Joseph O'Connor
CS 472 Lab 2
Task 3 Report
Fork repository link: https://github.com/Joeyo364/add_Joey

Unit Tests

In task 2.1 we were tasked with creating more unit tests for IntelliJ to expand coverage for the JPacman repository. For this I chose three methods to test: createBlinky (GhostFactory.java), nextMove (Ghost.java), and getScore (Player.Java). Two of these methods are tested in a single class, GhostTest.java, while the other is tested in PlayerTest.java. In the screenshots below you can see the entirety of the GhostTest class:

```
public class GhostTest {
    1 usage
    private static final PacManSprites sprites = new PacManSprites();
    1 usage
    private GhostFactory Factory = new GhostFactory(sprites);
    3 usages
    private final Ghost blinky = Factory.createBlinky();

    /* * Test that Blinky exists as an object */
    new *
    @Test
    void testBlinky(){ assertThat(blinky).isNotNull(); }

    /* * Use Blinky to test that nextMove is in correct possible directions */
    new *
    @Test
    void testNextMove(){
        if(blinky.hasSquare()){
            assertThat(blinky.nextMove()).isIn(Direction.NORTH, Direction.SOUTH,
                Direction.EAST, Direction.WEST);
        }
    }
}
```

The purpose of testBlinky is to ensure that the creation of the game object that is Blinky the ghost was instantiated correctly. The purpose of testNextMove is to ensure that the ghost AI does not try to move in an illegal direction set by the rules of the game, it uses Blinky to test for this. The unit test for the getScore method in PlayerTest.java can be seen below:

```
@Test
void testScore() { assertThat(ThePlayer.getScore()).isGreaterThanOrEqualTo( other: 0); }
```

This test simply asserts that a negative value is not assigned as the players score, as that cannot happen in the game.

Joseph O'Connor
CS 472 Lab 2
Task 3 Report

Questions

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

The results are not the same, this is because IntelliJ uses the unit tests that are provided to it by the developer (me and the rest of group 8) while JaCoCo uses its own set of tools to calculate the code coverage of the repository.

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

This is a helpful tool to discover gaps in unit tests and can be used to fix errors that might occur in different branches of the repository.

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

I personally prefer the IntelliJ report because I think it's more readable and I like how it looks visually. I like that the file structure is shown in IntelliJ's coverage, I dislike how JaCoCo uses the entire filepath to name the packages and that a new window/page must be opened to view what is contained in those packages.