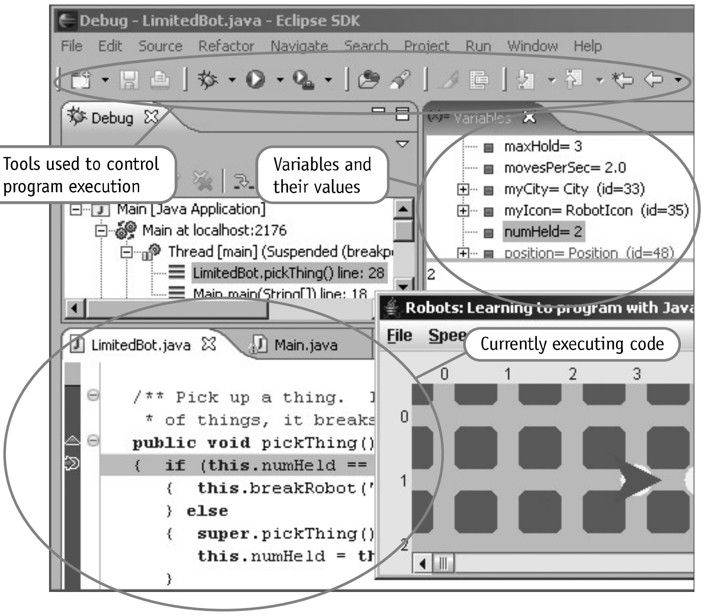
to be shown as well. The debugger even shows private instance variables in `LimitedBot`'s superclasses.

➤ After a program stops at a breakpoint, the toolbar shown in the upper-left corner of Figure 6-12 is used to continue execution. For example, the arrow on the far left continues execution until the next breakpoint is reached. Some of the other tools allow the programmer to step to the next statement. One tool treats a method call as one statement to execute while another steps into a method to execute the next statement.


Debug - LimitedBot.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Variables
Tools used to control program execution
Variables and their values
Currently executing code
LimitedBot.java
Main.java
Robots: Learning to program with Java
File Spee
0 1 2 3
0 1 2

```
Main [Java Application]
  Main at localhost:2176
    Thread [main] (Suspended (breakp
      LimitedBot.pickThing() line: 28
      Main.main(String[]) line: 18
```

```
maxHold= 3
movesPerSec= 2.0
myCity= City (id=33)
myIcon= RobotIcon (id=35)
numHeld= 2
position= Position (id=48)
2
```

```
/** Pick up a thing.  I
 * of things, it breaks
public void pickThing()
{  if (this.numHeld ==
   {  this.breakRobot('
   } else
   {  super.pickThing()
      this.numHeld = th
   }
```

(figure 6-12)

Eclipse debugger in use

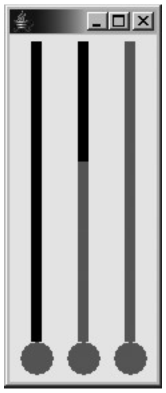
Debuggers are powerful tools that are worth learning. However, they are also complex and may distract beginning programmers from more important learning tasks.

6.7 GUI: Repainting

In this section, we'll create a new kind of graphical user interface component, a `Thermometer`. The major difference between a `Thermometer` and the `StickFigure` component created in Chapter 2 is that the `Thermometer` has an instance variable that controls its appearance. As Figure 6-13 shows, each thermometer can be set to show a different temperature.

(figure 6-13)

Three `Thermometer` *components, each with a different temperature setting*



The program in Listing 6-14 was used to create this image and may be used as a test harness during the development process. It creates three instances of the `Thermometer` class, displays them, and sets each to show a different temperature.

