import javax.swing.*;  // Packages used
import java.awt.*;
import java.awt.event.*;
import javax.swing.JScrollBar;
import java.lang.Math;
import java.awt.Graphics;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class functionApp extends JFrame implements AdjustmentListener, ActionListener
{
    Scrollbar cwSB = new Scrollbar(Scrollbar.VERTICAL, 0,1,0,25);
    private JPanel p1;
    private JPanel p2;
    private JPanel p3;
    private JPanel p4;
    private int freq = 10;
    private JButton startButton;

    public functionApp(String s)
    {
        super(s);

        getContentPane().setLayout( new GridLayout(2,2,1,1) );

        p1 = new JPanel();
        p1.setBackground(Color.cyan);
        JPanel p2 = new JPanel();
p2.setBackground(Color.cyan);
cwSB.addAdjustmentListener(this);
p2.add(cwSB);
JPanel p3 = new JPanel();
p3.setBackground(Color.cyan);
JPanel p4 = new JPanel();
p4.setBackground(Color.cyan);
startButton = new JButton("Start");
startButton.addActionListener(this);
p4.add(startButton);

getContentPane().add(p1);
getContentPane().add(p2);
getContentPane().add(p3);
getContentPane().add(p4);
}
public void actionPerformed(ActionEvent e)
{
    startButton.setText("Restart");
p1.repaint();
cWave cw = new cWave();
cw.start();
}

public static void main(String[] args)
{
    functionApp frame = new functionApp("Function App");
    frame.setSize(700,500);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
public void adjustmentValueChanged(AdjustmentEvent e) {
    if (e.getSource() == cwSB) {
        freq = cwSB.getValue();
    }
}

public class cWave extends Thread {
    public void run() {
        Graphics g = p1.getGraphics();
        g.setColor(Color.red);
        int y = 100;
        int i;
        for (i=0; i<500;i++) {
            // add 10ms delay between each drawn dot in curve
            try {
                Thread.sleep(10);
            } catch(InterruptedException ie){}
            drawLine method requires arguments x1,y1,x2,y2 as integer inputs
        }
    }
}
// cosine frequency is multiplied by 10
// cosine value is multiplied by 100 prior to casting as integer
// 100 is added to y1, y2 to shift downward on screen
    g.drawLine(i-1,y+100,i,(int)(100*Math.cos((freq*2*Math.PI/360.0)*i))+100);

    // Print out calculated cosine value to console
    System.out.println(100*Math.cos((freq*Math.PI/360.0)*i));

    // set previous value of y to current cosine calculation
    y = (int)(100*Math.cos((freq*2*Math.PI/360.0)*i));
    } //end of for-loop
    } //end of run
} //end of inner-class cWave
}