Weekly Lab 4 – **Rebound Revisited**

Maximum Points = 10

The files *Rebound.java* and *ReboundPanel.java* contain the program. This program has an image that moves around the screen, bouncing back when it hits the sides (you can use any GIF or JPEG you like). Save these files to your directory, then open *ReboundPanel.java* in the editor and observe the following:

- The constructor instantializes a Timer object with a delay of 20 (the time, in milliseconds, between generation of action events). The Timer object is started with its *start* method.
- The constructor also gives the initial position of the ball (*x* and *y*) and the distance it will move at each interval (*moveX* and *moveY*).
- The *actionPerformed* method in the *ReboundListener* inner class "moves" the image by adding the value of *moveX* to *x* and of *moveY* to *y*. This has the effect of moving the image to the right when *moveX* is positive and to the left when *moveX* is negative, and similarly (with down and up) with *moveY*. The *actionPerformed* method then checks to see if the ball has hit one of the sides of the panel—if it hits the left side (x <= 0) or the right side (x >= WIDTHIMAGE_SIZE) the sign of *moveX* is changed. This has the effect of changing the direction of the ball. Similarly the method checks to see if the ball hit the top or bottom.

Now do the following:

1. First experiment with the speed of the animation. This is affected by two things—the value of the DELAY constant and the amount the ball is moved each time.
   - Change DELAY to 100. Save, compile, and run the program. How does the speed compare to the original?
   - Change DELAY back to 20 and change *moveX* and *moveY* to 15. Save, compile, and run the program. Compare the motion to that in the original program.
   - Experiment with other combinations of values.
   - Change *moveX* and *moveY* back to 3. Use any value of DELAY you wish.
2. Now add a second image to the program by doing the following:
   - Declare a second ImageIcon object as an instance variable. You can use the same image as before or a new one.
   - Declare integer variables *x2* and *y2* to represent the location of the second image, and *moveX2* and *moveY2* to control the amount the second ball moves each time.
   - Initialize *moveX2* to 5 and *moveY2* to 8 (note—this image will have a different trajectory than the first—no longer a 45 degree angle).
   - In *actionPerformed*, move the second image by the amount given in the *moveX2* and *moveY2* variable, and add code to check to see if the second image has hit a side.
   - In *paintComponent*, draw the second image as well as the first.
   -
3. Compile and run the program. Make sure it is working correctly.

```
//********************************************************************
// Rebound.java Author: Lewis/Loftus
//
// Demonstrates an animation and the use of the Timer class.
//********************************************************************
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Rebound
{
//-----------------------------------------------------------------
// Displays the main frame of the program.
//-----------------------------------------------------------------
```

```
      public static void main (String[] args)
      {
            JFrame frame = new JFrame ("Rebound");
            frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
            frame.getContentPane().add(new ReboundPanel());
            frame.pack();
            frame.setVisible(true);
      }
}

//*****************************************************************
// ReboundPanel.java Author: Lewis/Loftus
//
// Represents the primary panel for the Rebound program.
//*****************************************************************
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ReboundPanel extends JPanel
{
      private final int WIDTH = 300, HEIGHT = 100;
      private final int DELAY = 20, IMAGE_SIZE = 35;
      private ImageIcon image;
      private Timer timer;
      private int x, y, moveX, moveY;

//-----------------------------------------------------------------
// Sets up the panel, including the timer for the animation.
//-----------------------------------------------------------------
      public ReboundPanel()
      {
            timer = new Timer(DELAY, new ReboundListener());
            image = new ImageIcon ("happyFace.gif");
            x = 0;
            y = 40;
            moveX = moveY = 3;
            setPreferredSize (new Dimension(WIDTH, HEIGHT));
            setBackground (Color.black);
            timer.start();
      }

//-----------------------------------------------------------------
// Draws the image in the current location.
//-----------------------------------------------------------------
      public void paintComponent (Graphics page)
      {
            super.paintComponent (page);
            image.paintIcon (this, page, x, y);
      }

//*****************************************************************
// Represents the action listener for the timer.
//*****************************************************************
      private class ReboundListener implements ActionListener
      {
```

```
//-------------------------------------------------------------
// Updates the position of the image and possibly the direction
// of movement whenever the timer fires an action event.
//-------------------------------------------------------------
            public void actionPerformed (ActionEvent event)
            {
                  x += moveX;
                  y += moveY;
                  if (x <= 0 || x >= WIDTH-IMAGE_SIZE)
                        moveX = moveX * -1;
                  if (y <= 0 || y >= HEIGHT-IMAGE_SIZE)
                        moveY = moveY * -1;
                  repaint();
            }
      }
}
```

(Due before end of the day on Friday, February 4, 2011) 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)