

Tutorial 7 – **Dental Payment** Application

Introducing JCheckBoxes and Message Dialogs

Outline

- 7.1 Test-Driving the **Dental Payment** Application
- 7.2 Constructing the **Dental Payment** Application
- 7.3 Using JCheckBoxes
- 7.4 Using a Dialog to Display a Message
- 7.5 Logical Operators
- 7.6 Wrap-Up



Objectives

- In this tutorial, you will learn to:
 - Use `JCheckBoxes` to allow users to select options.
 - Use dialogs to display messages.
 - Use logical operators to form more complex conditions.



7.1 Test-Driving the Dental Payment Application

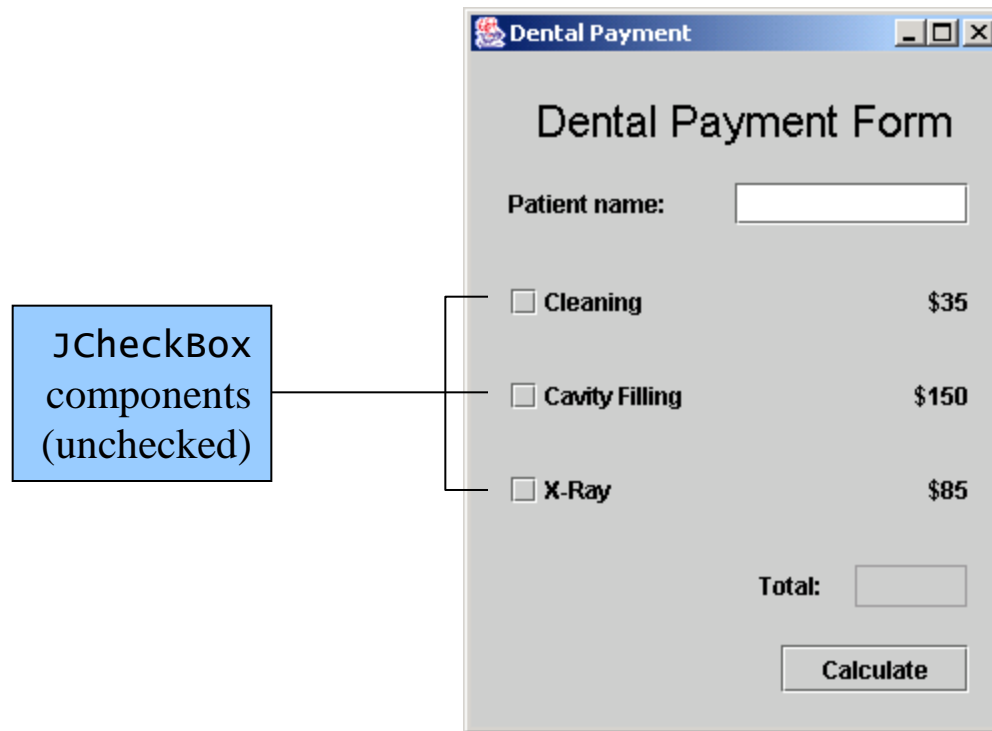
Application Requirements

A dentist has asked you to create an application that employees can use to bill patients. Your application must allow the user to enter the patient's name and specify which services were performed during the visit. Your application must then calculate the total charges. If a user attempts to calculate the total of a bill before any services are specified or before the patient's name is entered, an error message should be displayed.



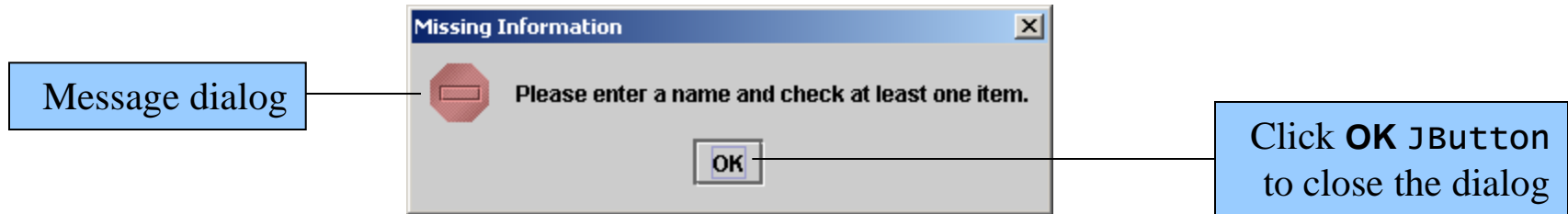
7.1 Test-Driving the Dental Payment Application (Cont.)

