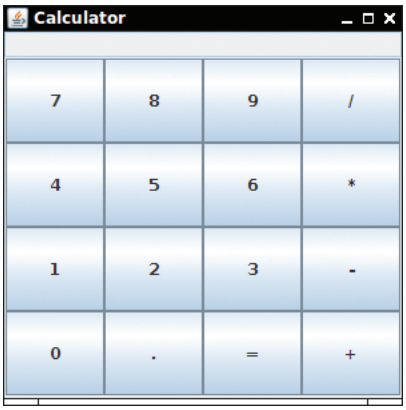
WORKED EXAMPLE 20.1 **Programming a Working Calculator**

Problem Statement Implement arithmetic and scientific operations for a calculator. Use the sample program from Section 20.1 as a starting point.

Arithmetic



In the calculator program of Section 20.1, the buttons for the arithmetic operations didn't do any work. It is actually a bit subtle to implement the behavior of a calculator. Imagine the user who has just entered `3 +`. At this point, we can't yet perform the addition because we don't have the second operand. We need to store the value (3) and the operator (+) and keep on going. Now the user continues:

```
3 + 4 *
```

As soon as the * button is clicked, we can get to work and *add* 3 and 4. That is, we take the saved value and the newly entered value, and combine them with the *saved* operator. Then we save the * so that it can be executed later.

(Here, we implement a common household calculator in which multiplication and addition have the same precedence. In Chapter 16, you saw how to implement a calculator in which multiplication has a higher precedence, as it does in mathematics.)

There is another subtlety, concerning the update of the calculator display. Consider the input

```
1 3 + 4 * 2 =
```

which arrives one button click at a time:

Button Clicked	Action	Display
1	Show 1 in display.	1
3	Add 3 to end of display.	13
+	Store 13 and + for later use.	13
4	Clear display, add 4.	4
*	Replace display with result of 13 + 4. Store 17 and * for later use.	17
2	Clear display, add 2.	2
=	Replace display with result of 17 * 2.	34

You may want to try this out with an actual calculator. Note the following:

- When an operator button is clicked and two operands are available, the display is updated with the result of the saved operation.

- The *first* digit button clicked after an operator clears the display. The other digit buttons append to the display. The display can't be cleared by the operator; it must be cleared by the first digit. (Otherwise, there would be no way for the user to see the result.)
- The = button puts the calculator into the same state as it was at the beginning, clearing the saved operation.

Now we have enough information to implement the arithmetic operator buttons. The calculator needs to remember

- the last value and operator.
- whether we are at the beginning or in the middle of entering a value.

We also need to remember the value that is currently being built up, but we can just take that from the `display` variable.

```
public class CalculatorFrame extends JFrame
{
   private JLabel display;
   . . .
   private double lastValue;
   private String lastOperator;
   private boolean startNewValue;

   public CalculatorFrame()
   {
      lastValue = 0;
      lastOperator = "=";
      startNewValue = true;
      . . .
   }
   . . .
}
```

The `actionPerformed` method of the digit button listeners appends the digit to the display; however, the display is cleared first if this is the first digit after an operator:

```
public void actionPerformed(ActionEvent event)
{
   if (startNewValue)
   {
      display.setText("");
      startNewValue = false;
   }
   display.setText(display.getText() + digit);
}
```

How does the method know which digit to use? It is passed to the constructor of the listener:

```
class DigitButtonListener implements ActionListener
{
   private String digit;

   public DigitButtonListener(String aDigit)
   {
      digit = aDigit;
   }
   . . .
}
```

We construct digit buttons with this helper method:

```
public JButton makeDigitButton(String digit)
{
   JButton button = new JButton(digit);
```

```
      ActionListener listener = new DigitButtonListener(digit);
      button.addActionListener(listener);
      return button;
   }
```

The helper method is called for each digit button:

```
   private void createButtonPanel()
   {
      JPanel buttonPanel = new JPanel();
      buttonPanel.setLayout(new GridLayout(4, 4));

      buttonPanel.add(makeDigitButton("7"));
      buttonPanel.add(makeDigitButton("8"));
      buttonPanel.add(makeDigitButton("9"));
      . . .
   }
```

We use the same strategy to pass the operator symbol to the `OperatorButtonListener`. Here is its `actionPerformed` method:

```
   public void actionPerformed(ActionEvent event)
   {
      if (!startNewValue)
      {
         double value = Double.parseDouble(display.getText());
         lastValue = calculate(lastValue, value, lastOperator);
         display.setText("" + lastValue);
         startNewValue = true;
      }

      lastOperator = operator;
   }
```

First, we check whether the operator follows a value. If a user clicked two operators in a row, as in `3 + * 4`, we assume that the intent was to replace an incorrectly entered operator.

In the normal case, we combine the last value with the display value, using the *last* operator. We update the display with the result, and get ready to receive the next value.

