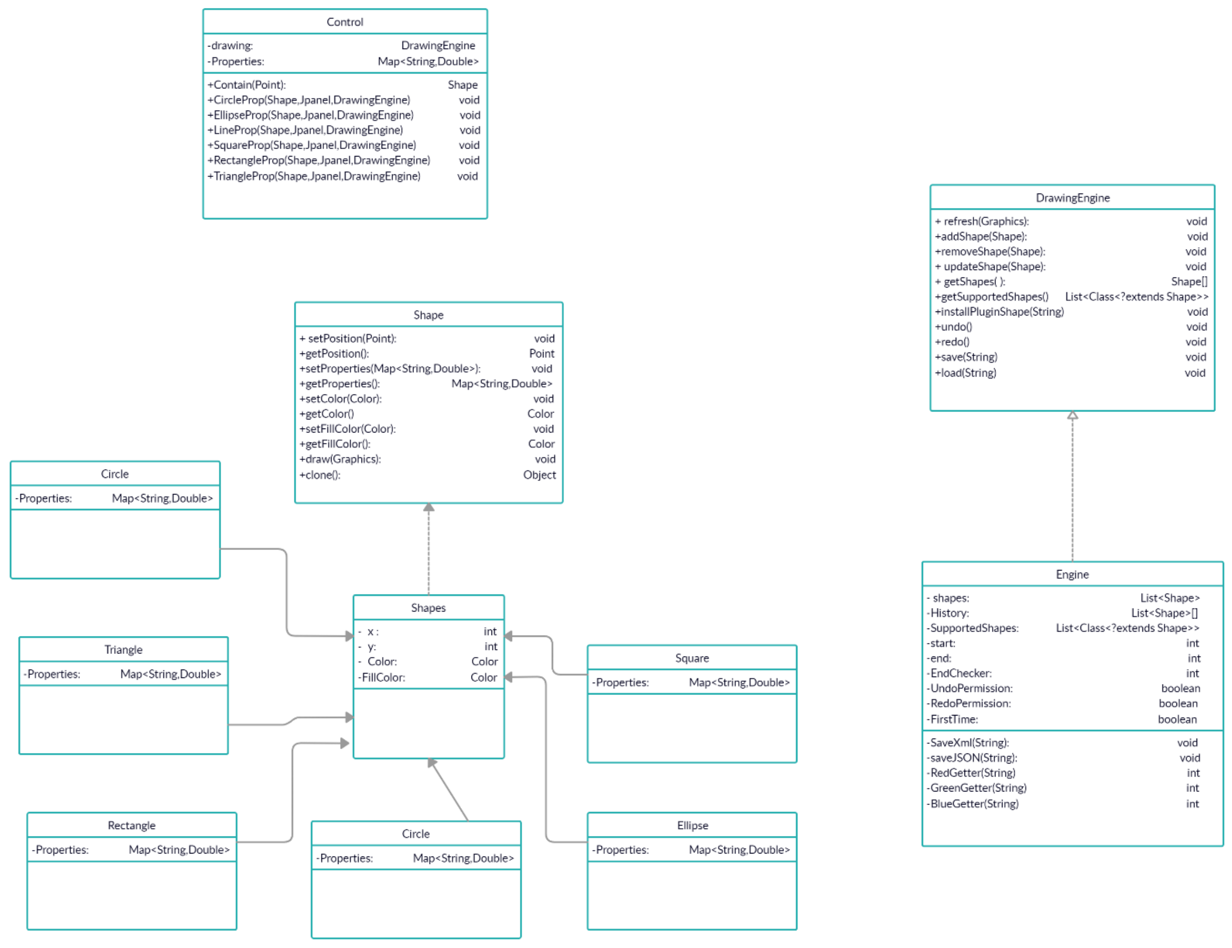
Vector based drawing application

Problem statement: A vector based drawing application was to be designed that supports 6 shapes: ellipse, circle, line segment, rectangle, square and triangle. A UML class diagram representing of the model. GUI with 2D Graphics capabilities. Enable of dynamic extensions.

Part1: UML (Unified Modeling Language):

As a first step to design the program we drew a UML class diagram in order to set plans on how we should work, taking into consideration the polymorphism and inheritance concepts.

