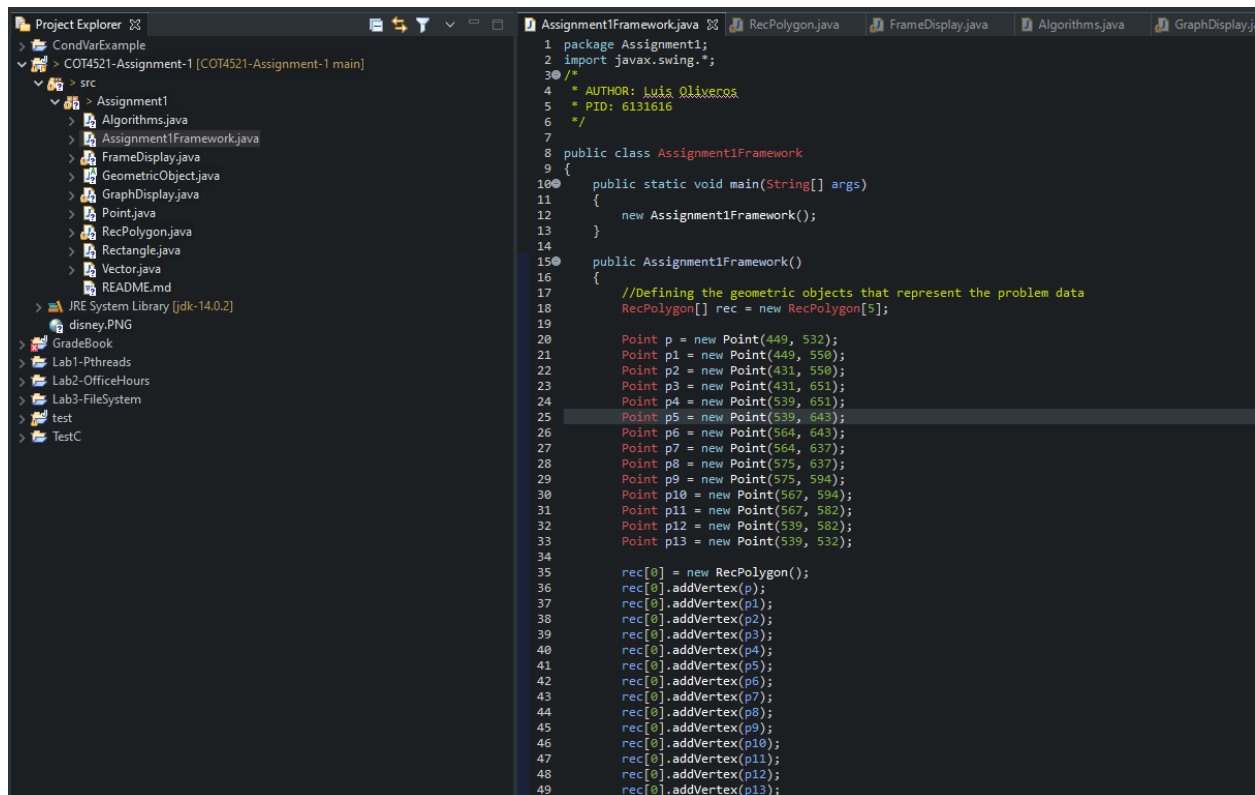


One of the challenges I had while implementing a solution to this project is working on the point containment algorithm for rectilinear polygons. Trying to implement this algorithm taught me that it is even more difficult to implement an algorithm for point containment of a polygon since not every vertical or horizontal line has to meet at a 90 degree angle. I would say that reviewing the lecture slides and the in class lectures helped with this problem to an extent.

Below are three images, two show the project 1 code in the Eclipse IDE, and the other shows the map when the program is executed.



The screenshot shows the Eclipse IDE with the Project Explorer on the left and the Editor on the right. The Project Explorer shows a project named 'COT4521-Assignment-1' with a 'src' folder containing several Java files. The Editor shows the code for 'Assignment1Framework.java'.

```
1 package Assignment1;
2 import javax.swing.*;
3
4 /*
5  * AUTHOR: Luis Oliveros
6  * PID: 6131616
7  */
8 public class Assignment1Framework
9 {
10     public static void main(String[] args)
11     {
12         new Assignment1Framework();
13     }
14
15     public Assignment1Framework()
16     {
17         //Defining the geometric objects that represent the problem data
18         RecPolygon[] rec = new RecPolygon[5];
19
20         Point p = new Point(449, 532);
21         Point p1 = new Point(449, 550);
22         Point p2 = new Point(431, 550);
23         Point p3 = new Point(431, 651);
24         Point p4 = new Point(539, 651);
25         Point p5 = new Point(539, 643);
26         Point p6 = new Point(564, 643);
27         Point p7 = new Point(564, 637);
28         Point p8 = new Point(575, 637);
29         Point p9 = new Point(575, 594);
30         Point p10 = new Point(567, 594);
31         Point p11 = new Point(567, 582);
32         Point p12 = new Point(539, 582);
33         Point p13 = new Point(539, 532);
34
35         rec[0] = new RecPolygon();
36         rec[0].addVertex(p);
37         rec[0].addVertex(p1);
38         rec[0].addVertex(p2);
39         rec[0].addVertex(p3);
40         rec[0].addVertex(p4);
41         rec[0].addVertex(p5);
42         rec[0].addVertex(p6);
43         rec[0].addVertex(p7);
44         rec[0].addVertex(p8);
45         rec[0].addVertex(p9);
46         rec[0].addVertex(p10);
47         rec[0].addVertex(p11);
48         rec[0].addVertex(p12);
49         rec[0].addVertex(p13);
50     }
51 }
```

```

110
111     double y = vertexList[i].getY();
112     if (y > max)
113     {
114         max = y;
115     }
116     return max;
117 }
118
119 public String toString()
120 {
121     String str = "POLY RECTANGLE " + super.toString() + "\n";
122     for (int i = 0; i < vertexNumber; i++)
123     {
124         str += vertexList[i] + "\n";
125     }
126     return str;
127 }
128
129 public double smallestX()
130 {
131     double min = vertexList[0].getX();
132     for (int i = 0; i < vertexNumber; i++)
133     {
134         double x = vertexList[i].getX();
135         if (x < min)
136         {
137             min = x;
138         }
139     }
140     return min;
141 }
142
143 public double smallestY()
144 {
145     double min = vertexList[0].getY();
146     for (int i = 0; i < vertexNumber; i++)
147     {
148         double y = vertexList[i].getY();
149         if (y < min)
150         {
151             min = y;
152         }
153     }
154     return min;
155 }
156
157 public void translate(vector v)
158 {
159     for (Point p: vertexList)
160     {
161         p.translate(v);
162     }
163 }
164
165 public boolean isPointInRecPolygon(Point p)
166 {
167     int j = 1;
168     for (int i = 0; i < vertexNumber; i++)
169     {
170

```

