Lab Assignment NO :7

Name:Swapnil Satish Kalshetti

Class: SE A

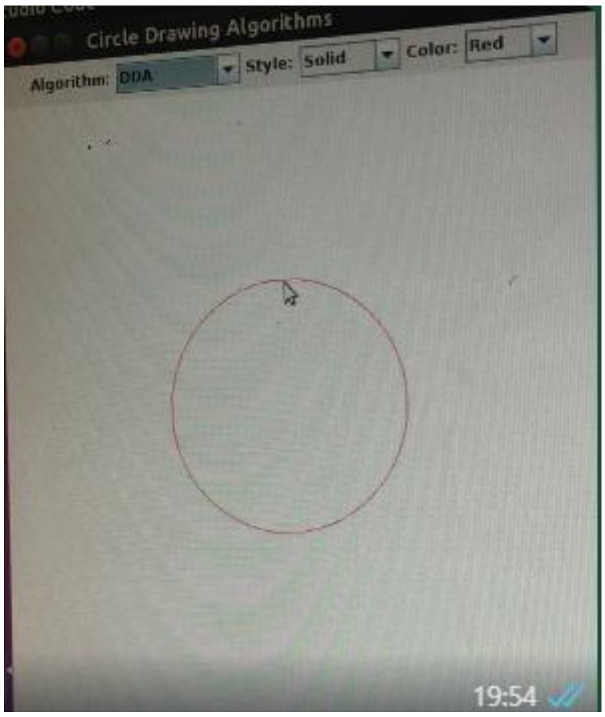
Roll no: CO2062

Subject: OOP&CG

```
import javax.swing.*;

import java.awt.*; import

java.awt.event.*;


public class CircleDrawer extends JFrame {


   private JComboBox<String> algorithmBox, styleBox, colorBox;     private

DrawPanel drawPanel;


   public CircleDrawer() {        setTitle("Circle Drawing

Algorithms");        setSize(600, 600);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

setLayout(new BorderLayout());


      // --- Top Control Panel ---

      JPanel controlPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));


      // Dropdown menus        algorithmBox = new JComboBox<>(new

String[]{"DDA", "Bresenham"});        styleBox = new JComboBox<>(new

String[]{"Solid", "Dashed", "Dotted"});        colorBox = new JComboBox<>(new

String[]{"Red", "Green", "Blue"});
```

```
        controlPanel.add(new JLabel("Algorithm:"));
controlPanel.add(algorithmBox);        controlPanel.add(new
JLabel("Style:"));        controlPanel.add(styleBox);
controlPanel.add(new JLabel("Color:"));
controlPanel.add(colorBox);

        add(controlPanel, BorderLayout.NORTH);

        // --- Drawing Panel ---        drawPanel =
new DrawPanel();        add(drawPanel,
BorderLayout.CENTER);

        // Repaint when selection changes
        ActionListener update = e -> drawPanel.repaint();
algorithmBox.addActionListener(update);
styleBox.addActionListener(update);        colorBox.addActionListener(update);
    }

    // Panel that draws the circle
class DrawPanel extends JPanel {
        @Override        protected void
paintComponent(Graphics g) {
super.paintComponent(g);

            // Get selected options
            String algorithm = (String) algorithmBox.getSelectedItem();
            String style = (String) styleBox.getSelectedItem();
            String color = (String) colorBox.getSelectedItem();

            // Set color            switch (color) {
case "Red" -> g.setColor(Color.RED);
case "Green" -> g.setColor(Color.GREEN);
case "Blue" -> g.setColor(Color.BLUE);
            }

            // Circle parameters
int xc = getWidth() / 2;
int yc = getHeight() / 2;
int r = 120;

            if (algorithm.equals("DDA")) {
drawCircleDDA(g, xc, yc, r, style);
            } else {
                drawCircleBresenham(g, xc, yc, r, style);
            }
        }

        // --- DDA Circle Algorithm ---         private void drawCircleDDA(Graphics g,
int xc, int yc, int r, String style) {             for (int theta = 0; theta < 360; theta++) {
double x = xc + r * Math.cos(Math.toRadians(theta));                double y = yc + r
* Math.sin(Math.toRadians(theta));

                // Apply style pattern                if (style.equals("Dashed"))
{                    if (theta % 15 < 8) g.drawLine((int)x, (int)y, (int)x,
(int)y);
                } else if (style.equals("Dotted")) {                    if (theta % 10
== 0) g.drawLine((int)x, (int)y, (int)x, (int)y);
                } else {
                    g.drawLine((int)x, (int)y, (int)x, (int)y);
                }
            }
        }
        // --- Bresenham Circle Algorithm ---         private void
drawCircleBresenham(Graphics g, int xc, int yc, int r, String style) {            int x = 0, y =
r;            int d = 3 - 2 * r;            int step = 0;

            while (x <= y) {                // Style handling
if (style.equals("Dashed")) {                    if (step % 15 <
8) drawPoints(g, xc, yc, x, y);
                } else if (style.equals("Dotted")) {                    if
(step % 10 == 0) drawPoints(g, xc, yc, x, y);
                } else {
drawPoints(g, xc, yc, x, y);
                }

                step++;                if (d <=
0) d = d + 4 * x + 6;                else {
d = d + 4 * (x - y) + 10;
                    y--
;                 }
x++;
            }
        }

        // Helper for 8-way symmetry         private void
drawPoints(Graphics g, int xc, int yc, int x, int y) {
g.drawLine(xc + x, yc + y, xc + x, yc + y);

            g.drawLine(xc - x, yc + y, xc - x, yc + y);
            g.drawLine(xc + x, yc - y, xc + x, yc - y);
            g.drawLine(xc - x, yc - y, xc - x, yc - y);
            g.drawLine(xc + y, yc + x, xc + y, yc + x);
            g.drawLine(xc - y, yc + x, xc - y, yc + x);
            g.drawLine(xc + y, yc - x, xc + y, yc - x);
            g.drawLine(xc - y, yc - x, xc - y, yc - x);
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new CircleDrawer().setVisible(true));
    }
}
```

Output:


Circle Drawing Algorithms
Algorithm: DDA
Style: Solid
Color: Red
19:54