

**Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки**

Лабораторна робота №2
з дисципліни
«Об'єктно-орієнтоване програмування»

Виконав:

студент групи ІМ-31
Литвиненко Сергій Андрійович
номер у списку групи: 11

Перевірив:

Порєв В. М.

Київ 2024

Варіант завдання

Ж - 11;

Масив - статичний;

Гумовий слід - пунктирна лінія чорного кольору;

Увід прямокутника - від центру до одного з кутів;

Відображення прямокутника - чорний контур з кольоровим заповненням;

Колір заповнення прямокутника - світло-зелений.

Увід еліпсу - по двом протилежним кутам;

Відображення еліпсу - чорний контур з білим заповненням;

Позначка поточного типу об'єкту - в заголовку вікна.

Файл Main.java.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
import javafx.fxml.FXMLLoader;

public class Main extends Application {
    final private String pathToView = "./resources/Main.fxml";
    final private String title = "Lab 2";

    static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage stage) throws Exception {
        final BorderPane root =
FXMLLoader.load(getClass().getResource(pathToView));
        final Scene scene = new Scene(root);
        final var pane = (AnchorPane)root.getCenter();
        scene.setOnKeyPressed((event) -> {
            if (event.isControlDown() && (event.getCode() == KeyCode.Z)) {
                final var childers = pane.getChildren();
                if (childers.size() > 0) pane.getChildren().removeLast();
            }
        });
    }
}
```

```
    stage.setWidth(960);  
    stage.setHeight(720);  
    stage.setScene(scene);  
    stage.setTitle(title);  
    stage.show();  
}  
}
```

Файл resources/Main.fxml.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.RadioMenuItem?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>

<BorderPane fx:id="borderPane" maxHeight="-Infinity" maxWidth="-Infinity"
minHeight="-Infinity" minWidth="-Infinity"
xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="controllers.MenuController">

    <top>

        <MenuBar id="menuBar" BorderPane.alignment="CENTER">

            <menus>

                <Menu mnemonicParsing="false" text="File">

                    <items>

                        <MenuItem mnemonicParsing="false" onAction="#exit"
text="Close" />

                    </items>

                </Menu>

                <Menu fx:id="objectsMenu" mnemonicParsing="false"
text="Objects">

                    <items>

                        <Menu mnemonicParsing="false" text="Rectangle">

                            <items>

                                <RadioMenuItem mnemonicParsing="false"
onAction="#rectangleCenter" text="From center" />

                            </items>

                        </Menu>

                    </items>

                </Menu>

            </menus>

        </MenuBar>

    </top>

</BorderPane>
```

```

        <RadioMenuItem mnemonicParsing="false"
onAction="#rectangleAngle" text="From corner" />
    </items>
</Menu>

<Menu mnemonicParsing="false" text="Ellipse">
    <items>
        <RadioMenuItem mnemonicParsing="false"
onAction="#ellipseCenter" text="From center" />
        <RadioMenuItem mnemonicParsing="false"
onAction="#ellipseAngle" text="From corner" />
    </items>
</Menu>

    <RadioMenuItem mnemonicParsing="false" onAction="#line"
text="Line" />
    <RadioMenuItem mnemonicParsing="false" onAction="#point"
text="Point" />
    <RadioMenuItem mnemonicParsing="false" onAction="#brush"
text="Brush" />
</items>
</Menu>

<Menu mnemonicParsing="false" text="Reference">
    <items>
        <MenuItem mnemonicParsing="false" text="About" />
    </items>
</Menu>

<Menu mnemonicParsing="false" text="Settings">
    <items>
        <Menu mnemonicParsing="false" fx:id="colors"
onAction="#colors" text="Colors">
            <items>
                </items>
            </items>
        </Menu>
    </items>
</Menu>

```

```
        </Menu>
        <RadioMenuItem mnemonicParsing="false" onAction="#fill"
text="Fill" />
    </items>
</Menu>
</menus>
</MenuBar>
</top>
<center>
    <AnchorPane fx:id="anchorPane" BorderPane.alignment="CENTER">
        <children>

            </children>
        </AnchorPane>
    </center>
</BorderPane>
```

