

## Lab 2 Report

Forked Repository Link: <https://github.com/mcebaniqued/jpacman>

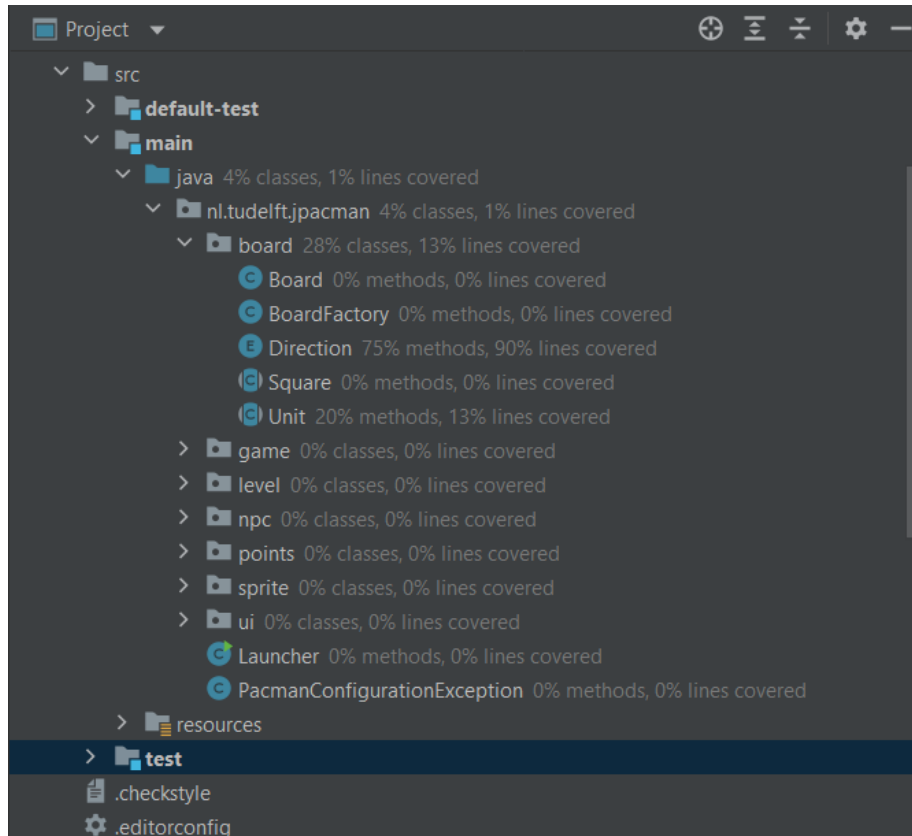
Main Repository: <https://github.com/CivBuilder/cs472project>

### Task 1: Initial Test Coverage

Coverage: Tests in 'jpacman' ×

3% classes, 1% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
META-INF			
nl	3% (4/110)	1% (10/616)	1% (28/2266)
scoreplugins			
sprite			



**Task 2: Added isAlive() Test**

Coverage: Tests in 'jpacman' ×

16% classes, 8% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
META-INF			
nl	16% (18/110)	9% (60/618)	8% (190/2298)
scoreplugins			
sprite			

Project

default-test

main

java 19% classes, 10% lines covered

nl.tudelft.jpacman 19% classes, 10% lines covered

board 28% classes, 13% lines covered

game 0% classes, 0% lines covered

level 16% classes, 4% lines covered

CollisionInteractionMap 0% methods, 0% lines covered

CollisionMap

DefaultPlayerInteractionMap 0% methods, 0% lines covered

Level 0% methods, 0% lines covered

LevelFactory 0% methods, 0% lines covered

MapParser 0% methods, 0% lines covered

Pellet 0% methods, 0% lines covered

Player 25% methods, 33% lines covered

PlayerCollisions 0% methods, 0% lines covered

PlayerFactory 100% methods, 100% lines covered

npc 0% classes, 0% lines covered

points 0% classes, 0% lines covered

sprite 100% classes, 59% lines covered

ui 0% classes, 0% lines covered

Launcher 0% methods, 0% lines covered

PacmanConfigurationException 0% methods, 0% lines covered

## Task 2.1: Added hasSquare(), getDirection(), and isInProgress() Tests

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

import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

import nl.tudelft.jpacman.npc.ghost.GhostFactory;
import nl.tudelft.jpacman.sprite.PacManSprites;
import nl.tudelft.jpacman.npc.Ghost;

public class GhostHasSquareTest {
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    private GhostFactory factory = new GhostFactory(SPRITE_STORE);

    private Ghost Blinky = factory.createBlinky();
    private Ghost Clyde = factory.createClyde();
    private Ghost Inky = factory.createInky();
    private Ghost Pinky = factory.createPinky();

    @Test
    void testBlinkyHasSquareIsFalse(){ assertThat(Blinky.hasSquare()).isEqualTo(false); }

