Lab Exercise: Java Programming Warm up

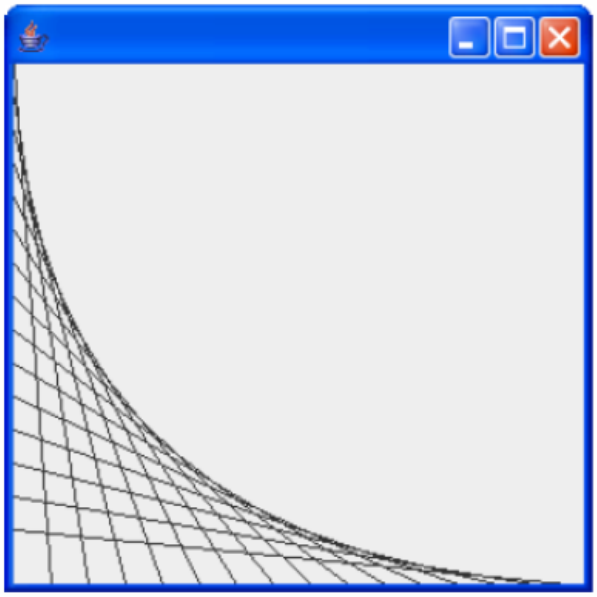
Student 1		Student 2	
Name	**CSUSM account ID**	**Name**	**CSUSM account ID**
Lauren Gonzalez	gonza823	Sirena Murphree	murph135

Lab Objectives

In this lab, you will learn/recall basic Java programming and understand how the GUI event dispatch thread works.

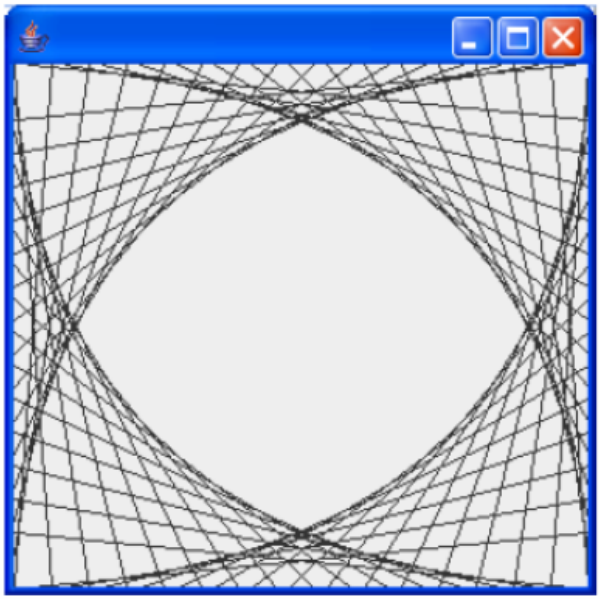
Problem Description

a. The example code given in this lab can produce the outcome as shown in the figure below.



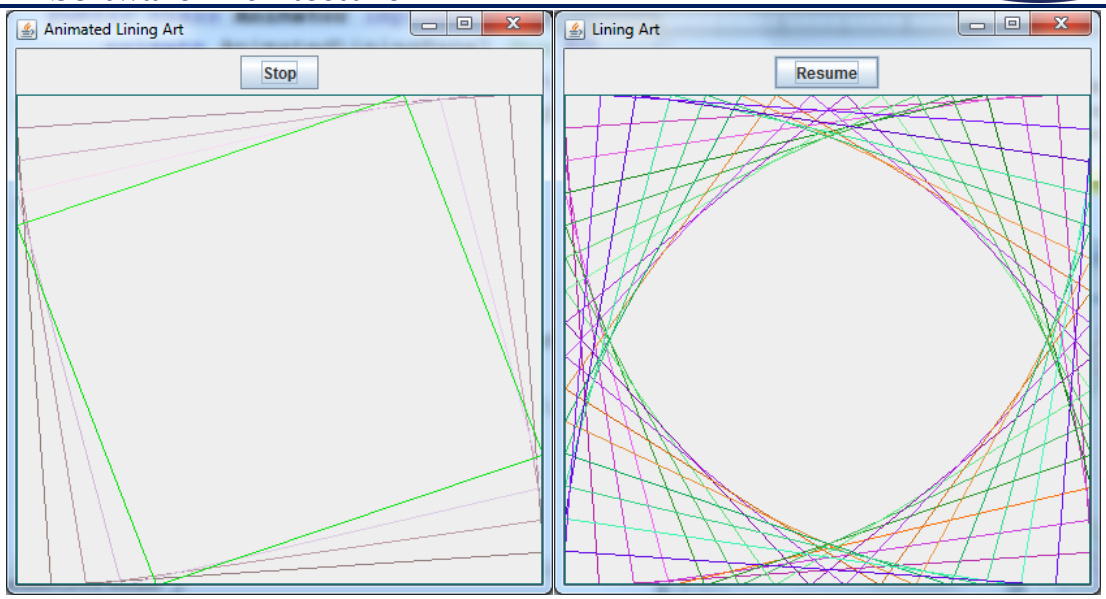
b. Study the code, make sure you understand how a JPanel object is used, on which lines are drawn. Each edge is divided into an equal number of increments (say, 15). The first line starts in the top-left corner and ends one step right on the bottom edge. For each successive line, move down one increment on the left edge and right one increment on the bottom edge. Continue drawing lines until you reach the bottom-right corner. The figure should scale as you resize the window so that the endpoints always touch the edges.