We also store the current operator so that it can be evaluated later.

The `calculate` method simply combines its inputs:

```
   public double calculate(double value1, double value2, String op)
   {
      if (op.equals("+"))
      {
         return value1 + value2;
      }
      else if (op.equals("-"))
      {
         return value1 - value2;
      }
      else if (op.equals("*"))
      {
         return value1 * value2;
      }
      else if (op.equals("/"))
      {
         return value1 / value2;
      }
      else // "="
      {
         return value2;
```

```
      }
   }
```

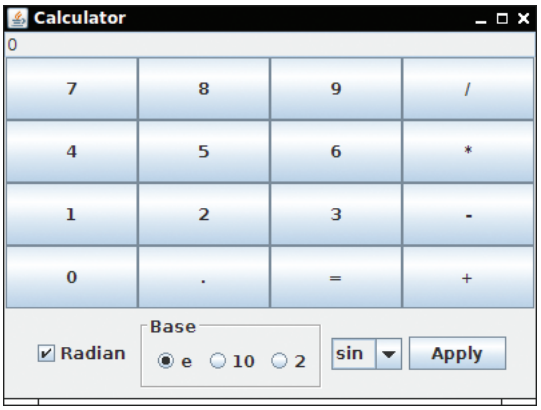
To understand the behavior for the = operator, think through an input `3 + 4 =` followed by `5 * 6`. When the = button is clicked, the last operator (+) is executed, and = becomes the last operator. When the * button is clicked, the `calculate` method receives the last value (7), the display value (5), and the last operator (=). It should simply return the second operand (5), which will later be combined with the 6.

This completes the implementation of the arithmetic operators.

Mathematical Functions

In order to practice working with user-interface components, we will enhance the calculator with a few mathematical functions. The trigonometric functions sin, cos, and tan take an argument that can be interpreted as radians or degrees. We provide a check box to select radians. (Perhaps two radio buttons for radians and degrees would be clearer, but we want to practice using a checkbox). For the log and exp functions, we provide radio buttons to select one of three bases: e, 10, and 2. We place the functions into a combo box.

Clicking the Apply button applies the selected function with the selected options.



First, we need to set up the user interface. We need

- A checkbox for radians
- Three radio buttons
- A button group for the radio buttons
- A border for the radio buttons
- A combo box for the functions
- An Apply button

Let's get the radio buttons out of the way first:

```
private JPanel createBaseButtons()
{
   baseeButton = new JRadioButton("e");
   base10Button = new JRadioButton("10");
   base2Button = new JRadioButton("2");

   baseeButton.setSelected(true);

   ButtonGroup group = new ButtonGroup();
   group.add(baseeButton);
   group.add(base10Button);
   group.add(base2Button);
```

```
      JPanel basePanel = new JPanel();
      basePanel.add(baseeButton);
      basePanel.add(base10Button);
      basePanel.add(base2Button);
      basePanel.setBorder(new TitledBorder(new EtchedBorder(), "Base"));

      return basePanel;
   }
```

Here we create three radio buttons, select one of them, and add them to a button group. Note that the buttons are instance variables—we need to query their state later. However, the button group is only used by the Swing library, not our program. Therefore, it can be a local variable.

Finally, we add the buttons into a panel so that we can apply a border.

The remainder of the user interface is simpler. We just need to add the checkbox, combo box, radio buttons, and Apply button to a panel, then add that panel to the southern area of the frame's border layout.

```
   private void createControlPanel()
   {
      radianCheckBox = new JCheckBox("Radian");
      radianCheckBox.setSelected(true);

      mathOpCombo = new JComboBox();
      mathOpCombo.addItem("sin");
      mathOpCombo.addItem("cos");
      mathOpCombo.addItem("tan");
      mathOpCombo.addItem("log");
      mathOpCombo.addItem("exp");

      mathOpButton = new JButton("Apply");
      mathOpButton.addActionListener(new MathOpListener());

      JPanel controlPanel = new JPanel();
      controlPanel.add(radianCheckBox);
      controlPanel.add(createBaseButtons());
      controlPanel.add(mathOpCombo);
      controlPanel.add(mathOpButton);

      add(controlPanel, BorderLayout.SOUTH);
   }
```

The only button that receives a listener is the Apply button. The other components change their state when they are clicked, and the listener of the Apply button reads the state when it calls the selected function.

