Weekly Lab 8 – **Coloring a Movable Circle**

Maximum Points = 10

File *MoveCircle.java* contains a program that uses *CirclePanel.java* to draw a circle and let the user move it by pressing buttons. Save these files to your directory and compile and run MoveCircle to see how it works. Then study the code, noting the following:

- CirclePanel uses a BorderLayout so that the buttons can go on the bottom. But the buttons are not added directly to the south of the main panel—if they were they would all be on top of each other, and only the last one would show. Instead, a new panel *buttonPanel* is created and the buttons are added to it. *buttonPanel* uses a flow layout (the default panel layout), so the buttons will appear next to each other and centered. This panel is added to the south of the main panel.
- The listeners for the buttons are all instances of the MoveListener class, which is also defined here. The parameters to the constructor tell how how many pixels in the x and y directions the circle should move when the button is pressed.
- The circle is not drawn in the constructor, as it is not a component. It is drawn in paintComponent, which provides a graphics context for drawing on CirclePanel.
- In MoveCircle the frame size is explicitly set so there will be room to move the circle around.

Modify the program as follows.

1. Modify CirclePanel so that in addition to moving the circle, the user can press a button to change its color. The color buttons should be on the top of the panel; have four color choices. You will need to do the following:
   - Create a button for each color you want to provide, and label them appropriately.
   - Write a new listener class ColorListener whose constructor takes the color the circle should change to. When the button is pressed, just change the circle's color and repaint.
   - Create a new ColorListener for each color button, and add the listeners to the buttons.
   - Create a panel for the color buttons to go on, and add them to it.
   - Add the color panel to the north part of the main panel.

   You do not need to make any changes to MoveCircle.

2. Set the background or text (you choose) of each button to be the color that it represents.

3. Add another button to the top that says "Choose Color."
   - Place the button in the middle of your other color buttons.
   - When pressed, this button should bring up a JColorChooser, and the circle color should become the color that the user chooses.
   - You can use the same ColorListener class that you used for the other buttons; just pass *null* for the color when the user wants to choose their own, and in the *actionPerformed* method bring up a JColorChooser if the color is *null*.
     - Remember that the easiest way to use a JColorChooser is to call its static *showDialog* method, passing three parameters: the component to add it to (the "Choose Color" button), a string to title the chooser window, and a default color (the current circle color).
   - Note that the "Choose Color" button will have to be an instance variable (instead of being local to the CirclePanel constructor like the other buttons) to be visible in the listener.

```
// ********************************************************************
//   MoveCircle.java
//
//   Uses CirclePanel to display a GUI that lets the user move
//   a circle by pressing buttons.
// ********************************************************************

import java.awt.*;
import javax.swing.*;

public class MoveCircle
{
    //-----------------------------------
    //  Set up a frame for the GUI.
    //-----------------------------------
    public static void main(String[] args)
    {
      JFrame frame = new JFrame ("MoveCircle");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setPreferredSize(new Dimension(400,300));

      frame.getContentPane().add(new CirclePanel(400,300));

      frame.pack();
      frame.setVisible(true);
    }
}
```

```
// ********************************************************************
//   CirclePanel.java
//
//   A panel with a circle drawn in the center and buttons on the
//   bottom that move the circle.
// ********************************************************************
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class CirclePanel extends JPanel
{
    private final int CIRCLE_SIZE = 50;
    private int x,y;
    private Color c;

    //-----------------------------------------------------------------
    // Set up circle and buttons to move it.
    //-----------------------------------------------------------------
    public CirclePanel(int width, int height)
    {
      // Set coordinates so circle starts in middle
      x = (width/2)-(CIRCLE_SIZE/2);
      y = (height/2)-(CIRCLE_SIZE/2);

      c = Color.green;

      // Need a border layout to get the buttons on the bottom
      this.setLayout(new BorderLayout());

      // Create buttons to move the circle
      JButton left = new JButton("Left");
      JButton right = new JButton("Right");
      JButton up = new JButton("Up");
      JButton down = new JButton("Down");

      // Add listeners to the buttons
      left.addActionListener(new MoveListener(-20,0));
      right.addActionListener(new MoveListener(20,0));
      up.addActionListener(new MoveListener(0,-20));
      down.addActionListener(new MoveListener(0,20));

      // Need a panel to put the buttons on or they'll be on
      // top of each other.
      JPanel buttonPanel = new JPanel();
      buttonPanel.add(left);
      buttonPanel.add(right);
      buttonPanel.add(up);
      buttonPanel.add(down);

      // Add the button panel to the bottom of the main panel
      this.add(buttonPanel, "South");

    }

    //-----------------------------------------------------------------
    // Draw circle on CirclePanel
```

```
    //-----------------------------------------------------------------
    public void paintComponent(Graphics page)
    {
      super.paintComponent(page);

      page.setColor(c);
      page.fillOval(x,y,CIRCLE_SIZE,CIRCLE_SIZE);
    }

    //-----------------------------------------------------------------
    // Class to listen for button clicks that move circle.
    //-----------------------------------------------------------------
    private class MoveListener implements ActionListener
    {
      private int dx;
      private int dy;

      //-----------------------------------------------------------------
      // Parameters tell how to move circle at click.
      //-----------------------------------------------------------------
      public MoveListener(int dx, int dy)
      {
          this.dx = dx;
          this.dy = dy;
      }

      //-----------------------------------------------------------------
      // Change x and y coordinates and repaint.
      //-----------------------------------------------------------------
      public void actionPerformed(ActionEvent e)
      {
          x += dx;
          y += dy;
          repaint();
      }
    }
}
```

(Due before end of the day on Friday, October 8, 2010) Submit your .java files containing your program to the dropbox in WebCT.

Grades are determined using the following scale:

- Runs correctly.....:___/3
- Correct output.....:___/2
- Design of output.....:___/1
- Design of logic.....:___/2
- Standards.....:___/1
- Documentation.....:___/1

Grading Rubric (Word document)