Файл settings/Color.java

```
package settings;

import java.util.Map;
import javafx.scene.canvas.GraphicsContext;
import java.util.Collection;

public class Color {

    static private javafx.scene.paint.Color currentColor =
    javafx.scene.paint.Color.BLACK;

    static private Map<String, javafx.scene.paint.Color> colors =
    Map.of(
        "black", javafx.scene.paint.Color.BLACK,
        "red", javafx.scene.paint.Color.RED,
        "blue", javafx.scene.paint.Color.BLUE,
        "green", javafx.scene.paint.Color.GREEN,
        "yellow", javafx.scene.paint.Color.YELLOW,
        "purple", javafx.scene.paint.Color.PURPLE,
        "pink", javafx.scene.paint.Color.PINK,
        "gold", javafx.scene.paint.Color.GOLD,
        "brown", javafx.scene.paint.Color.BROWN,
        "light blue", javafx.scene.paint.Color.LIGHTBLUE
    );

    static public void setColor(final String color) {
        if (!colors.containsKey(color)) return;
        currentColor = colors.get(color);
    }
}
```



```
static public void resetColor(final GraphicsContext context) {  
    currentColor = javafx.scene.paint.Color.BLACK;  
    context.setStroke(currentColor);  
    context.setFill(currentColor);  
}
```

```
static public void applyCurentColor(final GraphicsContext context)  
{  
    context.setStroke(currentColor);  
    context.setFill(currentColor);  
}
```

```
static public Collection<? extends String> getStringColors() {  
    return colors.keySet();  
}
```

```
static public Collection<? extends javafx.scene.paint.Color>  
getColors() {  
    return colors.values();  
}  
}
```

Файл settings/Fill.java

```
package settings;
```

```
public class Fill {
```

```
    private static boolean fill = false;
```

```
    public static boolean getFill() {
```

```
        return fill;
```

```
    }
```

```
    public static void setFill(final boolean flag) {
```

```
        fill = flag;
```

```
    }
```

```
}
```

Файл shapes/Shape.java

```
package shapes;

import javafx.scene.canvas.GraphicsContext;

public abstract class Shape {
    protected double x1, y1, x2, y2;

    Shape(double x1, double y1, double x2, double y2) {
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
    }

    Shape() {
        this(0, 0, 0, 0);
    }

    public void setCoords(double[] coords) {
        if (coords.length != 4) return;
        x1 = coords[0];
        y1 = coords[1];
        x2 = coords[2];
        y2 = coords[3];
    }

    public abstract void draw(final GraphicsContext context, boolean
fill);
```

}

Файл shapes/Rectangle.java

```
package shapes;

import javafx.scene.canvas.GraphicsContext;

public class Rectangle extends Shape {
    @Override
    public void draw(GraphicsContext context, boolean fill) {
        final var dx = Math.abs(x2 - x1);
        final var dy = Math.abs(y2 - y1);
        final var x = Math.min(x1, x2);
        final var y = Math.min(y1, y2);
        if (fill) context.fillRect(x, y, dx, dy);
        else context.strokeRect(x, y, dx, dy);
    }
}
```

Файл shapes/Point.java

```
package shapes;

import javafx.scene.canvas.GraphicsContext;

public class Point extends Shape {

    @Override
    public void draw(GraphicsContext context, boolean fill) {
        final var width = context.getLineWidth();
        context.fillOval(x2 - width, y2 - width, width * 2, width * 2);
    }

}
```

Файл shapes/Line.java

```
package shapes;
```

```
import javafx.scene.canvas.GraphicsContext;
```

```
public class Line extends Shape {
```

```
    @Override
```

```
    public void draw(GraphicsContext context, boolean fill) {
```

```
        context.strokeLine(x1, y1, x2, y2);
```

```
    }
```

```
}
```

Файл shapes/Elipse.java

```
package shapes;

import javafx.scene.canvas.GraphicsContext;

public class Elipse extends Shape {
    @Override
    public void draw(GraphicsContext context, boolean fill) {
        final double dx = Math.abs(x2 - x1);
        final double dy = Math.abs(y2 - y1);
        final double x = (x1 + x2 - dx) / 2;
        final double y = (y1 + y2 - dy) / 2;
        if (fill) context.fillOval(x, y, dx, dy);
        else context.strokeOval(x, y, dx, dy);
    }
}
```


