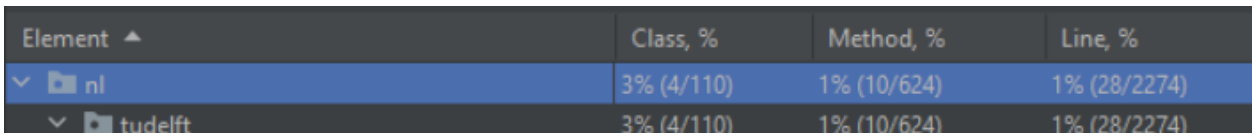
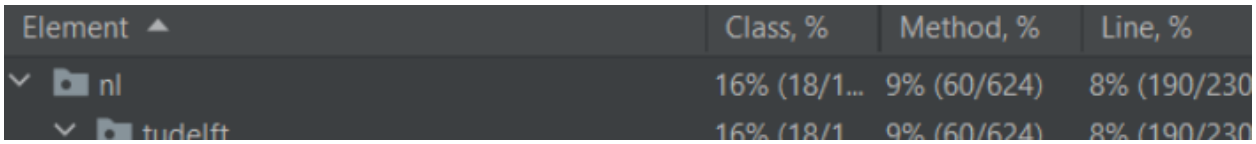
Testing Lab
Connor Mathis

Initial Test Coverage:



Element	Class, %	Method, %	Line, %
nl	3% (4/110)	1% (10/624)	1% (28/2274)
tudelft	3% (4/110)	1% (10/624)	1% (28/2274)

After Implementing isAlive() test:



Element	Class, %	Method, %	Line, %
nl	16% (18/1...	9% (60/624)	8% (190/230
tudelft	16% (18/1	9% (60/624)	8% (190/230

Final Test Coverage:

Element	Class, %	Method...	Line, %
nl	60% (6...	42% (26...	37% (88...
tudelft	60% (6...	42% (26...	37% (88...

JaCoCo Report:

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		67%		58%	73	155	103	344	21	69	4	12
nl.tudelft.jpacman.npc.ghost		71%		55%	56	105	43	181	5	34	0	8
nl.tudelft.jpacman.ui		77%		47%	54	86	21	144	7	31	0	6
default		0%		0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman.board		86%		58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman.sprite		88%		62%	29	70	10	113	5	38	0	5
nl.tudelft.jpacman		69%		25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.points		60%		75%	1	11	5	21	0	9	0	2
nl.tudelft.jpacman.game		87%		60%	10	24	4	45	2	14	0	3
nl.tudelft.jpacman.npc		100%		n/a	0	4	0	8	0	4	0	1
Total	1,204 of 4,694	74%	290 of 637	54%	291	590	227	1,039	51	268	6	47

According to IntelliJ, line coverage went from 1% of lines to 37% after implementing all of the tests. There is a difference in coverage reported between IntelliJ and JaCoCo. JaCoCo shows a much higher instruction coverage than the line coverage of IntelliJ. I prefer JaCoCo's representation of uncovered branches because it makes it easy to see exactly what in the source code is not being tested. This makes it easier to write tests that cover all branches of a method.

Tests Implemented:

src/main/java/nl/pacman/level/LevelFactory.java	createGhost()
src/main/java/nl/pacman/game/Game.java	start()
src/main/java/nl/pacman/level/PlayerCollisions.java	collide()

Code Snippets:

createGhost_test()

Tests that all four ghosts were created.

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

import nl.tudelft.jpacman.npc.ghost.GhostFactory;
import nl.tudelft.jpacman.points.DefaultPointCalculator;
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class LevelFactoryTest {
    private PacManSprites sprites = new PacManSprites();
    private LevelFactory factory = new LevelFactory(sprites, new
GhostFactory(sprites), new DefaultPointCalculator());

    @Test
    public void createGhost_test() {
        for(int i=0; i<4; i++) {
            assertThat(factory.createGhost()).isNotNull();
        }
    }
}
```

start_test()

Tests that the game is in progress after start() is executed.

```
package nl.tudelft.jpacman.game;

import nl.tudelft.jpacman.Launcher;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class GameTest {
    private Launcher launcher = new Launcher();
```

```
    private Game game = launcher.makeGame();

    @Test
    public void start_test() {
        game.start();
        assertThat(game.isInProgress()).isEqualTo(true);
    }
}
```

collide_test()

Tests that the player is not alive after colliding with a ghost.

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

import nl.tudelft.jpacman.npc.ghost.GhostFactory;
import nl.tudelft.jpacman.points.DefaultPointCalculator;
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class PlayerCollisionsTest {
    private PlayerFactory factory = new PlayerFactory(new
PacManSprites());
    private Player player = factory.createPacMan();

    private PacManSprites sprites = new PacManSprites();
    private GhostFactory gFactory = new GhostFactory(sprites);

    private PlayerCollisions collisions = new PlayerCollisions(new
DefaultPointCalculator());

    @Test
    public void collide_test() {
        collisions.collide(player, gFactory.createBlinky());
        assertThat(player.isAlive()).isEqualTo(false);
    }
}
```

Connor Mathis Repository Fork:
https://github.com/Syrodai/CS-472-Senior-Design-Project