Figure 7.1 **Dental Payment** application without input entered.



7.1 Test-Driving the Dental Payment Application (Cont.)

Figure 7.2 Message dialog appears when no name is entered and/or no JCheckBoxes are selected.

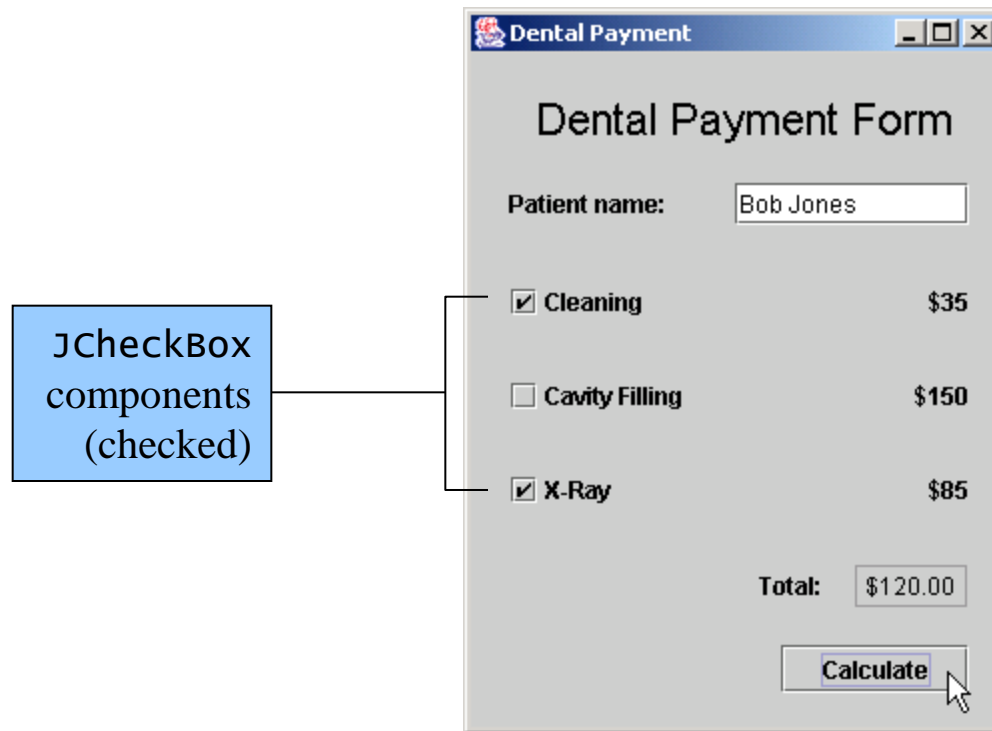


- Run the **Dental Payment** application
 - `cd c:\Examples\Tutorial07\CompletedApplication\DentalPayment`
 - `java DentalPayment`



7.1 Test-Driving the Dental Payment Application (Cont.)

Figure 7.3 **Dental Payment** application with input entered and total price of services displayed.



7.2 Constructing the Dental Payment Application

When the user clicks the Calculate JButton:

Input patient name

If user has not entered a patient name or has not selected any JCheckBoxes

Display error message in dialog

Else

If Cleaning JCheckBox is selected

Add cost of a cleaning to the total

If Cavity Filling JCheckBox is selected

Add cost of a cavity filling to the total

If X-Ray JCheckBox is selected

Add cost of an x-ray to the total

Display total price of services rendered in dollar format



7.2 Constructing the Dental Payment Application

Action	Component/Class	Event/Method
<i>Label the application's components</i>	dentalPaymentFormJLabel, patientNameJLabel, totalJLabel, cleaningPriceJLabel, cavityFillingPriceJLabel, xRayPriceJLabel, cleaningJCheckBox, cavityFillingJCheckBox, xRayJCheckBox	
<i>Input patient name</i>	patientNameJTextField	User clicks Calculate JButton
<i>If user has not entered a name or has not selected any JCheckBoxes Display error message in dialog</i>	cleaningJCheckBox, cavityFillingJCheckBox, xRayJCheckBox JOptionPane	
<i>Else If Cleaning JCheckBox is selected Add cost of cleaning to total</i>	cleaningJCheckBox	
<i>If Cavity Filling JCheckBox is selected Add cost of cavity filling to total</i>	cavityFillingJCheckBox	
<i>If X-Ray JCheckBox selected Add cost of an x-ray to total</i>	xRayJCheckBox	
<i>Display price of services</i>	totalJTextField	
Figure 7.4 ACE table for Dental Payment application.		