FIND THE CODE

ch06/thermometer/

Listing 6-14: *A test harness for the* `Thermometer` *class*

```java
import javax.swing.*;

/** Test a thermometer component.
 *
 * @author Byron Weber Becker */
public class Main extends Object
{
  public static void main(String[] args)
  { // Create three thermometer components.
    Thermometer t0 = new Thermometer();
    Thermometer t1 = new Thermometer();
    Thermometer t2 = new Thermometer();

    // Create a panel to hold the thermometers.
    JPanel contents = new JPanel();
    contents.add(t0);
    contents.add(t1);
    contents.add(t2);

    // Set up the frame.
    JFrame f = new JFrame();
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setContentPane(contents);
    f.pack();
    f.setVisible(true);

```

Listing 6-14: *A test harness for the* `Thermometer` *class* (continued)

```
      // Set the temperature of each thermometer.
      t0.setTemperature(0);
      t1.setTemperature(30);
      t2.setTemperature(50);
   }
}
```

6.7.1 Instance Variables in Components

In Section 2.7.3, we learned that the `paintComponent` method can be called by the Java system at any time. The user can resize a frame or expose a previously hidden frame. In either case, `paintComponent` will be called to repaint the contents of the frame. Therefore, the `paintComponent` method must be able to determine what the component should look like. For a `Thermometer`, this includes determining how high the alcohol (the modern replacement for mercury) should be drawn. It does so by consulting an instance variable. A client can set the instance variable to a given temperature with a small method called `setTemperature`.

Listing 6-15 shows the beginnings of the `Thermometer` class, complete with the instance variable used to store the current temperature. Most of the code in `paintComponent` must still be developed.

FIND THE CODE

ch06/thermometer/

Listing 6-15: *The beginnings of the* `Thermometer` *class*

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

/** A thermometer component to use in graphical user interfaces. It can
 *  display temperatures from MIN_TEMP to MAX_TEMP, inclusive.
 *
 *  @author Byron Weber Becker */
public class Thermometer extends JComponent
{
  public final int MIN_TEMP = 0;
  public final int MAX_TEMP = 50;
  private int temp = MIN_TEMP;

```

Listing 6-15: *The beginnings of the* `Thermometer` *class* (continued)

```
14    /** Construct a new thermometer. */
15    public Thermometer()
16    { super();
17      this.setPreferredSize(new Dimension(50, 250));
18    }
19
20    /** Paint the thermometer to show the current temperature. */
21    public void paintComponent(Graphics g)
22    { super.paintComponent(g);
23
24      // paint the thermometer
25    }
26
27     /** Set the thermometer's temperature.
28     *  @param newTemp the new temperature. */
29     public void setTemperature(int newTemp)
30     { this.temp = newTemp;
31     }
32  }
```

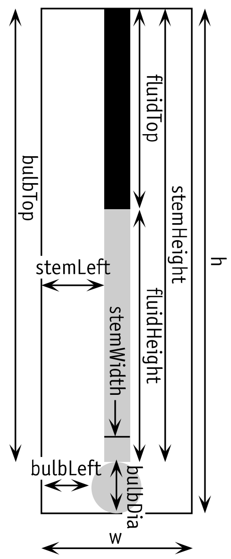
Recall that the preferred size, set in line 17, is used by the frame to determine how large the thermometer should be. Forgetting to set the preferred size will make the component so small that it is almost invisible.

LOOKING AHEAD

Programming Project 6.15 asks you to improve upon the hard-coded minimum and maximum.

This version of the class fixes the minimum and maximum temperature the thermometer can display with two named constants.

Working out the actual code for `paintComponent` is somewhat tedious. It helps to declare temporary variables initialized with significant values. The diagram in Figure 6-14 illustrates the meaning of those used in Listing 6-16.



(figure 6-14)

`Thermometer` *calculations*

The height and width of the component are found first in lines 5 and 6 and stored in variables to make using them more convenient. All the calculations should ultimately depend on the height and width so that the thermometer is drawn appropriately as the component is resized.

The variables with names ending in `Left` and `Top` hold values specifying the location of a shape. Variables with names ending in `Height`, `Width`, and `Dia` (short for "diameter") hold values specifying the size of a shape.

FIND THE CODE

ch06/thermometer/

Listing 6-16: *The finished implementation of* `paintComponent`

