Aslı Aykan
2016400222

CMPE 496 | HW1

Description

It is an implementation of an object-oriented drawing editor that allows users to create, move and delete rectangles, squares, circles, and lines in an interactive graphics.

Implementation Details

For this project, I preferred to use Java GUI. I created a class called *Editor.java*, in which the main method exists, and all the events are handled. For the shapes, I created four classes which are *MySquare.java*, *MyRectangle.java*, *MyCircle.java* and *MyLine.java.* So, the project consists of 5 classes.

In the window editor, I created a large panel, which is the drawing area, and radio buttons for users to choose any action. The actions are as follows:

-Add Rectangle
-Add Line
-Add Square
-Add Circle
- Move Shape
- Delete Shape
- Enlarge
- Shrink
- Rotate

The *add* actions are to create any shape. By choosing any of this action and clicking on the drawing area, the related shape will be created in the place of the click action.

The *move shape* action is to move any shape. By choosing this action, clicking on any shape and dragging it will move the shape to the place that the dragging action ends in.

The *delete shape* action is to delete any shape. By choosing this action, clicking on any shape will delete the shape from the drawing area.

The *enlarge* action is to enlarge any shape. By choosing this action, clicking on any shape will enlarge the shape.

The *shrink* action is to shrink any shape. By choosing this action, clicking on any shape will shrink the shape.

The *rotate* action is to shrink any shape. By choosing this action, clicking on any shape will rotate the shape. (It rotates 90 degrees, which will rotate the line and the rectangle.)

On the top left corner of the window, there exists a text area, in which the information for which radio button is for which action is written.

I also added a color palette to the left side of the window editor. The default color is red for my drawing editor. By clicking any other color in the color palette, users can create new shapes with different colors.

All the classes for shapes extend *JComponent* class and they contain overridden methods which are *paintComponent(Graphics g)*, *contains(Point p)*, and *setLocation(Point p).*

paintComponent(Graphics g): draws the shape on the Graphics object
contains(Point p): check whether the shape contains the point p

setLocation(Point p): move the shape on the Graphics object so that the point p is the new center of that shape.

The classes for the shapes have some additional methods which are *delete()*, *enlarge()* and *shrink().*

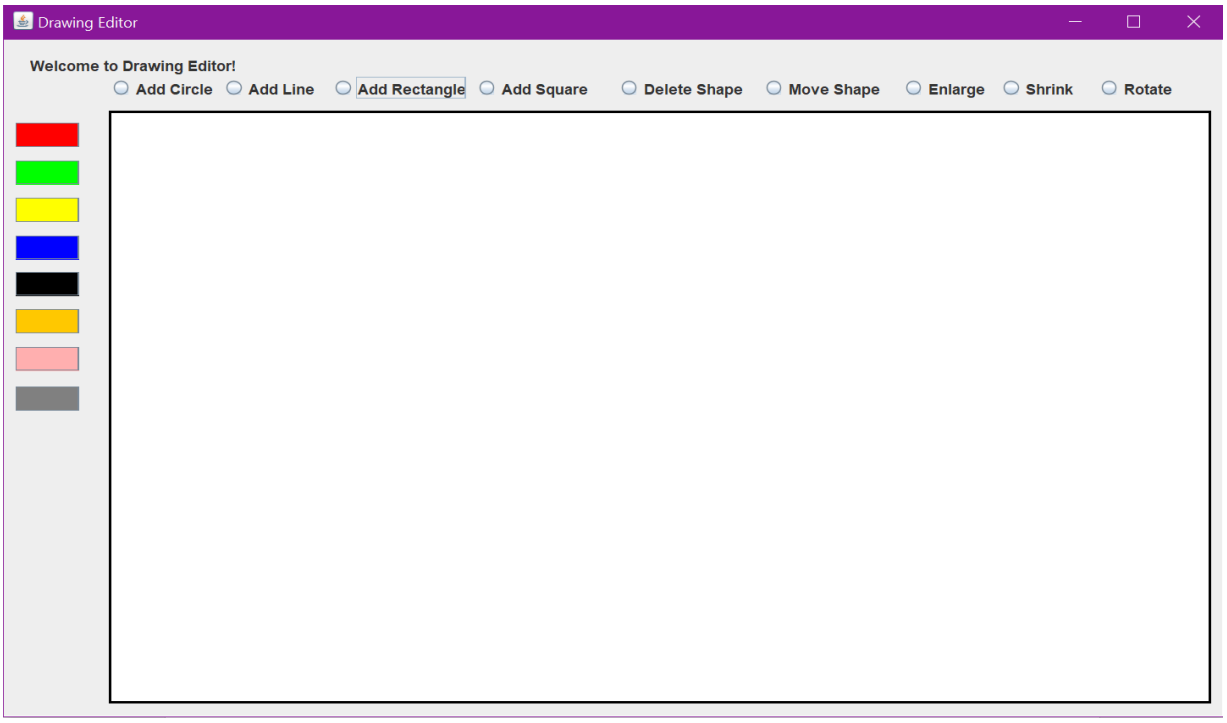
delete(): deletes the shape on Graphics object
enlarge(): enlarges the shape by deleting the old shape from Graphics objects and redrawing on it with new coordinates. It calls *paintComponent()* for redrawing.
shrink(): shrinks the shape by deleting the old shape from Graphics objects and redrawing on it with new coordinates. It calls *paintComponent()* for redrawing.