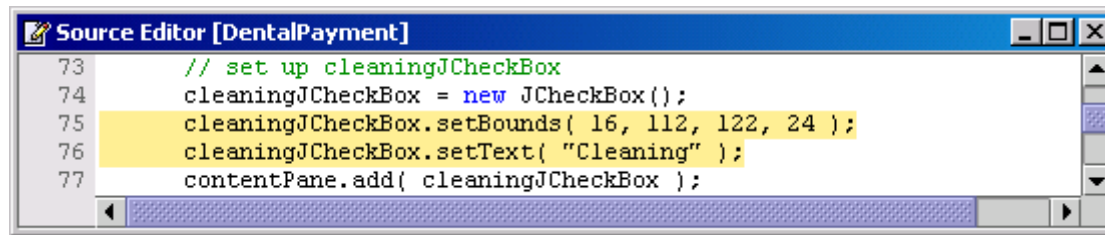
7.3 Using JCheckBoxes

- JCheckBoxes
 - small gray square either blank or containing a small checkmark
 - Properties:
 - *selected* property: whether or not a JCheckBox is selected
 - *text* property: the text just beside a JCheckBox
 - *bounds* property: location of JCheckBox
 - Methods:
 - `isSelected` – returns true if JCheckBox is selected
 - `setText` – sets text of JCheckBox
 - `setBounds` – sets bounds of JCheckBox
- **state button**: a button that has either an on or off state



7.3 Using JCheckBoxes (Cont.)

Figure 7.5 Changing the *text* and *bounds* properties of cleaningJCheckBox.



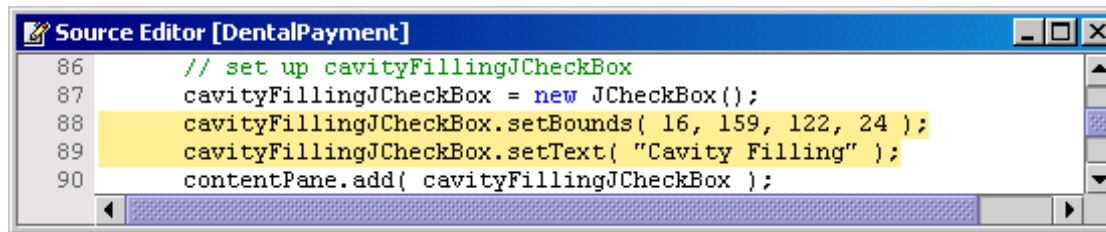
```
Source Editor [DentalPayment]
73 // set up cleaningJCheckBox
74 cleaningJCheckBox = new JCheckBox();
75 cleaningJCheckBox.setBounds( 16, 112, 122, 24 );
76 cleaningJCheckBox.setText( "Cleaning" );
77 contentPane.add( cleaningJCheckBox );
```

- Use methods `setBounds` and `setText` to change `JCheckBox` properties



7.3 Using JCheckBoxes (Cont.)

Figure 7.6 Changing *text* and *bounds* properties of cavityFillingJCheckBox.

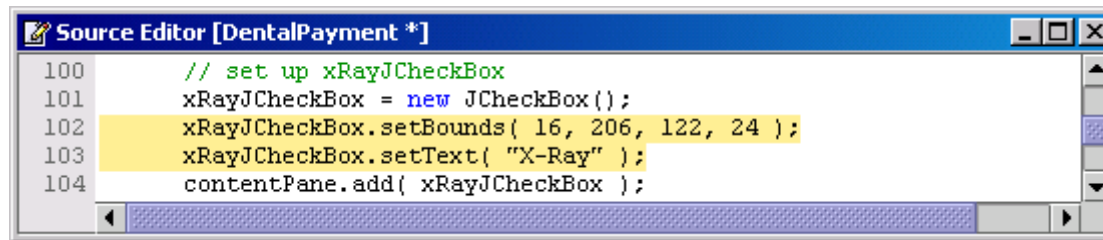


```
Source Editor [DentalPayment]
86 // set up cavityFillingJCheckBox
87 cavityFillingJCheckBox = new JCheckBox();
88 cavityFillingJCheckBox.setBounds( 16, 159, 122, 24 );
89 cavityFillingJCheckBox.setText( "Cavity Filling" );
90 contentPane.add( cavityFillingJCheckBox );
```