Файл editors/BrushEditor.java

```
package editors;

import javafx.scene.layout.Pane;
import settings.Color;
import javafx.scene.canvas.GraphicsContext;

public class BrushEditor extends Editor {

    private GraphicsContext context;

    public BrushEditor(Pane pane) {
        super(pane, null);
    }

    @Override
    protected double[] getCoords(double startX, double startY, double
x, double y) {
        return new double[]{ startX, startY, x, y };
    }

    public void onLeftButtonDown(double x, double y) {
        context = super.createContext();
        Color.applyCurentColor(context);
        context.beginPath();
        context.moveTo(x, y);
    }

    public void onMouseMove(double x, double y) {
```

```
    context.lineTo(x, y);  
    context.stroke();  
}
```

```
public void onLeftButtonUp(double x, double y) {  
  
}  
}
```

Файл editors/Editor.java

```
package editors;

import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.layout.Pane;
import settings.Color;
import settings.Fill;
import shapes.Shape;

public abstract class Editor {
    private static double lineWidth = 2.5;
    private static double lineDashes = 10;

    private final Pane pane;
    protected double startX = 0;
    protected double startY = 0;
    protected boolean drawing = false;
    protected final Shape shape;

    public Editor(final Pane pane, final Shape shape) {
        this.pane = pane;
        this.shape = shape;
    }

    protected GraphicsContext createContext() {
        final var width = pane.getWidth();
        final var height = pane.getHeight();
```

```
final var canvas = new Canvas(width, height);
pane.getChildren().add(canvas);
final var context = canvas.getGraphicsContext2D();
context.setLineWidth(lineWidth);
Color.applyCurentColor(context);
return context;
}
```

```
protected void deletelastCanvas() {
    final var childrens = pane.getChildren();
    if (childrens.size() > 0) childrens.removeLast();
}
```

```
public void onLeftButtonDown(double x, double y) {
    startX = x;
    startY = y;
}
```

```
public void onMouseMove(double x, double y) {
    if (drawing) deletelastCanvas();
    else drawing = true;
    final var coords = getCoords(startX, startY, x, y);
    shape.setCoords(coords);
    final var context = createContext();
    context.setLineDashes(lineDashes);
    shape.draw(context, false);
}
```

```
public void onLeftButtonUp(double x, double y) {  
    deleteLastCanvas();  
    final var coords = getCoords(startX, startY, x, y);  
    shape.setCoords(coords);  
    final var context = createContext();  
    context.setLineDashes(0);  
    shape.draw(context, Fill.getFill());  
    drawing = false;  
}
```

```
protected abstract double[] getCoords(double startX, double  
startY, double x, double y);  
}
```

Файл editors/ElipseCenterEditor.java

```
package editors;

import javafx.scene.layout.Pane;
import shapes.Elipse;

public class ElipseCenterEditor extends Editor {

    public ElipseCenterEditor(Pane pane) {
        super(pane, new Elipse());
    }

    @Override
    protected double[] getCoords(double startX, double startY, double
x, double y) {
        return new double[]{ 2 * startX - x, 2 * startY - y, x, y };
    }

}
```

Файл editors/ElipseCornerEditor.java

```
package editors;
```

```
import javafx.scene.layout.Pane;
```

```
import shapes.Elipse;
```

```
public class ElipseCornerEditor extends Editor {
```

```
    public ElipseCornerEditor(Pane pane) {
```

```
        super(pane, new Elipse());
```

```
    }
```

```
    @Override
```

```
    protected double[] getCoords(double startX, double startY, double  
x, double y) {
```

```
        return new double[]{ startX, startY, x, y };
```

```
    }
```

```
}
```

Файл editors/LineEditor.java

```
package editors;

import javafx.scene.layout.Pane;
import shapes.Line;

public class LineEditor extends Editor {

    public LineEditor(final Pane pane) {
        super(pane, new Line());
    }

    @Override
    protected double[] getCoords(double startX, double startY, double
x, double y) {
        return new double[]{ startX, startY, x, y };
    }
}
```


