	}


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

	@Ignore
	//testing Database.play() method.
	@Test(expected =  Exception.class)
```

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


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


}
```

2. JukeBoxTest

```
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
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.Jukebox;
import com.qa.jukebox.Song;

public class JukeBoxNegativeTest {

	private Jukebox testJB;
	private Database db;
	private java.util.List songListJB;
	private Song song1, song2;
	private int ccn = 123;

	public JukeBoxNegativeTest() {
	}

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

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

	@Before
	public void setUp() {
```

```
		song1 = new Song("Kadhal Cricket", "Kharesma Ravichandran", "Thani Oruvan",
"Cricket.mp3", "Mp3", 214);
		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;
	}

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

	}

	@Ignore
	@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());
  }


	// testing Database obj - getter method
	@Test
	public void testGetDB() {
		assertFalse("", testJB.getDb().getClass() != db.getClass());// object expected,
object actual
```

```
	}

	// testing Database obj - setter method
	@Test
	public void testSetDB() {
		Song testSong1 = new Song("Kannala Kannala", "Kaushik Krish", "Thani Oruvan",
"Kannala.mp3", "Mp3", 215);
		List songListTest = new ArrayList<Song>();
		songListTest.add(testSong1);
		Database testDBSetter = new Database();
		testDBSetter.setSongList(songListTest); /// method to be tested
		Jukebox testJBsetter = new Jukebox(testDBSetter, 444);

		assertFalse("", testJBsetter.getDb() != testDBSetter);// object expected, object
actual

	}

	//testing creditcard - setter method
   @Test
   public void testSetCreditCard() {
	testJB.setCreditCard(234);
      assertFalse(" ", testJB.getCreditCard() != 234);
   }

   //testing creditcard - getter method
   @Test
   public void testGetCreditcard() {
	assertFalse("" , testJB.getCreditCard() != 123);
   }

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


	@Ignore
	//testing Database.play(index) method.
	@Test(expected =  Exception.class)
   public void testExceptionPLAYIndex() {
		Song song=db.getSongList(0);
```

```
        song.play();
    }

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


}
```

3. SongTest

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

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;

public class SongNegativeTest {

	private Song testSong1, testSong2;


  public SongNegativeTest() {
  }


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

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

  @Before
  public void setUp() {

    testSong1 = new Song("Kadhal Cricket", "Kharesma Ravichandran",
	 "Thani Oruvan", "Cricket.mp3", "Mp3", 214);
    testSong2 = new Song("Kadhal Cricket", "Kharesma Ravichandran",
	 "Thani Oruvan", "Cricket.mp3", "Mp3", 214);
  }
```

```
@After
public void tearDown() {

   testSong1 = null;
   testSong2 = null;
}




@Ignore
@Test
public void testSongDefaultConstructor() {

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

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


 @Test
 public void testIsLong1() {
        testSong1.setDuration(20);
  assertFalse("Song is long",testSong1.isLong());
 }



 //Testing Play() method in Song.java
 //Reading a local file that was no longer available.
 @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.");
        }
```

```
  }

  //testing name - setter method
  @Test
  public void testSetSongName() {
     Song song = new Song();
     song.setName("cricket");
     assertFalse(" ",song.getName() == null);

  }

  //testing name - getter method
  @Test
  public void testGetSongName() {
         Song song = new Song();
     song.setName(null);
     assertFalse(song.getName() == "hai ra hai ra");
  }


 //testing artist - setter method
  @Test
  public void testSetArtistName() {
     Song song = new Song();
     song.setArtist("rehman");
     assertFalse(song.getArtist() == null);
  }

  //testing artist - getter method
  @Test
  public void testGetArtistName() {
         Song song = new Song();
     song.setArtist(null);
     assertFalse(song.getArtist() == "balu");
  }


 //testing album - setter method
  @Test
  public void testSetAlbumName() {
     Song song = new Song();
     song.setAlbum("jeans");
     assertFalse(song.getAlbum() == "99");
  }
```

```
  //testing album - getter method
  @Test
  public void testGetAlbumName() {
        Song song = new Song();
     song.setAlbum("roja");
     assertFalse(song.getAlbum() == "mock");
  }

 //testing url - setter method
  @Test
  public void testSetUrlName() {
     Song song = new Song();
     song.setUrl(null);
     assertFalse(song.getUrl() == "jeans.mp3");
  }

  //testing url - getter method
  @Test
  public void testGetUrlName() {
        Song song = new Song();
     song.setUrl("roja.mp3");
     assertFalse(song.getUrl() == null);
  }


 //testing format - setter method
  @Test
  public void testSetFormatName() {
     Song song = new Song();
     song.setFormat(null);
     assertFalse(song.getFormat() == "Mp3");
  }

  //testing format - getter method
  @Test
  public void testGetFormatName() {
        Song song = new Song();
     song.setFormat(null);
     assertFalse(song.getFormat() == "p3");
  }

 //testing duration - setter method
  @Test
```

```
    public void testSetDurationName() {
       Song song = new Song();
       song.setDuration(0);
       assertFalse(song.getDuration() == 214);
    }

    //testing duration - getter method
    @Test
    public void testGetDurationName() {
           Song song = new Song();
       song.setDuration(20);
       assertFalse(song.getDuration() == 1000);
    }

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


}
```