7.3 Using JCheckBoxes (Cont.)

Figure 7.7 Changing *text* and *bounds* properties of xRayJCheckBox.

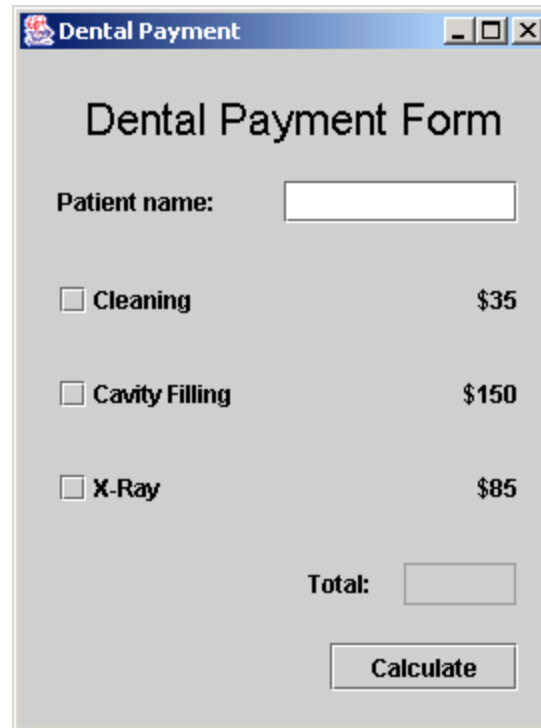


```
Source Editor [DentalPayment *]
100 // set up xRayJCheckBox
101 xRayJCheckBox = new JCheckBox();
102 xRayJCheckBox.setBounds( 16, 206, 122, 24 );
103 xRayJCheckBox.setText( "X-Ray" );
104 contentPane.add( xRayJCheckBox );
```



7.3 Using JCheckBoxes (Cont.)

Figure 7.8 Application running after customizing three JCheckBoxes.



The screenshot shows a Java Swing window titled "Dental Payment". The window contains a form titled "Dental Payment Form". The form has the following elements:

- A text field labeled "Patient name:".
- Three JCheckBoxes, each with a corresponding price:
 - Cleaning \$35
 - Cavity Filling \$150
 - X-Ray \$85
- A text field labeled "Total:".
- A "Calculate" button.



7.3 Using JCheckBoxes (Cont.)

- Use method `isSelected` to determine if a `JCheckBox` is selected.



7.3 Using JCheckBoxes (Cont.)

Figure 7.9 Determining if a JCheckBox has been selected.

```
Source Editor [DentalPayment]
152 // calculate cost of patient's visit
153 private void calculateJButtonActionPerformed((ActionEvent event)
154 {
155     double total = 0.0; // sum of all services provided
156
157     // if patient had a cleaning
158     if ( cleaningJCheckBox.isSelected() )
159     {
160         total += 35; // add 35 to total
161     }
162
163     // if patient had cavity filled
164     if ( cavityFillingJCheckBox.isSelected() )
165     {
166         total += 150; // add 150 to total
167     }
168
169     // if patient had x-ray taken
170     if ( xRayJCheckBox.isSelected() )
171     {
172         total += 85; // add 85 to total
173     }
174
175     // specify display format
176     DecimalFormat dollars = new DecimalFormat( "$0.00" );
177
178     // display total
179     totalJTextField.setText( dollars.format( total ) );
180
181 } // end method calculateJButtonActionPerformed
```

Calling method
isSelected

Add this code to
add the proper
amount to total



7.3 Using JCheckBoxes (Cont.)

Figure 7.10 Application running without input.

