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 №4

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 - 12;

Динамічний об’єкт класу;

Файл Main.java.

```
import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.layout.AnchorPane;

import javafx.scene.layout.BorderPane;

import javafx.stage.Stage;

import javafx.fxml.FXMLLoader;

import javafx.scene.canvas.Canvas;


public class Main extends Application {

  final private String pathToView = "./resources/Main.fxml";

  final private String title = "Lab 3";


  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)((BorderPane)root.getCenter()).getCenter();

    final var canvas = (Canvas)pane.getChildren().getFirst();

    canvas.widthProperty().bind(pane.widthProperty());

    canvas.heightProperty().bind(pane.heightProperty());

    stage.setScene(scene);

    pane.setPrefHeight(700);
    pane.setPrefWidth(900);
    stage.setTitle(title);
    stage.show();
  }
}
```

Файл resources/Main.fxml

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


<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Menu?>

<?import javafx.scene.control.MenuBar?>

<?import javafx.scene.control.MenuItem?>

<?import javafx.scene.control.RadioMenuItem?>

<?import javafx.scene.control.ToolBar?>

<?import javafx.scene.control.Tooltip?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.canvas.Canvas?>


<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="#saveAs"
text="Save as..." />

            <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 id="rectangleCenter"
mnemonicParsing="false" text="From center" />

                <RadioMenuItem id="rectangleCorner"
mnemonicParsing="false" text="From corner" />

              </items>

            </Menu>

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

              <items>

                <RadioMenuItem id="ellipseCenter"
mnemonicParsing="false" text="From center" />

                <RadioMenuItem id="ellipseCorner"
mnemonicParsing="false" text="From corner" />

              </items>

            </Menu>

            <RadioMenuItem id="cube" mnemonicParsing="false"
text="Cube" />

            <RadioMenuItem id="line" mnemonicParsing="false"
text="Line" />

            <RadioMenuItem id="line-ellipse" mnemonicParsing="false"
text="Line Ellipse" />

            <RadioMenuItem id="point" mnemonicParsing="false"
text="Point" />

            <RadioMenuItem id="brush" mnemonicParsing="false"
text="Brush" />

          </items>
        </Menu>

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

          <items>

            <MenuItem mnemonicParsing="false" text="About" />

          </items>

        </Menu>

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

          <items>

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

              <items>

              </items>

            </Menu>

            <RadioMenuItem mnemonicParsing="false" onAction="#fill"
text="Fill" />

          </items>

        </Menu>

      </menus>

    </MenuBar>

  </top>

   <center>

    <BorderPane BorderPane.alignment="CENTER">

      <top>

        <ToolBar BorderPane.alignment="CENTER" fx:id="toolBar">

          <items>

            <Button id="rectangleCenter-button"
mnemonicParsing="false">

              <graphic>

                <ImageView fitHeight="32.0" fitWidth="32.0">

                  <image>
                    <Image url="@icons/rectangle-center.png" />
                  </image>
                </ImageView>
              </graphic>
              <tooltip>
                <Tooltip text="Rectangle Center" />
              </tooltip>
            </Button>
            <Button id="rectangleCorner-button"
mnemonicParsing="false">
              <graphic>
                <ImageView fitHeight="32.0" fitWidth="32.0">
                  <image>
                    <Image url="@icons/rectangle-corner.png" />
                  </image>
                </ImageView>
              </graphic>
              <tooltip>
                <Tooltip text="Rectangle Corner" />
              </tooltip>
            </Button>
            <Button id="cube-button" mnemonicParsing="false">
              <graphic>
                <ImageView fitHeight="32.0" fitWidth="32.0">
                  <image>
                    <Image url="@icons/cube.png" />
                  </image>
                </ImageView>
              </graphic>
```