Файл editors/PointEditor.java

```
package editors;

import javafx.scene.layout.Pane;
import shapes.Point;

public class PointEditor extends Editor {

    public PointEditor(Pane pane) {
        super(pane, new Point());
    }

    @Override
    protected double[] getCoords(double startX, double startY, double
x, double y) {
        return new double[]{ startX, startY, x, y };
    }

    @Override
    public void onLeftButtonDown(double x, double y) {
        super.onLeftButtonDown(x, y);
        super.onMouseMove(x, y);
    }
}
```

Файл editors/RctangleCenterEditor.java

```
package editors;

import javafx.scene.layout.Pane;
import shapes.Rectangle;

public class RectangleCenterEditor extends Editor {

    public RectangleCenterEditor(final Pane pane) {
        super(pane, new Rectangle());
    }

    @Override
    protected double[] getCoords(double startX, double startY, double
x, double y) {
        return new double[]{ 2 * startX - x, 2 * startY - y, x, y };
    }
}
```

Файл editors/RctangleCornerEditor.java

```
package editors;

import javafx.scene.layout.Pane;
import shapes.Rectangle;

public class RectangleCornerEditor extends Editor {

    public RectangleCornerEditor(final Pane pane) {
        super(pane, new Rectangle());
    }

    @Override
    protected double[] getCoords(double startX, double startY, double
x, double y) {
        return new double[]{ startX, startY, x, y };
    }
}
```

Файл editors/MenuController.java

```
package controllers;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
import javafx.scene.control.Menu;
import javafx.scene.control.RadioMenuItem;
import javafx.scene.input.MouseButton;
import javafx.scene.input.MouseEvent;
import javafx.scene.control.MenuItem;
import javafx.application.Platform;
import java.util.ArrayList;
import java.util.List;
import javafx.util.Pair;
import settings.Color;
import settings.Fill;
import editors.*;

public class MenuController {

    @FXML
    private BorderPane borderPane;

    @FXML
    private Menu objectsMenu;
```

```
@FXML
```

```
private AnchorPane anchorPane;
```

```
@FXML
```

```
private Menu colors;
```

```
private final List<Pair<Menu, List<RadioMenuItem>>> menuItems =  
new ArrayList<>();
```

```
private boolean isPrimary(final MouseEvent event) {  
    return event.getButton().equals(MouseButton.PRIMARY);  
}
```

```
private void processEvent(final Editor editor, final ActionEvent  
parentEvent) {  
    final var item = (RadioMenuItem)(parentEvent.getSource());  
    anchorPane.setOnMousePressed((event) -> {  
        if (isPrimary(event) && item.isSelected()) {  
            editor.onLeftButtonDown(event.getX(), event.getY());  
        }  
    });  
    anchorPane.setOnMouseDragged((event) -> {  
        if (isPrimary(event) && item.isSelected()) {  
            editor.onMouseMove(event.getX(), event.getY());  
        }  
    });  
    anchorPane.setOnMouseReleased((event) -> {  
        if (isPrimary(event) && item.isSelected()) {
```

```
        editor.onLeftButtonUp(event.getX(), event.getY());
    }
});
}
```

@FXML

```
private void rectangleCenter(final ActionEvent event) {
    processEvent(new RectangleCenterEditor(anchorPane), event);
}
```

@FXML

```
private void rectangleAngle(final ActionEvent event) {
    processEvent(new RectangleCornerEditor(anchorPane), event);
}
```

@FXML

```
private void ellipseCenter(final ActionEvent event) {
    processEvent(new EllipseCenterEditor(anchorPane), event);
}
```

@FXML

```
private void ellipseAngle(final ActionEvent event) {
    processEvent(new EllipseCornerEditor(anchorPane), event);
}
```