The screenshot shows a window titled "Dental Payment" with a form titled "Dental Payment Form". The form contains a text field for "Patient name:", three checkboxes for "Cleaning" (\$35), "Cavity Filling" (\$150), and "X-Ray" (\$85), and a "Total:" field displaying "\$0.00". A "Calculate" button is located at the bottom. A blue callout box points to the total field with the text: "Application calculates a bill of \$0.00 when no JCheckBoxes are checked".



7.3 Using JCheckBoxes (Cont.)

Figure 7.11 Application running with services selected, but no patient name entered.

The screenshot shows a Java Swing window titled "Dental Payment" with a standard Windows-style title bar. The window content is a form titled "Dental Payment Form". At the top, there is a label "Patient name:" followed by an empty text input field. Below this, there are three service items, each with a checkbox, a label, and a price:

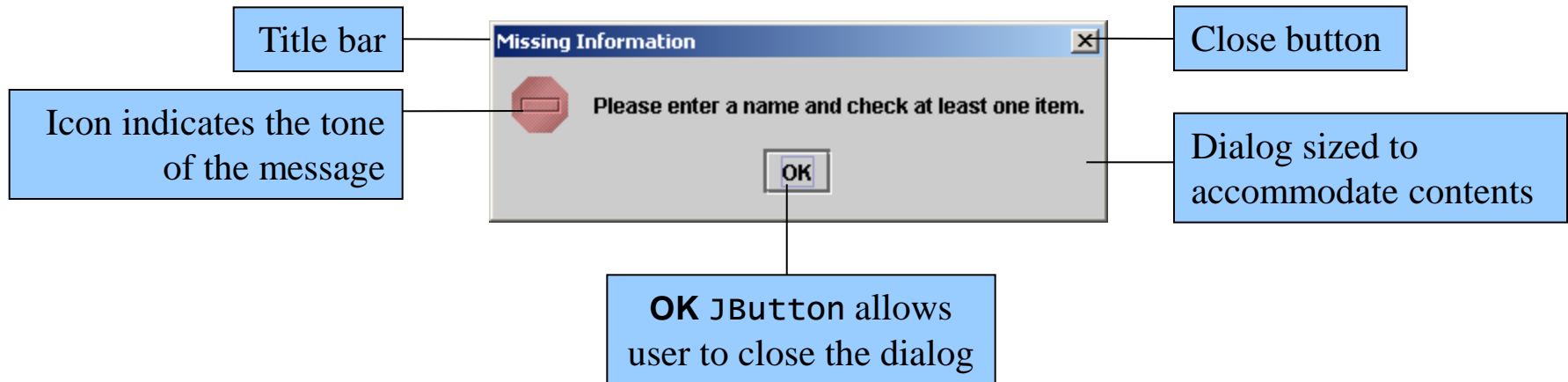
Service	Price
<input checked="" type="checkbox"/> Cleaning	\$35
<input type="checkbox"/> Cavity Filling	\$150
<input type="checkbox"/> X-Ray	\$85

At the bottom right of the form, there is a label "Total:" followed by a text input field containing the value "\$35.00". Below the total field is a button labeled "Calculate". A mouse cursor is pointing at the "Calculate" button.



7.4 Using a Dialog to Display a Message

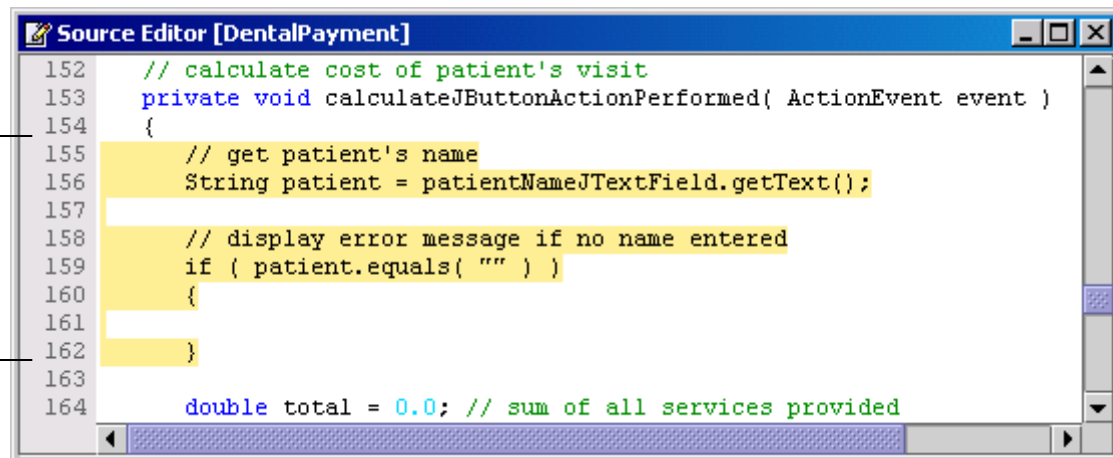
Figure 7.12 Message dialog displayed by the application.