```
 1  /** Paint the thermometer to show the current temperature. */
 2  public void paintComponent(Graphics g)
 3  { super.paintComponent(g);
 4
 5    final int w = this.getWidth();
 6    final int h = this.getHeight();
 7
 8    final int bulbDia = h/10;
 9    final int bulbLeft = w/2 - bulbDia/2;
10    final int bulbTop = h - bulbDia;
11
12    final int stemWidth = bulbDia/3;
13    final int stemLeft = w/2 - stemWidth/2;
```

Listing 6-16: *The finished implementation of* `paintComponent` (continued)

```
  final int stemHeight = h - bulbDia;

  final int fluidHeight = stemHeight *
        (this.temp - MIN_TEMP) / (MAX_TEMP - MIN_TEMP);
  final int fluidTop = stemHeight - fluidHeight;

  // paint the fluid
  g.setColor(Color.RED);
  g.fillOval(bulbLeft, bulbTop, bulbDia, bulbDia);
  g.fillRect(stemLeft, fluidTop, stemWidth, fluidHeight);

  // paint the stem above the fluid
  g.setColor(Color.BLACK);
  g.fillRect(stemLeft, 0, stemWidth, fluidTop);
}
```

6.7.2 Triggering a Repaint

If you run the test harness with the current version of `Thermometer`, you will notice that the thermometers are painted as though the temperature is 0 rather than the temperatures set in the test harness. However, if you resize the frame, forcing the thermometers to be repainted, then they will be drawn with the correct temperatures. In other words, the thermometers display a temperature change only when they are repainted.

KEY IDEA

Call `repaint` *when instance variables affecting the image change.*

Somehow we need to be able to trigger the repainting of the component whenever the temperature changes. We do so with an inherited method, `repaint`. Calling `repaint` after we have reset the instance variable informs the Java system that it should call `paintComponent` as soon as possible. The revised version of `setTemperature` is:

```
public void setTemperature(int newTemp)
{ this.temp = newTemp;
  this.repaint();
}
```

6.7.3 Animating the `Thermometer`

Adding the following code to the end of the test harness will cause the thermometer to show a steadily increasing temperature—just like the temperature climbing on a hot summer's morning.

```
for(int temp = t0.MIN_TEMP; temp <= t0.MAX_TEMP; temp = temp
+ 1)
{  t0.setTemperature(temp);
   Utilities.sleep(50);
}
```

The call to `Utilities.sleep` causes the current thread to pause for 50 milliseconds, or 0.050 seconds, to give the Java system a chance to repaint the screen—and so you have time to see the change in the thermometer.

The `sleep` method should *not* be called inside the `paintComponent` method. `paintComponent` is called by the Java system; it has many important things to do and should not be forced to wait for anything.

6.8 Patterns

Every time you are writing an expression, you need values. These values could come from any of the constructs discussed in this chapter. In almost every situation, one of the constructs is a better choice than the others. Carefully consider which of the following patterns best describes your situation and is best suited to solve your problem.

6.8.1 The Named Constant Pattern

Name: Named Constant

Context: You have a literal value used one or more times in your program. The value is known when you write the program and does not change while the program is running.

Solution: Use a named constant, as suggested by the following examples:

```
private static final int DAYS_IN_WEEK = 7;
private static final int COST_PER_MOVE = 25;
```

In general, a named constant has the following form:

```
«accessModifier» static final «type» «name» = «value»;
```

where *«accessModifier»* is `public`, `protected`, or `private`. Use `private` if the value is used only within the class where it is defined. Use `public` if other classes might need it—for example, as an actual parameter to a method defined within the class.

«type» is the type of the value stored in the constant. So far, we have discussed only integers, but any type (including a class name) is possible. *«name»* is the name of the variable, and *«value»* is the first (and last) value assigned to it.

Graphics programs often use many constants in the course of drawing a picture. (See `paintComponent` in Section 2.7.3 for an example.) Having a named constant for each