```
static Color activeColor;
        static JButton[] buttons;
        public static void main(String[] argos) {

                activeColor = Color.WHITE;
                buttons = new JButton[5];
                buttons[0] = new JButton("GRAY");
                buttons[1] = new JButton("RED");
                buttons[2] = new JButton("BLUE");
                buttons[3] = new JButton("WHITE");
                buttons[4] = new JButton("BLACK");

                JFrame jf = new JFrame("Graphics Demo");
                jf.setSize(400, 400);
                jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                //MyPanel mp = new MyPanel();
                //jf.add(mp);
                BetterPanel bp = new BetterPanel();
                bp.addMouseListener(new PressListener(bp));
                jf.add(bp);
                JPanel side = new JPanel(); // a panel to select color
                side.setSize(100, 400) // Usually doesn't work
                side.setLayout(new GridLayout(5, 1, 5, 5));
                ButtonListener b1 = new ButtonListener();

                for (JButton b: buttons) {
                        b.addActionListener(b1);
                        side.add(b);
                }
                JPanel main = new JPanel();
                main.setLayout(new BorderLayout());
                jf.add(main);
                main.add(bp, BorderLayout.WEST);
                main.add(side, BorderLayout.EAST);
                jfsetVisible(true);
        }
        class ButtonListener implements ActionListener {
                @Override
                public void actionPerformed(ActionEvent e) {
                        JButton button = (JButton) e.getSource();
                        if (button.getText() == "GRAY") {
                                CS3913Spring2024Graphics.activeColor = Color.GRAY;
                        } else if (button.getText == "RED") {
                                CS3913Spring2024Graphics.activeColor = Color.RED;
                }
                else if (button.getText == "BLUE") {
                        CS3913Spring2024Graphics.activeColor = Color.BLUE;
                }
                else if (button.getText == "WHITE") {
                        CS3913Spring2024Graphics.activeColor = Color.WHITE;
                }
                else if (button.getText == "BLACK") {
                        CS3913Spring2024Graphics.activeColor = Color.BLACK;
                }
        }
}
class PressListener extends MouseAdapter { // detect presses on the panel
        BetterPanel bp;
        PressListener(BetterPanel newbp) {
                bp = newbp;
        }
        public void mouseClicked(MouseEvent e) {
                // Recognize that mouse was clicked
                bp.addPoint(e.getX(), e.getY());
                bp.repaint();
        }
        @Override
        public void mouseClicked(MouseEvent e) {
                bp.points.add
        }
}
class BetterPanel extends JPanel {
        class Location {
                int x;
                int y;
                Location (int newx, int newy, Color newc) {
                        x = newx;
                        y = newy;
                        c = newc; // each point has a color
                }
        }
        ArrayList<Location> points;
        BetterPanel() {
                super();
                pointer = new ArrayList();

        }
        public void addPoint(int x, int y) {
```

```
		points.add(new Location(x,y));
	}
	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		for (Location l : points) {
			g.setColor(Color.red);
			g.fillOval(l.x - 5, l.y - 5, 10, 10);
		}
	}
}
class MyPanel extends JPanel {
	int calls;
	boolean firstDraw;
	MyPanel() { super(); calls = 0; firstDraw = true; }
	public void paintComponent(Graphics g) {
		super paintComponent(g);

		int height = this.getSize().height;
		int width = this.getSize().width; // JPanels have this attribute
		if (firstDraw) {
			g.fillRect(0, 0, width / 2, height / 2);
			firstDraw = false;
		}
		// reference point is the upper left for fill rect
		// 0,0 is the upper left corner, param 3 is width, param 4 is height
		g.fillRect(0, 0, width / 2, height / 2);
		// fills Q2 of screen
		// resizing the screen will format accordingly
		g.setFont(new Font("Arial", Font.BOLD, 40));
		g.setColor(Color.BLUE);
		g.drawString(""+calls, width / 2, height / 2);
		// reference point is the lower left corner for drawstring
		// places string starting at (3rd param, 4 param) coordinates
		// calls increase after the screen repaints bc of resizing
	}
}
```