7.4 Using a Dialog to Display a Message (Cont.)

Figure 7.13 Adding an if statement to event handler calculateJButtonActionPerformed.

Add this code to
verify patient
name was entered

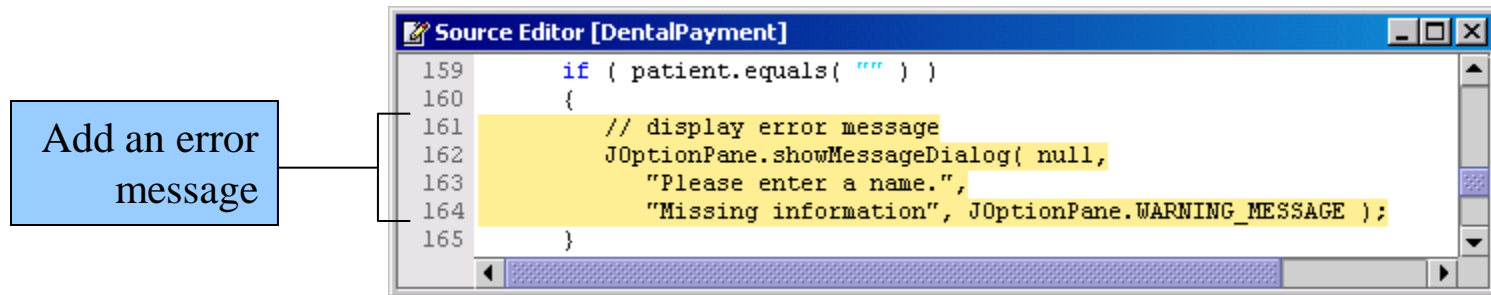


```
Source Editor [DentalPayment]
152 // calculate cost of patient's visit
153 private void calculateJButtonActionPerformed( ActionEvent event )
154 {
155     // get patient's name
156     String patient = patientNameJTextField.getText();
157
158     // display error message if no name entered
159     if ( patient.equals( "" ) )
160     {
161
162     }
163
164     double total = 0.0; // sum of all services provided
```



7.4 Using a Dialog to Display a Message (Cont.)

Figure 7.14 Message dialog code that displays a message to users.



- Use method `showMessageDialog` to display error message in separate window



7.4 Using a Dialog to Display a Message (Cont.)

Figure 7.15 Adding an else part to an if statement.

The screenshot shows a Source Editor window titled "Source Editor [DentalPayment]". The code is as follows:

```
164         "Missing information", JOptionPane.WARNING_MESSAGE );
165     }
166     else // otherwise, do calculations
167     {
168         double total = 0.0; // sum of all services provided
169
170         // if patient had a cleaning
171         if ( cleaningJCheckBox.isSelected() )
172         {
173             total += 35; // add 35 to total
174         }
175
176         // if patient had cavity filled
177         if ( cavityFillingJCheckBox.isSelected() )
178         {
179             total += 150; // add 150 to total
180         }
181
182         // if patient had x-ray taken
183         if ( xRayJCheckBox.isSelected() )
184         {
185             total += 85; // add 85 to total
186         }
187
188         // specify display format
189         DecimalFormat dollars = new DecimalFormat( "$0.00" );
190
191         // display total
192         totalJTextField.setText( dollars.format( total ) );
193
194     } // end else
195
196 } // end method calculateJButtonActionPerformed
```

