Tutorial 11 – **Security Panel** Application

Introducing the `switch` Multiple-Selection Statement, `Date` and `DateFormat`

<u>Outline</u>

Objectives

- In this tutorial, you will learn to:
  - Use the `switch` multiple-selection statement.
  - Use `case` labels.
  - Display a date and time.
  - Use a `JPasswordField`.
  - Use a `Date` to determine the system's current date and time.
  - Use a `DateFormat` to format the date and time.

11.1 Test Driving the **Security Panel** Application

Application Requirements

A pharmaceutical company wants to install a security panel outside its laboratory facility. Only authorized personnel may enter the lab, using their security codes. The following are the valid security codes (also called access codes) and the groups of employees they represent:

Values	*Groups*
1645	*Technicians*
8345	*Custodians*
9998, 1006–1008	*Scientists*

When a security code is entered, it should not be visible to anyone standing near the security panel. For each security code, access is either granted or denied. All access attempts are displayed in a screen below the keypad. If access is granted, the date, time and group (scientists, custodians, etc.) are displayed on the screen. If access is denied, the date, the time and a message, "Access Denied," are displayed on the screen. Furthermore, an employee can enter the access code 7, 8 or 9 to summon a security guard for assistance. The date, the time and a message, "Restricted Access," are then displayed on the screen to indicate that the request has been received.

11.1 Test Driving the **Security Panel** Application (Cont.)

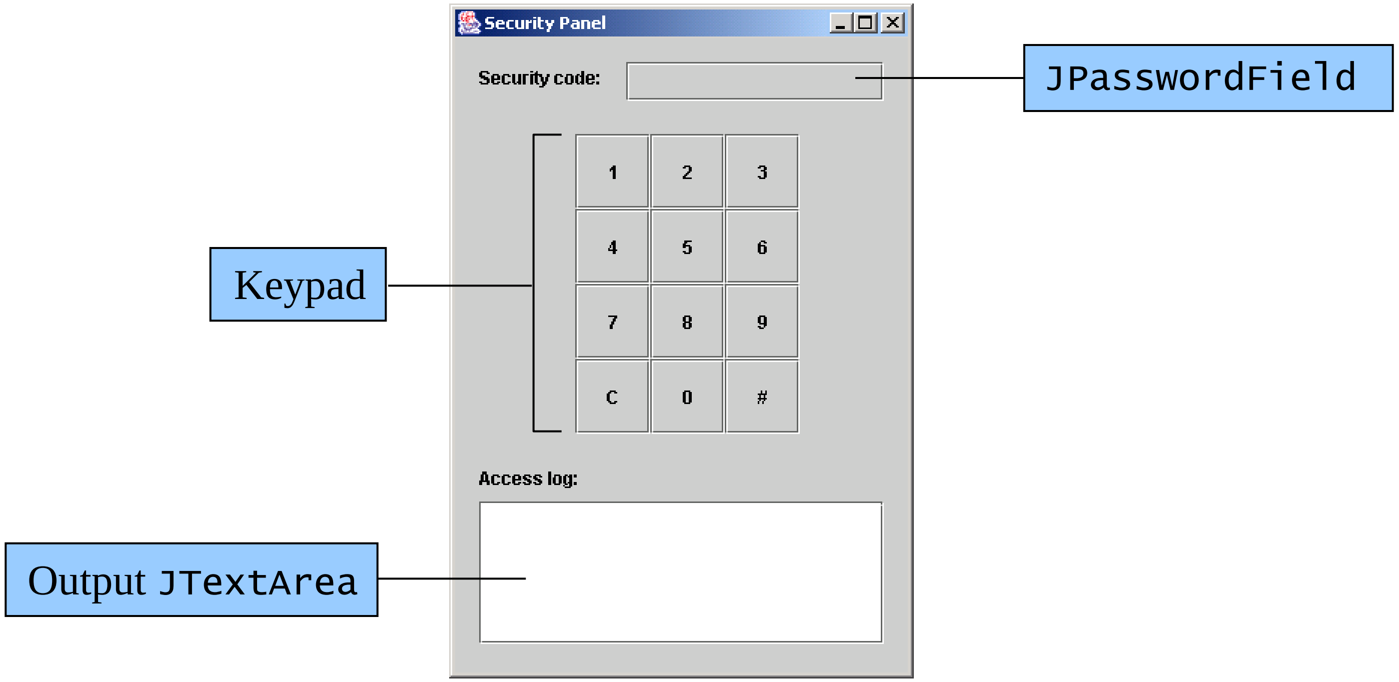
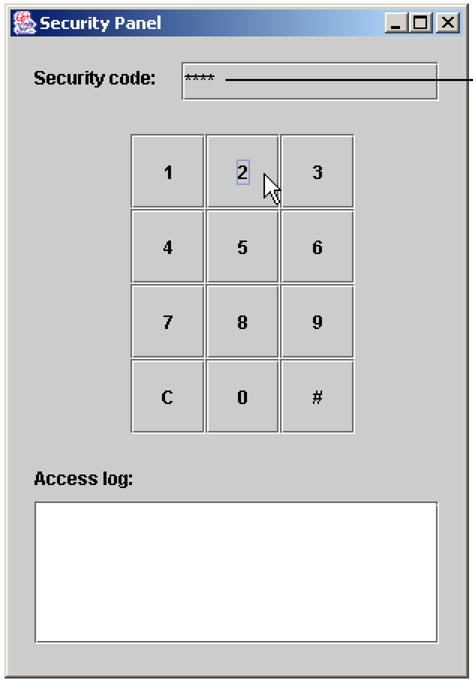


