MARINA CASTELLÀ OLIVELLA
NAIARA GARMENDIA OLABARRIA

LAB 4

The aim of this session is to implement the design from seminar 4. We have to implement regions, with different geometrical shapes and be able to draw and to move them. In this case we have a class entity, that is an abstract class, and is the class that all the shapes that can be drawn inherits from. In this class we have the two abstract methods ('translate' and 'draw') that are going to be implemented in every class that inherits from there.

The first step in this lab was to create the classes 'MyPoint' and 'MyVector'. The point class was the same as the one we used in the previous labs, but we have to add two new methods, 'translate' and 'difference'.

```
//transalate method
public void translate( int dx, int dy ) {
    x = x + dx;
    y = y + dy;
}
//difference method
public MyVector difference(MyPoint p){
    int coord1 = p.getX() - x;
    int coord2 = p.getY() - y;
    MyVector vector = new MyVector(coord1, coord2);
    return vector;

}
```

In the class 'MyVector' the method we have to implement is the one that calculates the cross product of two vectors.

```
//cross product method
public double crossProduct(MyVector vector1){
    int op1 = x * vector1.getY();
    int op2 = y * vector1.getX();
    double result = op1 - op2;
    return result;
}
```

When we've already coded these two classes, we start by creating the classes that inherit from the class region, which are all the different shapes that can be drawn in the window that is going to open. We have the polygonal region and the ellipsoidal region, in this two

classes, we have to implement the ‘draw method’, ‘translate method’ and the ‘isPointInside method’

- Draw: in the draw method, we use the methods that are already in a library of Java such as g.fillPolygon, fillOval…. We also use the java.awt library, to get the different colors that already exist in it.
- IsPointInside: in EllipsoidalRegion, we only have to apply the formula that they gave us in the pdf. For PolygonalRegion is a little bit different, we have to use the method from the classes ‘MyPoint’ and ‘MyVector’, to see if the cross product of every pair of points has the same signal.

MyPolygon

```
@Override
public boolean isPointInside(MyPoint p) {
    int pSize = points.size();

    MyPoint p1 = points.get(pSize - 1);
    MyPoint p2 = points.get(0);

    MyVector d1 = p2.difference(p1);
    MyVector d2 = p.difference(p1);

    double a = d1.crossProduct(d2);

    for (int i = 0; i<pSize - 1; i++){
        p1 = points.get(i);
        p2 = points.get(i+1);
        d1 = p2.difference(p1);
        d2 = p.difference(p1);
        double b = d1.crossProduct(d2);

        if(a*b < 0){
            return false;
        }
    }
    return true;
}
```

EllipsoidalRegion

```
@Override
public boolean isPointInside(MyPoint p) {
    double num1 = p.getX()-c.getX();
    double num2 = p.getY()-c.getY();
    double part1 =(Math.pow(num1, 2)/Math.pow(r1,2))+(Math.pow(num2, 2)/Math.pow(r2,2));
    if (part1 <= 1){
        return true;
    } else{
        return false;
    }


}
```

- Translate: The functionality of this function is to move the region along the x and y axes.

```
@Override
public void translate( int dx, int dy ) {
    for (int i = 0; i < points.size(); i++ ) {
        int coordX = points.get(i).getX();
        int coordY = points.get(i).getY();

        coordX = coordX + dx;
        coordY = coordY + dy;

        points.get(i).setX(coordX);
        points.get(i).setY(coordY);
    }
}
```

```
@Override
public void translate(int dx, int dy) {
    int sx = c.getX();
    int sy = c.getY();
    sx = sx + dx;
    sy = sy + dy;
    c.setX(sx);
    c.setY(sy);
}
```

The classes Triangular Region, Rectangular Region and Circular region, only need the constructor method, because they inherit the methods from their parents.

After creating all the classes, we create the testing class, in this class we create an instance of EntityDrawer and we create several entities and add them to the graphical application. This was the main class, and with this we were able to see the resulting window of our code.

Overall, this lab was a little difficult for us, because even though we knew how each class and method should work, when implementing it, some errors appeared that were a little bit difficult for us to solve. After solving it we have to change some things in order to get the program to work in the way that we want.

At the end we were able to solve almost everything but we have one issue with moving our circles. In the computer it appears like the points have changed their position, but then in the window, the circles are not moving. We were not able to fix this, because we really don't know what is wrong with our code.