Here is the listener code:

```
   class MathOpListener implements ActionListener
   {
      public void actionPerformed(ActionEvent event)
      {
         double value = Double.parseDouble(display.getText());
         String mathOp = (String) mathOpCombo.getSelectedItem();

         double base = 10;
         if (baseeButton.isSelected()) { base = Math.E; }
         else if (base2Button.isSelected()) { base = 2; }

         boolean radian = radianCheckBox.isSelected();
         if (!radian && (mathOp.equals("sin")
               || mathOp.equals("cos") || mathOp.equals("tan")))
         {
```

```
            value = Math.toRadians(value);
         }

         if (mathOp.equals("sin"))
         {
            value = Math.sin(value);
         }
         else if (mathOp.equals("cos"))
         {
            value = Math.cos(value);
         }
         else if (mathOp.equals("tan"))
         {
            value = Math.tan(value);
         }
         else if (mathOp.equals("log"))
         {
            value = Math.log(value) / Math.log(base);
         }
         else if (mathOp.equals("exp"))
         {
            value = Math.pow(base, value);
         }
         display.setText("" + value);

         startNewValue = true;
      }
   }
```

First, we get the function's argument from the display, and the base from the radio buttons. If we need to call a trigonometric function with degrees, we convert the argument to radians. That is what the Java library expects.

Then we execute the selected function and update the display. Finally, we set the `startNewValue` flag. If the user clicks a digit button, the display is cleared and the button becomes the first digit of a new value.

worked_example_1/CalculatorViewer.java

```
1  import javax.swing.JFrame;
2
3  public class CalculatorViewer
4  {
5     public static void main(String[] args)
6     {
7        JFrame frame = new CalculatorFrame();
8        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9        frame.setTitle("Calculator");
10       frame.setVisible(true);
11    }
12 }
```

