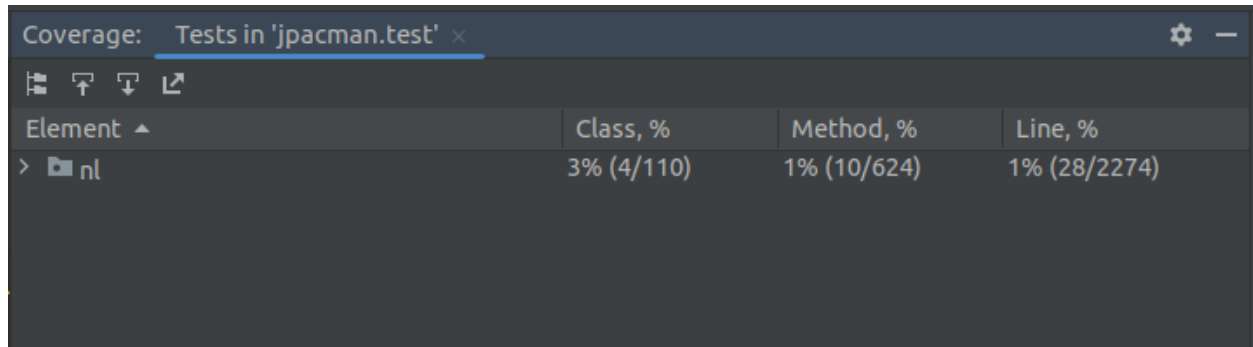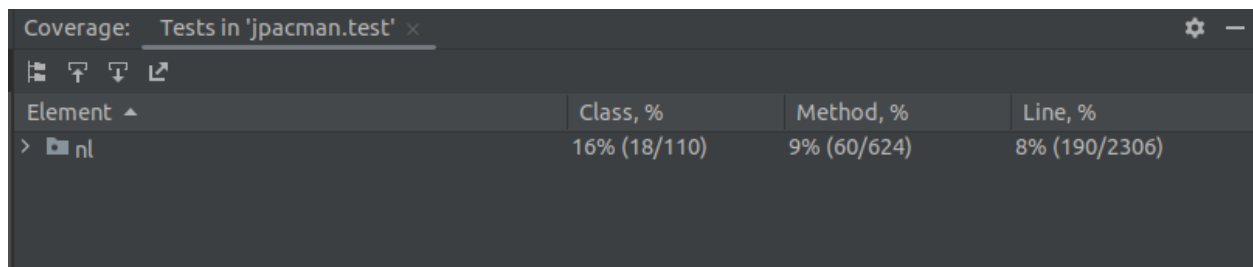# Unit Testing Lab Report

Task 1

After running the test with coverage, there was 3% coverage. The coverage is not good enough but 3% is too low.

Coverage:	Tests in 'jpacman.test' ×			✿ —
Element ▲		Class, %	Method, %	Line, %
> ▣ nl		3% (4/110)	1% (10/624)	1% (28/2274)

Task 2

Coverage:	Tests in 'jpacman.test' ×			✿ —
Element ▲		Class, %	Method, %	Line, %
> ▣ nl		16% (18/110)	9% (60/624)	8% (190/2306)

Task 2.1

setKiller Test

```java
public class SetKillerTest {
    2 usages
    Unit killer;
    1 usage
    private static final PacManSprites newSprite = new PacManSprites();
    1 usage
    private final PlayerFactory Factory = new PlayerFactory(newSprite);
    2 usages
    private final Player thePlayer = Factory.createPacMan();
    no usages
    @Test
    void testSetKiller() {
        thePlayer.setKiller(killer);
        assertThat(thePlayer.getKiller()).isEqualTo(killer);
    }
}
```

Before testing setKiller method

Coverage:	Tests in 'jpacman.test' ×			⚙ —

Element ▲	Class, %	Method, %	Line, %
> ▦ nl	16% (18/110)	9% (60/624)	8% (190/2306)

After testing setKiller method (Method % increases 1%)

Element ▲	Class, %	Method, %	Line, %
> ▦ nl	16% (18/110)	10% (64/624)	8% (196/2306)

setAlive Test

```java
public class SetAliveTest {
    1 usage
    private static final PacManSprites newSprite = new PacManSprites();
    1 usage
    private final PlayerFactory Factory = new PlayerFactory(newSprite);
    2 usages
    private final Player thePlayer = Factory.createPacMan();
    no usages
    @Test
    void testSetAlive() {
        thePlayer.setAlive(true);
        assertThat(thePlayer.isAlive()).isEqualTo( expected: true);
    }
}
```

Before testing setAlive method

Element ▲	Class, %	Method, %	Line, %
> ▦ nl	16% (18/110)	10% (64/624)	8% (196/2306)

After testing setAlive method (Line % increases 1%)

Coverage:	Tests in 'jpacman.test' ✕			⚙ ―
Element ▲		Class, %	Method, %	Line, %
❯ 📁 nl		16% (18/110)	10% (66/624)	9% (208/2306)

Pellet Constructor Test

```java
public class PelletTest {
    2 usages
    Sprite newSprite;
    2 usages
    int points;
    no usages
    @Test
    void testPelletConstructor() {
        Pellet newPellet = new Pellet(points, newSprite);
        assertThat(newPellet.getSprite()).isEqualTo(newSprite);
        assertThat(newPellet.getValue()).isEqualTo(points);
    }
}
```

Before testing Pellet constructor

Coverage:	Tests in 'jpacman.test' ✕			⚙ ―
Element ▲		Class, %	Method, %	Line, %
❯ 📁 nl		16% (18/110)	10% (66/624)	9% (208/2306)

After testing Pellet constructor (Class % increases 2%, Method % increases 1%)

Coverage:	Tests in 'jpacman.test' ✕			⚙ ―
Element ▲		Class, %	Method, %	Line, %
❯ 📁 nl		18% (20/110)	11% (72/624)	9% (220/2308)

Task 3

- The coverage results on JaCoCo are similar to the results on IntelliJ but the results on JaCoCo seem to be higher than the IntelliJ results.
- The source code visualization on JaCoCo is helpful because you can easily see which lines are not covered (highlighted red) and which lines are covered (highlighted green). You can easily identify which methods/lines need unit tests.
- I liked the JaCoCo report better than the IntelliJ coverage window because there is more information and the visualization is easier to understand. The highlighting of uncovered lines in the source code is also helpful.

Forked Repo Link: https://github.com/JaneenSudiacal/cs472project