Lab 09 – GUI Basics and Graphics

Objectives:

- Understand the hierarchy of Swing components
- Learn to control layout by using various layout managers
- Build simple graphical interface with common GUI components and
- Customize GUI components' properties
- Learn to paint on a GUI component directly

Java uses *LayoutManager* to manage GUI components' layout. A *LayoutManager* considers various factors of the graphical context and computes the proper size of a component in a container. Therefore, different *LayoutManager* use different algorithms to compute the proper size of a component based on but not limited to the following factors:

- The preferred size defined by a graphical component
- The size of any fonts involved
- The text that might appear in a component
- Settings such as number of rows or columns
- The amount of data in a component
- The look and feel in question
- Borders that have been applied to a component
- The layout manager that's managing the contents of a container
- The size of an image to be displayed in a component

1. To illustrate the above mentioned effect, create the three following classes *TestFlowLayout*, *TestGridLayout* and *TestBorderLayout* separately. Execute these three classes and inspect the output respectively. Although the buttons are set to 100x100 in all three examples, do you find any differences between them?

```
// Using BorderLayout
public class TestBorderLayout extends JFrame {

    public TestBorderLayout() {
        setLayout(new BorderLayout(5, 5));
        JButton jbt = new JButton("I am a Button");
        jbt.setPreferredSize(new Dimension(100, 100));
        add(jbt, BorderLayout.CENTER);
    }

    /** Main method */
    public static void main(String[] args) {
        TestBorderLayout frame = new TestBorderLayout();
        frame.setTitle("TestBorderLayout");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}


// Using FlowLayout
public class TestFlowLayout extends JFrame {

    public TestFlowLayout() {
        setLayout(new FlowLayout(FlowLayout.LEFT, 5, 5));
        JButton jbt = new JButton("I am a Button");
        jbt.setPreferredSize(new Dimension(100, 100));
        add(jbt);
    }

    /** Main method */
    public static void main(String[] args) {
```

```
        TestFlowLayout frame = new TestFlowLayout();
        frame.setTitle("TestFlowLayout");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}


// Using GridLayout
public class TestGridLayout extends JFrame {

    public TestGridLayout() {
        setLayout(new GridLayout(1, 0, 5, 5));
        JButton jbt = new JButton("I am a Button");
        jbt.setPreferredSize(new Dimension(100, 100));
        add(jbt);
    }

    /** Main method */
    public static void main(String[] args) {
        TestGridLayout frame = new TestGridLayout();
        frame.setTitle("TestGridLayout");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```

2. When designing User Interface, you very often need to customize your font style. The application relies on the installed font types on your system. To check this, you can use the class *GraphicsEnvironment* like below:

```
public class AvailableFonts {

    public static void main(String[] args) {
        GraphicsEnvironment e = GraphicsEnvironment.getLocalGraphicsEnvironment();
        String[] fontnames = e.getAvailableFontFamilyNames();