In addition to these methods, all classes for shapes have a constructor, in which the coordinates and the color for the shapes is defined.

In the *Editor.java* class, a *JFrame* is initialized with a *JPanel.* All radio buttons and toggle buttons for color palette are also added to this frame. With the help of action listeners, color can be changed by clicking on any color and user can change its action by clicking different radio buttons.
Panel has also a mouse listener, which overrides *mousePressed(Event e)* and *mouseReleased(Event e)* methods. In *mousePressed(Event e)* method, according to the selected radio button, appropriate methods are called. When new shapes are added, it is added to the panel and when a shape is deleted, it is deleted from the panel. In order to understand which shape is clicked for move, delete, enlarge, shrink and rotate actions, there exists loops which iterate on the components that are in the panel and call related *contains(Point p)* method to find the shape. In *mouseReleased(Event e)* method, the dragged point is sent as parameter to the related *setLocation(Point p)* function to move the shape.

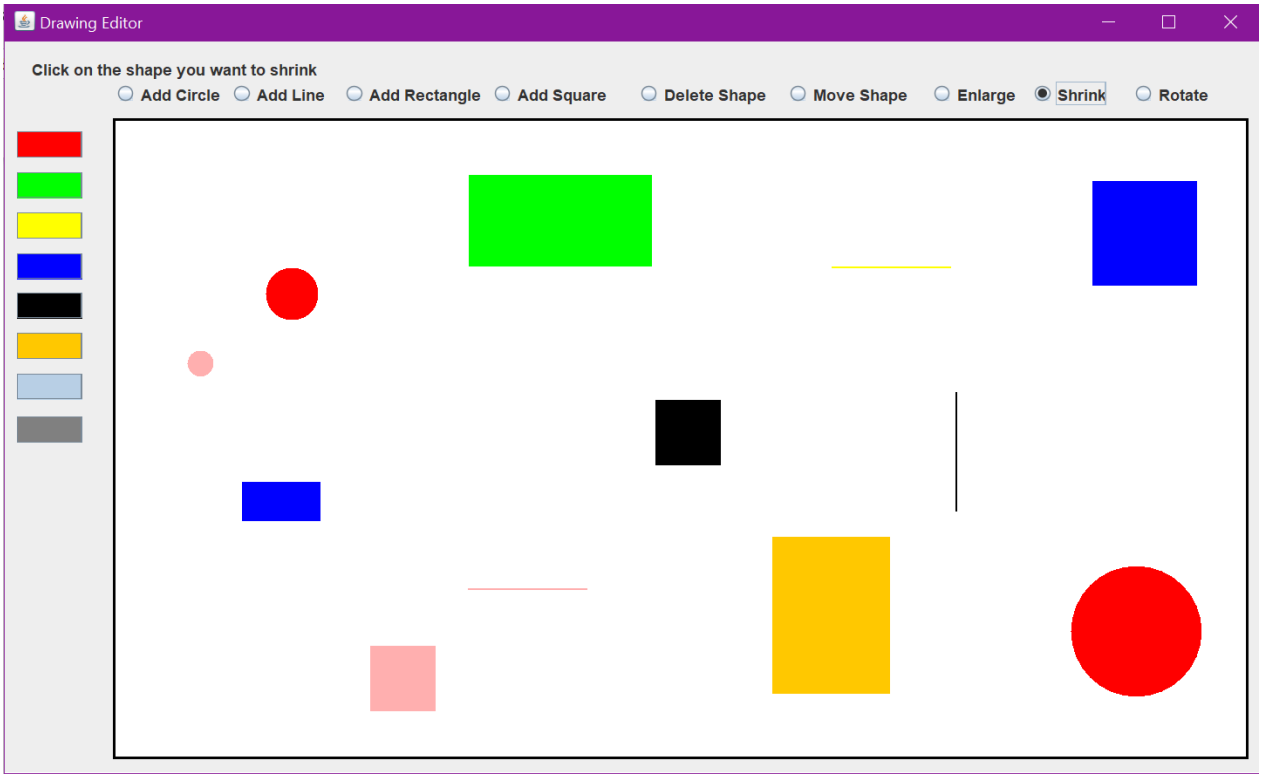
Images of the Drawing Editor

initially empty editor:



Aslı Aykan
2016400222

the drawing editor after uses adds different shapes in different colors, rotates, enlarges, and shrinks:



Code

Code screenshots are on the next pages.

Aslı Aykan
2016400222

```java
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Point;

import javax.swing.JComponent;

public class MyRectangle extends JComponent {

    private Graphics graphic; // graphics object to be used for painting the shape
    private Color color; // color of the shape
    private int leftX; // x coordinate of the left edge of the rectangle
    private int rightX; // x coordinate of the right edge of the rectangle
    private int topY; // y coordinate of the top edge of the rectangle
    private int bottomY; // y coordinate of the bottom edge of the rectangle
    private int oX; // x coordinate of the clicked point
    private int oY; // y coordinate of the clicked point
    private int width = 100; // width of the rectangle
    private int height = 50; // height of the rectangle

    // constructor
    public MyRectangle(int x,int y, Graphics g, Color _color) {