```
              <tooltip>
                <Tooltip text="Cube" />
              </tooltip>
            </Button>
            <Button id="ellipseCenter-button"
mnemonicParsing="false">
              <graphic>
                <ImageView fitHeight="32.0" fitWidth="32.0">
                  <image>
                    <Image url="@icons/ellipse-center.png" />
                  </image>
                </ImageView>
              </graphic>
              <tooltip>
                <Tooltip text="Ellipse Center" />
              </tooltip>
            </Button>
            <Button id="ellipseCorner-button"
mnemonicParsing="false">
              <graphic>
                <ImageView fitHeight="32.0" fitWidth="32.0">
                  <image>
                    <Image url="@icons/ellipse-corner.png" />
                  </image>
                </ImageView>
              </graphic>
              <tooltip>
                <Tooltip text="Elipce Corner" />
              </tooltip>
            </Button>
            <Button id="line-button" mnemonicParsing="false">
              <graphic>
                <ImageView fitHeight="32.0" fitWidth="32.0">
                  <image>
                    <Image url="@icons/line.png" />
                  </image>
                </ImageView>
              </graphic>
              <tooltip>
                <Tooltip text="Line" />
              </tooltip>
            </Button>
            <Button id="line-ellipse-button"
mnemonicParsing="false">
              <graphic>
                <ImageView fitHeight="32.0" fitWidth="32.0">
                  <image>
                    <Image url="@icons/line-ellipse.png" />
                  </image>
                </ImageView>
              </graphic>
              <tooltip>
                <Tooltip text="Line ELlipse" />
              </tooltip>
            </Button>
            <Button id="point-button" mnemonicParsing="false">
              <graphic>
                <ImageView fitHeight="32.0" fitWidth="32.0">
                  <image>
                    <Image url="@icons/point.png" />
                  </image>
                </ImageView>
              </graphic>
              <tooltip>
                <Tooltip text="Point" />
              </tooltip>
            </Button>
            <Button id="brush-button" mnemonicParsing="false">
              <graphic>
                <ImageView fitHeight="32.0" fitWidth="32.0">
                  <image>
                    <Image url="@icons/brush.png" />
                  </image>
                </ImageView>
              </graphic>
              <tooltip>
                <Tooltip text="Brush" />
              </tooltip>
            </Button>
          </items>
        </ToolBar>
      </top>
      <center>
        <AnchorPane fx:id="anchorPane"
BorderPane.alignment="CENTER">
          <Canvas fx:id="canvas" />
        </AnchorPane>
```

```
          </center>
        </BorderPane>
      </center>
</BorderPane>
```

Файл settings/Color.java

```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

```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;

import javafx.scene.paint.Color;

import java.util.List;


public abstract class Shape {

  protected List<Double> coords;

  public Color color = Color.BLACK;

  public boolean fill = false;

  public double dashes = 0;

  public boolean useDashes = true;


  protected void prepareContext(final GraphicsContext context) {

    context.setStroke(color);

    context.setFill(color);

    context.setLineDashes(dashes);

  }


  public abstract void draw(final GraphicsContext context);


  public abstract void setCoords(double x1, double y1, double x2,
double y2);

  public void onStart(GraphicsContext context, double x, double y) {


  }

}
```

Файл shapes/RectangleCorner.java

```java
package shapes;

import javafx.scene.canvas.GraphicsContext;
import java.util.List;
import java.util.ArrayList;

public class RectangleCorner extends Shape implements Rectangable {
  public RectangleCorner() {
    super();
    coords = new ArrayList<>(List.of(0.0, 0.0, 0.0, 0.0));
  }

  @Override
  public void draw(GraphicsContext context) {
    prepareContext(context);
    Rectangable.super.drawRectangle(
      context,
      coords.get(0),
      coords.get(1),
      coords.get(2),
      coords.get(3),
      fill
    );
  }
```

Файл shapes/RectangleCenter.java

```
package shapes;


public class RectangleCenter extends RectangleCorner {
  @Override
  public void setCoords(double x1, double y1, double x2, double y2)
{
    super.setCoords(2 * x1 - x2, 2 * y1 - y2, x2, y2);
  }
}


  @Override
  public void setCoords(double x1, double y1, double x2, double y2)
{
    coords.set(0, Math.min(x1, x2));
    coords.set(1, Math.min(y1, y2));
    coords.set(2, Math.abs(x2 - x1));
    coords.set(3, Math.abs(y2 - y1));
  }
}
```