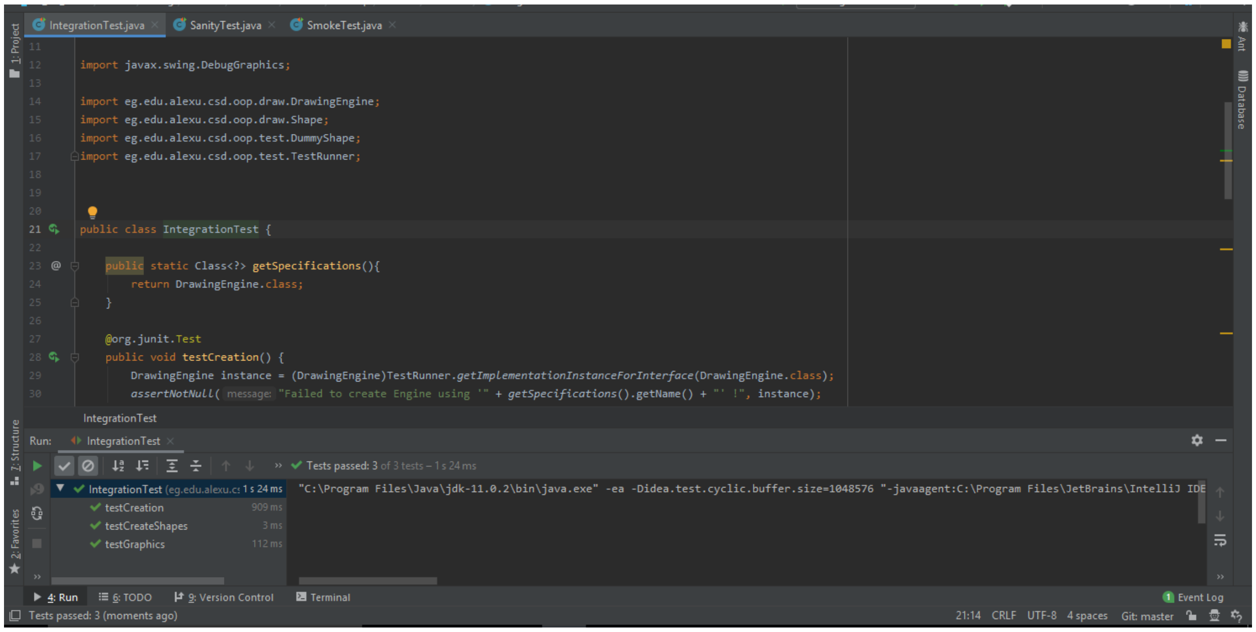
An interface called "Shape" includes main methods like setting and getting : positions, color, fill color and a hashmap of properties, and at last a draw and a clone methods. We created an abstract class "Shapes" that implements this interface and from which extends all other shapes like : square, triangle, ellipse, etc. Another interface "DrawingEngine" and an "Engine" class that implements it and contain the organizing part of the project : "refresh" to redraw all shapes, "addShape" to add the new shape in the list of shapes, "removeShape" to remove it from the list, an "updateShape" which replaces a shape with another shape of the same family but with different properties, "getSupportedShapes" which returns a list of all shapes that we can draw including the dynamically loading classes, "undo" & "redo" methods, "save" & "load" and finally a "getShapes" which returns an array of the drawn shapes. In order not to write methods in the GUI class we created a “Control” class which contains methods related to the GUI : “Contains” when clicked on a shape returns the shape to edit it and a show properties to edit ,clone and delete shapes.



Control
-drawing: DrawingEngine
-Properties: Map<String,Double>
+Contain(Point): Shape
+CircleProp(Shape,Jpanel,DrawingEngine) void
+EllipseProp(Shape,Jpanel,DrawingEngine) void
+LineProp(Shape,Jpanel,DrawingEngine) void
+SquareProp(Shape,Jpanel,DrawingEngine) void
+RectangleProp(Shape,Jpanel,DrawingEngine) void
+TriangleProp(Shape,Jpanel,DrawingEngine) void
DrawingEngine
+ refresh(Graphics): void
+addShape(Shape): void
+removeShape(Shape): void
+ updateShape(Shape): void
+ getShapes( ): Shape[]
+getSupportedShapes() List<Class<?extends Shape>>
+installPluginShape(String) void
+undo() void
+redo() void
+save(String) void
+load(String) void
Shape
+ setPosition(Point): void
+getPosition(): Point
+setProperties(Map<String,Double>): void
+getProperties(): Map<String,Double>
+setColor(Color): void
+getColor() Color
+setFillColor(Color): void
+getFillColor(): Color
+draw(Graphics): void
+clone(): Object
Circle
-Properties: Map<String,Double>
Shapes
- x : int
- y: int
- Color: Color
-FillColor: Color
Triangle
-Properties: Map<String,Double>
Square
-Properties: Map<String,Double>
Rectangle
-Properties: Map<String,Double>
Circle
-Properties: Map<String,Double>
Ellipse
-Properties: Map<String,Double>
Engine
- shapes: List<Shape>
-History: List<Shape>[]
-SupportedShapes: List<Class<?extends Shape>>
-start: int
-end: int
-EndChecker: int
-UndoPermission: boolean
-RedoPermission: boolean
-FirstTime: boolean
-SaveXml(String): void
-saveJSON(String): void
-RedGetter(String) int
-GreenGetter(String) int
-BlueGetter(String) int

We were more concerned with finishing the logic first and passing the test cases instead of creating a fancy GUI with a wrong logic.

