Ashley Bertrand
Executive Summary – A3

Testing Approach

The primary goal of my testing approach was to ensure complete test coverage. In order to do so, I set up a test plan so that the test harness would cover all of the instructions and branches that each method under test has, including all loops, if/else statements, switch statement cases, etc. After setting up each test case, in the assert statements, I called all related methods on the objects to test for correct behavior. I then checked for missed instructions and missed branches in JaCoCoverage from EclEmma.

Number of Tests

I wrote eight testing methods and each had independent assert statements called.

- testEnterDoor(): 6 assert calls
- testGo(): 2 assert calls
- testPickUp(): 15 assert calls
- testDrop(): 11 assert calls
- testAddItem(): 2 assert calls
- testRemoveItem(): 1 assert call
- testEnterRoom(): 2 assert calls
- testExitRoom(): 2 assert calls

Coverage

Class	Method	Missed Instructions Coverage	Missed Branches Coverage
Door	enter()	100%	83%
Player	go()	100%	n/a
Player	pickUp()	100%	100%
Player	drop()	100%	89%
Room	addItem()	100%	n/a
Room	removeItem()	100%	n/a
Room	enter()	100%	n/a
Room	exit	100%	n/a

In class Door, method enter(), EclEmma indicates the line `else if (p.getLoc() == inSite)` as yellow, giving a warning of incomplete coverage. This is because there is no final 'else' in this block of code to serve as a catch-all for remaining input conditions within the method, and complete test coverage cannot be met.

In class Player, method drop(), EclEmma indicates the line `switch (itemNum)` as yellow, giving a warning of incomplete coverage. This is because there is no 'default' case in this switch statement to serve as a catch-all for remaining input conditions called on the method, and complete test coverage cannot be met.

Failures

Through testing, there were no failures. However, two of the methods, as described above, did not reach 100% coverage, and therefore, full test coverage is not reached. This has to do with the structure of the methods, not because the actual functionality is incorrect.