Файл shapes/Rectangable.java

```
package shapes;


import javafx.scene.canvas.GraphicsContext;


public interface Rectangable {
  default void drawRectangle(GraphicsContext context, double x,
double y, double dx, double dy, boolean fill) {
    final var width = context.getLineWidth();
    if (fill) context.fillRect(x, y, dx + width, dy + width);
    else context.strokeRect(x, y, dx + width, dy + width);
  }
}
```

Файл shapes/Point.java

```
package shapes;


import java.util.ArrayList;

import java.util.List;


import javafx.scene.canvas.GraphicsContext;


public class Point extends Shape {


  public Point() {

    super();

    coords = new ArrayList<>(List.of(0.0, 0.0));

  }


  @Override

  public void onStart(GraphicsContext context, double x, double y) {

    this.setCoords(0, 0, x, y);

    this.draw(context);

  }


  @Override

  public void draw(GraphicsContext context) {

    prepareContext(context);

    final var x = coords.get(0);

    final var y = coords.get(1);

    final var width = context.getLineWidth();

    context.fillOval(x - width, y - width, width * 2, width * 2);
  }


  @Override
  public void setCoords(double x1, double y1, double x2, double y2)
{
    coords.set(0, x2);
    coords.set(1, y2);
  }
}
```

Файл shapes/LineEllipse.java

```
package shapes;

import java.util.ArrayList;
import java.util.List;

import javafx.scene.canvas.GraphicsContext;

public class Point extends Shape {

  public Point() {
    super();
    coords = new ArrayList<>(List.of(0.0, 0.0));
  }

  @Override
  public void onStart(GraphicsContext context, double x, double y) {
    this.setCoords(0, 0, x, y);
    this.draw(context);
  }

  @Override
  public void draw(GraphicsContext context) {
    prepareContext(context);
    final var x = coords.get(0);
    final var y = coords.get(1);
    final var width = context.getLineWidth();
    context.fillOval(x - width, y - width, width * 2, width * 2);
  }


  @Override
  public void setCoords(double x1, double y1, double x2, double y2)
{
    coords.set(0, x2);
    coords.set(1, y2);
  }
}
```

Файл shapes/Line.java

```
package shapes;


import java.util.ArrayList;

import java.util.List;


import javafx.scene.canvas.GraphicsContext;


public class Line extends Shape implements Linable {


  public Line() {

    super();

    coords = new ArrayList<>(List.of(0.0, 0.0, 0.0, 0.0));

  }


  @Override

  public void draw(GraphicsContext context) {

    prepareContext(context);

    Linable.super.drawLine(

      context,

      coords.get(0),

      coords.get(1),

      coords.get(2),

      coords.get(3)

    );

  }


  @Override
  public void setCoords(double x1, double y1, double x2, double y2)
{

    coords.set(0, x1);

    coords.set(1, y1);

    coords.set(2, x2);

    coords.set(3, y2);

  }

}
```

Файл shapes/Linable.java

```
package shapes;


import javafx.scene.canvas.GraphicsContext;


public interface Linable {
  public default void drawLine(GraphicsContext context, double x1,
double y1, double x2, double y2) {
    context.strokeLine(x1, y1, x2, y2);
  }
}
```

Файл shapes/ElipseCorner.java

```
package shapes;


import java.util.ArrayList;

import java.util.List;


import javafx.scene.canvas.GraphicsContext;


public class EllipseCorner extends Shape implements Ellipsable {


  public EllipseCorner() {

    super();

    coords = new ArrayList<>(List.of(0.0, 0.0, 0.0, 0.0));

  }


  @Override

  public void draw(GraphicsContext context) {

    prepareContext(context);

    Ellipsable.super.drawEllipse(

      context,

      coords.get(0),

      coords.get(1),

      coords.get(2),

      coords.get(3),

      fill

    );

  }
  @Override

  public void setCoords(double x1, double y1, double x2, double y2)
{

    final double dx = Math.abs(x2 - x1);

    final double dy = Math.abs(y2 - y1);

    coords.set(0, (x1 + x2 - dx) / 2);

    coords.set(1, (y1 + y2 - dy) / 2);

    coords.set(2, dx);

    coords.set(3, dy);

  }

}
```

