SE 471 Software Architecture

REVIEWED
By Simon at 8:53 am, Feb 03, 2023

Lab Exercise: Java Programming Warm up

Student 1		Student 2	
Name	**CSUSM account ID**	**Name**	**CSUSM account ID**
EJ Lilagan	200413348	**Dalynna Nguyen**	200982020

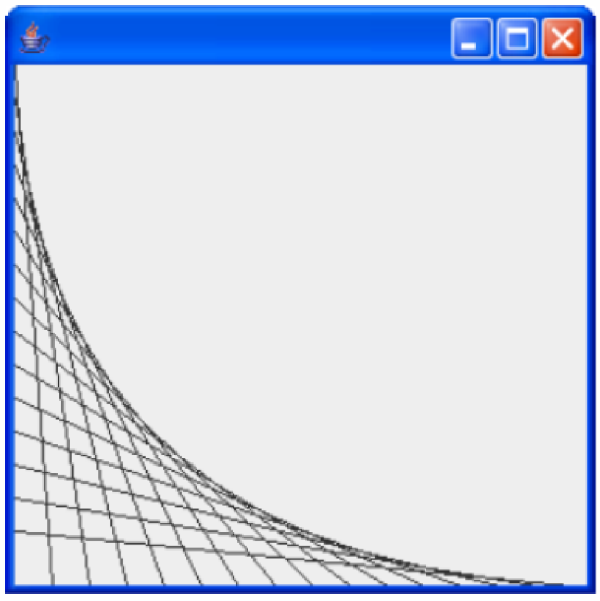
100

Lab Objectives

In this lab, you will learn/recall basic Java programming and understand how the GUI event dispatch thread works. Only two students are allowed to work together in this lab. Try to practice pair programming if possible.