        graphic = g;
        oX = x;
        oY = y;
        color = _color;
        leftX = x-width/2;
        rightX = x+width/2;
        topY = y-height/2;
        bottomY = y+height/2;
        paintComponent(graphic);
    }

    // overridden paintComponent function to paint the shape
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        graphic.setColor(color);
        graphic.fillRect(leftX,topY,width,height);
    }

    // overridden contains function to check whether the clicked point is in the shape
    @Override
    public boolean contains(Point p) {
        super.contains(p);
        if(leftX<=p.getX() && p.getX()<=rightX && topY<=p.getY() && p.getY()<=bottomY) {
            return true;
        } else {
            return false;
        }

    }

    // overridden setLocation function to set new location for moved shape
    @Override
    public void setLocation(Point p) {
        super.setLocation(p);
        graphic.setColor(Color.WHITE);
        graphic.fillRect(leftX,topY,width,height);
        leftX = (int) (p.getX()-width/2);
        rightX = (int) (p.getX()+width/2);
        topY = (int) (p.getY()-height/2);
        bottomY = (int) (p.getY()+height/2);
        oX = (leftX+rightX)/2;
        oY = (topY+bottomY)/2;
        paintComponent(graphic);
    }

    // delete function to delete the shape
    public void delete() {
        graphic.setColor(Color.WHITE);
        graphic.fillRect(leftX,topY,width,height);
    }

    // enlarge function to enlarge the shape
    public void enlarge() {
        delete();
        leftX -=10;
        rightX +=10;
        width+=20;
        topY-=5;
        bottomY+=5;
        height+=10;
        paintComponent(graphic);
    }

    // shrink function to shrink the shape
    public void shrink() {
        delete();
        leftX +=10;
        rightX -=10;
        width-=20;
        topY+=5;
        bottomY-=5;
        height-=10;
        paintComponent(graphic);
    }

    // rotate function to rotate the shape
    public void rotate() {
        delete();
        leftX = oX-height/2;
        rightX = oX+height/2;
        topY = oY-width/2;
        bottomY = oY+width/2;
        int twidth = width;
        width = height;
        height = twidth;
        paintComponent(graphic);
    }

}
```