Файл editors/EllipseCenter.java

```
package shapes;


public class EllipseCenter extends EllipseCorner {
  @Override
  public void setCoords(double x1, double y1, double x2, double y2)
{
    super.setCoords(2 * x1 - x2, 2 * y1 - y2, x2, y2);
  }
}
```

Файл editors/Ellipsable.java

```
package shapes;


import javafx.scene.canvas.GraphicsContext;


public interface Ellipsable {
  public default void drawEllipse(GraphicsContext context, double x,
double y, double dx, double dy, boolean fill) {
    final var width = context.getLineWidth();
    if (fill) context.fillOval(x, y, dx + width, dy + width);
    else context.strokeOval(x, y, dx + width, dy + width);
  }
}
```

Файл editors/Cube.java

```
package shapes;


import java.util.ArrayList;

import java.util.List;


import javafx.scene.canvas.GraphicsContext;


public class Cube extends Shape implements Linable, Rectangable {


  private static final int deltaX = 50;

  private static final int deltaY = -40;


  public Cube() {

    super();

    coords = new ArrayList<>(List.of(0.0, 0.0, 0.0, 0.0));

  }


  @Override

  public void draw(GraphicsContext context) {

    prepareContext(context);

    fill = false;

    final var x1 = coords.get(0);

    final var y1 = coords.get(1);

    final var dx = coords.get(2);

    final var dy = coords.get(3);

    Rectangable.super.drawRectangle(context, x1, y1, dx, dy, fill);

    Rectangable.super.drawRectangle(context, x1 + deltaX, y1 +
deltaY, dx, dy, fill);
```

```
    Linable.super.drawLine(context, x1, y1, x1 + deltaX, y1 +
deltaY);

    Linable.super.drawLine(context, x1 + dx, y1, x1 + dx + deltaX,
y1 + deltaY);

    Linable.super.drawLine(context, x1, y1 + dy, x1 + deltaX, y1 +
dy + deltaY);

    Linable.super.drawLine(context, x1 + dx, y1 + dy, x1 + dx +
deltaX, y1 + dy + deltaY);

  }


  @Override

  public void setCoords(double x1, double y1, double x2, double y2)
{

    coords.set(0, Math.min(x1, x2));

    coords.set(1, Math.min(y1, y2));

    coords.set(2, Math.abs(x2 - x1));

    coords.set(3, Math.abs(y2 - y1));

  }

}
```

Файл editors/Brush.java

```
package shapes;


import java.util.ArrayList;


import javafx.scene.canvas.GraphicsContext;


public class Brush extends Shape {


  public Brush() {
    super();
    coords = new ArrayList<>();
    useDashes = false;
  }


  @Override
  public void onStart(GraphicsContext context, double x, double y) {
    this.setCoords(0, 0, x, y);
  }


  @Override
  public void draw(GraphicsContext context) {
    prepareContext(context);
    final var size = coords.size();
    if (size <= 2) return;
    var prevX = coords.get(0);
    var prevY = coords.get(1);
    for (int i = 2; i < size; i += 2) {
      final var x = coords.get(i);
      final var y = coords.get(i + 1);
      context.strokeLine(prevX, prevY, x, y);
      prevX = x;
      prevY = y;
    }
  }

  @Override
  public void setCoords(double x1, double y1, double x2, double y2)
{
    coords.add(x2);
    coords.add(y2);
  }
}
```