    @Test
    void testClydeHasSquareIsFalse(){ assertThat(Clyde.hasSquare()).isEqualTo(false); }

    @Test
    void testInkyHasSquareIsFalse(){ assertThat(Inky.hasSquare()).isEqualTo(false); }

    @Test
    void testPinkyHasSquareIsFalse(){ assertThat(Pinky.hasSquare()).isEqualTo(false); }
}
```

Coverage: Tests in 'jpacman' ×

29% classes, 11% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
META-INF			
nl	29% (32/110)	15% (94/618)	11% (268/2310)
scoreplugins			
sprite			

```

package nl.tudelft.jpacman.level;

import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

import nl.tudelft.jpacman.npc.ghost.GhostFactory;
import nl.tudelft.jpacman.sprite.PacManSprites;
import nl.tudelft.jpacman.npc.Ghost;
import nl.tudelft.jpacman.board.Direction;

public class GhostGetDirection {
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    private GhostFactory Factory = new GhostFactory(SPRITE_STORE);

    private Ghost Blinky = Factory.createBlinky();
    private Ghost Clyde = Factory.createClyde();
    private Ghost Inky = Factory.createInky();
    private Ghost Pinky = Factory.createPinky();

    @Test
    void testBlinkyGetDirectionIsEast(){ assertThat(Blinky.getDirection()).isEqualTo(Direction.EAST); }

    @Test
    void testClydeGetDirectionIsEast(){ assertThat(Clyde.getDirection()).isEqualTo(Direction.EAST); }

    @Test
    void testInkyGetDirectionIsEast(){ assertThat(Inky.getDirection()).isEqualTo(Direction.EAST); }

    @Test
    void testPinkyGetDirectionIsEast(){ assertThat(Pinky.getDirection()).isEqualTo(Direction.EAST); }
}

```

Coverage: Tests in 'jpacman' ×

29% classes, 11% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
META-INF			
nl	29% (32/110)	15% (96/618)	11% (270/2310)
scoreplugins			
sprite			

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

import nl.tudelft.jpacman.board.Board;
import nl.tudelft.jpacman.board.Square;
import nl.tudelft.jpacman.npc.Ghost;

import static org.mockito.Mockito.mock;

import org.assertj.core.util.Lists;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class LevelIsInProgress {
    private Level level;
    private Ghost ghost = mock(Ghost.class);
    private Board board = mock(Board.class);
    private Square square1 = mock(Square.class);
    private Square square2 = mock(Square.class);
    private final CollisionMap collisions = mock(CollisionMap.class);

    @Test
    void testLevelIsInProgressIsFalse(){
        level = new Level(board, Lists.newArrayList(ghost), Lists.newArrayList(square1, square2), collisions);
        assertThat(level.isInProgress()).isEqualTo(false);
    }
}
```

Coverage: Tests in 'jpacman' ×

34% classes, 13% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
META-INF			
nl	34% (38/110)	17% (106/618)	13% (314/2312)
scoreplugins			
sprite			

### Task 3: JaCoCo Report on JPacman

#### jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
<a href="#">nl.tudelft.jpacman.level</a>		67%		57%	74	155	104	344	21	69	4	12
<a href="#">nl.tudelft.jpacman.npc.ghost</a>		71%		55%	56	105	43	181	5	34	0	8
<a href="#">nl.tudelft.jpacman.ui</a>		77%		47%	54	86	21	144	7	31	0	6
<a href="#">default</a>		0%		0%	12	12	21	21	5	5	1	1
<a href="#">nl.tudelft.jpacman.board</a>		86%		59%	43	93	2	110	0	40	0	7
<a href="#">nl.tudelft.jpacman.sprite</a>		86%		59%	30	70	11	113	5	38	0	5
<a href="#">nl.tudelft.jpacman</a>		69%		25%	12	30	18	52	6	24	1	2
<a href="#">nl.tudelft.jpacman.points</a>		60%		75%	1	11	5	21	0	9	0	2
<a href="#">nl.tudelft.jpacman.game</a>		87%		60%	10	24	4	45	2	14	0	3
<a href="#">nl.tudelft.jpacman.npc</a>		100%		n/a	0	4	0	8	0	4	0	1
Total	1,212 of 4,694	74%	292 of 637	54%	292	590	229	1,039	51	268	6	47

The result from IntelliJ's coverage differs greatly from JaCoCo's coverage. JaCoCo shows 74% coverage while IntelliJ shows 34% coverage. I do find JaCoCo's UI to be helpful because it adds a visualization of how much missed instruction and branches for each element, and you can compare it with other elements. It also gives more information about lines, methods, classes, and the overall for each element. I think the most helpful out of all would be the highlighting of missed and covered lines. It tells you exactly which lines are being used for tests so it's easy to point out which line or method still needs to be tested. Overall, I would prefer JaCoCo's report over IntelliJ.