Figure 11.1 **Security Panel** application.

11.1 Test Driving the **Security Panel** Application (Cont.)

Figure 11.2 Asterisks displayed in the **SecurityCode:** `JPasswordField`.

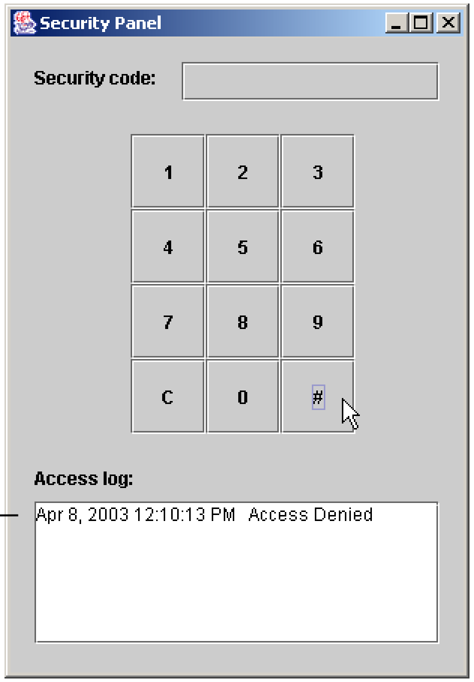


`JPasswordField` displays one asterisk (*) for each numeric key the user presses (so no one can see the actual security code entered)

- Enter the security code `1212`
- `JPasswordField` displays asterisks rather than the typed characters

11.1 Test Driving the **Security Panel** Application (Cont.)

Figure 11.3 **Security Panel** displaying the **Access Denied** message.



