Ace Aceron
Testing Assignment
9/25/2023

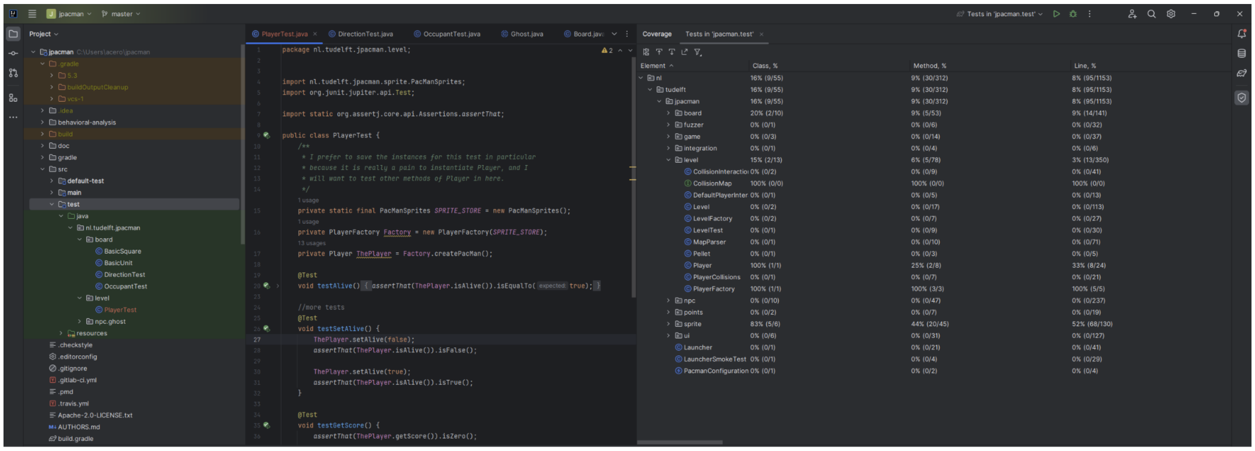
Link to fork repo: https://github.com/Hoodi3ac3/jpacman

Task 1:

Coverage is not good enough its only 1%

Task 2.1:

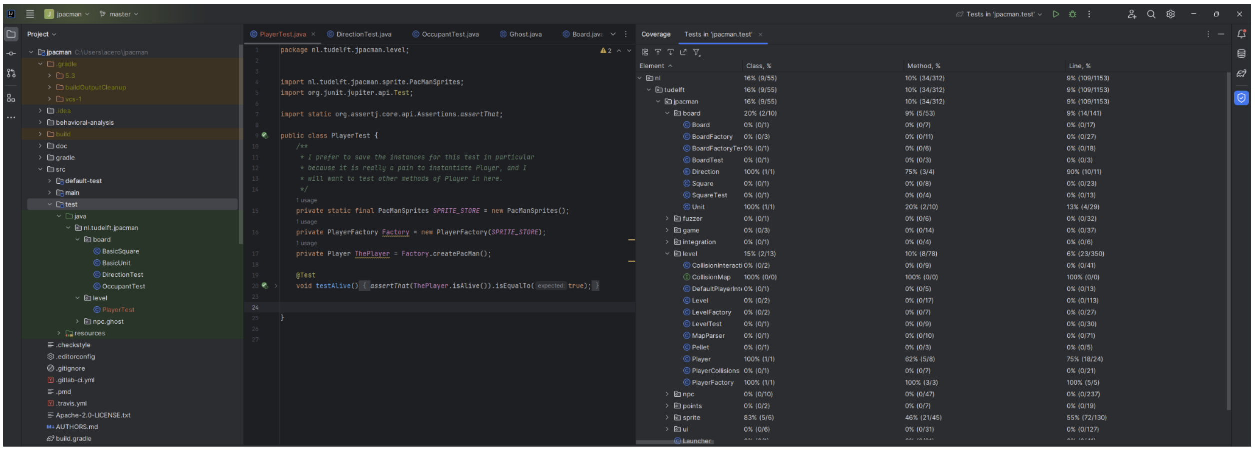
Before Adding Unit Tests



```java
package nl.tudelft.jpacman.level;


import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;

import static org.assertj.core.api.Assertions.assertThat;

public class PlayerTest {
    /**
     * I prefer to save the instances for this test in particular
     * because it is really a pain to instantiate Player, and I
     * will want to test other methods of Player in here.
     */
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    private Player ThePlayer = Factory.createPacMan();

    @Test
    void testAlive() { assertThat(ThePlayer.isAlive()).isEqualTo( true); }

    //more tests
    @Test
    void testSetAlive() {
        ThePlayer.setAlive(false);
        assertThat(ThePlayer.isAlive()).isFalse();

        ThePlayer.setAlive(true);
        assertThat(ThePlayer.isAlive()).isTrue();
    }

    @Test
    void testGetScore() {
        assertThat(ThePlayer.getScore()).isZero();
```

