Arturo Arroyo
CS 472-1001
Testing Report

Task 2.1

The initial code coverage for jpacman.test file was shown on the following picture:

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (60/624)	8% (190/2306)
tudelft	16% (18/110)	9% (60/624)	8% (190/2306)
jpacman	16% (18/110)	9% (60/624)	8% (190/2306)
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	15% (4/26)	6% (10/156)	3% (26/700)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
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)

I created test cases for the following methods:
src/main/java/nl/tudelft/jpacman/board/Board/getWidth
src/main/java/nl/tudelft/jpacman/level/Pellet/getValue
src/main/java/nl/tudelft/jpacman/board/BoardFactory/createBoard

For my first test coverage, I looked to test the getWidth function using the following code:

```
    @Test
    void test() {
        int width = TheBoard.getWidth();
        assertThat( actual: TheBoard.getWidth() == width);
    }
}
```

After I ran the test I achieved the same amount of coverage

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (60/624)	8% (190/2306)
tudelft	16% (18/110)	9% (60/624)	8% (190/2306)
jpacman	16% (18/110)	9% (60/624)	8% (190/2306)
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	15% (4/26)	6% (10/156)	3% (26/700)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
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)
LauncherSmokeTe	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurat	0% (0/1)	0% (0/2)	0% (0/4)

For my second test coverage, I test the getValue method on the Pellet.java file.

```
    @Test
    void test(){
        int value = 10;
        assertThat( actual: value == p.getValue());
    }
}
```

After the test the coverage went up by 8% in the level package.

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)
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)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmok	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigu	0% (0/1)	0% (0/2)	0% (0/4)

For my third and final test, I tested out the createBoard method in the BoardFactory.java file.

```
no usages   new *
public class BoardFactoryTest {

    2 usages
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    1 usage
    private PlayerFactory pf = new PlayerFactory(SPRITE_STORE);
    no usages
    private Player ThePlayer = pf.createPacMan();
    1 usage
    private BoardFactory bf = new BoardFactory(SPRITE_STORE);
    1 usage
    private Square [][] sq = new Square[10][10];

    no usages   new *
    @Test
    void test(){

        Board b = bf.createBoard(sq);
    }
}
```

The overall test coverage increased by an additional 7% percent after running the test

Element	Class, %	Method, %	Line, %
nl	25% (28/112)	13% (86/620)	11% (260/2310)
tudelft	25% (28/112)	13% (86/620)	11% (260/2310)
jpacman	25% (28/112)	13% (86/620)	11% (260/2310)
board	54% (12/22)	29% (30/102)	30% (86/284)
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	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)	46% (42/90)	53% (138/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSm	0% (0/1)	0% (0/4)	0% (0/29)
PacmanCon	0% (0/1)	0% (0/2)	0% (0/4)

Task 3

Question 1

- IntelliJ and JaCoCo do not report the same test coverage results. The reason for this is because JaCoCo reports much less methods than IntelliJ therefore causing it to show up as having more coverage.

Question 2

- Yes it can be useful to give a visual respresentations of all uncovered branches.\

Question 3

- Both JaCoCo and IntelliJ offer unique feature for running tests on programs. I prefer JaCoCo because it intuitively you tells which lines in the code have not been covered with an easy to read color schematic.

Team Fork Repo Link https://github.com/arroya2/cs472-team6.git

Jpacman Fork Repo Link https://github.com/arroya2/jpacman.git