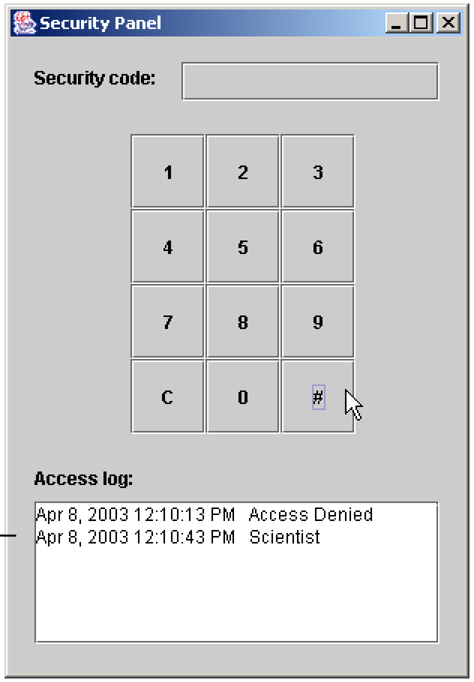
Apr 8, 2003 12:10:13 PM Access Denied

Message indicating that an invalid security code was entered

- Press # to submit your security code
- Press C to clear your security code

11.1 Test Driving the **Security Panel** Application (Cont.)

Figure 11.4 **Security Panel** application confirming a valid security code entry.



```
Apr 8, 2003 12:10:13 PM   Access Denied
Apr 8, 2003 12:10:43 PM   Scientist
```

Message displayed when a valid security code is entered

- Enter 1006 to log on with a valid security code

11.2 Introducing the `switch` Multiple-Selection Statement

- Multiple selections with a nested if … else statement

```
if ( grade == 'A' )
{
   displayJLabel.setText( "Excellent!" );
}
else if ( grade == 'B' )
{
   displayJLabel.setText( "Very good!" );
}
else if ( grade == 'C' )
{
   displayJLabel.setText( "Good." );
}
else if ( grade == 'D' )
{
   displayJLabel.setText( "Poor." );
}
else if ( grade == 'F' )
{
   displayJLabel.setText( "Failure." );
}
else
{
   displayJLabel.setText( "Invalid grade." );
}
```

11.2 Introducing the `switch` Multiple-Selection Statement (Cont.)

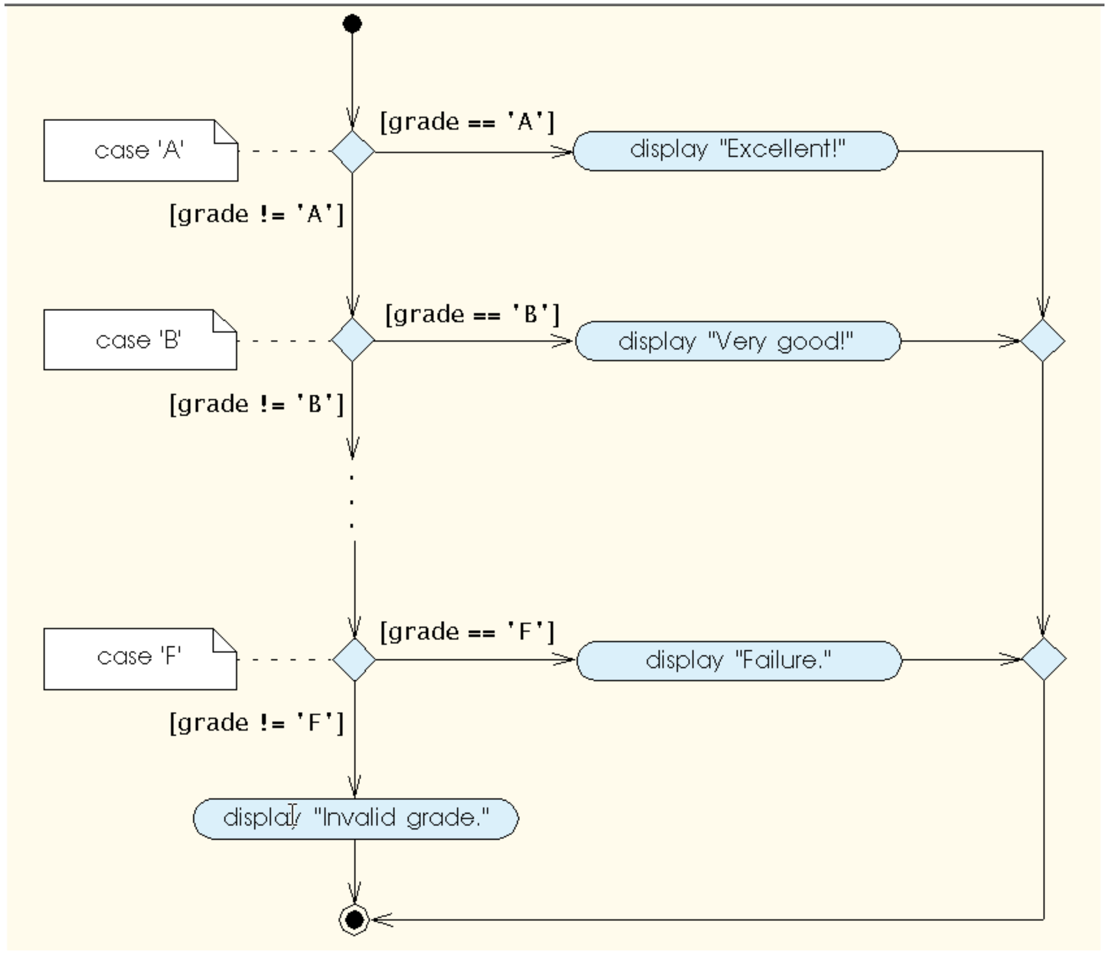
- `switch` statement: multiple selection statement
  - Controlling expression
  - `case` labels
  - `default` case
  - Only types `char`, `byte`, `short`, and `int` can be tested in a switch statement
  - `break` statement
- `Char`
  - one of Java's eight primitive types
  - Character constant (character literal)
  - Represented as a character within single quotes