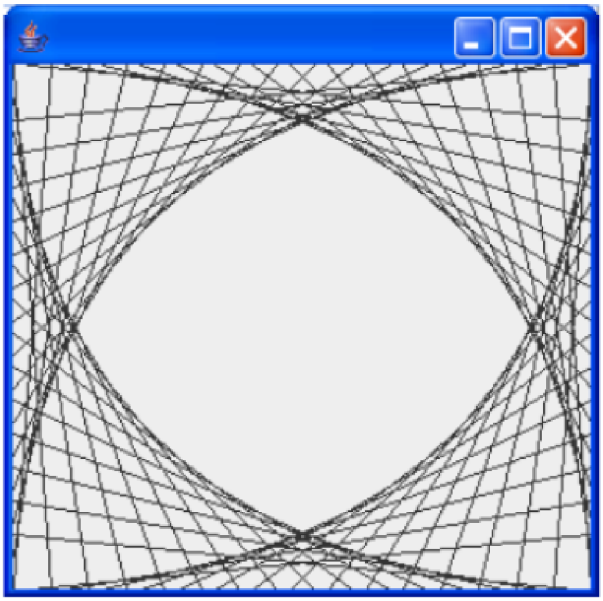
Problem Description

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

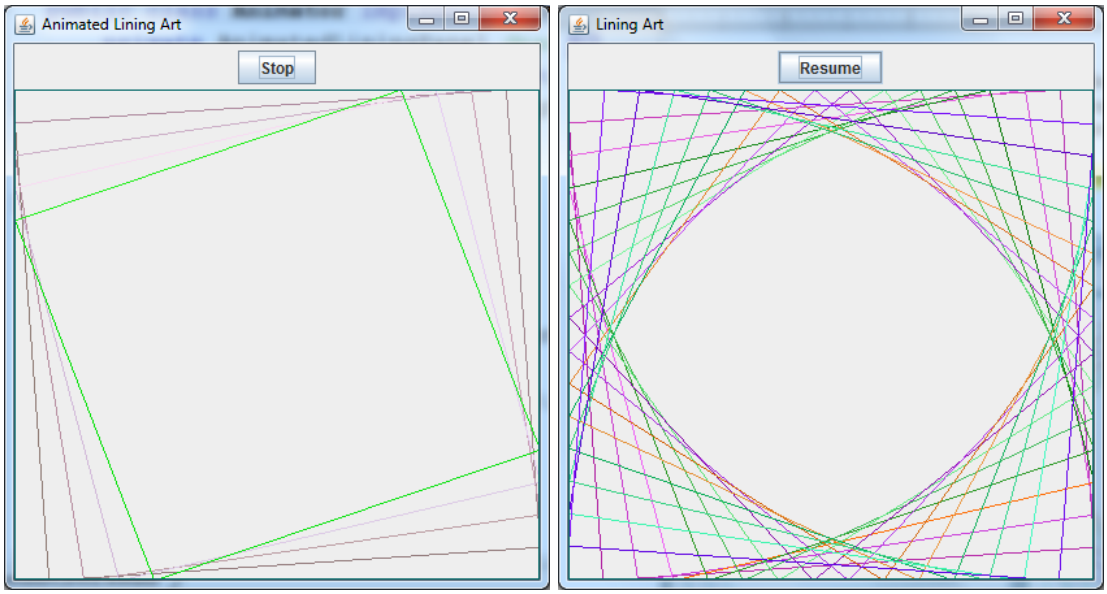


b. Study the code, make sure you understand how a JPanel object is used, on which lines are drawn. Each edge is divided into an equal number of increments (say, 15). The first line starts in the top-left corner and ends one step right on the bottom edge. For each successive line, move down one increment 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**]



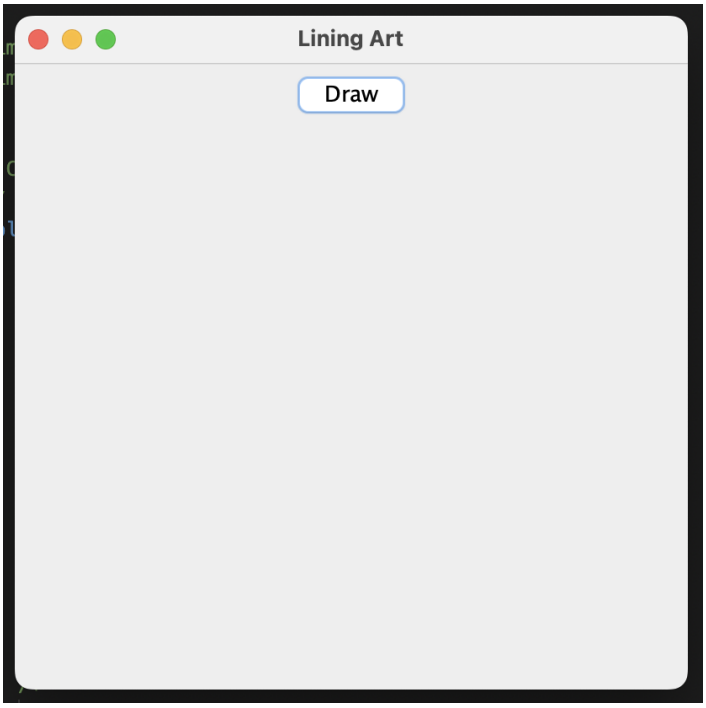
d. Add random colors to the lines [**20 points**]
e. Use a Runnable thread (you should take time to learn how to use the Runnable interface to create threads) to animate the drawing of lines, and add a button so that it can be pressed at any time to stop and resume the drawing (the text of the button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

