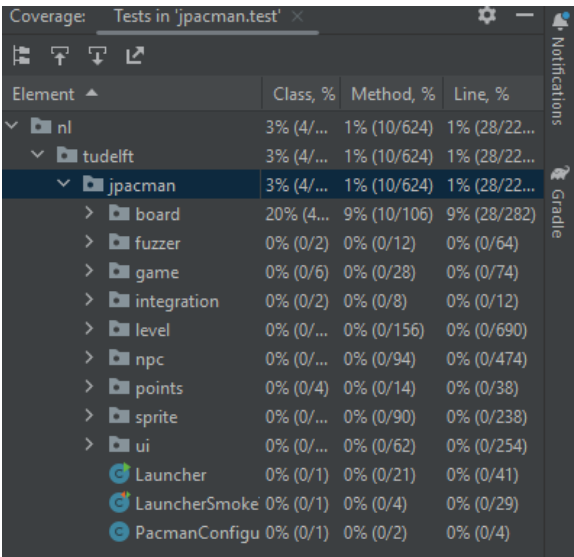


UNIT TESTING LAB

INTELLIJ

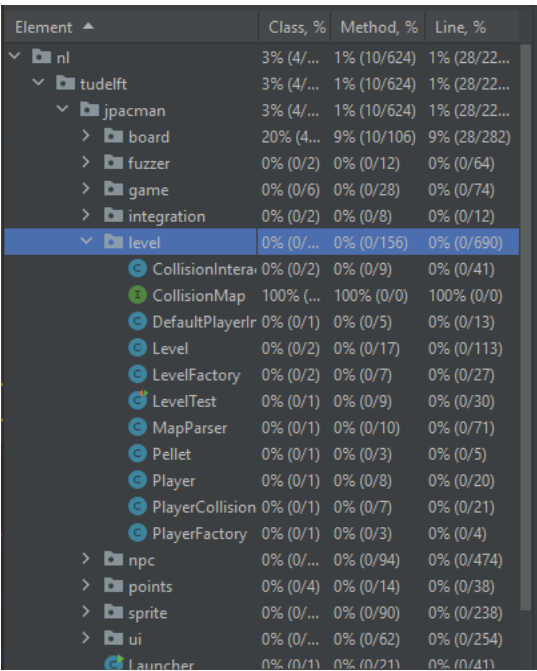
- I. Initial Testing results  
Testing results using the IntelliJ IDE test coverage plugin with no modifications. All coverage tests were passed successfully but not much is actually working in this version. I think this is not an okay coverage for the base of the game, and lacks a lot of functionality.



Coverage: Tests in 'jpacman.test' x

Element ^	Class, %	Method, %	Line, %
nl	3% (4/...	1% (10/624)	1% (28/22...
tudelft	3% (4/...	1% (10/624)	1% (28/22...
jpacman	3% (4/...	1% (10/624)	1% (28/22...
board	20% (4...	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/...	0% (0/156)	0% (0/690)
npc	0% (0/...	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	0% (0/...	0% (0/90)	0% (0/238)
ui	0% (0/...	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmoke	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigu	0% (0/1)	0% (0/2)	0% (0/4)

- II. Testing coverage increase  
Next I increased the coverage on JPacman various features.
  - a. The first feature tested was the player movement. The coverage for collision detection increased.



Element ^	Class, %	Method, %	Line, %
nl	3% (4/...	1% (10/624)	1% (28/22...
tudelft	3% (4/...	1% (10/624)	1% (28/22...
jpacman	3% (4/...	1% (10/624)	1% (28/22...
board	20% (4...	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/...	0% (0/156)	0% (0/690)
CollisionIntera	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (...)	100% (0/0)	100% (0/0)
DefaultPlayerlr	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	0% (0/1)	0% (0/8)	0% (0/20)
PlayerCollision	0% (0/1)	0% (0/7)	0% (0/21)
PlayerFactory	0% (0/1)	0% (0/3)	0% (0/4)
npc	0% (0/...	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	0% (0/...	0% (0/90)	0% (0/238)
ui	0% (0/...	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)

- b. Next was testing the points earned by a player. This also increased in coverage.

