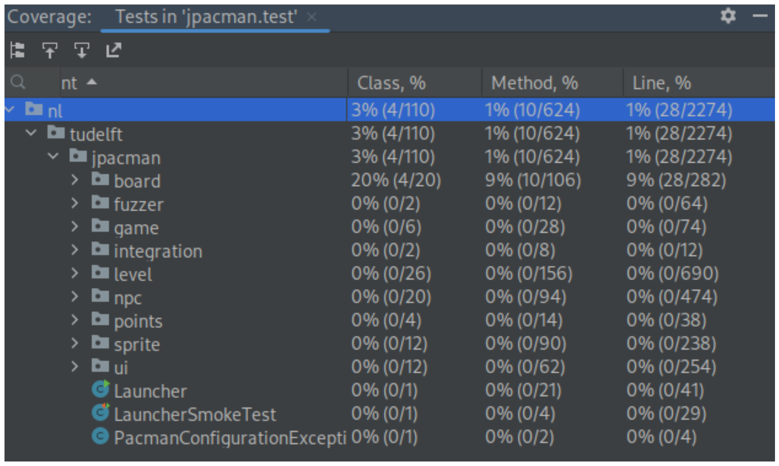
Unit Testing Report

[Forked Repository]

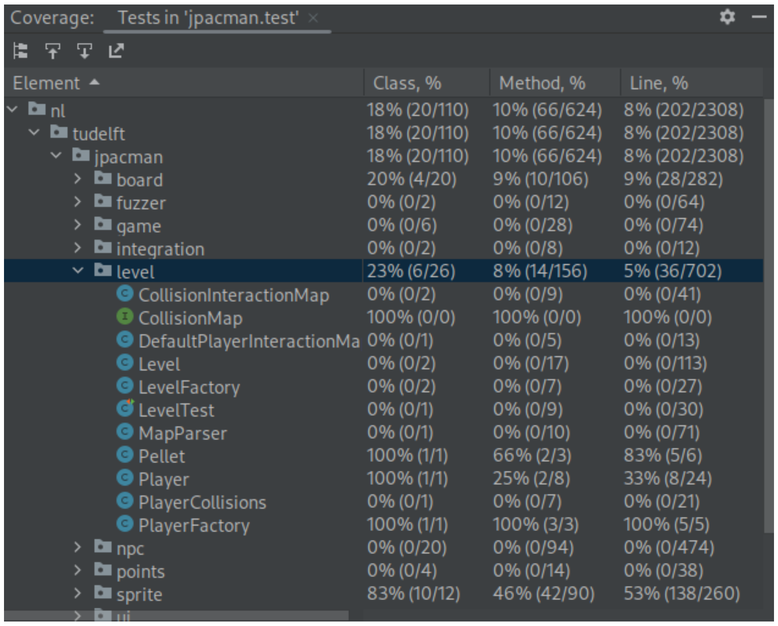
After running the initial tests I found that the test coverage for the JPacman project was lacking. Here is an image of the original test coverage before adding any unit tests.


Coverage: Tests in 'jpacman.test'

nt	Class, %	Method, %	Line, %
nl	3% (4/110)	1% (10/624)	1% (28/2274)
tudelft	3% (4/110)	1% (10/624)	1% (28/2274)
jpacman	3% (4/110)	1% (10/624)	1% (28/2274)
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	0% (0/26)	0% (0/156)	0% (0/690)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	0% (0/12)	0% (0/90)	0% (0/238)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)

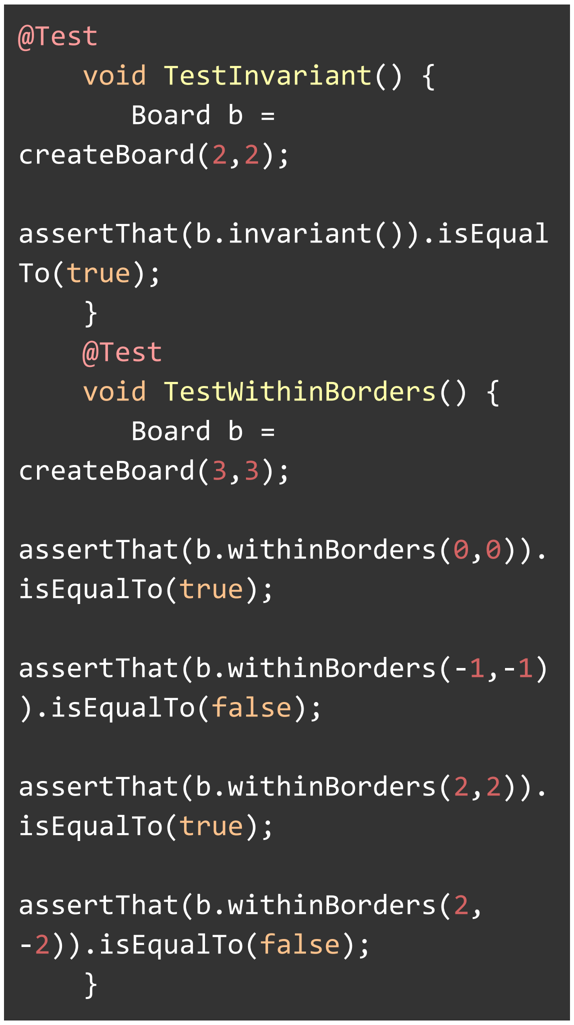
I decided to first add a simple unit test for the pellet class, here is a snippet of the code added alongside an updated image of the test coverage.

```
public class PelletValueTest {
    private static final PacManSprites
SPRITE_STORE = new PacManSprites();

    private Pellet pellet = new Pellet(2,
SPRITE_STORE.getPelletSprite());

    @Test
    void testPelletValue()
{assertThat(pellet.getValue()).isEqualTo(2);}
}
```