Aslı Aykan
2016400222

```java
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Point;

import javax.swing.JComponent;

public class MyCircle extends JComponent {

    private int xCoord; // calculated x coordinate to draw circle
    private int yCoord; // calculated y coordinate to draw circle
    private int oX; // x coordinate of the clicked point
    private int oY; // y coordinate of the clicked point
    private Graphics graphic; // graphics object to be used for painting the shape
    private int radius;  // radius of the circle
    private Color color; // color of the shape

    // constructor
    public MyCircle(int x, int y, Graphics g, Color _color) {
        radius = 40;
        oX = x;
        oY = y;
        xCoord = x - radius / 2;
        yCoord = y - radius / 2;
        graphic = g;
        color = _color;
        paintComponent(g);

    }

    // overridden paintComponent function to paint the shape
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(graphic);

        graphic.setColor(color);
        graphic.fillOval(xCoord, yCoord, radius, radius);

    }

    // overridden contains function to check whether the clicked point is in the shape
    @Override
    public boolean contains(Point p) {
        super.contains((Point) p);
        double distance = Math.sqrt(Math.pow((p.getX() - xCoord), 2) + Math.pow((p.getY() - yCoord),
2));
        if (distance <= radius) {
            return true;
        } else {
            return false;
        }

    }

    // overridden setLocation function to set new location for moved shape
    @Override
    public void setLocation(Point p) {
        super.setLocation(p);
        graphic.setColor(Color.WHITE);
        graphic.fillOval(xCoord, yCoord, radius, radius);
        xCoord = (int) (p.getX() - radius / 2);
        yCoord = (int) (p.getY() - radius / 2);
        paintComponent(graphic);
    }

    // delete function to delete the shape
    public void delete() {
        graphic.setColor(Color.WHITE);
        graphic.fillOval(xCoord, yCoord, radius, radius);
    }

    // enlarge function to enlarge the shape
    public void enlarge() {
        delete();
        radius = radius + 10;
        xCoord = oX - radius / 2;
        yCoord = oY - radius / 2;
        paintComponent(graphic);
    }

    // shrink function to shrink the shape
    public void shrink() {
        delete();
        radius = radius - 10;
        xCoord = oX - radius / 2;
        yCoord = oY - radius / 2;
        paintComponent(graphic);
    }

}
```

