Anthony Romaine
CS 472 – 1001

Testing Report

Task 2

The initial code coverage report from IntelliJ can be seen below. Initially, the code coverage is very poor and insufficient only having 3% class coverage, 1% method, and 1% line coverage for the project.

Element	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)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Even after just adding a single test case for the isAlive method the code coverage is significantly better as seen below with 16% class, 9% method, and 8% line 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)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

isAlive test case snippet:

```
1 usage
private static final PacManSprites PAC_MAN_SPRITES = new PacManSprites();
1 usage
private PlayerFactory playerFactory = new PlayerFactory(PAC_MAN_SPRITES);
1 usage
private Player player = playerFactory.createPacMan();
@Test
void testIsAlive() { assertThat(player.isAlive()).isEqualTo( expected: true); }
```

Task 2.1

I chose to create test cases for the following methods:

src/main/java/nl/tudelft/jpacman/board/Direction.java/getDeltaX
src/main/java/nl/tudelft/jpacman/level/Player.java/getScore
src/main/java/nl/tudelft/jpacman/level/Pellet.java/getValue

getDeltaX

Coverage results before and after creating a test case for getDeltaX can be seen below. Project coverage went from 16% (18/110) class, 9% (60/624) method, 8% (190/2306) line to 16% (18/110) class, 9% (62/624) method, and 8% (192/2306) line coverage.

Before getDeltaX test case:

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)

After getDeltaX test case:

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (62/624)	8% (192/2306)
tudelft	16% (18/110)	9% (62/624)	8% (192/2306)
jpacman	16% (18/110)	9% (62/624)	8% (192/2306)
board	20% (4/20)	11% (12/106)	10% (30/282)
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)

getDeltaX snippet:

```
@Test
void testGetDeltaX() {
    Direction test = Direction.EAST;
    assertThat(test.getDeltaX()).isEqualTo( expected: 1);
}
```

getScore

Coverage results before and after creating a test case for the getScore method can be seen below. Project coverage went from 16% (18/110) class, 9% (62/624) method, 8% (192/2306) line to 16% (18/110) class, 10% (66/624) method, and 8% (198/2306) line coverage.

Before getScore test case:

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (62/624)	8% (192/2306)
tudelft	16% (18/110)	9% (62/624)	8% (192/2306)
jpacman	16% (18/110)	9% (62/624)	8% (192/2306)
board	20% (4/20)	11% (12/106)	10% (30/282)
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)

After getScore test case:

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	10% (66/624)	8% (198/2306)
tudelft	16% (18/110)	10% (66/624)	8% (198/2306)
jpacman	16% (18/110)	10% (66/624)	8% (198/2306)
board	20% (4/20)	11% (12/106)	10% (30/282)
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)	8% (14/156)	4% (32/700)
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)

getScore snippet:

```
@Test
void testGetScore() {
    int testValue = 5;
    int oldPlayerScore = player.getScore();
    player.addPoints(testValue);
    assertThat(player.getScore()).isEqualTo( expected: oldPlayerScore+testValue);
}
```

getValue

Coverage results before and after creating a test case for the getValue method can be seen below. Project coverage went from 16% (18/110) class, 10% (66/624) method, 8% (198/2306) line to 18% (20/110) class, 11% (72/624) method, and 9% (210/2308) line coverage.

Before getValue test case:

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	10% (66/624)	8% (198/2306)
tudelft	16% (18/110)	10% (66/624)	8% (198/2306)
jpacman	16% (18/110)	10% (66/624)	8% (198/2306)
board	20% (4/20)	11% (12/106)	10% (30/282)
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)	8% (14/156)	4% (32/700)
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)

After getValue test case:

Element	Class, %	Method, %	Line, %
nl	18% (20/110)	11% (72/624)	9% (210/2308)
tudelft	18% (20/110)	11% (72/624)	9% (210/2308)
jpacman	18% (20/110)	11% (72/624)	9% (210/2308)
board	20% (4/20)	11% (12/106)	10% (30/282)
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)	11% (18/156)	5% (42/702)
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)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

getScore snippet:

```
public class PelletTest {
    1 usage
    private static final Sprite PELLET_SPRITE = new PacManSprites().getPelletSprite();
    2 usages
    private static final int TEST_POINT_VALUE = 10;
    1 usage
    private Pellet pellet = new Pellet(TEST_POINT_VALUE, PELLET_SPRITE);
    @Test
    void testGetValue() {
        assertThat(pellet.getValue()).isEqualTo(TEST_POINT_VALUE);
    }
}
```

Task 3

Are the coverage results from `JaCoCo` similar to the ones you got from `IntelliJ` in the last task? Why so or why not?

The JaCoCo coverage results are not similar to the IntelliJ coverage results. The JaCoCo results report back much less methods and classes. For example, it only reported 268 total methods versus 624 total methods reported by IntelliJ. The JaCoCo results also report greater code coverage than IntelliJ. For example, JaCoCo reports 67% instruction coverage of the level package versus 23% class, 11% method, and 5% line coverage reported by IntelliJ.

Did you find helpful the source code visualization from `JaCoCo` on uncovered branches?

I think that the source code visualization is very useful especially for uncovered branches. This can prove very useful in debugging and help have more extensive and complete code coverage.

Which visualization did you prefer and why? `IntelliJ`'s coverage window or `JaCoCo`'s report?

I prefer the JaCoCo's visualization of the source code and highlighting uncovered branches. I think that this is especially useful in achieving complete code coverage, as well as catching hidden or overlooked edge cases. However, for viewing percentage of code completion I prefer the IntelliJ coverage window. I prefer this because from the classes that I counted the count seem more accurate, also I think that it is more convenient to view the coverage results within the IDE especially if you are working on multiple test cases and frequent changes. Whereas the visualization of JaCoCo's is useful when doing a thorough investigation of code coverage.

Team Fork Repo Link: https://github.com/Lettuce05/cs472-team6
jpacman Fork Repo Link: https://github.com/Lettuce05/jpacman/tree/TestingLab