Matthew Ocampo

CS 472

Section 1001

2/5/2024

https://github.com/tbg510/cs472-group4

Software Testing: Unit Testing Report

**Task 2**

For this task 3 unit tests were created for 3 methods in the Pellet and AnimatedSprite classes: the Pellet constructor in the Pellet class, and the restart and split methods in the AnimatedSprite class. Before the AnimatedSprite tests, an AnimatedSprite object was created.

```
@Test
void testPellet()
{
    // Make new pellet object
    Pellet pellet = new Pellet( points: 10, PELLET_SPRITE);

    // Check if pellet object actually created
    assertThat(pellet).isNotNull();
}
```

The constructor test created a new pellet object and tested to see if the object was actually created (if not null).

The restart test simply called the AnimatedSprite object's restart method.

```
@Test
void testRestart()
{
    // Call AnimatedSprite object restart method
    SPRITE_CHECK.restart();
}
```

```
@Test
void testSplit()
{
    // Check if split returns valid Sprite
    assertThat(SPRITE_CHECK.split( x: 1, y: 1, width: 10, height: 10)).isNotNull();
}
```

The split test checked to see if the AnimatedSprite object's split method returned a valid Sprite object.

After these unit tests were created, the test coverage increased by a marginal amount (95 lines to 113 lines), since these methods contained a relatively small amount of lines (18).

**Task 3**

The JaCoCo report is not similar to the IntelliJ test coverage. The JaCoCo report states that only 6 classes were not tested, while in the IntelliJ test coverage 46 classes were not tested. Additionally, the JaCoCo report did not test 51 methods, while the IntelliJ test coverage did not test 278 methods.

JaCoCo's source code visualization on uncovered branches was helpful. It allowed an easy representation of several untested branches throughout the project. Overall, JaCoCo's report seemed preferable when visualization test coverage since it is easy to look through and see methods that are tested or not.