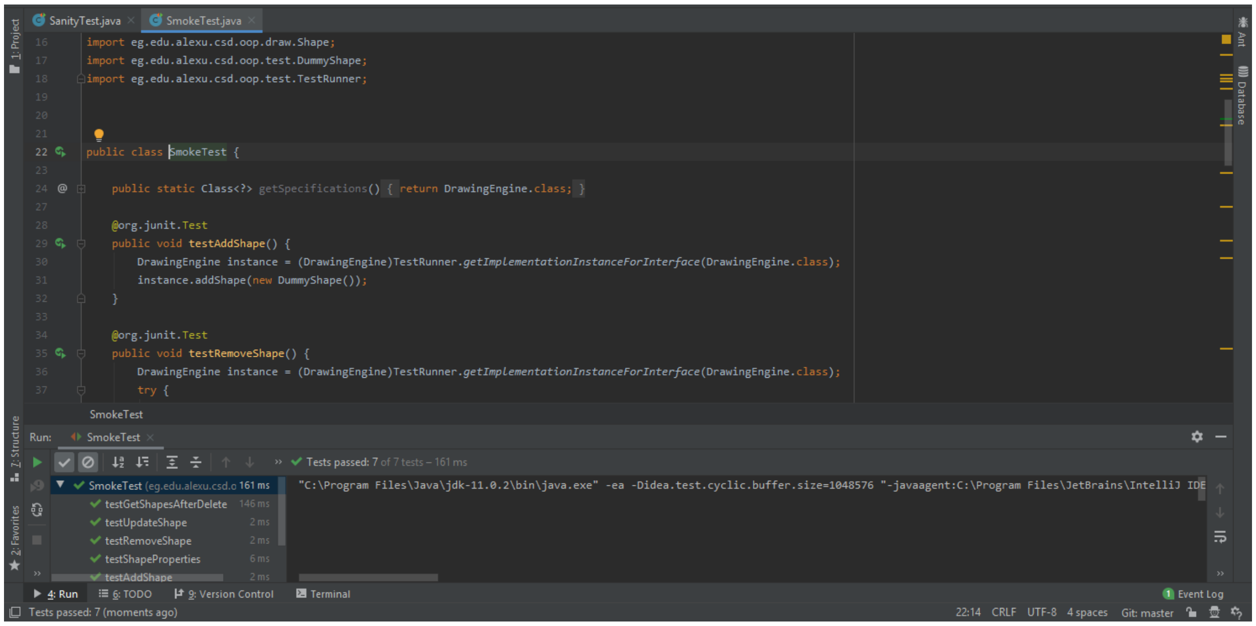
Integration test:



```java
import javax.swing.DebugGraphics;

import eg.edu.alexu.csd.oop.draw.DrawingEngine;
import eg.edu.alexu.csd.oop.draw.Shape;
import eg.edu.alexu.csd.oop.test.DummyShape;
import eg.edu.alexu.csd.oop.test.TestRunner;


public class IntegrationTest {

    public static Class<?> getSpecifications(){
        return DrawingEngine.class;
    }

    @org.junit.Test
    public void testCreation() {
        DrawingEngine instance = (DrawingEngine)TestRunner.getImplementationInstanceForInterface(DrawingEngine.class);
        assertNotNull( message: "Failed to create Engine using '" + getSpecifications().getName() + "' !", instance);
```

```
Tests passed: 3 of 3 tests – 1 s 24 ms
IntegrationTest (eg.edu.alexu.c: 1 s 24 ms
    testCreation        909 ms
    testCreateShapes      3 ms
    testGraphics        112 ms
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE
Tests passed: 3 (moments ago)
```

Smoke test:

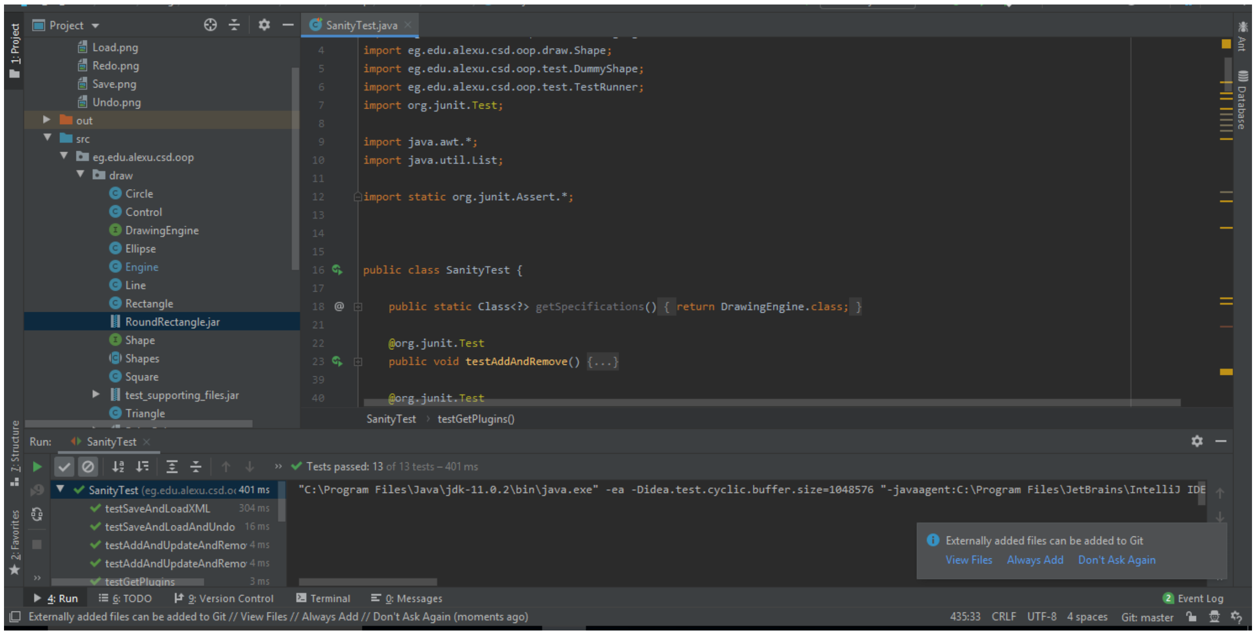


```java
import eg.edu.alexu.csd.oop.draw.Shape;
import eg.edu.alexu.csd.oop.test.DummyShape;
import eg.edu.alexu.csd.oop.test.TestRunner;


public class SmokeTest {

    public static Class<?> getSpecifications() { return DrawingEngine.class; }

    @org.junit.Test
    public void testAddShape() {
        DrawingEngine instance = (DrawingEngine)TestRunner.getImplementationInstanceForInterface(DrawingEngine.class);
        instance.addShape(new DummyShape());
    }

    @org.junit.Test
    public void testRemoveShape() {
        DrawingEngine instance = (DrawingEngine)TestRunner.getImplementationInstanceForInterface(DrawingEngine.class);
        try {
```

```
Tests passed: 7 of 7 tests – 161 ms
SmokeTest (eg.edu.alexu.csd.o 161 ms
    testGetShapesAfterDelete  146 ms
    testUpdateShape             2 ms
    testRemoveShape             2 ms
    testShapeProperties         6 ms
    testAddShape                2 ms
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE
Tests passed: 7 (moments ago)
```

