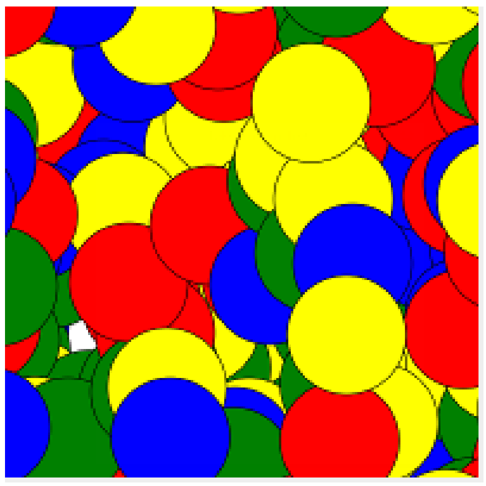
Object Oriented Programming with Java

Lab Practice:12

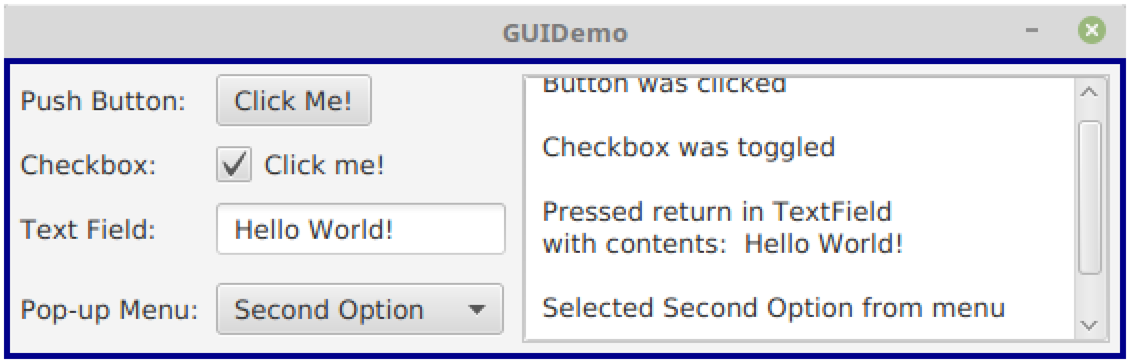
1. Draw a large number of randomly colored, randomly positioned, filled circles. For simplicity you can randomly select color red, green, blue or yellow. A simple switch statement can do the job.

```
switch (…….) {
  case 0:
    g.setFill( Color.RED );
    break;
  case 1:
    g.setFill( Color.GREEN );
    break;
  case 2:
    g.setFill( Color.BLUE );
    break;
  case 3:
    g.setFill( Color.YELLOW );
    break;
}
```

   I will choose the center points of the circles at random. Let's say that the width of the drawing area is given by a variable, width. Then we want a random value in the range 0 to width-1 for the horizontal position of the center. Similarly, the vertical position of the center will be a random value in the range 0 to height-1. That leaves the size of the circle to be determined; I will make the radius of each circle equal to 30 pixels. Th typical output of the program should be



2. Lets create a simple program that contains several GUI components that are available in the Java FX library. The program shows a window containing a button, a text input box, a choice box (pop-up menu), and a text area. The text area is used for a "transcript" that records interactions of the user with the other components.



```
Button was clicked

Checkbox was toggled

Pressed return in TextField
with contents:  Hello World!

Selected Second Option from menu
```

There are four components in the window with which the user can interact: a button, a checkbox, a text field, and a pop-up menu. The labels themselves are components (even though you can't interact with them). The right half of the window is a text area component, which can display multiple lines of text. A scrollbar component appears alongside the text area when the number of lines of text becomes larger than will fit in the text area. And in fact, in Java terminology, the whole window is itself considered to be a "component."

When a user interacts with GUI components, "events" are generated. For example, clicking a push button generates an event, and pressing a key on the keyboard generates an event. Each time an event is generated, a message is sent to the program telling it that the event has occurred, and the program responds according to its program. In fact, a typical GUI program consists largely of "event handlers" that tell the program how to respond to various types of events. In the above example, the program has been programmed to respond to each event by displaying a message in the text area. In a more realistic example, the event handlers would have more to do.

```
import javafx.application.Application;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.geometry.Insets;

public class GUIDemo extends Application {
    public static void main(String[] args) {
        launch(args);
    }

    //--------------------------------------------------------------------------------
    public void start(Stage stage) {

        GridPane root = new GridPane();
            // Put the transcript area in the right half of the
            // pane. The left half will be occupied by a grid of 4 rows
            // and two columns. Each row contains a component and
            // a label for that component.
.....
......
.......
//create different components and add them to the root.

}
}
```

**** This problem.2 seems more advanced than our lecture till now. However, try to solve as much as you can.**