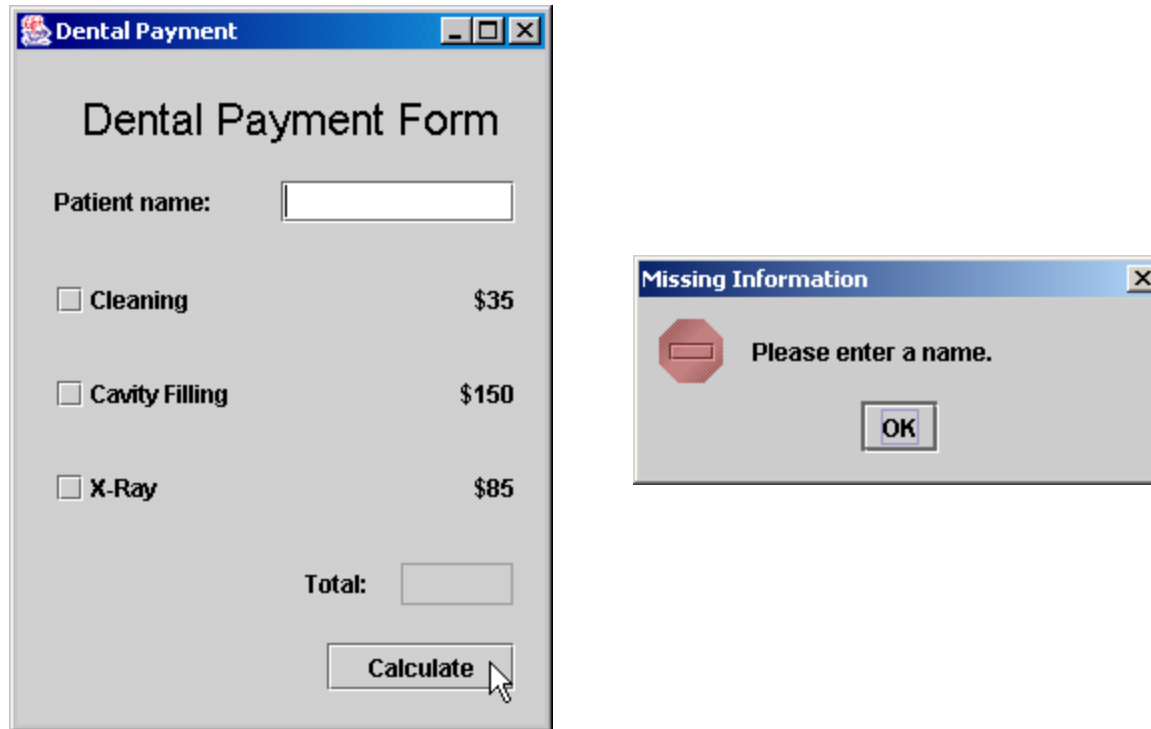
Two blue callout boxes with black text and lines pointing to the code:

- The first box, labeled "Add an else statement", points to line 166.
- The second box, labeled "Add the right brace to end else statement", points to line 194.



7.4 Using a Dialog to Display a Message (Cont.)

Figure 7.17 Application running without a name entered.



7.4 Using a Dialog to Display a Message (Cont.)

Figure 7.18 Application running with a name entered, but without any JCheckBoxes selected.

Dental Payment Form

Patient name:

Cleaning \$35

Cavity Filling \$150

X-Ray \$85

Total:

User must now enter a name

Application calculates a bill of \$0.00 when a name is entered and no JCheckBoxes are checked



7.4 Using a Dialog to Display a Message (Cont.)





JOptionPane Constants	Icon	Description
<code>JOptionPane.ERROR_MESSAGE</code>		Icon containing a stop sign. Typically used to alert the user of errors or critical situations.
<code>JOptionPane.INFORMATION_MESSAGE</code>		Icon containing the letter “i.” Typically used to display information about the state of the application.
<code>JOptionPane.QUESTION_MESSAGE</code>		Icon containing a question mark. Typically used to ask the user a question.
<code>JOptionPane.WARNING_MESSAGE</code>		Icon containing an exclamation point. Typically used to caution the user against potential problems.
<code>JOptionPane.PLAIN_MESSAGE</code>	no icon	Displays a dialog that simply contains a message, with no icon.

Figure 7.19 JOptionPane dialog icon constants.



7.5 Logical Operators

- Logical operators
 - `&&` (conditional AND)
 - `||` (conditional OR)
 - `^` (boolean logical exclusive OR)
 - `!` (logical negation)



7.5 Logical Operators (Cont.)

Using Conditional AND (&&)

```
if ( gender.equals( "Female" ) && age >= 65 )
{
    seniorFemales += 1;
}
```

expression1	expression2	expression1 && expression2
false	false	false
false	true	false
true	false	false
true	true	true

Figure 7.20 Truth table for the && operator.