Aslı Aykan
2016400222

```java
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Point;

import javax.swing.JComponent;

public class MyLine extends JComponent {


    private int firstX;  // x coordinate of the start point
    private int secondX; // x coordinate of the end point
    private int firstY;  // y coordinate of the start point
    private int secondY; // y coordinate of the end point
    private int oX; // x coordinate of the clicked point
    private int oY; // y coordinate of the clicked point
    private int length = 90;  // length of line
    private Graphics graphic; // graphics object to be used for painting the shape
    private Color color; // color of the shape

    // constructor
    public MyLine(int x, int y, Graphics g, Color _color) {
        oX = x;
        oY = y;
        firstX = x - length / 2;
        secondX = x + length / 2;
        firstY = y;
        secondY = y;
        graphic = g;
        color = _color;
        paintComponent(graphic);
    }

    // overridden paintComponent function to paint the shape
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        graphic.setColor(color);
        graphic.drawLine(firstX, firstY, secondX, secondY);
    }

    // overridden contains function to check whether the clicked point is in the shape
    @Override
    public boolean contains(Point p) {
        super.contains((Point) p);
        if (firstX - 2 <= p.getX() && p.getX() <= secondX + 2 && firstY - 2 <= p.getY() && p.getY() <=
secondY + 2) {
            return true;
        } else {
            return false;
        }

    }

    // overridden setLocation function to set new location for moved shape
    @Override
    public void setLocation(Point p) {
        super.setLocation(p);
        delete();
        if (firstY == secondY) {
            firstX = (int) (p.getX() - length / 2);
            secondX = (int) (p.getX() + length / 2);
            firstY = (int) (p.getY());
            secondY = (int) (p.getY());
        } else {
            firstX = (int) (p.getX());
            secondX = (int) (p.getX());
            firstY = (int) (p.getY() - length / 2);
            secondY = (int) (p.getY() + length / 2);
        }
        paintComponent(graphic);
    }

    // delete function to delete the shape
    public void delete() {
        graphic.setColor(Color.WHITE);
        graphic.drawLine(firstX, firstY, secondX, secondY);
    }

    // enlarge function to enlarge the shape
    public void enlarge() {
        delete();
        if (firstY == secondY) {
            firstX -= 5;
            secondX += 5;
        } else {
            firstY -= 5;
            secondY += 5;
        }
        length += 10;
        paintComponent(graphic);
    }

    // shrink function to shrink the shape
    public void shrink() {
        delete();
        if (firstY == secondY) {
            firstX += 5;
            secondX -= 5;
        } else {
            firstY += 5;
            secondY -= 5;
        }
        length -= 10;
        paintComponent(graphic);
    }

    // rotate function to rotate the shape
    public void rotate() {
        delete();
        if (firstY == secondY) {
            firstX = oX;
            secondX = oX;
            firstY = oY - length / 2;
            secondY = oY + length / 2;
        } else {
            firstX = oX - length / 2;
            secondX = oX + length / 2;
            firstY = oY;
            secondY = oY;
        }

        paintComponent(graphic);
    }
}
```

Aslı Aykan
2016400222

```java
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Point;

import javax.swing.JComponent;

public class MySquare extends JComponent {

    private int leftX;  // x coordinate of the left edge of the square
    private int rightX; // x coordinate of the right edge of the square
    private int topY;   // y coordinate of the top edge of the square
    private int bottomY; // y coordinate of the bottom edge of the square
    private int length = 50; // length of each edge of the square
    private Graphics graphic; // graphics object to be used for painting
    private Color color; // color of the shape

    // constructor
    public MySquare(int x, int y, Graphics g, Color _color) {
        leftX = x - length / 2;
        rightX = x + length / 2;
        topY = y - length / 2;
        bottomY = y + length / 2;
        graphic = g;
        color = _color;
        paintComponent(graphic);
    }

    // overridden paintComponent function to paint the shape
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        graphic.setColor(color);
        graphic.fillRect(leftX, topY, length, length);
    }

    // overridden contains function to check whether the clicked point is in the shape
    @Override
    public boolean contains(Point p) {
        super.contains(p);
        if (leftX <= p.getX() && p.getX() <= rightX && topY <= p.getY() && p.getY() <= bottomY) {
            return true;
        } else {
            return false;
        }

    }

    // overridden setLocation function to set new location for moved shape
    @Override
    public void setLocation(Point p) {
        super.setLocation(p);
        delete();
        leftX = (int) (p.getX() - length / 2);
        rightX = (int) (p.getX() + length / 2);
        topY = (int) (p.getY() - length / 2);
        bottomY = (int) (p.getY() + length / 2);
        paintComponent(graphic);
    }

    // delete function to delete shape
    public void delete() {
        graphic.setColor(Color.WHITE);
        graphic.fillRect(leftX, topY, length, length);
    }

    // enlarge function to enlarge the shape
    public void enlarge() {
        delete();
        leftX -= 5;
        rightX += 5;
        topY -= 5;
        bottomY += 5;
        length += 10;
        paintComponent(graphic);
    }

    // shrink function to shrink the shape
    public void shrink() {
        delete();
        leftX += 5;
        rightX -= 5;
        topY += 5;
        bottomY -= 5;
        length -= 10;
        paintComponent(graphic);
    }

}
```