Sanity test:

***Note concerning the "Sanity test" in order to pass these tests the RoundRectangle class was loaded in compile time, the path needs to be changed from a PC to another.

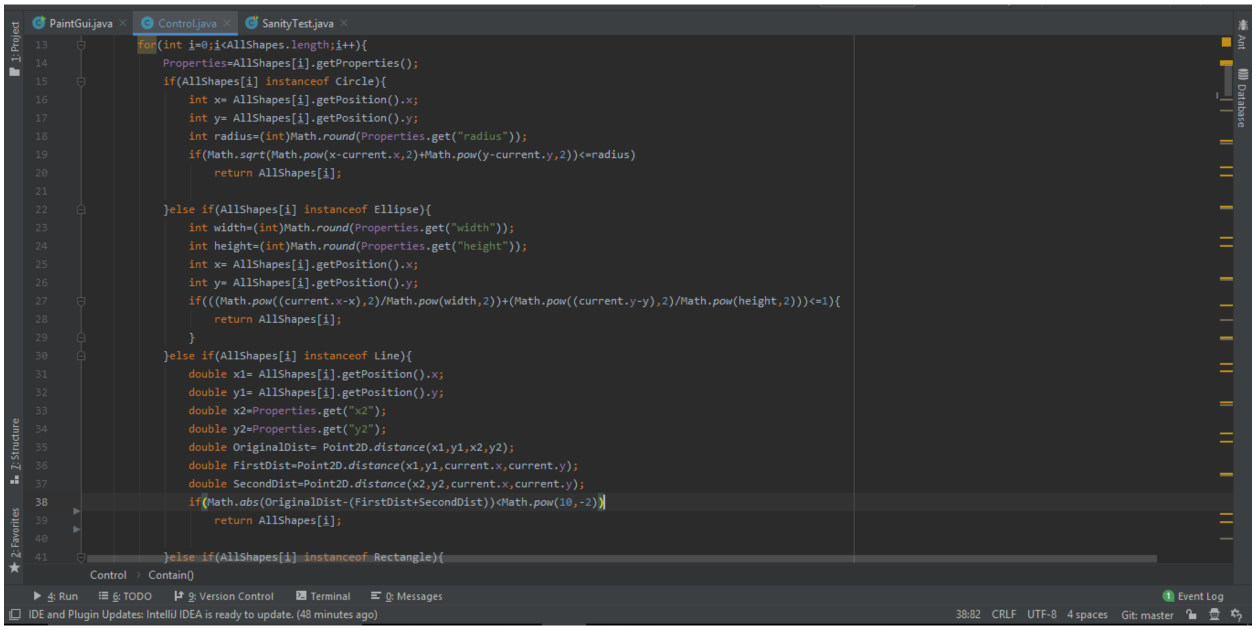


```java
import eg.edu.alexu.csd.oop.draw.Shape;
import eg.edu.alexu.csd.oop.test.DummyShape;
import eg.edu.alexu.csd.oop.test.TestRunner;
import org.junit.Test;

import java.awt.*;
import java.util.List;

import static org.junit.Assert.*;



public class SanityTest {

    public static Class<?> getSpecifications() { return DrawingEngine.class; }

    @org.junit.Test
    public void testAddAndRemove() {...}

    @org.junit.Test
```

```
Tests passed: 13 of 13 tests – 401 ms
SanityTest (eg.edu.alexu.csd.o( 401 ms
  testSaveAndLoadXML            304 ms
  testSaveAndLoadAndUndo         16 ms
  testAddAndUpdateAndRemo         4 ms
  testAddAndUpdateAndRemo         4 ms
  testGetPlugins                  3 ms

"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ IDE

Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (moments ago)
```

