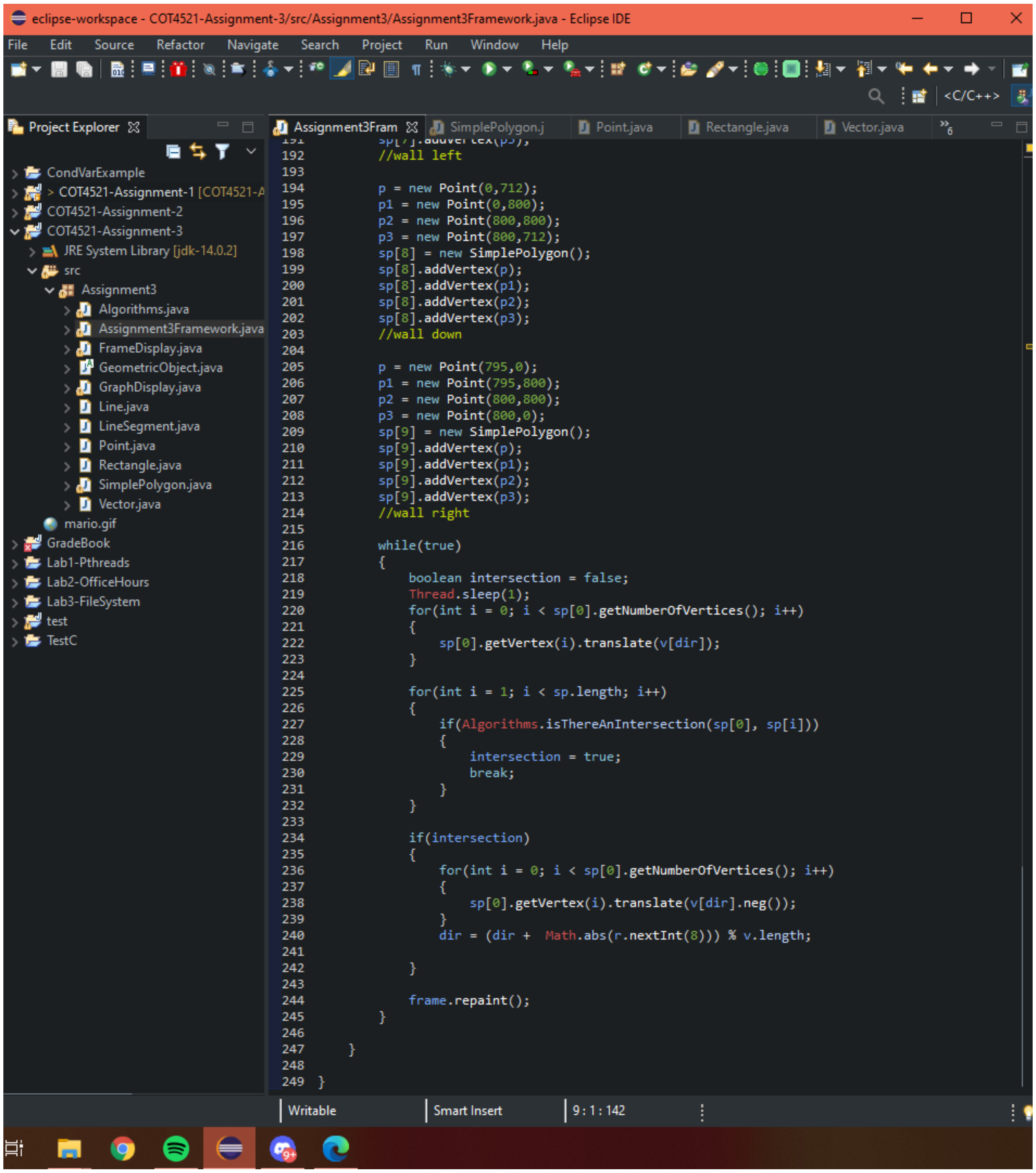
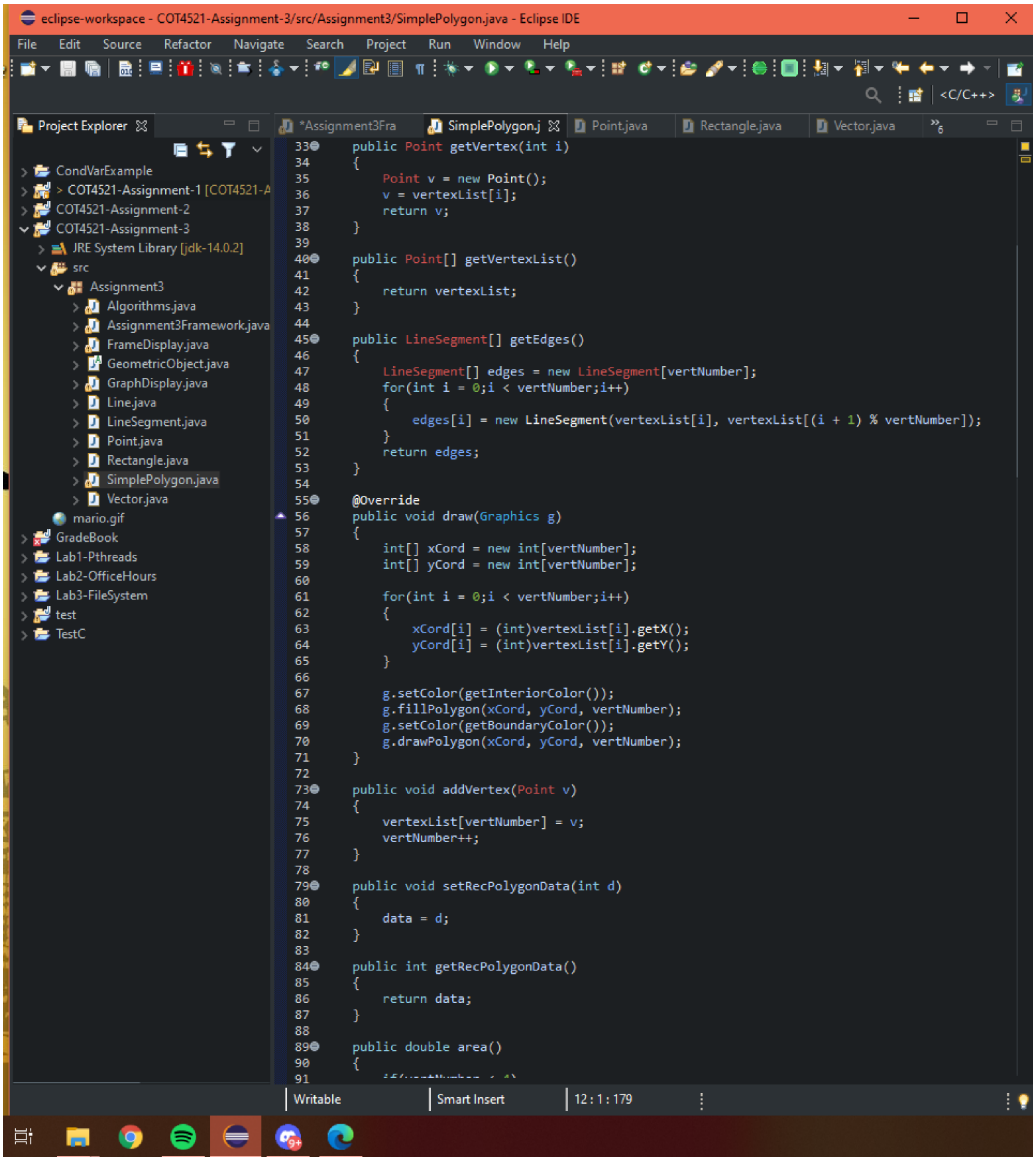
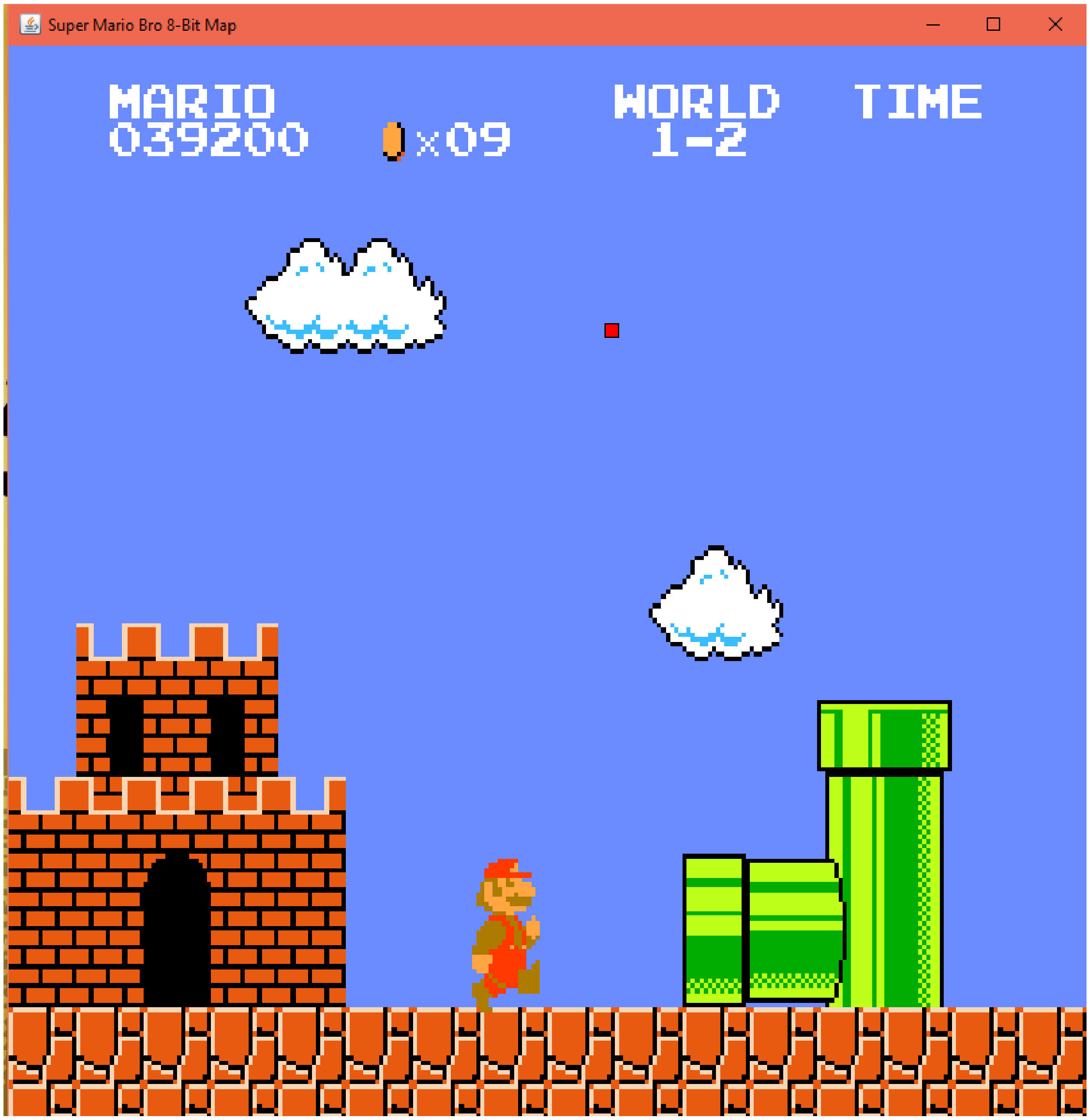
Project 3 was full of hurdles that I had to overcome while implementing the solution. The most challenging component of the project was figuring out the watchman route algorithm, that uses the area of polygons and a specific watchman polygon i.e the first polygon object and travels the obstacles. Then if it intersects a polygon it must know to change directions without traversing said polygon. 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 3 code in the Eclipse IDE, and the other shows the map when the program is executed to do the watchman route algorithm on simple polygon objects.



```
        //wall left

        p = new Point(0,712);
        p1 = new Point(0,800);
        p2 = new Point(800,800);
        p3 = new Point(800,712);
        sp[8] = new SimplePolygon();
        sp[8].addVertex(p);
        sp[8].addVertex(p1);
        sp[8].addVertex(p2);
        sp[8].addVertex(p3);
        //wall down

        p = new Point(795,0);
        p1 = new Point(795,800);
        p2 = new Point(800,800);
        p3 = new Point(800,0);