Aslı Aykan
2016400222

```java
import java.awt.EventQueue;
import java.awt.Point;
import javax.swing.JFrame;
import javax.swing.ButtonGroup;
import javax.swing.JComponent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.awt.event.ActionEvent;
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.Component;
import javax.swing.border.LineBorder;
import javax.swing.JRadioButton;
import javax.swing.JToggleButton;
import javax.swing.JLabel;

public class Editor implements ActionListener {

    private JFrame frame; // frame of the application
    private Color color; // selected color for drawing


    // Launching the application.
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Editor window = new Editor();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }


    // Creating the application
    public Editor() {
        initialize();
    }


    // Initializing the contents of the frame
    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 975, 597);
        frame.setTitle("Drawing Editor");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);
        color = Color.RED;

        JPanel panel = new JPanel();
        panel.setOpaque(true);
        panel.setLayout(null);
        panel.setBorder(new LineBorder(new Color(0, 0, 0), 2));
        panel.setBackground(Color.WHITE);
        panel.setBounds(83, 59, 868, 490);
        panel.setVisible(true);
        frame.getContentPane().add(panel);

        // adding action buttons
        JRadioButton rdbtnRectangle = new JRadioButton("Add Rectangle");
        rdbtnRectangle.setBounds(258, 29, 111, 23);
        frame.getContentPane().add(rdbtnRectangle);

        JRadioButton rdbtnCircle = new JRadioButton("Add Circle");
        rdbtnCircle.setBounds(83, 29, 87, 23);
        frame.getContentPane().add(rdbtnCircle);

        JRadioButton rdbtnLine = new JRadioButton("Add Line");
        rdbtnLine.setBounds(172, 29, 84, 23);
        frame.getContentPane().add(rdbtnLine);

        JRadioButton rdbtnMoveShape = new JRadioButton("Move Shape");
        rdbtnMoveShape.setBounds(597, 29, 97, 23);
        frame.getContentPane().add(rdbtnMoveShape);

        JRadioButton rdbtnDeleteShape = new JRadioButton("Delete Shape");
        rdbtnDeleteShape.setBounds(483, 29, 111, 23);
        frame.getContentPane().add(rdbtnDeleteShape);

        JRadioButton rdbtnSquare = new JRadioButton("Add Square");
        rdbtnSquare.setBounds(371, 29, 102, 23);
        frame.getContentPane().add(rdbtnSquare);

        JRadioButton rdbtnEnlarge = new JRadioButton("Enlarge");
        rdbtnEnlarge.setBounds(707, 29, 75, 23);
        frame.getContentPane().add(rdbtnEnlarge);

        JRadioButton rdbtnShrink = new JRadioButton("Shrink");
        rdbtnShrink.setBounds(784, 29, 75, 23);
        frame.getContentPane().add(rdbtnShrink);

        JRadioButton rdbtnRotate = new JRadioButton("Rotate");
        rdbtnRotate.setBounds(861, 29, 75, 23);
        frame.getContentPane().add(rdbtnRotate);

