Solutions for Exercises Unit 7

This is a non-graded exercise which should be posted to your learning journal.

The *StopWatchLabel* component from section 12.4.5 in the textbook displays the text "Timing..." when the stopwatch is running. It would be nice if it displayed the elapsed time since the stopwatch was started. For that, you need to create a *Timer* (see section 6.5.1 in the textbook.) Add a Timer to the original source code, *StopWatchLabel.java* in the code directory, to drive the display of the elapsed time in seconds. Create the timer in the mousePressed() routine when the stopwatch is started. Stop the timer in the mousePressed() routine when the stopwatch is stopped. The elapsed time won't be very accurate anyway, so just show the integral number of seconds. You only need to set the text a few times per second. For the *Timer* method, use a delay of 200 milliseconds for the timer. Here is an applet that tests one possible solution to this exercise:

Discussion

This one is almost too easy. (The hardest part is that the stopwatch was referred to as a "timer" in the program, and this could be confused with a *Timer* object, so some of the old references to "timer" were changed to "stopwatch".)

An instance variable named timer of *type* Timer is added to the class. The timer is started when the stopwatch is started and is stopped when the stopwatch is stopped. Both of these things happen in the mousePressed() method. A new timer could have been created each time the stopwatch is started, but the solution reuses a single timer. The first time the stopwatch is started, a Timer object is created and started. After that, the same *Timer* object is simply restarted. You can tell the difference between these two cases since the first time, the timer will be null:

```
if (timer == null) {
   timer = new Timer(200,this);
   timer.start();
}
else
   timer.restart();
```

The class is declared to implement *ActionListener* so that it can respond to events from the timer. (As usual, it would probably be better style to create another object to do the listening.) The actionPerformed() method just has to set the text on the label to show how much time has passed since the stopwatch was started. The starting time of the stopwatch is in the instance variable startTime. The current time is given by System.currentTimeMillis(), so the elapsed time, in milliseconds, is just System.currentTimeMillis([illegible]rtTime. This has to be divided by 1000 to give the number of seconds. (Remember that dividing an integer by an integer always gives an integer. The answer is truncated by dropping the fractional part, giving the integral number of seconds.) The actionPerformed() method simply does this calculation and then sets the text of the label.

The complete source code is shown below, followed by the source code for the little applet that tests the component. The applet sets the fonts and colors of the stopwatch component.

The Solution

The timer component, with changes from the original shown in red:

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

/**
 * A custom component that acts as a simple stop-watch.  When the user clicks
 * on it, this component starts timing.  When the user clicks again,
 * it displays the time between the two clicks.  Clicking a third time
 * starts another timer, etc.  While it is timing, the label displays
 * the integral number of seconds since it was started.
 */
public class StopWatchLabel2 extends JLabel
                           implements MouseListener, ActionListener {

  private long startTime;   // Start time of timer.
                           //   (Time is measured in milliseconds.)

  private boolean running;  // True when the timer is running.

  private Timer timer;      // Generates events that cause the component
                           // to be repainted periodically while the
                           // stop watch is running.

  /**
   * Constructor sets initial text on the label to
   * "Click to start timer." and sets up a mouse listener
   * so the label can respond to clicks.
   */
  public StopWatchLabel2() {
    super("  Click to start timer.  ", JLabel.CENTER);
    addMouseListener(this);
  }


  /**
   * Tells whether the timer is currently running.
   */
  public boolean isRunning() {
    return running;
  }


  /**
   * This will be called when an event from the timer is received.  It just
   * sets the stop watch to show the amount of time that
   * it has been running. Time is rounded down to the nearest second.
   */
  public void actionPerformed(ActionEvent evt) {
    long time = (System.currentTimeMillis() - startTime) / 1000;
    setText("Running:  " + time + " seconds");
   }


  /**
   * React when the user presses the mouse by starting
   * or stopping the stop watch and changing the text that
   * is shown on the label.
   */
  public void mousePressed(MouseEvent evt) {
    if (running == false) {
         // Record the time and start the stop watch.
      running = true;
      startTime = evt.getWhen();  // Time when mouse was clicked.
      setText("Running:  0 seconds");
      if (timer == null) {
         timer = new Timer(100,this);
         timer.start();
      }
```

```
         else
            timer.restart();
      }
      else {
            // Stop the stop watch.  Compute the elapsed time since the
            // stop watch was started and display it.
         timer.stop();
         running = false;
         long endTime = evt.getWhen();
         double seconds = (endTime - startTime) / 1000.0;
         setText("Time: " + seconds + " sec.");
      }
   }

   public void mouseReleased(MouseEvent evt) { }
   public void mouseClicked(MouseEvent evt) { }
   public void mouseEntered(MouseEvent evt) { }
   public void mouseExited(MouseEvent evt) { }

}
```

The code for the applet that tests the component:

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

/**
 * A trivial applet that tests the StopWatchLabel2 component.
 * The applet just creates and shows a StopWatchLabel2.
 */

public class TestStopWatch2 extends JApplet {

   public void init() {

      StopWatchLabel2 watch = new StopWatchLabel2();
      watch.setFont( new Font("SansSerif", Font.BOLD, 24) );
      watch.setBackground(Color.WHITE);
      watch.setForeground( new Color(180,0,0) );
      watch.setOpaque(true);
      getContentPane().add(watch, BorderLayout.CENTER);

   }

}
```

Last modified: Wednesday, 13 May 2020, 4:20 PM