        sp[9] = new SimplePolygon();
        sp[9].addVertex(p);
        sp[9].addVertex(p1);
        sp[9].addVertex(p2);
        sp[9].addVertex(p3);
        //wall right

        while(true)
        {
            boolean intersection = false;
            Thread.sleep(1);
            for(int i = 0; i < sp[0].getNumberOfVertices(); i++)
            {
                sp[0].getVertex(i).translate(v[dir]);
            }

            for(int i = 1; i < sp.length; i++)
            {
                if(Algorithms.isThereAnIntersection(sp[0], sp[i]))
                {
                    intersection = true;
                    break;
                }
            }

            if(intersection)
            {
                for(int i = 0; i < sp[0].getNumberOfVertices(); i++)
                {
                    sp[0].getVertex(i).translate(v[dir].neg());
                }
                dir = (dir +  Math.abs(r.nextInt(8))) % v.length;

            }

            frame.repaint();
        }

    }

}
```



```java
    public Point getVertex(int i)
    {
        Point v = new Point();
        v = vertexList[i];
        return v;
    }

    public Point[] getVertexList()
    {
        return vertexList;
    }

    public LineSegment[] getEdges()
    {
        LineSegment[] edges = new LineSegment[vertNumber];
        for(int i = 0;i < vertNumber;i++)
        {
            edges[i] = new LineSegment(vertexList[i], vertexList[(i + 1) % vertNumber]);
        }
        return edges;
    }

    @Override
    public void draw(Graphics g)
    {
        int[] xCord = new int[vertNumber];
        int[] yCord = new int[vertNumber];

        for(int i = 0;i < vertNumber;i++)
        {
            xCord[i] = (int)vertexList[i].getX();
            yCord[i] = (int)vertexList[i].getY();
        }

        g.setColor(getInteriorColor());
        g.fillPolygon(xCord, yCord, vertNumber);
        g.setColor(getBoundaryColor());
        g.drawPolygon(xCord, yCord, vertNumber);
    }

    public void addVertex(Point v)
    {
        vertexList[vertNumber] = v;
        vertNumber++;
    }

    public void setRecPolygonData(int d)
    {
        data = d;
    }

    public int getRecPolygonData()
    {
        return data;
    }

    public double area()
    {
```


Super Mario Bro 8-Bit Map
MARIO
039200
x09
WORLD
1-2
TIME