        ButtonGroup bgroup = new ButtonGroup();
        bgroup.add(rdbtnRectangle);
        bgroup.add(rdbtnCircle);
        bgroup.add(rdbtnLine);
        bgroup.add(rdbtnMoveShape);
        bgroup.add(rdbtnDeleteShape);
        bgroup.add(rdbtnSquare);
        bgroup.add(rdbtnEnlarge);
        bgroup.add(rdbtnShrink);
        bgroup.add(rdbtnRotate);

        // adding color buttons
        ButtonGroup bgroup2 = new ButtonGroup();
        JToggleButton redbtn = new JToggleButton();
        redbtn.setBounds(10, 69, 50, 20);
        redbtn.setBackground(Color.RED);
        redbtn.setName("red");
        JToggleButton grnbtn = new JToggleButton();
        grnbtn.setBounds(10, 100, 50, 20);
        grnbtn.setBackground(Color.GREEN);
        grnbtn.setName("green");
        JToggleButton ylwbtn = new JToggleButton();
        ylwbtn.setBounds(10, 131, 50, 20);
        ylwbtn.setBackground(Color.YELLOW);
        ylwbtn.setName("yellow");
        JToggleButton bluebtn = new JToggleButton();
        bluebtn.setBounds(10, 162, 50, 20);
        bluebtn.setBackground(Color.BLUE);
        bluebtn.setName("blue");
        JToggleButton blackbtn = new JToggleButton();
        blackbtn.setBounds(10, 192, 50, 20);
        blackbtn.setBackground(Color.BLACK);
        blackbtn.setName("black");
        JToggleButton ornbtn = new JToggleButton();
        ornbtn.setName("orange");
        ornbtn.setBackground(Color.ORANGE);
        ornbtn.setBounds(10, 223, 50, 20);
        JToggleButton pkbtn = new JToggleButton();
        pkbtn.setName("pink");
        pkbtn.setBackground(Color.PINK);
        pkbtn.setBounds(10, 254, 50, 20);
        JToggleButton graybtn = new JToggleButton();
        graybtn.setName("gray");
        graybtn.setBackground(Color.GRAY);
        graybtn.setBounds(10, 287, 50, 20);
        frame.getContentPane().add(redbtn);
        frame.getContentPane().add(grnbtn);
        frame.getContentPane().add(ylwbtn);
        frame.getContentPane().add(bluebtn);
        frame.getContentPane().add(graybtn);
        frame.getContentPane().add(blackbtn);
        frame.getContentPane().add(ornbtn);
        frame.getContentPane().add(pkbtn);
        bgroup2.add(redbtn);
        bgroup2.add(grnbtn);
        bgroup2.add(ylwbtn);
        bgroup2.add(bluebtn);
        bgroup2.add(graybtn);
        bgroup2.add(blackbtn);
        bgroup2.add(ornbtn);
        bgroup2.add(pkbtn);

        JLabel text = new JLabel("Welcome to Drawing Editor!");
        text.setBounds(21, 11, 235, 20);
        frame.getContentPane().add(text);

        // adding action listeners
        redbtn.addActionListener(this);
        bluebtn.addActionListener(this);
        graybtn.addActionListener(this);
        ylwbtn.addActionListener(this);
        grnbtn.addActionListener(this);
        blackbtn.addActionListener(this);
        ornbtn.addActionListener(this);
        pkbtn.addActionListener(this);

        ....
```