7.5 Logical Operators (Cont.)

Using Conditional OR (||)

```
if ( semesterAverage >= 90 || finalExam >= 90 )
{
    JOptionPane.showMessageDialog( null, "Student grade is A.",
        "Student Grade", JOptionPane.INFORMATION_MESSAGE );
}
```

expression1	expression2	expression1 expression2
false	false	false
false	true	true
true	false	true
true	true	true

Figure 7.21 Truth table for the || operator.



7.5 Logical Operators (Cont.)

Using Boolean Logical Exclusive OR (\wedge)

expression1	expression2	expression1 \wedge expression2
false	false	false
false	true	true
true	false	true
true	true	false

Figure 7.22 Truth table for the boolean logical exclusive OR (\wedge) operator.



7.5 Logical Operators (Cont.)

Using Logical Negation (!)

```
if ( !( grade == value ) )
{
    displayLabel.setText( "They are not equal!" );
}
```

expression	! expression
false	true
true	false

Figure 7.23 Truth table for the ! operator (logical NOT).



7.5 Logical Operators (Cont.)

Figure 7.24 Using the && and || logical operators.

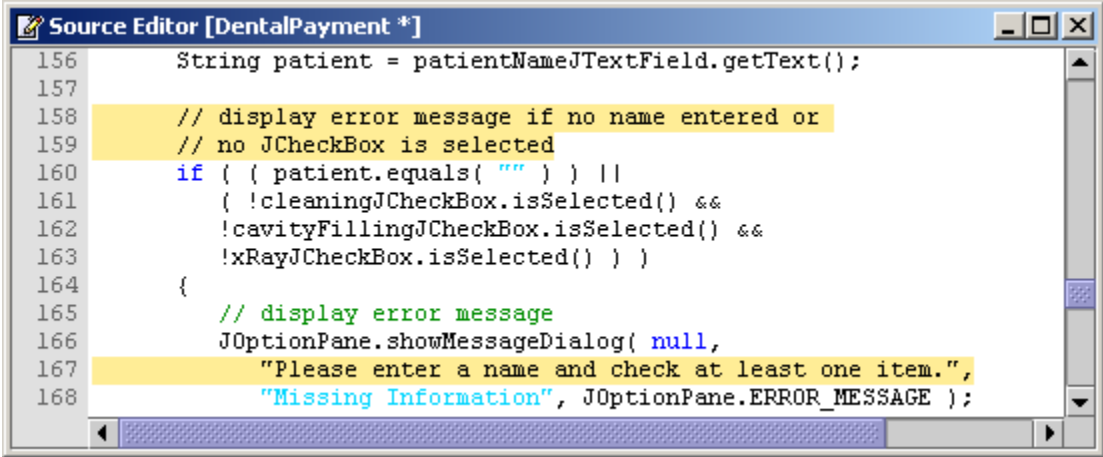
Inserting a complex expression

```
Source Editor [DentalPayment]
156     String patient = patientNameJTextField.getText();
157
158     // display error message if no name entered
159     if ( ( patient.equals( "" ) ) ||
160         ( !cleaningJCheckBox.isSelected() &&
161           !cavityFillingJCheckBox.isSelected() &&
162           !xRayJCheckBox.isSelected() ) )
163     {
164         // display error message
```



7.5 Logical Operators (Cont.)

Figure 7.25 Message dialog displayed if no name is entered and no JCheckBoxes are selected.



```
Source Editor [DentalPayment *]
156     String patient = patientNameJTextField.getText();
157
158     // display error message if no name entered or
159     // no JCheckBox is selected
160     if ( ( patient.equals( "" ) ) ||
161         ( !cleaningJCheckBox.isSelected() &&
162           !cavityFillingJCheckBox.isSelected() &&
163           !xRayJCheckBox.isSelected() ) )
164     {
165         // display error message
166         JOptionPane.showMessageDialog( null,
167         "Please enter a name and check at least one item.",
168         "Missing Information", JOptionPane.ERROR_MESSAGE );

```



7.5 Logical Operators (Cont.)

Figure 7.26 Message dialog displayed for incorrect input.