Element	Class, %	Method, %	Line, %
nl	16% (9/55)	9% (30/312)	8% (95/1153)
tudelft	16% (9/55)	9% (30/312)	8% (95/1153)
jpacman	16% (9/55)	9% (30/312)	8% (95/1153)
board	20% (2/10)	9% (5/53)	9% (14/141)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	15% (2/13)	6% (5/78)	3% (13/350)
CollisionInteractio	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInter	0% (0/1)	0% (0/5)	0% (0/13)
Level	0% (0/2)	0% (0/17)	0% (0/113)
LevelFactory	0% (0/2)	0% (0/7)	0% (0/27)
LevelTest	0% (0/1)	0% (0/9)	0% (0/30)
MapParser	0% (0/1)	0% (0/10)	0% (0/71)
Pellet	0% (0/1)	0% (0/3)	0% (0/5)
Player	100% (1/1)	25% (2/8)	33% (8/24)
PlayerCollisions	0% (0/1)	0% (0/7)	0% (0/21)
PlayerFactory	100% (1/1)	100% (3/3)	100% (5/5)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	83% (5/6)	44% (20/45)	52% (68/130)
ui	0% (0/6)	0% (0/31)	0% (0/127)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfiguration	0% (0/1)	0% (0/2)	0% (0/4)

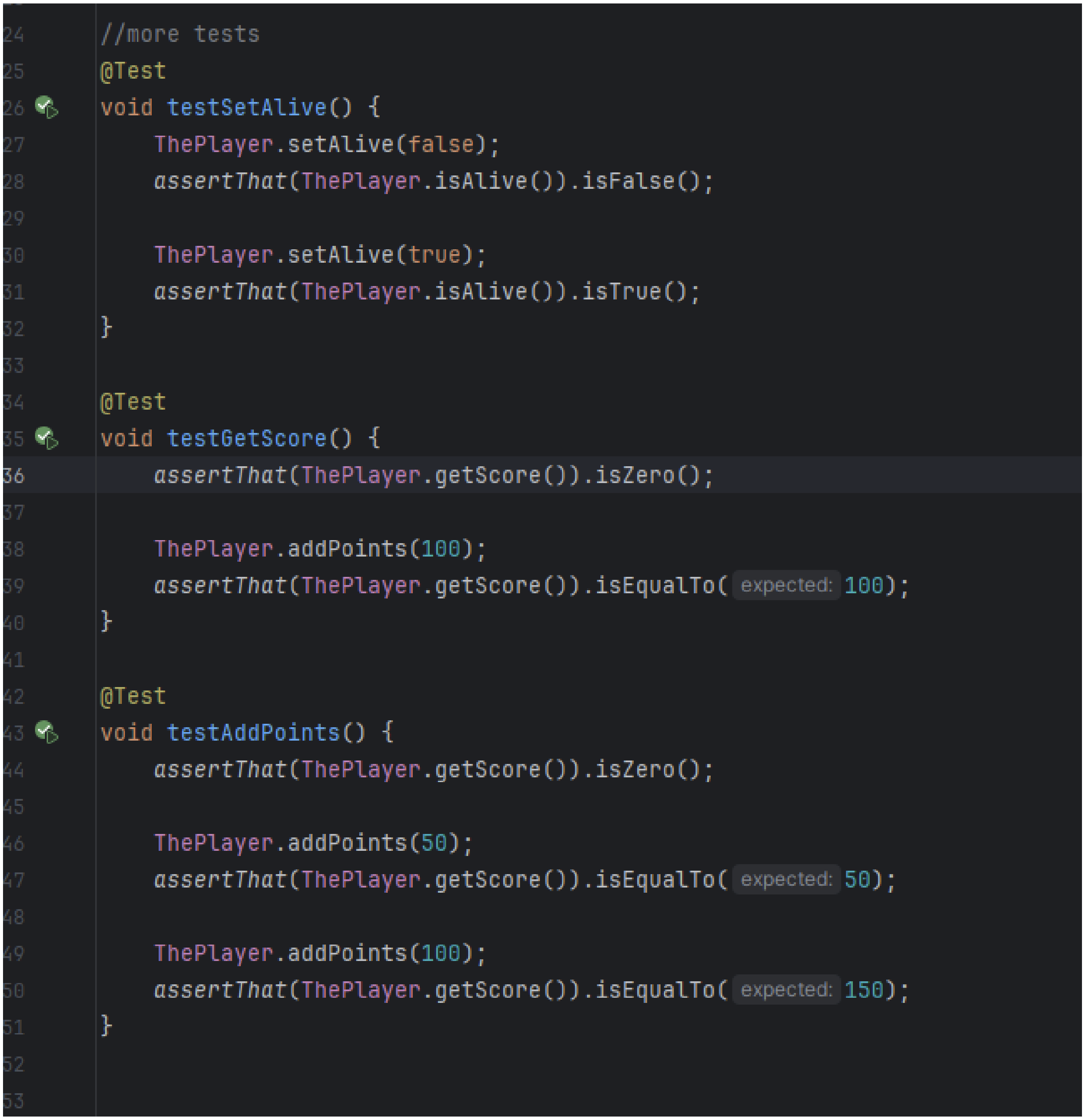
After Adding Unit Tests



```java
package nl.tudelft.jpacman.level;


import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;

import static org.assertj.core.api.Assertions.assertThat;

public class PlayerTest {
    /**
     * I prefer to save the instances for this test in particular
     * because it is really a pain to instantiate Player, and I
     * will want to test other methods of Player in here.
     */
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    private Player ThePlayer = Factory.createPacMan();

    @Test
    void testAlive() { assertThat(ThePlayer.isAlive()).isEqualTo( true); }


}
```