11.2 Introducing the `switch` Multiple-Selection Statement (Cont.)

```
switch ( grade )
{
   case 'A':
      displayJLabel.setText( "Excellent!" );
      break;

   case 'B':
      displayJLabel.setText( "Very good!" );
      break;

   case 'C':
      displayJLabel.setText( "Good." );
      break;

   case 'D':
      displayJLabel.setText( "Poor." );
      break;

   case 'F':
      displayJLabel.setText( "Failure." );
      break;

   default:
      displayJLabel.setText( "Invalid grade." );
}
```

11.2 Introducing the `switch` Multiple-Selection Statement (Cont.)

Figure 11.5 `switch` multiple-selection statement UML activity diagram.



11.3 Constructing the **Security Panel** Application

```
When the user clicks a numeric JButton
    Get the JButton’s digit
    Append the digit to the text in the JPasswordField

When the user clicks the # JButton
    Get the security code input by the user from the JPasswordField
    Clear the JPasswordField

    switch based on the security code variable
        case where access code is 7, 8 or 9
            Store text “Restricted Access” in a String variable
        case where access code equals 1645
            Store text “Technician” in a String variable
        case where access code equals 8345
            Store text “Custodian” in a String variable
        case where access code equals 9998 or is in the range 1006 to 1008
            Store text “Scientist” in a String variable
        default case where none of the preceding cases match
            Store text “Access Denied” in a String variable

    Display a message in the JTextArea with current time and the String
        variable’s contents
```