worked_example_1/CalculatorFrame.java

```
1  import java.awt.BorderLayout;
2  import java.awt.GridLayout;
3  import java.awt.event.ActionListener;
4  import java.awt.event.ActionEvent;
5  import javax.swing.ButtonGroup;
6  import javax.swing.JButton;
7  import javax.swing.JCheckBox;
```

```
 8  import javax.swing.JComboBox;
 9  import javax.swing.JFrame;
10  import javax.swing.JPanel;
11  import javax.swing.JRadioButton;
12  import javax.swing.JLabel;
13  import javax.swing.border.EtchedBorder;
14  import javax.swing.border.TitledBorder;
15
16  /**
17     This frame contains a panel that displays buttons
18     for a calculator and a panel with a text fields to
19     specify the result of calculation.
20  */
21  public class CalculatorFrame extends JFrame
22  {
23     private JLabel display;
24     private JCheckBox radianCheckBox;
25     private JRadioButton baseeButton;
26     private JRadioButton base10Button;
27     private JRadioButton base2Button;
28     private JComboBox mathOpCombo;
29     private JButton mathOpButton;
30
31     private double lastValue;
32     private String lastOperator;
33     private boolean startNewValue;
34
35     private static final int FRAME_WIDTH = 400;
36     private static final int FRAME_HEIGHT = 300;
37
38     public CalculatorFrame()
39     {
40        createButtonPanel();
41        createControlPanel();
42
43        display = new JLabel("0");
44        add(display, BorderLayout.NORTH);
45
46        lastValue = 0;
47        lastOperator = "=";
48        startNewValue = true;
49
50        setSize(FRAME_WIDTH, FRAME_HEIGHT);
51     }
52
53     /**
54        Creates the control panel with the text field
55        and buttons on the frame.
56     */
57     private void createButtonPanel()
58     {
59        JPanel buttonPanel = new JPanel();
60        buttonPanel.setLayout(new GridLayout(4, 4));
61
62        buttonPanel.add(makeDigitButton("7"));
63        buttonPanel.add(makeDigitButton("8"));
64        buttonPanel.add(makeDigitButton("9"));
65        buttonPanel.add(makeOperatorButton("/"));
66        buttonPanel.add(makeDigitButton("4"));
67        buttonPanel.add(makeDigitButton("5"));
```

```
68          buttonPanel.add(makeDigitButton("6"));
69          buttonPanel.add(makeOperatorButton("*"));
70          buttonPanel.add(makeDigitButton("1"));
71          buttonPanel.add(makeDigitButton("2"));
72          buttonPanel.add(makeDigitButton("3"));
73          buttonPanel.add(makeOperatorButton("-"));
74          buttonPanel.add(makeDigitButton("0"));
75          buttonPanel.add(makeDigitButton("."));
76          buttonPanel.add(makeOperatorButton("="));
77          buttonPanel.add(makeOperatorButton("+"));
78
79          add(buttonPanel, BorderLayout.CENTER);
80       }
81
82       class MathOpListener implements ActionListener
83       {
84          public void actionPerformed(ActionEvent event)
85          {
86             double value = Double.parseDouble(display.getText());
87             String mathOp = (String) mathOpCombo.getSelectedItem();
88
89             double base = 10;
90             if (baseeButton.isSelected()) { base = Math.E; }
91             else if (base2Button.isSelected()) { base = 2; }
92
93             boolean radian = radianCheckBox.isSelected();
94             if (!radian && (mathOp.equals("sin")
95                   || mathOp.equals("cos") || mathOp.equals("tan")))
96             {
97                value = Math.toRadians(value);
98             }
99
100            if (mathOp.equals("sin"))
101            {
102               value = Math.sin(value);
103            }
104            else if (mathOp.equals("cos"))
105            {
106               value = Math.cos(value);
107            }
108            else if (mathOp.equals("tan"))
109            {
110               value = Math.tan(value);
111            }
112            else if (mathOp.equals("log"))
113            {
114               value = Math.log(value) / Math.log(base);
115            }
116            else if (mathOp.equals("exp"))
117            {
118               value = Math.pow(base, value);
119            }
120            display.setText("" + value);
121
122            startNewValue = true;
123         }
124      }
125
126      private JPanel createBaseButtons()
127      {
```

```
128         baseeButton = new JRadioButton("e");
129         base10Button = new JRadioButton("10");
130         base2Button = new JRadioButton("2");
131
132         baseeButton.setSelected(true);
133
134         ButtonGroup group = new ButtonGroup();
135         group.add(baseeButton);
136         group.add(base10Button);
137         group.add(base2Button);
138
139         JPanel basePanel = new JPanel();
140         basePanel.add(baseeButton);
141         basePanel.add(base10Button);
142         basePanel.add(base2Button);
143         basePanel.setBorder(new TitledBorder(new EtchedBorder(), "Base"));
144
145         return basePanel;
146      }
147
148      private void createControlPanel()
149      {
150         radianCheckBox = new JCheckBox("Radian");
151         radianCheckBox.setSelected(true);
152
153         mathOpCombo = new JComboBox();
154         mathOpCombo.addItem("sin");
155         mathOpCombo.addItem("cos");
156         mathOpCombo.addItem("tan");
157         mathOpCombo.addItem("log");
158         mathOpCombo.addItem("exp");
159
160         mathOpButton = new JButton("Apply");
161         mathOpButton.addActionListener(new MathOpListener());
162
163         JPanel controlPanel = new JPanel();
164         controlPanel.add(radianCheckBox);
165         controlPanel.add(createBaseButtons());
166         controlPanel.add(mathOpCombo);
167         controlPanel.add(mathOpButton);
168
169         add(controlPanel, BorderLayout.SOUTH);
170      }
171
172      /**
173         Combines two values with an operator.
174         @param value1 the first value
175         @param value2 the second value
176         @param op an operator (+, -, *, /, or =)
177      */
178      public double calculate(double value1, double value2, String op)
179      {
180         if (op.equals("+"))
181         {
182            return value1 + value2;
183         }
184         else if (op.equals("-"))
185         {
186            return value1 - value2;
```

```
187         }
188         else if (op.equals("*"))
189         {
190            return value1 * value2;
191         }
192         else if (op.equals("/"))
193         {
194            return value1 / value2;
195         }
196         else // "="
197         {
198            return value2;
199         }
200      }
201
202      class DigitButtonListener implements ActionListener
203      {
204         private String digit;
205
206         /**
207            Constructs a listener whose actionPerformed method adds a digit
208            to the display.
209            @param aDigit the digit to add
210         */
211         public DigitButtonListener(String aDigit)
212         {
213            digit = aDigit;
214         }
215
216         public void actionPerformed(ActionEvent event)
217         {
218            if (startNewValue)
219            {
220               display.setText("");
221               startNewValue = false;
222            }
223            display.setText(display.getText() + digit);
224         }
225      }
226
227      /**
228         Makes a button representing a digit of a calculator.
229         @param digit the digit of the calculator
230         @return the button of the calculator
231      */
232      public JButton makeDigitButton(String digit)
233      {
234         JButton button = new JButton(digit);
235         ActionListener listener = new DigitButtonListener(digit);
236         button.addActionListener(listener);
237         return button;
238      }
```

```
239
240     class OperatorButtonListener implements ActionListener
241     {
242        private String operator;
243
244        /**
245           Constructs a listener whose actionPerformed method
246           schedules an operator for execution.
247        */
248        public OperatorButtonListener(String anOperator)
249        {
250           operator = anOperator;
251        }
252
253        public void actionPerformed(ActionEvent event)
254        {
255           if (!startNewValue)
256           {
257              double value = Double.parseDouble(display.getText());
258              lastValue = calculate(lastValue, value, lastOperator);
259              display.setText("" + lastValue);
260              startNewValue = true;
261           }
262
263           lastOperator = operator;
264        }
265     }
266
267     /**
268        Makes a button representing an operator of a calculator.
269        @param op the operator of the calculator
270        @return the button of the calculator
271     */
272     public JButton makeOperatorButton(String op)
273     {
274        JButton button = new JButton(op);
275        ActionListener listener = new OperatorButtonListener(op);
276        button.addActionListener(listener);
277        return button;
278     }
279  }
```