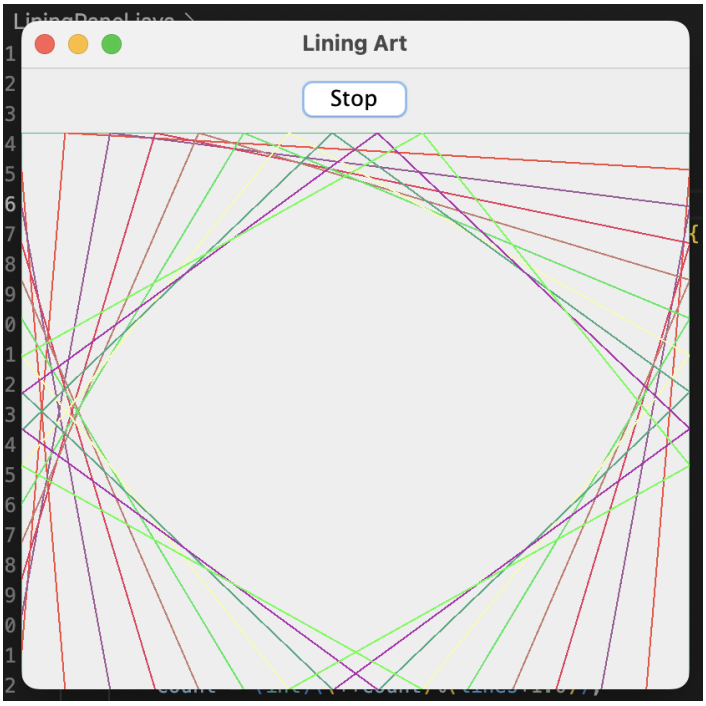


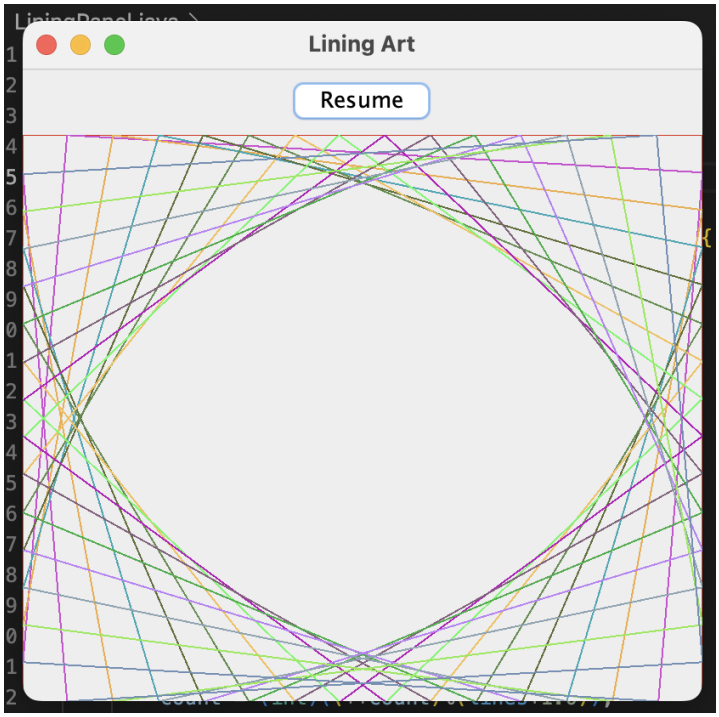
Submission:

- 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 your 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".

Screenshots (Draw -> Stop -> Resume)







Source Code:

Animator:

```
public class Animator implements Runnable{
   private LiningPanel dpanel;
   private boolean stop;
   public Animator(LiningPanel panel){
       dpanel = panel;
       stop = false;
   }
   public boolean doStop(){
       return stop;
   }
   public void set_stop(boolean stop2){
       this.stop = stop2;
   }
   @Override
   public void run() {
       while(true){
           if(!stop){ //if it is still true (or running)
               dpanel.counter(); //implemented from LiningPanel.java
               dpanel.repaint(); //not implemented
           }
           try {
               Thread.sleep(300);
           } catch (InterruptedException e) {
               e.printStackTrace();
           }
       }
   }
}
```

Button Control:

```
public class button_control extends javax.swing.JPanel implements ActionListener {
   private Animator animate;
   private JButton control_button;
   public button_control(Animator control) {
       animate = control;
       control_button = new JButton("Draw");

       animate.set_stop(true);

       control_button.addActionListener(this);
       this.add(control_button);

   }
   @Override
   public void actionPerformed(ActionEvent e) {
       if (animate.doStop()) {
           animate.set_stop(false);
           control_button.setText("Stop");
       } else {
           animate.set_stop(true);
           control_button.setText("Resume");
       }
   }
}
```

Line Drawing Test:

```
public class LineDrawingTest {

   public static void main(String[] args) {
       /* Given */
       JFrame application = new JFrame();
       LiningPanel panel = new LiningPanel();
       Animator animation = new Animator(panel);

       application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       application.getContentPane().add(new button_control(animation), BorderLayout.NORTH);
       application.getContentPane().add(panel, BorderLayout.CENTER);
       application.add(panel);
       application.setSize(400, 400);
       application.setTitle("Lining Art");
       application.setVisible(true);
       animation.run();
   }

}
```

Lining Panel:

```
public class LiningPanel extends javax.swing.JPanel {
   private Color colorArray[];
   private static final double lines = 15.0;
   private int count;

   public LiningPanel() {
       count = 0;
       colorArray = new Color[(int)lines+1];

       colors();
   }

   public void counter(){
       count = (int)((++count)%(lines+1.0));
       colors();
   }

   private void colors() {
       Random rd = new Random();
       int c1 = rd.nextInt(255); //color 1
       int c2 = rd.nextInt(255); //color 2
       int c3 = rd.nextInt(255); //color 3
       colorArray[count] = new Color(c1,c2,c3);
   }

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

       //double lines = 15.0; //increment the spaces
       for(int i = 0; i < count; i++)
       {
```

```
            int w2 = (int)((i/lines)*w);
            int h2 = (int)((i/lines)*h);

            g.setColor(colorArray[i]);// setting color
            g.drawLine(0,  h2, w2, h); //bottom left
            g.drawLine(w2,  0, 0, h - h2); //top left
            g.drawLine(w,  h2, w2, 0); //top right
            g.drawLine(w - w2,  h, w, h2); //bottom right
        }
    }
}
```