11.3 Constructing the **Security Panel** Application (Cont.)

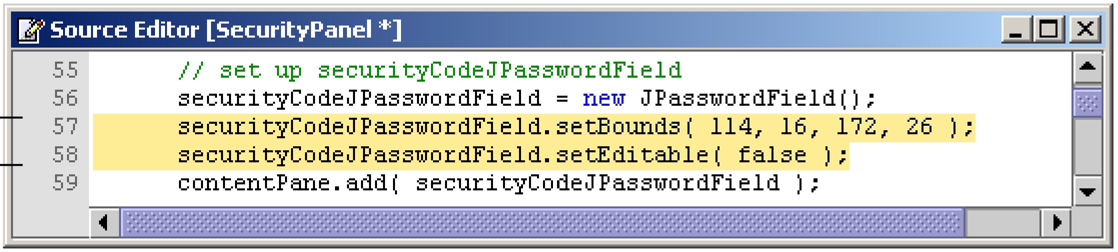
Action	Component/Object	Event/Method
Label all the application's components	`securityCodeJLabel`, `accessLogJLabel`	Application is run
Get the JButton's digit	`zeroJButton`, `oneJButton`, `twoJButton`, `threeJButton`, `fourJButton`, `fiveJButton`, `sixJButton`, `sevenJButton`, `eightJButton`, `nineJButton`	User clicks a numeric `JButton`
Append the digit to the text in the JPasswordField	`securityCodeJPasswordField`	
Clear JPasswordField	`securityCodeJPasswordField`	User clicks the **C** `JButton`
Get the security code input from the JPasswordField	`securityCodeJPasswordField`	User clicks the **#** `JButton`
Clear the JPasswordField	`securityCodeJPasswordField`	
Determine whether security code is valid	`message (String)`	
Display message in the JTextArea	`accessLogJTextArea`	

Figure 11.6 ACE table for **Security Panel** application.

11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.7 Setting the `securityCodeJPasswordField`'s *bounds* and *editable* properties.

Set the *bounds* and *editable* properties of the `JPasswordField`



```
// set up securityCodeJPasswordField
securityCodeJPasswordField = new JPasswordField();
securityCodeJPasswordField.setBounds( 114, 16, 172, 26 );
securityCodeJPasswordField.setEditable( false );
contentPane.add( securityCodeJPasswordField );
```

- Echo characters (masked characters)
- Set character displayed using the `setEchoChar` method

11.3 Constructing the **Security Panel** Application (Cont.)

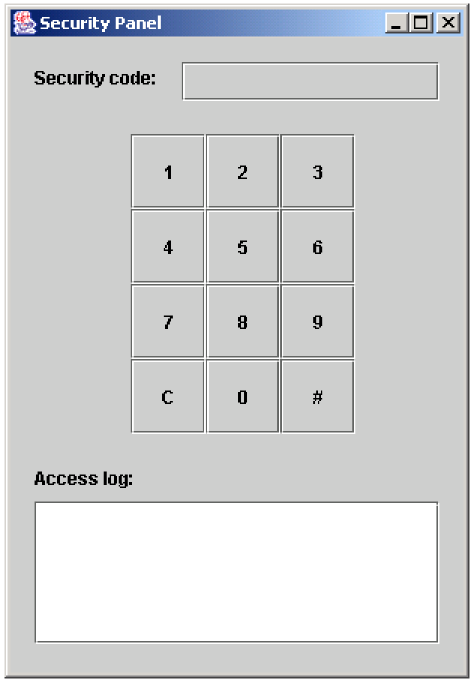
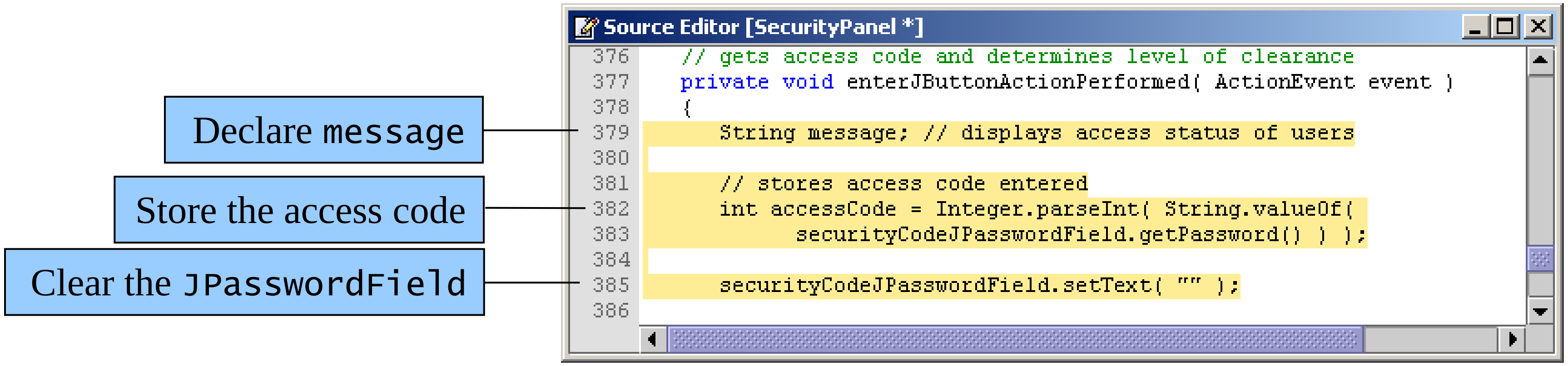