c. Modify the code to mirror the design in all four corners, as shown in the figure below. [**60 points**]



d. Add random colors to the lines [**20 points**]

e. Use a Runnable thread (you should take time to learn how to use the Runnable interface to create threads) to animate the drawing of lines, and add a button so that it can be pressed at any time to stop and resume the drawing (the text of the button should be able to change to reflect the current situation: say, from "draw" to "stop" to "resume"). [**20 points**]

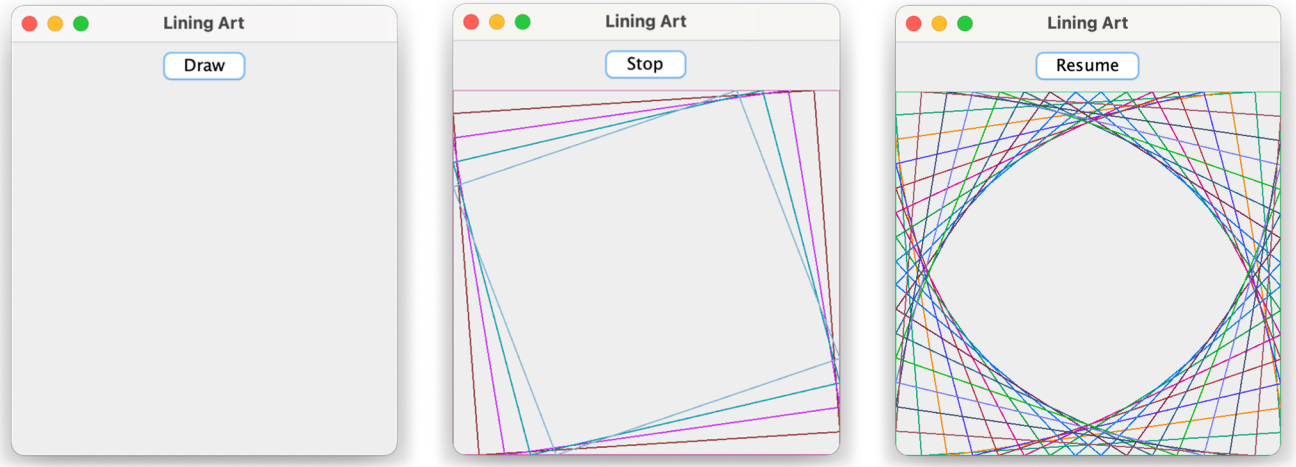


Submission:

- First, zip the src folder of your project and submit the zip file to the ungraded assignment named "Lab0CodeSubmission". One submission from each team.

- Paste all you source code here **inside this lab report, so that I could comment on your code if applicable.**
- **Paste a screenshot of a run of your program (like the ones given above) inside this lab report.**
- Save this report in PDF, then **each student** needs to submit the pdf report to the graded assignment named "Lab0ReportSubmission".