@FXML

```
private void line(final ActionEvent event) {
    processEvent(new LineEditor(anchorPane), event);
}
```

```
}
```

```
@FXML
```

```
private void brush(final ActionEvent event) {  
    processEvent(new BrushEditor(anchorPane), event);  
}
```

```
@FXML
```

```
private void point(final ActionEvent event) {  
    processEvent(new PointEditor(anchorPane), event);  
}
```

```
@FXML
```

```
private void exit() {  
    Platform.exit();  
}
```

```
@FXML
```

```
private void colors(final ActionEvent event) {  
    final var item = (MenuItem)event.getTarget();  
    final var text = item.getText();  
    Color.setColor(text);  
}
```

```
@FXML
```

```
private void fill() {  
    final var fill = Fill.getFill();  
    Fill.setFill(!fill);  
}
```

```
}
```

```
private void makeItemsUnique(final String text) {  
    for (final var pair: menuItems) {  
        final var items = pair.getValue();  
        for (final var item: items) {  
            final var selected = item.isSelected();  
            final var equals = item.getText().equals(text);  
            if (selected && !equals) item.setSelected(false);  
        }  
    }  
}
```

```
private String getFullName(final MenuItem selected, final Menu  
root) {  
    final StringBuilder result = new StringBuilder(root.getText() +  
" -> ");  
    boolean find = false;  
    for (final MenuItem item: root.getItems()) {  
        if (item instanceof final Menu menu) {  
            final var subpath = getFullName(selected, menu);  
            if (subpath.length() == 0) continue;  
            find = true;  
            result.append(subpath);  
            break;  
        }  
        if (!item.equals(selected)) continue;  
        find = true;  
        result.append(item.getText());  
    }  
}
```



```

        break;
    }
    return find ? result.toString() : "";
}

```

```

private void buildMenuItems(final Menu root) {
    final List<RadioMenuItem> items = new ArrayList<>();
    for (final MenuItem item: root.getItems()) {
        if (item instanceof final Menu menu) {
            buildMenuItems(menu);
            continue;
        }
        items.addLast((RadioMenuItem)item);
    }
    menuItems.addLast(new Pair<Menu, List<RadioMenuItem>>(root,
items));
}

```

```

private void addColors() {
    final var items = new ArrayList<MenuItem>();
    for (final var color: Color.getStringColors()) {
        items.addLast(new MenuItem(color));
    }
    colors.getItems().addAll(items);
}