Файл editors/Editor.java

```
package editors;


import shapes.Shape;
import java.util.Stack;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import settings.Color;
import settings.Fill;


public class Editor {
  private static double lineDashes = 10;
  private double startX = 0;
  private double startY = 0;
  private boolean drawing = false;
  private static Stack<Shape> shapes = new Stack<>();
  private final Canvas canvas;
  private GraphicsContext context;


  @SuppressWarnings("unused")
  public Editor(final Canvas canvas) {
    this.canvas = canvas;
    context = canvas.getGraphicsContext2D();
    canvas.widthProperty().addListener((event) -> {
      clear();
      drawAll();
    });
    canvas.heightProperty().addListener((event) -> {
```

```
    clear();
    drawAll();
  });
}

public void drawAll() {
  for (final var shape: shapes) shape.draw(context);
}

public void clear() {
  context.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());
}

public void add(final Shape shape) {
  shapes.add(shape);
}

public void pop() {
  if (shapes.size() > 0) shapes.pop();
}

public void onLeftButtonDown(double x, double y) {
  startX = x;
  startY = y;
  final var shape = shapes.peek();
  shape.dashes = shape.useDashes ? lineDashes : 0;
  shape.color = Color.getCurrentColor();
  shape.fill = Fill.getFill();
```

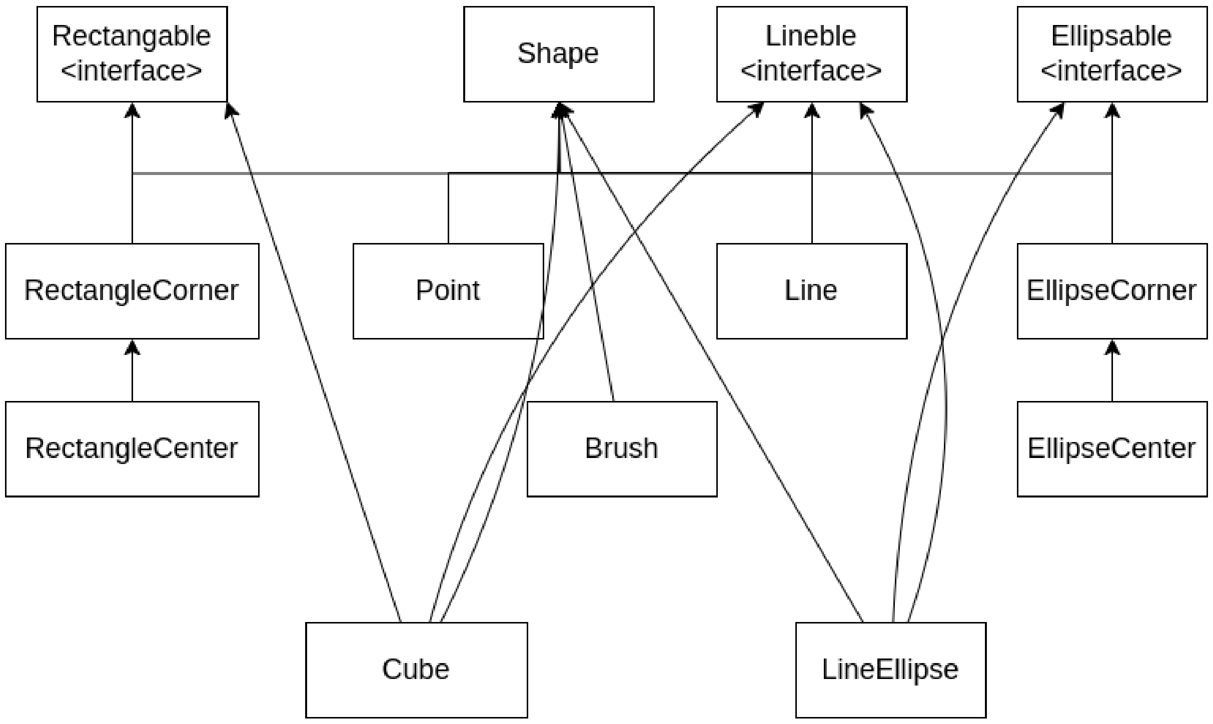
```
    shape.onStart(context, x, y);
  }

  public void onMouseMove(double x, double y) {
    if (drawing) clear();
    else drawing = true;
    shapes.peek().setCoords(startX, startY, x, y);
    drawAll();
  }

  public void onLeftButtonUp(double x, double y) {
    clear();
    final var shape = shapes.peek();
    shape.setCoords(startX, startY, x, y);
    shape.dashes = 0;
    drawAll();
    drawing = false;
  }
}
```