Aslı Aykan
2016400222

```
// text message of add rectangle button
        rdbtnRectangle.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("Click the drawing area to add rectangle");

            }
        });

        // text message of add circle button
        rdbtnCircle.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("Click the drawing area to add circle");

            }
        });

        // text message of add line button
        rdbtnLine.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("Click the drawing area to add line");

            }
        });

        // text message of add square button
        rdbtnSquare.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("Click the drawing area to add square");

            }
        });

        // text message of move shape button
        rdbtnMoveShape.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("Click on the shape and drag it to move");

            }
        });

        // text message of delete shape button
        rdbtnDeleteShape.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("Click on the shape you want to delete");

            }
        });

        // text message of enlarge button
        rdbtnEnlarge.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("Click on the shape you want to enlarge");

            }
        });

        // text message of shrink button
        rdbtnShrink.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("Click on the shape you want to shrink");

            }
        });

        // text message of rotate button
        rdbtnRotate.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                text.setText("Click on the shape you want to rotate");

            }
        });

        // mouse listener for the panel
        panel.addMouseListener(new MouseAdapter() {
            private JComponent moved;
            private boolean isMoved = false;

            @Override
            public void mousePressed(MouseEvent e) {

                // adding circle to the panel
                if (rdbtnCircle.isSelected()) {
                    MyCircle circle = new MyCircle(e.getX(), e.getY(), panel.getGraphics(), color);
                    panel.add(circle);
                // adding line to the panel
                } else if (rdbtnLine.isSelected()) {
                    MyLine line = new MyLine(e.getX(), e.getY(), panel.getGraphics(), color);
                    panel.add(line);
                // adding rectangle to the panel
                } else if (rdbtnRectangle.isSelected()) {
                    MyRectangle rect = new MyRectangle(e.getX(), e.getY(), panel.getGraphics(), color);
                    panel.add(rect);
                // adding square to the panel
                } else if (rdbtnSquare.isSelected()) {
                    MySquare sqr = new MySquare(e.getX(), e.getY(), panel.getGraphics(), color);
                    panel.add(sqr);
                // finding which shape to be moved
                } else if (rdbtnMoveShape.isSelected()) {
                    panel.addMouseMotionListener(new MouseMotionAdapter() {
                        public void mouseDragged(MouseEvent e) {
                            for (Component c : panel.getComponents()) {
                                if (c.contains(e.getPoint())) {
                                    moved = (JComponent) c;
                                    isMoved = true;
                                    break;
                                }
                            }
                        }

                    });
                // deleting the shape from panel
                } else if (rdbtnDeleteShape.isSelected()) {
                    for (Component c : panel.getComponents()) {
                        if (c.contains(e.getPoint())) {
                            panel.remove(c);
                            if (c instanceof MyCircle) {
                                ((MyCircle) c).delete();
                            } else if (c instanceof MyRectangle) {
                                ((MyRectangle) c).delete();
                            } else if (c instanceof MySquare) {
                                ((MySquare) c).delete();
                            } else if (c instanceof MyLine) {
                                ((MyLine) c).delete();
                            }
                            break;
                        }
                    }
                // finding which shape to enlarge
                } else if (rdbtnEnlarge.isSelected()) {
                    for (Component c : panel.getComponents()) {
                        if (c.contains(e.getPoint())) {
                            if (c instanceof MyCircle) {
                                ((MyCircle) c).enlarge();
                            } else if (c instanceof MyRectangle) {
                                ((MyRectangle) c).enlarge();
                            } else if (c instanceof MySquare) {
                                ((MySquare) c).enlarge();
                            } else if (c instanceof MyLine) {
                                ((MyLine) c).enlarge();
                            }
                        }
                    }
                // finding which shape to shrink
                } else if (rdbtnShrink.isSelected()) {
                    for (Component c : panel.getComponents()) {
                        if (c.contains(e.getPoint())) {
                            if (c instanceof MyCircle) {
                                ((MyCircle) c).shrink();
                            } else if (c instanceof MyRectangle) {
                                ((MyRectangle) c).shrink();
                            } else if (c instanceof MySquare) {
                                ((MySquare) c).shrink();
                            } else if (c instanceof MyLine) {
                                ((MyLine) c).shrink();
                            }
                        }
                    }
                // finding which shape to rotate
                } else if (rdbtnRotate.isSelected()) {
                    for (Component c : panel.getComponents()) {
                        if (c.contains(e.getPoint())) {
                            if (c instanceof MyRectangle) {
                                ((MyRectangle) c).rotate();
                            } else if (c instanceof MyLine) {
                                ((MyLine) c).rotate();
                            }
                        }
                    }
                }

            }

            // setting location of the moved shape
            @Override
            public void mouseReleased(MouseEvent e) {
                if (moved != null && isMoved) {
                    Point moveP = new Point(e.getPoint());
                    moved.setLocation(moveP);
                    isMoved = false;
                }
            }
        });
    }

    // setting the color for drawing
    @Override
    public void actionPerformed(ActionEvent e) {
        String btnName = ((Component) e.getSource()).getName();
        if (btnName.equals("red")) {
            color = Color.RED;
        } else if (btnName.equals("green")) {
            color = Color.GREEN;
        } else if (btnName.equals("yellow")) {
            color = Color.YELLOW;
        } else if (btnName.equals("blue")) {
            color = Color.BLUE;
        } else if (btnName.equals("gray")) {
            color = Color.GRAY;
        } else if (btnName.equals("black")) {
            color = Color.BLACK;
        } else if (btnName.equals("pink")) {
            color = Color.PINK;
        } else if (btnName.equals("orange")) {
            color = Color.ORANGE;
        }

    }
}
```