The screenshot shows the `PointCalculator` class in the `nl.tudelft.jpacman.points` package. It has three methods: `collidedWithAGhost`, `consumedAPellet`, and `pacmanMoved`. Each method is marked with "no usages" and has a coverage of 100% (0/0). The `consumedAPellet` method is highlighted, showing the line `player.addPoints(pellet.getValue());`.

```
package nl.tudelft.jpacman.points;

import nl.tudelft.jpacman.board.Direction;
import nl.tudelft.jpacman.level.Pellet;
import nl.tudelft.jpacman.level.Player;
import nl.tudelft.jpacman.npc.Ghost;

public class PointCalculator {

    no usages
    public void collidedWithAGhost(Player player, Ghost ghost) {
    }

    no usages
    public void consumedAPellet(Player player, Pellet pellet) { player.addPoints(pellet.getValue()); }

    no usages
    public void pacmanMoved(Player player, Direction direction) {
    }
}
```

- c. Next the levels were improved. This coverage also increased

▼ jpacman	15% (18/118)	8% (60/670)	7% (190/2560)
> board	20% (4/20)	9% (10/106)	9% (28/284)
> 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	13% (4/30)	5% (10/190)	2% (26/936)
> npc	0% (0/20)	0% (0/94)	0% (0/474)
> points	0% (0/8)	0% (0/26)	0% (0/54)
> sprite	83% (10/12)	44% (40/90)	52% (136/260)
> 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 screenshot shows the `Level` class constructor in the `jpacman` package. It takes four arguments: `Board board`, `List<Ghost> ghosts`, `List<Square> startPositions`, and `CollisionMap collisionMap`. The constructor includes three assertions, several field assignments, and a loop to initialize the `npcs` map.

```
public Level(Board board, List<Ghost> ghosts, List<Square> startPositions,
             CollisionMap collisionMap) {
    assert board != null;
    assert ghosts != null;
    assert startPositions != null;

    this.board = board;
    this.inProgress = false;
    this.npcs = new HashMap<>();
    for (Ghost ghost : ghosts) {
        npcs.put(ghost, null);
    }
    this.startSquares = startPositions;
    this.startSquareIndex = 0;
    this.players = new ArrayList<>();
    this.collisions = collisionMap;
    this.observers = new HashSet<>();
}
```

- d. Finally was testing the direction detection. Coverage was increased.  
Overall this is the total coverage increase:

	Class, %	Method, %	Line, %
▲	14% (18/1...	8% (60/740)	6% (190/2840)
nl.tudelft	14% (18/1...	8% (60/740)	6% (190/2840)
jpacman	14% (18/1...	8% (60/740)	6% (190/2840)
> board	20% (4/20)	9% (10/106)	9% (28/284)
> fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
> game	0% (0/8)	0% (0/36)	0% (0/110)
> integration	0% (0/2)	0% (0/8)	0% (0/12)
> level	13% (4/30)	5% (10/190)	2% (26/938)
> npc	0% (0/20)	0% (0/94)	0% (0/476)
> points	0% (0/8)	0% (0/26)	0% (0/54)
> sprite	83% (10/12)	44% (40/90)	52% (136/260)
> ui	0% (0/16)	0% (0/82)	0% (0/392)
Launcher	0% (0/1)	0% (0/21)	0% (0/46)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationExcept	0% (0/1)	0% (0/2)	0% (0/4)

```
package nl.tudelft.jpacman.board;

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




















import org.junit.jupiter.api.Test;

no usages
public class DirectionTest {
    no usages
    @Test
    void testNorth() {
        Direction north = Direction.valueOf( name: "NORTH");
        assertThat(north.getDeltaY()).isEqualTo( expected: -1);
    }
}
```

## JaCoCo

Here are the coverage results when using JaCoCo

### jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.
<a href="#">nl.tudelft.jpacman.level</a>		67%		57%
<a href="#">nl.tudelft.jpacman.npc_ghost</a>		71%		55%
<a href="#">nl.tudelft.jpacman.ui</a>		77%		47%
<a href="#">default</a>		0%		0%
<a href="#">nl.tudelft.jpacman.board</a>		86%		58%
<a href="#">nl.tudelft.jpacman.sprite</a>		86%		59%
<a href="#">nl.tudelft.jpacman</a>		69%		25%
<a href="#">nl.tudelft.jpacman.points</a>		60%		75%
<a href="#">nl.tudelft.jpacman.game</a>		87%		60%
<a href="#">nl.tudelft.jpacman.npc</a>		100%		n/a
Total	1,213 of 4,694	74%	293 of 637	54%

## Summary

Overall, coverage increased and was listed as higher on JaCoCo vs. through IntelliJ coverage testing. JaCoCo's visual representation of coverage was more helpful than IntelliJ's. It was easier to visualize how far the project is from being completed. I am not sure which to trust, however. It seems IntelliJ is more strict, while JaCoCo allows for more wriggle room. This can either be a good or bad thing depending on the needs of the project.