Part 2 drawing shapes:

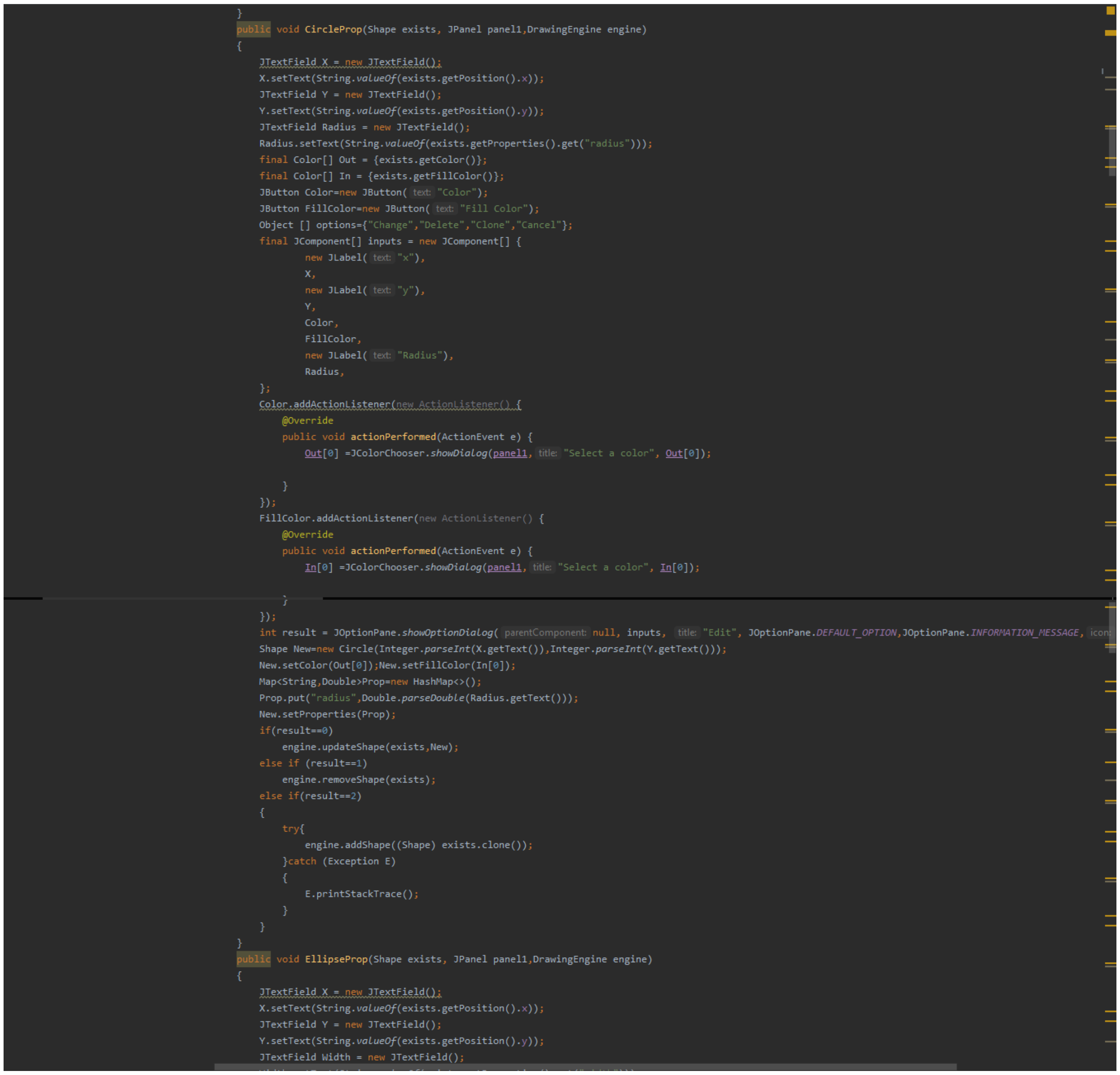
All shapes are drawn with the mouse using the mouse drag and release with black as the circumference color and white as the fill color. In order to avoid the problems drawing rectangle and square as the start point must be the top left corner we let the user draw these shapes with other shapes from their center. Mathematical equations with loops in the existing shapes to test if the clicked coordinate is included in the shape or not in the control class.



```java
        for(int i=0;i<AllShapes.length;i++){
            Properties=AllShapes[i].getProperties();
            if(AllShapes[i] instanceof Circle){
                int x= AllShapes[i].getPosition().x;
                int y= AllShapes[i].getPosition().y;
                int radius=(int)Math.round(Properties.get("radius"));
                if(Math.sqrt(Math.pow(x-current.x,2)+Math.pow(y-current.y,2))<=radius)
                    return AllShapes[i];

            }else if(AllShapes[i] instanceof Ellipse){
                int width=(int)Math.round(Properties.get("width"));
                int height=(int)Math.round(Properties.get("height"));
                int x= AllShapes[i].getPosition().x;
                int y= AllShapes[i].getPosition().y;
                if(((Math.pow((current.x-x),2)/Math.pow(width,2))+(Math.pow((current.y-y),2)/Math.pow(height,2)))<=1){
                    return AllShapes[i];
                }
            }else if(AllShapes[i] instanceof Line){
                double x1= AllShapes[i].getPosition().x;
                double y1= AllShapes[i].getPosition().y;
                double x2=Properties.get("x2");
                double y2=Properties.get("y2");
                double OriginalDist= Point2D.distance(x1,y1,x2,y2);
                double FirstDist=Point2D.distance(x1,y1,current.x,current.y);
                double SecondDist=Point2D.distance(x2,y2,current.x,current.y);
                if(Math.abs(OriginalDist-(FirstDist+SecondDist))<Math.pow(10,-2))
                    return AllShapes[i];

            }else if(AllShapes[i] instanceof Rectangle){
```