Файл controller/MenuController.java

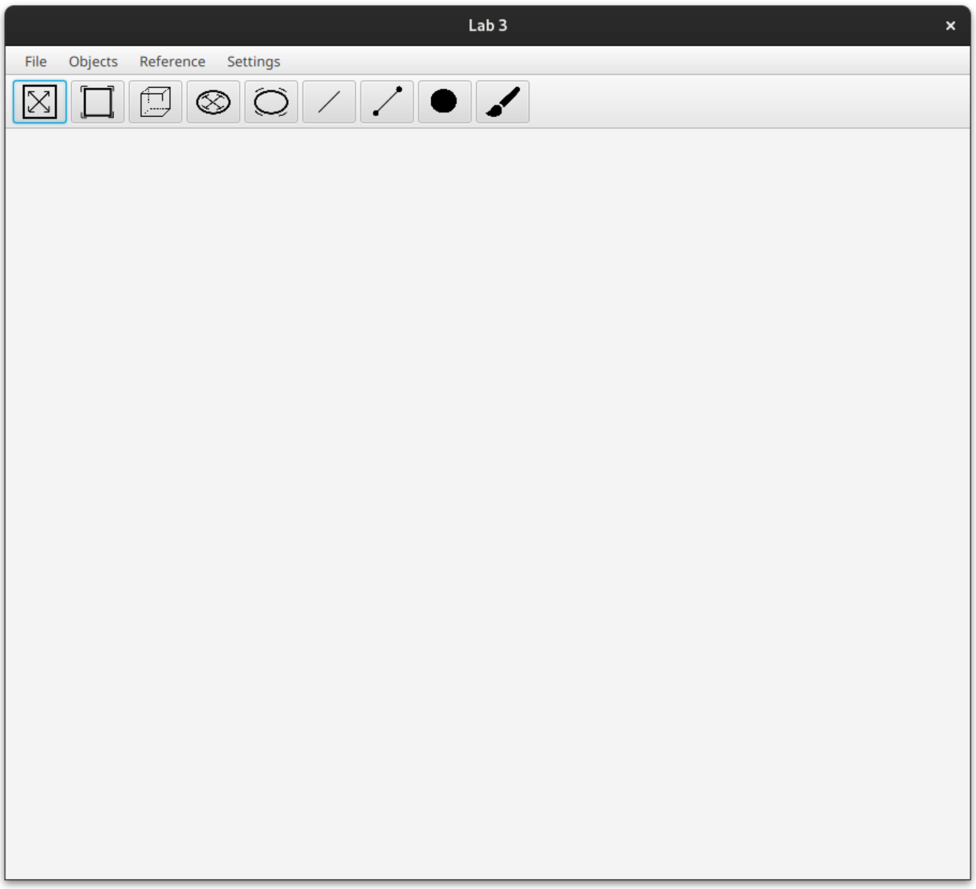
```
package controllers;


import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.BorderPane;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.scene.canvas.Canvas;
import javafx.scene.control.Button;
import javafx.scene.control.Menu;
import javafx.scene.control.RadioMenuItem;
import javafx.scene.control.ToolBar;
import javafx.scene.image.WritableImage;
import javafx.scene.input.MouseButton;
import javafx.scene.input.MouseEvent;
import javafx.scene.control.MenuItem;
import javafx.application.Platform;
import javafx.embed.swing.SwingFXUtils;
import javafx.scene.input.KeyCode;
import java.io.IOException;
import java.util.ArrayList;
import settings.Color;
import settings.Fill;
import shapes.*;
import java.util.Map;
import javax.imageio.ImageIO;

import editors.Editor;

public class MenuController {

  @FXML
  private BorderPane borderPane;

  @FXML
  private Menu objectsMenu;

  @FXML
  private AnchorPane anchorPane;

  @FXML
  private Menu colors;

  @FXML
  private RadioMenuItem lastSelected = null;

  @FXML
  private Canvas canvas;

  @FXML
  private ToolBar toolBar;

  private Editor editor;
  private final Map<String, Class<? extends Shape>> editors =
Map.of(
    "rectangleCenter", RectangleCenter.class,
    "rectangleCorner", RectangleCorner.class,
    "ellipseCenter", EllipseCenter.class,
    "ellipseCorner", EllipseCorner.class,
    "line", Line.class,
    "point", Point.class,
    "brush", Brush.class,
    "line-ellipse", LineEllipse.class,
    "cube", Cube.class
  );

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

  private void processEvent(final Shape shape, final RadioMenuItem
item) {
    anchorPane.setOnMousePressed((event) -> {
      if (isPrimary(event) && item.isSelected()) {
        editor.add(shape);
        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());
        processEvent(getShape(item.getId()), item);
      }
    });
  }

  @FXML
  private void exit() {
    Platform.exit();
  }

  @FXML
  private void saveAs() throws IOException {
    final var stage = (Stage)borderPane.getScene().getWindow();
    final var savefile = new FileChooser();
    savefile.setTitle("Save File");
    final var file = savefile.showSaveDialog(stage);
    if (file == null) return;
    final var writableImage = new
WritableImage((int)canvas.getWidth(), (int)canvas.getHeight());
    canvas.snapshot(null, writableImage);
    final var renderedImage =
SwingFXUtils.fromFXImage(writableImage, null);
    ImageIO.write(renderedImage, "png", file);
  }
```

