**Jess Rudolph**

**Project 3**

**Task 2 – Getting Started**

The error in the Fibonacci.java file was that “case 0” of the switch statement needed to equal 0 instead of 1. I figured this out by glancing at the testFibonacci method.

**Task 3 – Rectangle**

When I ran the Junit Test on the RectangleTest.java class, the error showed that the expected value did not match the actual value. In order to find the error, I traced the code and found that the Point method had this.x =y so I changed it to this.x =x.

The Rectangle class was refactored to eliminate duplicated code for the points for the axes. I did this by using Eclipse’s “Extract Local Variable”.

public class Rectangle {

/\*\* The p2. \*/

private Point p1, p2;

private double xAxis;

private double yAxis;

/\*\*

\* Instantiates a new rectangle.

\*

\* @param p1 the p1

\* @param p2 the p2

\*/

Rectangle(Point p1, Point p2) {

this.p1 = p1;

this.p2 = p2;

xAxis = Math.abs(p2.x - p1.x);

yAxis = Math.abs(p2.y - p1.y);

}

/\*\*

\* Gets the area.

\*

\* @return the area

\*/

public Double getArea() {

return (xAxis \* yAxis);

}

/\*\*

\* Gets the diagonal.

\*

\* @return the diagonal

\*/

public Double getDiagonal() {

return Math.sqrt(Math.pow((xAxis), 2) + Math.pow((yAxis), 2));

}

}

**Task 4 – A Vending Machine**

I did not find any bugs.

**Task 5 – Summing it All up**

In completing this project, I learned how to write test cases. I understand that a test case needs to be present for each conceivable scenario. The biggest learning for me was to only have one assertion method per test case. I began writing the test cases in the format of the Rectangle project. I later realized the Junit cheat sheet stated to only have one per method and had to refactor my code.

I like Junit’s support for writing test cases. I like that I can select which methods to add test cases for based on the list of methods in that class. I like that it also provides the setup and teardown if you opt to include those.