IDE and Plugin Updates: IntelliJ IDEA is ready to update. (48 minutes ago)

If an object is returned when clicked a pop-up box appears to adjust properties manually , each shape has its own one. A screenshot of a circle properties is included.

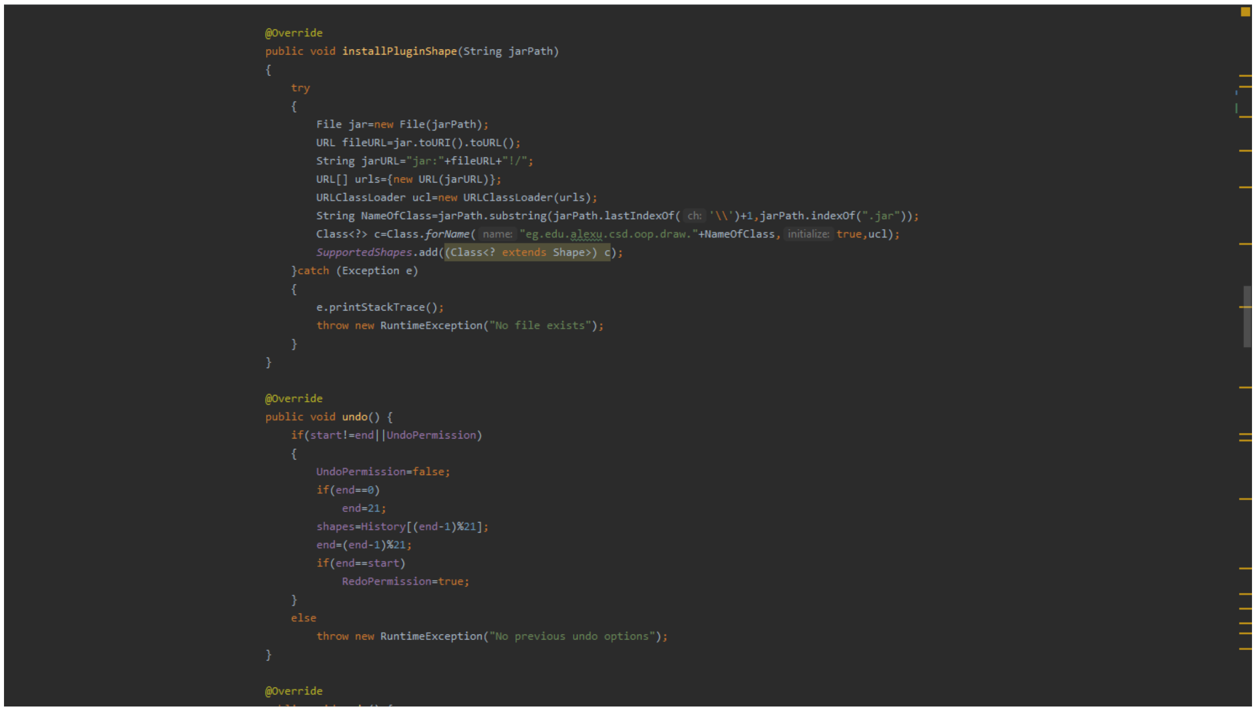


```java
}
public void CircleProp(Shape exists, JPanel panel1,DrawingEngine engine)
{
    JTextField X = new JTextField();
    X.setText(String.valueOf(exists.getPosition().x));
    JTextField Y = new JTextField();
    Y.setText(String.valueOf(exists.getPosition().y));
    JTextField Radius = new JTextField();
    Radius.setText(String.valueOf(exists.getProperties().get("radius")));
    final Color[] Out = {exists.getColor()};
    final Color[] In = {exists.getFillColor()};
    JButton Color=new JButton( text: "Color");
    JButton FillColor=new JButton( text: "Fill Color");
    Object [] options={"Change","Delete","Clone","Cancel"};
    final JComponent[] inputs = new JComponent[] {
            new JLabel( text: "x"),
            X,
            new JLabel( text: "y"),
            Y,
            Color,
            FillColor,
            new JLabel( text: "Radius"),
            Radius,
    };
    Color.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            Out[0] =JColorChooser.showDialog(panel1, title: "Select a color", Out[0]);

        }
    });
    FillColor.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            In[0] =JColorChooser.showDialog(panel1, title: "Select a color", In[0]);

        }
    });
    int result = JOptionPane.showOptionDialog( parentComponent: null, inputs,  title: "Edit", JOptionPane.DEFAULT_OPTION,JOptionPane.INFORMATION_MESSAGE, icon:
    Shape New=new Circle(Integer.parseInt(X.getText()),Integer.parseInt(Y.getText()));
    New.setColor(Out[0]);New.setFillColor(In[0]);
    Map<String,Double>Prop=new HashMap<>();
    Prop.put("radius",Double.parseDouble(Radius.getText()));
    New.setProperties(Prop);
    if(result==0)
        engine.updateShape(exists,New);
    else if (result==1)
        engine.removeShape(exists);
    else if(result==2)
    {
        try{
            engine.addShape((Shape) exists.clone());
        }catch (Exception E)
        {
            E.printStackTrace();
        }
    }
}
public void EllipseProp(Shape exists, JPanel panel1,DrawingEngine engine)
{
    JTextField X = new JTextField();
    X.setText(String.valueOf(exists.getPosition().x));
    JTextField Y = new JTextField();
    Y.setText(String.valueOf(exists.getPosition().y));
    JTextField Width = new JTextField();
```

Adding shape adds a shape in the list of shapes, updating the shape means remove the old shape and adds the new one, remove the shape removes that shape in the list

Part 3 Dynamic loading:

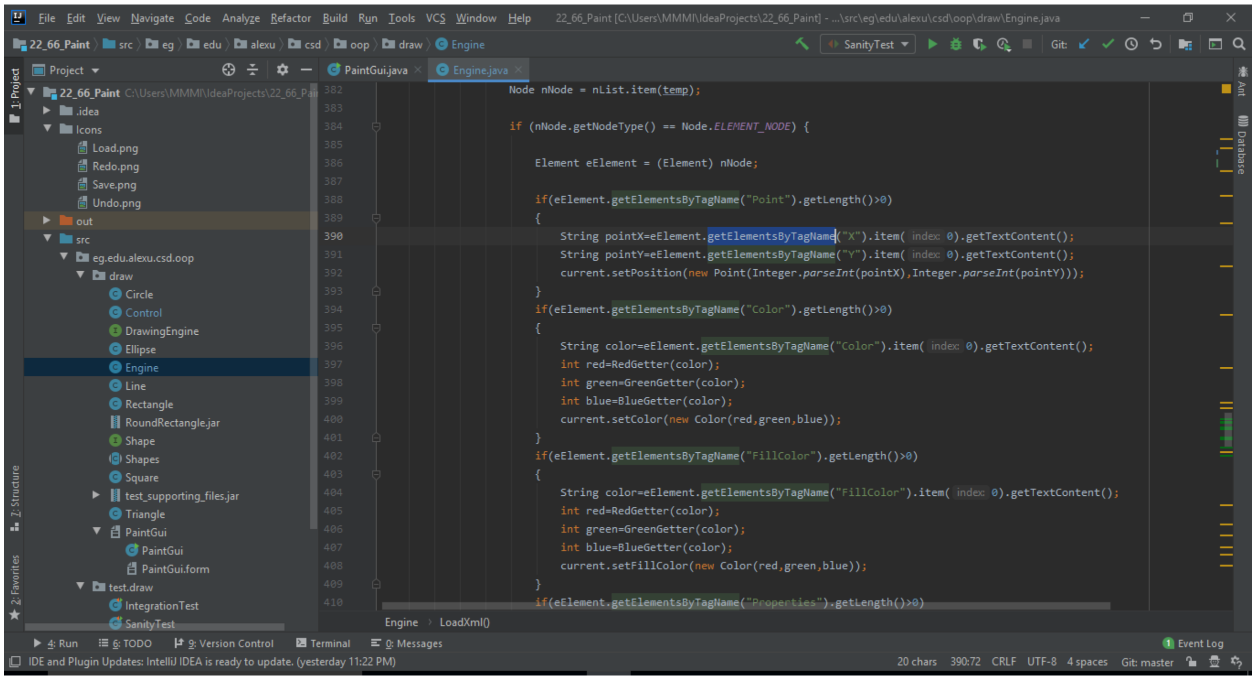
The dynamic loading is adding a library (jar file) during run time. We used ClassLoader to load the jar file as an external library



```java
@Override
public void installPluginShape(String jarPath)
{
    try
    {
        File jar=new File(jarPath);
        URL fileURL=jar.toURI().toURL();
        String jarURL="jar:"+fileURL+"!/";
        URL[] urls={new URL(jarURL)};
        URLClassLoader ucl=new URLClassLoader(urls);
        String NameOfClass=jarPath.substring(jarPath.lastIndexOf( ch: '\\')+1,jarPath.indexOf(".jar"));
        Class<?> c=Class.forName( name: "eg.edu.alexu.csd.oop.draw."+NameOfClass, initialize: true,ucl);
        SupportedShapes.add((Class<? extends Shape>) c);
    }catch (Exception e)
    {
        e.printStackTrace();
        throw new RuntimeException("No file exists");
    }
}

@Override
public void undo() {
    if(start!=end||UndoPermission)
    {
        UndoPermission=false;
        if(end==0)
            end=21;
        shapes=History[(end-1)%21];
        end=(end-1)%21;
        if(end==start)
            RedoPermission=true;
    }
    else
        throw new RuntimeException("No previous undo options");
}

@Override
```

Part 4 save and load:

We had to let the user to save in .xml or .json without using external library. We used DOM parser to save .xml and to parse it using getElementsByTagName



```java
                Node nNode = nList.item(temp);

                if (nNode.getNodeType() == Node.ELEMENT_NODE) {

                    Element eElement = (Element) nNode;

                    if(eElement.getElementsByTagName("Point").getLength()>0)
                    {
                        String pointX=eElement.getElementsByTagName("X").item(0).getTextContent();
                        String pointY=eElement.getElementsByTagName("Y").item(0).getTextContent();
                        current.setPosition(new Point(Integer.parseInt(pointX),Integer.parseInt(pointY)));
                    }
                    if(eElement.getElementsByTagName("Color").getLength()>0)
                    {
                        String color=eElement.getElementsByTagName("Color").item(0).getTextContent();
                        int red=RedGetter(color);
                        int green=GreenGetter(color);
                        int blue=BlueGetter(color);
                        current.setColor(new Color(red,green,blue));
                    }
                    if(eElement.getElementsByTagName("FillColor").getLength()>0)
                    {
                        String color=eElement.getElementsByTagName("FillColor").item(0).getTextContent();
                        int red=RedGetter(color);
                        int green=GreenGetter(color);
                        int blue=BlueGetter(color);
                        current.setFillColor(new Color(red,green,blue));
                    }
                    if(eElement.getElementsByTagName("Properties").getLength()>0)
```

Unfortunately there doesn't exist any built-in library to save JSON or parse it so we used https://json.org/example.html to learn how a JSON file should be written and implemented our own method "saveJSON".

Assumptions:

- Dynamically loaded classes drawn shapes its properties can't be adjusted
- The screenshots in the user manually shows how the program should work with dynamic loading but in order to pass the test cases we had to load it in the constructor as mentioned before.