        for (int i = 0; i < fontnames.length; i++) {
            System.out.println(fontnames[i]);
        }
    }
}
```

Run this program to see what types of fonts are available in your system now.

3. Based on your observation, try to reproduce the following graphical user interface as similar as possible.

   Let's think about the following quesitons:
   - What are the components used?
   - How many sub-containers?
   - What are the layout managers used?

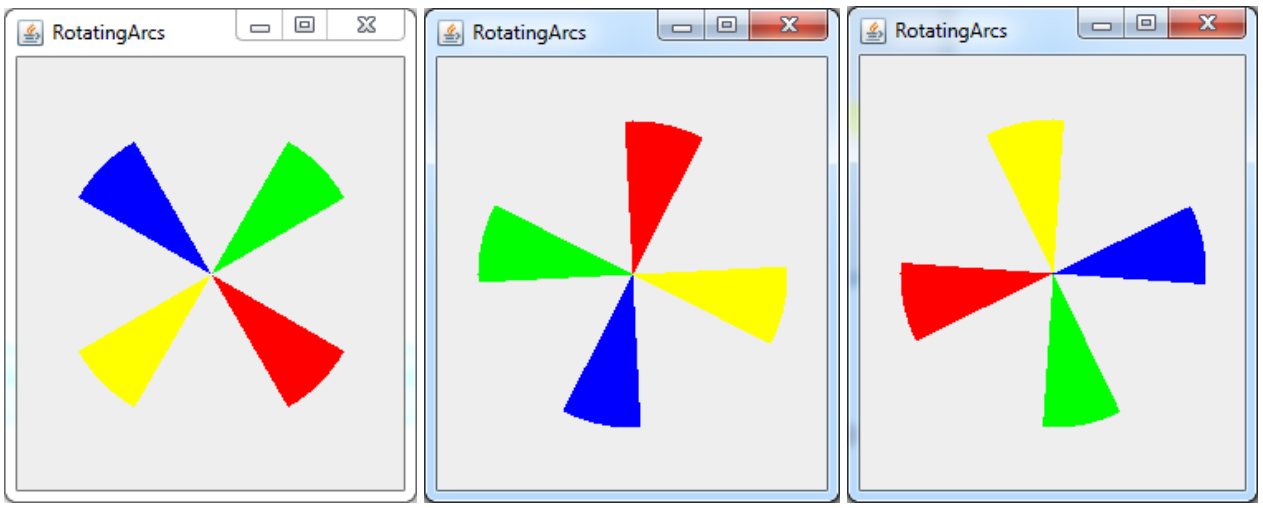
   Google logo source: https://www.google.com/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png

   You can use the ImageIcon class and URL class to load the Google logo image from Internet.

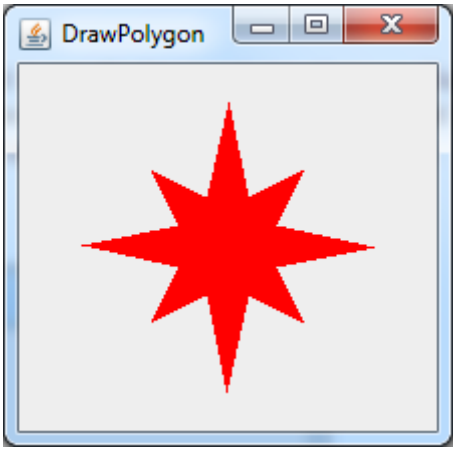
4. Based on the example program ***DrawArcs***, animate the "fan" and keep it rotating as shown. You may need to use the following APIs:

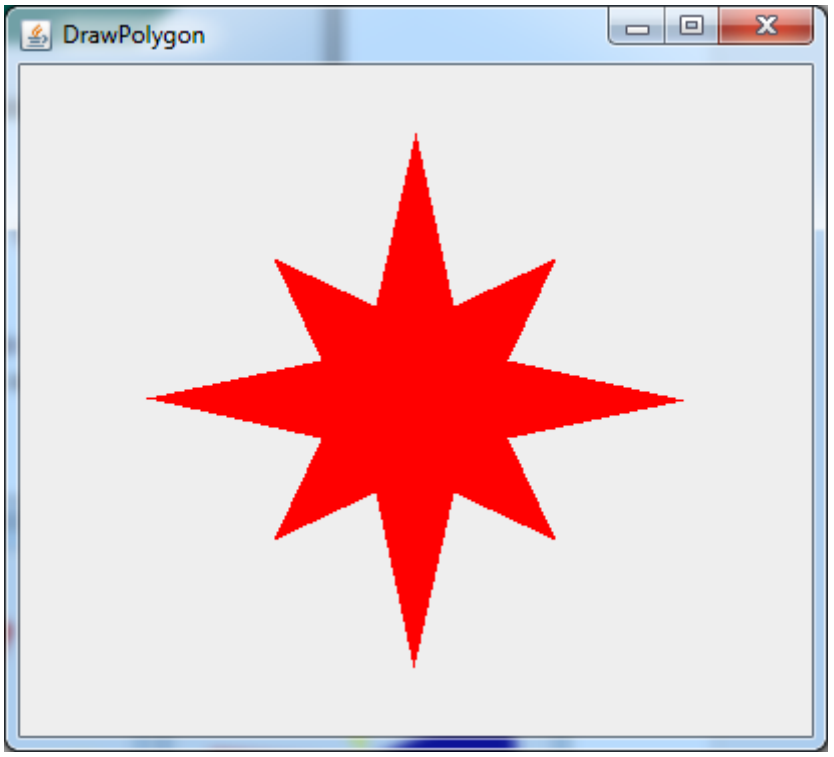
   `Thread.sleep(millisecond)` - pause the program for certain number of milliseconds; the pause time interval is used to control the refresh/repaint rate

   `JFrame.repaint()` - redraw or refresh the JFrame as well as its components



5. Revise the example program ***DrawPolygon*** such that it can display a red star as illustrated. The star should always be positioned in the center of the window and its size is dynamically scaled based on the window size. You should adjust its arms' length to make it look close to the shape shown below.





- END -