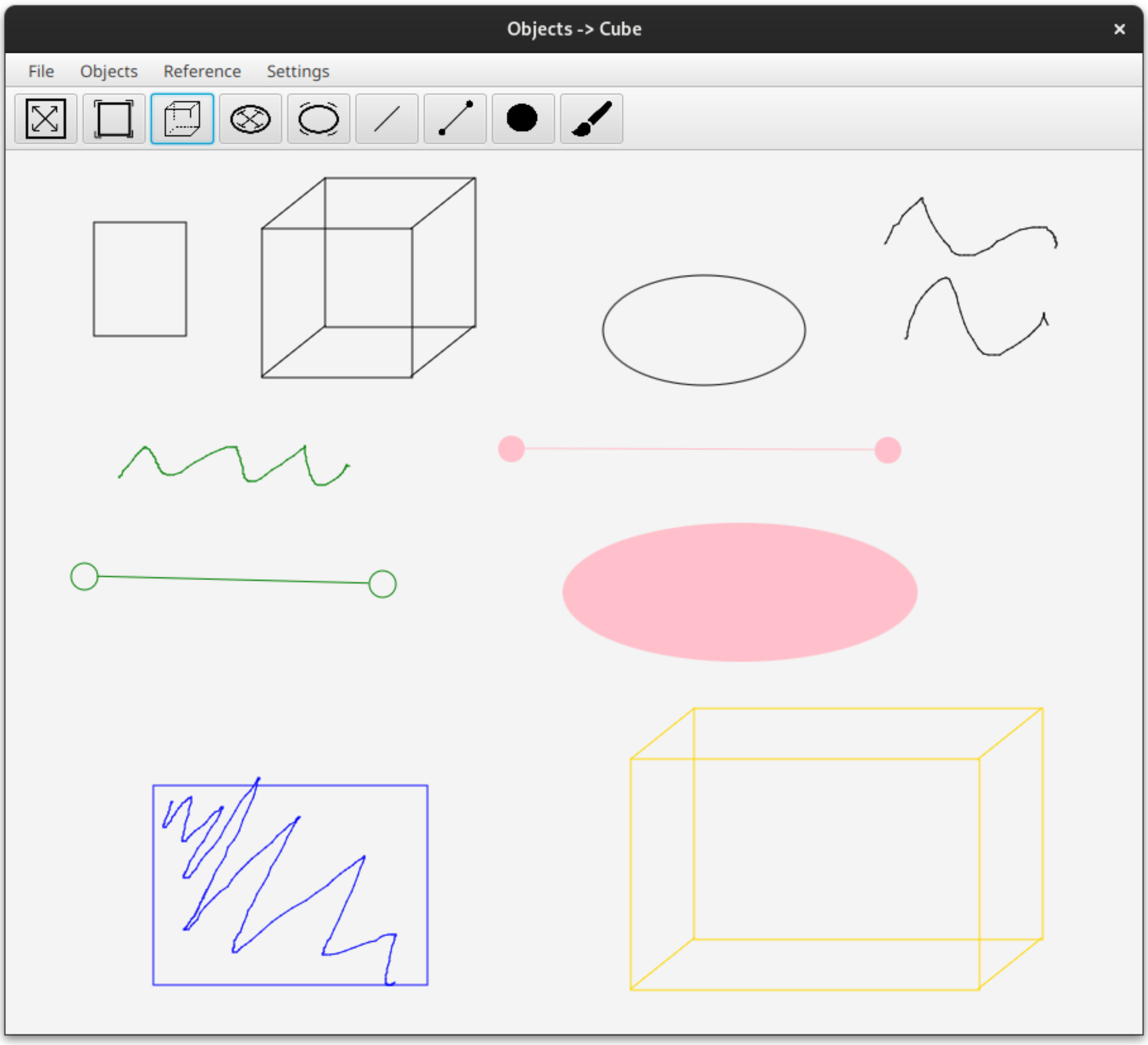
```
@FXML
private void colors(final ActionEvent event) {
  final var item = (MenuItem)event.getTarget();
  final var text = item.getText();
  Color.setColor(text);
}

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

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

private Shape getShape(final String id) {
  final var constructor = editors.get(id);
  try {
    final var declared = constructor.getDeclaredConstructor();
    final var shape = declared.newInstance();
    return shape;
    } catch (Exception e) {
      e.printStackTrace();
      return null;
    }
  }

  @SuppressWarnings("unused")
  private void addItemsEvenets(final Menu root) {
    for (final var item: root.getItems()) {
      if (item instanceof Menu menu) {
        addItemsEvenets(menu);
        continue;
      }
      final var selected = (RadioMenuItem)item;
      final var fullPath = getFullName(selected, objectsMenu);
      item.setOnAction((event) -> {
        if (lastSelected != null) lastSelected.setSelected(false);
        selected.setSelected(true);
        lastSelected = selected;
        final var window = (Stage)borderPane.getScene().getWindow();
        window.setTitle(fullPath);
        final var shape = getShape(selected.getId());
        processEvent(shape, selected);
      });
      final var buttonId = selected.getId() + "-button";
      final var button = (Button)toolBar.getItems().filtered((node)
-> {
        return node.getId().equals(buttonId);
      }).getFirst();
      button.setOnAction((event) -> item.fire());
    }
  }

  @SuppressWarnings("unused")
  @FXML
  private void initialize() {
    addColors();
    addItemsEvenets(objectsMenu);
    editor = new Editor(canvas);
    borderPane.sceneProperty().addListener((property) -> {
      final var scene = borderPane.getScene();
      scene.setOnKeyPressed((event) -> {
      if (event.isControlDown() && (event.getCode() == KeyCode.Z)) {
        editor.pop();
        editor.clear();
        editor.drawAll();
      };
    });
    });
  }

  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() : "";
  }
}
```

Діаграма наслідування



Скріншоти виконання





Висновки

Під час виконання лабораторної роботи я здобув навички використання інкапсуляції, абстрактних типів, успадкування та поліморфізму, множинного успадкування, інтерфейсів, 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 та множинне наслідування.