Element	Class, %	Method, %	Line, %
nl	16% (9/55)	10% (34/312)	9% (109/1153)
tudelft	16% (9/55)	10% (34/312)	9% (109/1153)
jpacman	16% (9/55)	10% (34/312)	9% (109/1153)
board	20% (2/10)	9% (5/53)	9% (14/141)
Board	0% (0/1)	0% (0/7)	0% (0/17)
BoardFactory	0% (0/3)	0% (0/11)	0% (0/27)
BoardFactoryTe	0% (0/1)	0% (0/6)	0% (0/18)
BoardTest	0% (0/1)	0% (0/3)	0% (0/3)
Direction	100% (1/1)	75% (3/4)	90% (10/11)
Square	0% (0/1)	0% (0/8)	0% (0/23)
SquareTest	0% (0/1)	0% (0/4)	0% (0/13)
Unit	100% (1/1)	20% (2/10)	13% (4/29)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	15% (2/13)	10% (8/78)	6% (23/350)
CollisionInteracti	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInt	0% (0/1)	0% (0/5)	0% (0/13)
Level	0% (0/2)	0% (0/17)	0% (0/113)
LevelFactory	0% (0/2)	0% (0/7)	0% (0/27)
LevelTest	0% (0/1)	0% (0/9)	0% (0/30)
MapParser	0% (0/1)	0% (0/10)	0% (0/71)
Pellet	0% (0/1)	0% (0/3)	0% (0/5)
Player	100% (1/1)	62% (5/8)	75% (18/24)
PlayerCollisions	0% (0/1)	0% (0/7)	0% (0/21)
PlayerFactory	100% (1/1)	100% (3/3)	100% (5/5)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	83% (5/6)	46% (21/45)	55% (72/130)
ui	0% (0/6)	0% (0/31)	0% (0/127)
Launcher			

The Tests I did



```
//more tests
@Test
void testSetAlive() {
    ThePlayer.setAlive(false);
    assertThat(ThePlayer.isAlive()).isFalse();

    ThePlayer.setAlive(true);
    assertThat(ThePlayer.isAlive()).isTrue();
}

@Test
void testGetScore() {
    assertThat(ThePlayer.getScore()).isZero();

    ThePlayer.addPoints(100);
    assertThat(ThePlayer.getScore()).isEqualTo( expected: 100);
}

@Test
void testAddPoints() {
    assertThat(ThePlayer.getScore()).isZero();

    ThePlayer.addPoints(50);
    assertThat(ThePlayer.getScore()).isEqualTo( expected: 50);

    ThePlayer.addPoints(100);
    assertThat(ThePlayer.getScore()).isEqualTo( expected: 150);
}
```

It is clear that the coverage for methods in level/player class increased after adding the unit tests.

Task 3:

1. There are differences between using JaCoco instead of IntelliJ because they use different mechanisms to collect coverage data.
2. Yes, the source code visualization from JaCoCo on uncovered branches can be extremely helpful. It provides detailed insights into which branches of the code are covered (or partially covered) by tests and which branches are not.

3. I like IntelliJ better because it provides real-time feedback within the IDE during development and testing.