

## Unit Test Report

For the first unit test I tested the createPacman function in the Player Factory class. It was a simple test just to make sure that Player object was created.

```
package nl.tudelft.jpacman.level;
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;
import nl.tudelft.jpacman.sprite.Sprite;

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

public class PlayerFactoryTest {

    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    private Player ThePlayer = Factory.createPacman();

    @Test
    void testCreatePacman()
    {
        assertThat(ThePlayer != null);
    }
}
```

The screenshot shows an IDE window with the following components:

- Project Explorer:** Shows the project structure with a 'test' directory containing 'PlayerTest'.
- Code Editor:** Displays the source code for 'PlayerTest.java'.
- Coverage View:** A table showing coverage data for 'Tests in 'jpacman.test''.
- Test Results:** A summary of test results at the bottom.

Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	14% (8/...	9% (30/313)	8% (93/114...	4% (23/575)
> board	20% (2/1...	9% (5/53)	9% (14/141)	1% (1/96)
> fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/16)
> game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/14)
> integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
> level	15% (2/13)	6% (5/78)	3% (13/348)	0% (0/179)
> npc	0% (0/10)	0% (0/48)	0% (0/234)	0% (0/120)
> points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/12)
> sprite	66% (4/6)	44% (20/45)	51% (66/128)	31% (22/70)
> ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/60)

**Test Results:** 2 sec 902 ms. Tests passed: 44 of 44 tests - 2 sec 902 ms.

**Tasks:**

- > Task :compileJava UP-TO-DATE
- > Task :processResources UP-TO-DATE
- > Task :classes UP-TO-DATE

Furthermore for my second test I wanted to test the consuming of the pellets and ensure the score was correctly updated for the player.

```

package points;
import nl.tudelft.jpacman.level.Pellet;
import nl.tudelft.jpacman.level.Player;
import nl.tudelft.jpacman.level.PlayerFactory;
import nl.tudelft.jpacman.points.DefaultPointCalculator;
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;
import nl.tudelft.jpacman.sprite.Sprite;
import static org.assertj.core.api.Assertions.assertThat;

public class DefaultPointCalculatorTest {
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    private Player ThePlayer = Factory.createPacMan();
    private static final Sprite pelletSprite= SPRITE_STORE.getPelletSprite();
    private static final Pellet pellet=new Pellet(10,pelletSprite);
    private static final DefaultPointCalculator pointCalc= new
DefaultPointCalculator();

    @Test
    void testConsumedAPellet(){
        pointCalc.consumedAPellet(ThePlayer,pellet);
        assertThat(ThePlayer.getScore()).isEqualTo(10);
    }
}

```

The screenshot shows an IDE with a code coverage window open. The window displays a table of coverage data for the tests in 'jpacman.test'. The table has columns for Element, Class, Method, Line, and Branch, each with a percentage and a count of hits/misses. The data is as follows:

Element	Class	Method	Line	Branch
nl.tudelft.jpacman	16% (9/...	10% (32/...	8% (98/11...	4% (23/5...
board	20% (2/...	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/16)
game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	23% (3/...	7% (6/78)	4% (17/3...	0% (0/179)
CollisionInteractionMap	0% (0/2)	0% (0/9)	0% (0/39)	0% (0/24)
CollisionMap	100% (...)	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInteractionMap	0% (0/1)	0% (0/5)	0% (0/13)	100% (0/0)

The IDE also shows a task list at the bottom with tasks: :compileJava UP-TO-DATE, :processResources UP-TO-DATE, and :classes UP-TO-DATE. The status bar at the bottom indicates 16:43, LF, UTF-8, and 4 spaces.

For the last unit test I tested getting the value of individual pellets and ensuring the values were correctly being stored.

```

package nl.tudelft.jpacman.level;
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;
import nl.tudelft.jpacman.sprite.Sprite;
import static org.assertj.core.api.Assertions.assertThat;

public class PelletTest {
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    private static final Sprite pelletSprite= SPRITE_STORE.getPelletSprite();
    private static final Pellet pellet=new Pellet(10,pelletSprite);

    @Test
    void testGetValue(){

        assertThat(pellet.getValue()==10);
    }
}

```

Coverage - jpacman

Coverage Tests in 'jpacman.test' Tests in 'jpacman.test' Tests in 'jpacman.test' Tests in 'jpacman.test' Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	18% (10/55)	11% (35/313)	9% (103/1144)	4% (23/575)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/16)
game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	23% (3/13)	10% (8/78)	5% (20/349)	0% (0/179)
npc	0% (0/10)	0% (0/48)	0% (0/234)	0% (0/120)
points	50% (1/2)	14% (1/7)	10% (2/20)	0% (0/12)
DefaultPointCalculator	100% (1/1)	33% (1/3)	50% (2/4)	100% (0/0)
PointCalculator	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
PointCalculatorLoader	0% (0/1)	0% (0/4)	0% (0/16)	0% (0/12)
sprite	66% (4/6)	46% (21/45)	52% (67/128)	31% (22/70)
ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/60)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)	0% (0/6)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/2)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

Type here to search

76°F 8:33 PM 9/17/2024

In looking over the Jacoco files the coverage was different most likely to Jacoco considering the branch coverage as well. I do prefer the Jacoco representation as it gives a lot more information and easier navigate by quickly seeing code and methods.