```

```

@FXML
private void initialize() {
    buildMenuItems(objectsMenu);
}

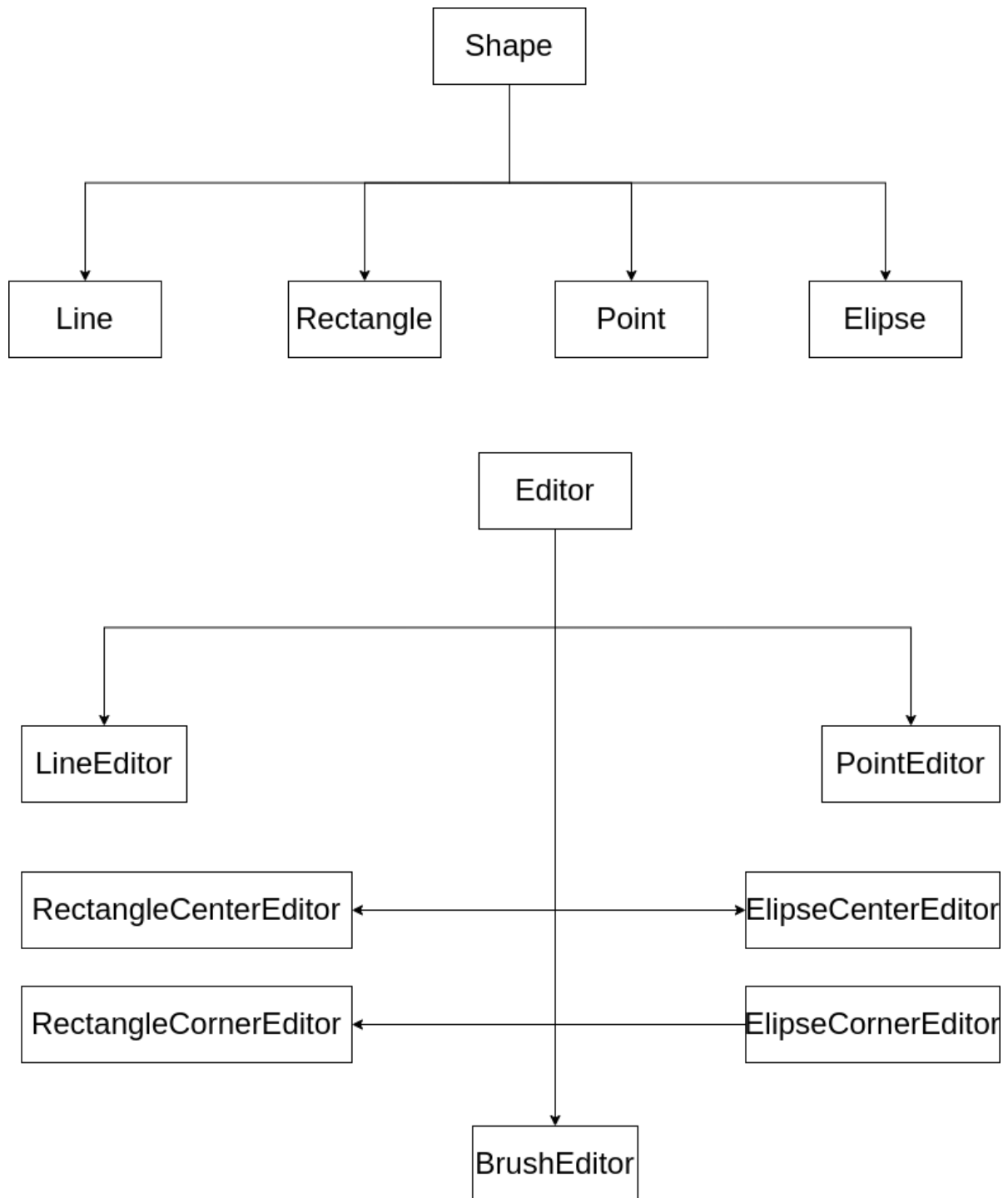
```

```

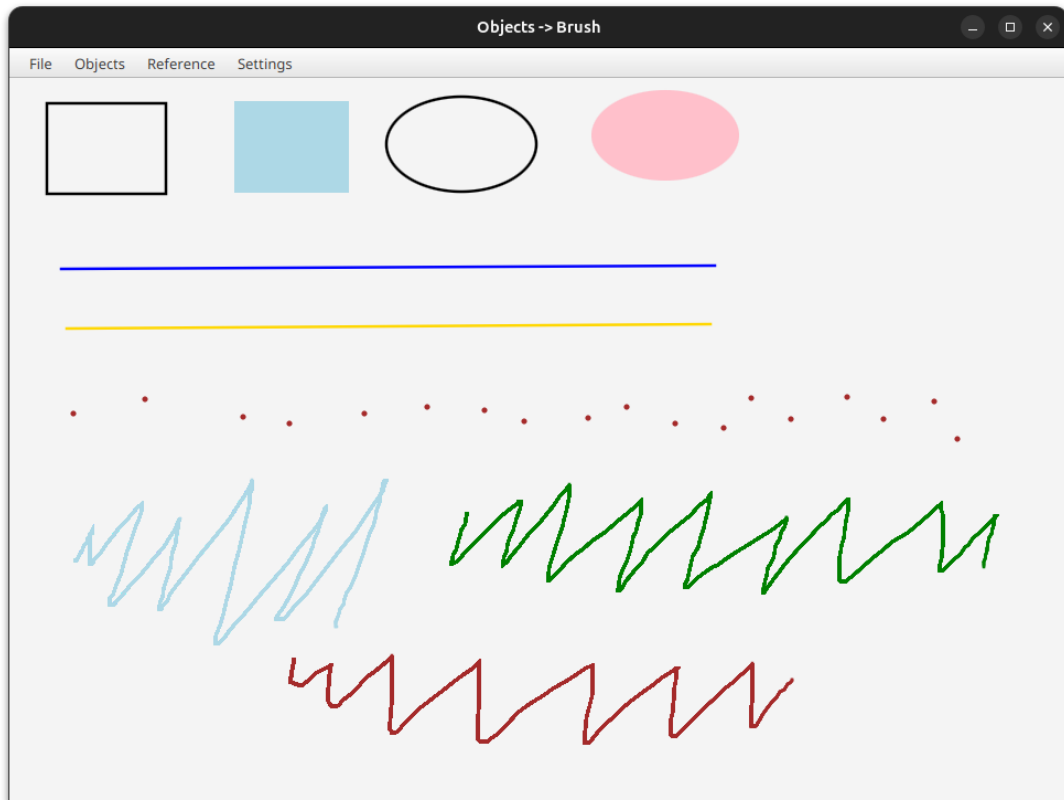
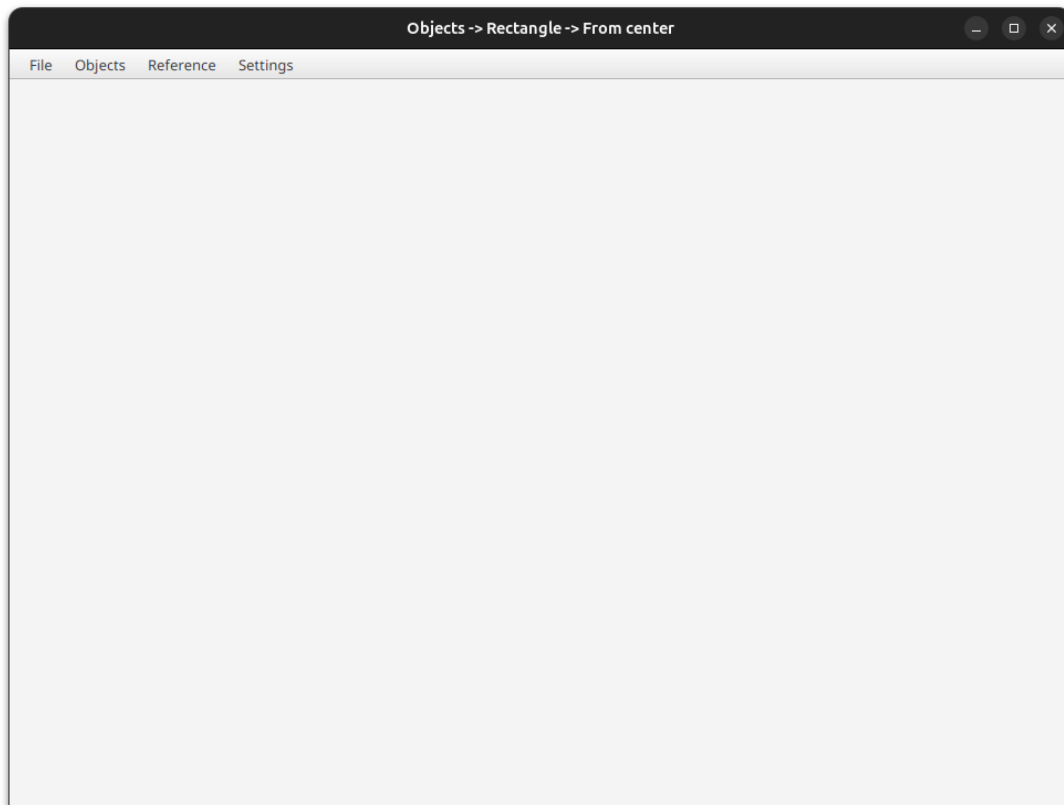
addColors();
for (final var pair: menuItems) {
    final var menu = pair.getKey();
    menu.setOnAction((action) -> {
        if (action.getTarget() instanceof Menu) return;
        final var selected = (RadioMenuItem)action.getTarget();
        final var text = selected.getText();
        makeItemsUnique(text);
        final var window = (Stage)borderPane.getScene().getWindow();
        final var fullPath = getFullName(selected, objectsMenu);
        window.setTitle(fullPath);
    });
}
}
}

```

Діаграма наслідування



Скріншоти виконання



Висновки

Під час виконання лабораторної роботи я здобув навички використання інкапсуляції, абстрактних типів, успадкування та поліморфізму, створив простий графічний редактор та вдосконалив свої вміння програмування на Java. Протягом виконання я отримав теоретичні знання з архітектури розробки графічних додатків, та дізнався про кращі практики написання коду в об'єктно орієнтованому стилі використовуючи поліморфізм.