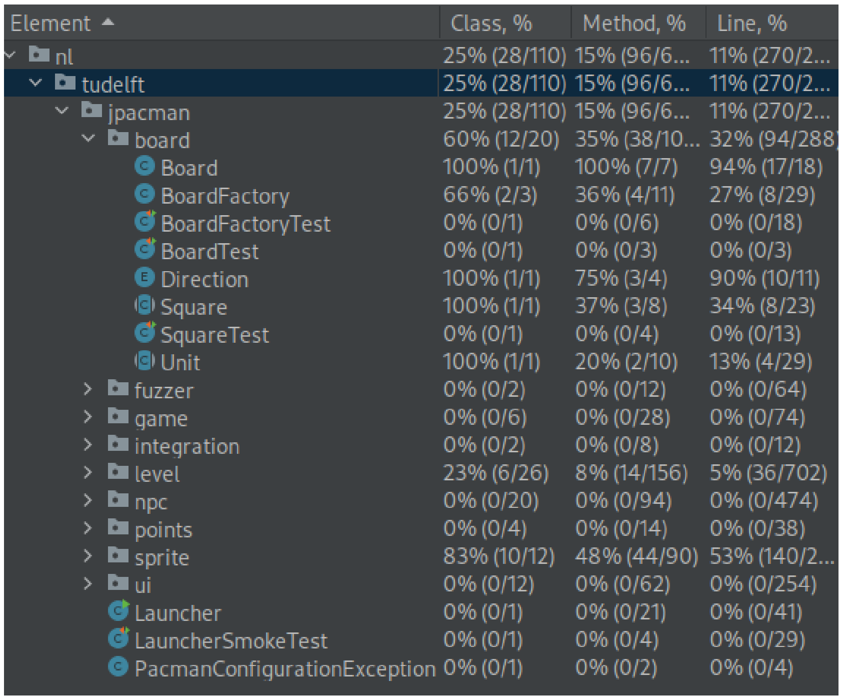

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	18% (20/110)	10% (66/624)	8% (202/2308)
tudelft	18% (20/110)	10% (66/624)	8% (202/2308)
jpacman	18% (20/110)	10% (66/624)	8% (202/2308)
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	23% (6/26)	8% (14/156)	5% (36/702)
CollisionInteractionMap	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInteractionMa	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	100% (1/1)	66% (2/3)	83% (5/6)
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% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	46% (42/90)	53% (138/260)

Next I added unit tests for all of the methods in the Board class. Here are a couple of those tests as well as the updated coverage.



```
@Test
    void TestInvariant() {
        Board b =
createBoard(2,2);

assertThat(b.invariant()).isEqual
To(true);
    }
    @Test
    void TestWithinBorders() {
        Board b =
createBoard(3,3);

assertThat(b.withinBorders(0,0)).
isEqualTo(true);

assertThat(b.withinBorders(-1,-1)
).isEqualTo(false);

assertThat(b.withinBorders(2,2)).
isEqualTo(true);

assertThat(b.withinBorders(2,
-2)).isEqualTo(false);
    }
```



Element	Class, %	Method, %	Line, %
nl	25% (28/110)	15% (96/6…	11% (270/2…
tudelft	25% (28/110)	15% (96/6…	11% (270/2…
jpacman	25% (28/110)	15% (96/6…	11% (270/2…
board	60% (12/20)	35% (38/10…	32% (94/288)
Board	100% (1/1)	100% (7/7)	94% (17/18)
BoardFactory	66% (2/3)	36% (4/11)	27% (8/29)
BoardFactoryTest	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	100% (1/1)	37% (3/8)	34% (8/23)
SquareTest	0% (0/1)	0% (0/4)	0% (0/13)
Unit	100% (1/1)	20% (2/10)	13% (4/29)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	23% (6/26)	8% (14/156)	5% (36/702)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	48% (44/90)	53% (140/2…
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

The coverage results generated by JaCoCo were mostly similar to the results from Intellij. I did notice that JaCoCo reported more coverage over certain classes that Intellij did not report. The visualization on uncovered branches was useful as it is a feature that was lacking in Intellij. Overall I preferred the Intellij coverage window over JaCoCo, it has a more modern interface and I liked the ability to navigate directly to the classes from the window.