Screen shots



Code

LineDrawingTest.java

```
/**
 * SE471 – Lab0
 * Purpose:
 *  1)  learn/recall basic Java programming and understand how the GUI event dispatch thread
works.
 *  2)  Use a Runnable  thread
 *
 * Lauren Gonzalez  gonza823
 * Sirena Murphree  murph135
 */
package LineDrawing;
import javax.swing.JFrame;
import java.awt.BorderLayout;

public class LineDrawingTest {

    public static void main(String[] args) {

        JFrame application = new JFrame();
        LiningPanel panel = new LiningPanel();
        FlipBook animation = new FlipBook(panel);

        application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        application.getContentPane().add(new ControlPanel(animation), BorderLayout.NORTH);
        application.getContentPane().add(panel,  BorderLayout.CENTER);
        application.setSize(300, 350);
        application.setTitle("Lining Art");
        application.setVisible(true);

        animation.run();
    }
}
```

LiningPanel.java

```
package LineDrawing;
import java.awt.Color;
import java.awt.Graphics;
import java.util.Random;
import javax.swing.JPanel;

public class LiningPanel extends javax.swing.JPanel{
    /**
     * const LIMIT
     * the upper limit of the nmber of lines to be drawn
     */
    private static final double LIMIT = 15.0;

    /**
     * the current number of lines to be drawn
     */
    private int lines;

    /**
     * array<Color>
     */
    private Color rgbColor[];

    /**
     * Constructor
     */
    public LiningPanel() {
        lines = 0;
        rgbColor = new Color[(int)LIMIT+1];
        newRandomColor();
    }

    /**
```

```
     * incriments the number of lines to be drawn, resets when the limit is hit
     * invokes newRandomColor()
     */
    public void moreLines(){
        lines = (int)((++lines)%(LIMIT+1.0));
        newRandomColor();
    }

    /**
     * makes a new color at the index of lines
     */
    private void newRandomColor(){
        Random random = new Random();
        int red = random.nextInt(255);
        int green = random.nextInt(255);
        int blue = random.nextInt(255);
        rgbColor[lines] = new Color(red, green, blue);
    }

    @Override
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        int w = getWidth();
        int h = getHeight();

        for(int i = 0; i < lines; i++)
        {
            int w2 = (int)((i/LIMIT)*w);
            int h2 = (int)((i/LIMIT)*h);

        //d.    Add random colors to the lines
            g.setColor(rgbColor[i]);

        // c.   Modify the code to mirror the design in all four corners, as shown in the
figure below
        //  g.drawLine(x1, y1, x2, y2) <-- order of lines
            g.drawLine(0,  h2, w2, h);
            g.drawLine(w2,  h, w, h-h2);
            g.drawLine(w,  h-h2, w-w2, 0);
            g.drawLine(w-w2, 0, 0, h2);
        }
    }
}
```

ControlPanel.java

```
package LineDrawing;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;

public class ControlPanel extends javax.swing.JPanel implements ActionListener {

    private FlipBook animation;
    private JButton controlButton;

    /**
     * Constructor
     * @param itemBeingControlled
     *      Runnable object that animates a JPanel
     */
    public ControlPanel(FlipBook itemBeingControlled) {
        animation = itemBeingControlled;
        controlButton = new JButton("Draw");
        animation.setStopped(true);

        controlButton.addActionListener(this);
        this.add(controlButton);
    }

    /**
     * Control Button:
     * onClick the animation will resume if stopped or stop if running
     */
    @Override
    public void actionPerformed(ActionEvent e) {
        if(animation.isStopped()){
            animation.setStopped(false);
            controlButton.setText("Stop");
        }else{
            animation.setStopped(true);
            controlButton.setText("Resume");
        }
    }
}
```

FlipBook.java

```
package LineDrawing;

public class FlipBook implements Runnable{
    private LiningPanel sketch;
    private boolean stopped;

    /**
     * constructor
     * @param panelToBeSketched
     * the Jpanel object
     */
    public FlipBook(LiningPanel panelToBeSketched){
        sketch = panelToBeSketched;
        stopped = false;
    }

    /**
     * see if the FlipBook animation is running or stopped
     * @return stopped
     *      true = the FlipBook animation is NOT running
     *      false = the FlipBook animation IS running
     */
    public boolean isStopped() {
        return stopped;
    }

    /**
     * indicate to stop or run the FlipBook animation
     * @param stopped
     *      true = STOP the FlipBook animation
     *      false = RESUME the FlipBook animation
     */
    public void setStopped(boolean stopped) {
        this.stopped = stopped;
    }

    /**
     *  runs the FlipBook animation
     */
    @Override
    public void run() {
        while(true){
            if(!stopped){
                sketch.moreLines();
```

```
                sketch.repaint();
            }
            try {
                Thread.sleep(125);
            } catch (Exception e) {}
        }
    }
}
```