```
package test;
import gameLogic.Game;
import gameLogic.GameState;
import gameLogic.Player;
import gameLogic.PlayerManager;
import gameLogic.map.IPositionable;
import gameLogic.map.Position;
import gameLogic.map.Station;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;
public class SnapshotTest extends LibGdxTest {
    private PlayerManager pm;
    private Game game;
    private Player p1;
    private Station a;
    @Before
    public void setUpGame() throws Exception {
        game = Game.getInstance();
        game.getPlayerManager();
        pm = game.getPlayerManager();
        p1 = pm.getCurrentPlayer();
        a = new Station("Rome", new Position(100, 200));
    }
    @Test
    public void testSnapshots() throws Exception {
        game.setDestination(a);
        assertTrue("The new destination is set for this state",
game.getDestination().equals(a));
        game.setState(GameState.NORMAL);
        assertTrue("Game State is initally set to NORMAL", game.getState() ==
GameState.NORMAL);
        game.setState(GameState.ANIMATING);
        assertTrue("Game State is now set to WAITING", game.getState() ==
GameState.ANIMATING);
        assertTrue("There should be 2 snapshots of the Game already.",
game.getSnapshotsNumber() == 2);
        game.setDestination(null);
        assertTrue("The new destination is NOT for this new state",
game.getDestination() == null);
        assertTrue("You are now NOT in replay mode", !game.replayMode);
        game.replaySnapshot(0);
        assertTrue("Game state is now again NORMAL", game.getState() ==
GameState.NORMAL);
        assertTrue("The  destination is set for this state ",
game.getDestination().equals(a));
        assertTrue("You are now in replay mode", game.replayMode);
    }
    @Test
    public void testSetReplaySpeed() throws Exception {
        assertEquals("Game speed should start at 1.0f", 1.0f, game.getGameSpeed(), 0);
        //Set the speed to five times faster
        game.setGameSpeed(5);
        assertEquals("Game speed should now be five times faster", 5.0f,
game.getGameSpeed(), 0);
        //Set the speed to a quarter of original
        game.setGameSpeed(0.25f);
        assertEquals("Game speed should now be a quarter", 0.25f, game.getGameSpeed(),
0);
    }
}
```

```java
package test;
import gameLogic.map.Connection;
import gameLogic.map.Map;
import gameLogic.map.Position;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;
public class MapTest extends LibGdxTest{
    private Map map;
    String name1 = "station1";
    String name2 = "station2";
    String name3 = "station3";
    String name4 = "station4";
    @Before
    public void mapSetup() throws Exception {
        map = new Map();
    }
    @Test
    public void intersectingConnectionsTest() throws Exception {
        //Tests two lines that intersect within the map
        map.addStation(name1, new Position(45, 45));
        map.addStation(name2, new Position(90, 90));
        map.addConnection(name1, name2);
        map.addStation(name3, new Position(90, 45));
        map.addStation(name4, new Position(45, 90));
        map.addConnection(name3, name4);
        Connection connectionOne = map.getConnection(name1, name2);
        Connection connectionTwo = map.getConnection(name3, name4);
        //Connection One and Two should intersect
        assertTrue("Connections intersecting should be true",
connectionOne.intersect(connectionTwo));
        assertTrue("Connections intersecting should be true",
connectionTwo.intersect(connectionOne));
        //Connection should not intersect itself
        assertFalse("Connection should not intersect itself",
connectionOne.intersect(connectionOne));
        assertFalse("Connection should not intersect itself",
connectionTwo.intersect(connectionTwo));
    }
    @Test
    public void nonIntersectingTestOne() throws Exception {
        //Tests two lines that are not parallel but do not intersect
        map.addStation(name1, new Position(10, 10));
        map.addStation(name2, new Position(40, 40));
        map.addConnection(name1, name2);
        map.addStation(name3, new Position(10, 5));
        map.addStation(name4, new Position(39, 39));
        map.addConnection(name3, name4);
        Connection connectionOne = map.getConnection(name1, name2);
        Connection connectionTwo = map.getConnection(name3, name4);
        //Connection One and Two should not intersect
        assertFalse("Connections intersecting should be false",
connectionOne.intersect(connectionTwo));
        assertFalse("Connections intersecting should be false",
connectionTwo.intersect(connectionOne));
    }
    @Test
    public void nonIntersectingTestTwo() throws Exception {
        //Tests two parallel lines
        map.addStation(name1, new Position(10, 40));
        map.addStation(name2, new Position(40, 40));
        map.addConnection(name1, name2);
        map.addStation(name3, new Position(5, 20));
        map.addStation(name4, new Position(20, 20));
        map.addConnection(name3, name4);
        Connection connectionOne = map.getConnection(name1, name2);
        Connection connectionTwo = map.getConnection(name3, name4);
        //Connection One and Two should not intersect
        assertFalse("Connections intersecting should be false",
connectionOne.intersect(connectionTwo));
        assertFalse("Connections intersecting should be false",
connectionTwo.intersect(connectionOne));
    }
}
```

```java
package test;
import gameLogic.Player;
import gameLogic.PlayerManager;
import gameLogic.resource.ConnectionModifier;
import gameLogic.resource.ResourceManager;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;
public class ResourceManagerTest extends LibGdxTest {
    PlayerManager pm;
    Player player;
    ResourceManager rm;
    @Before
    public void resourceManagerSetup() throws Exception {
        pm = new PlayerManager();
        player = new Player(pm,1);
        rm = new ResourceManager();
    }

@Test
public void testAddConnectionModifierToPlayer() throws  Exception {
    ConnectionModifier connectionModifier = new ConnectionModifier("Connection
Modifier", player);
    player.addResource(connectionModifier);
    assertTrue(player.getResources().contains(connectionModifier));
    assertTrue(player.getConnectionModifiers().contains(connectionModifier));
    assertEquals(1, player.getConnectionModifiers().size());
}
```