Figure 11.8 **Security Panel** application after customizing the `JPasswordField`.

11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.9 Storing the access code and clearing the **Security code:** `JPasswordField`.

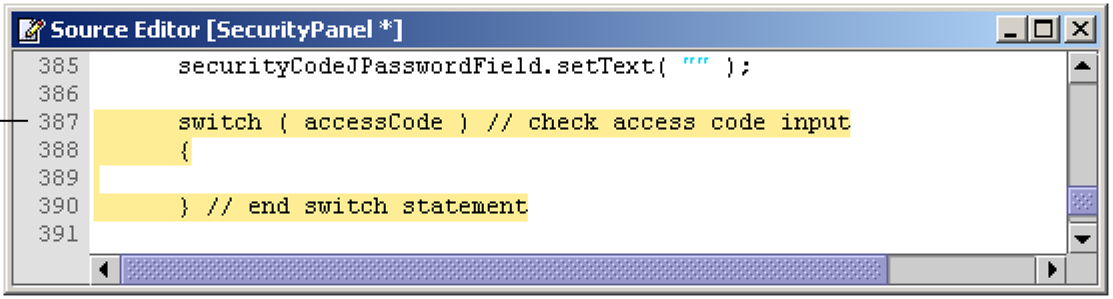


```
   // gets access code and determines level of clearance
   private void enterJButtonActionPerformed( ActionEvent event )
   {
      String message; // displays access status of users

      // stores access code entered
      int accessCode = Integer.parseInt( String.valueOf(
            securityCodeJPasswordField.getPassword() ) );

      securityCodeJPasswordField.setText( "" );
```

11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.10 Adding a `switch` statement to the method.

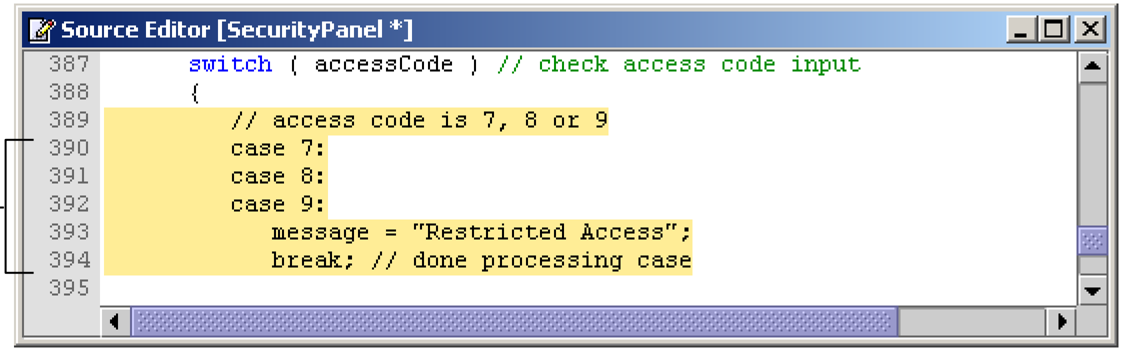
Beginning of the `switch` statement



```
      securityCodeJPasswordField.setText( "" );

      switch ( accessCode ) // check access code input
      {

      } // end switch statement

```

11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.11 Adding `case` labels to the `switch` statement.

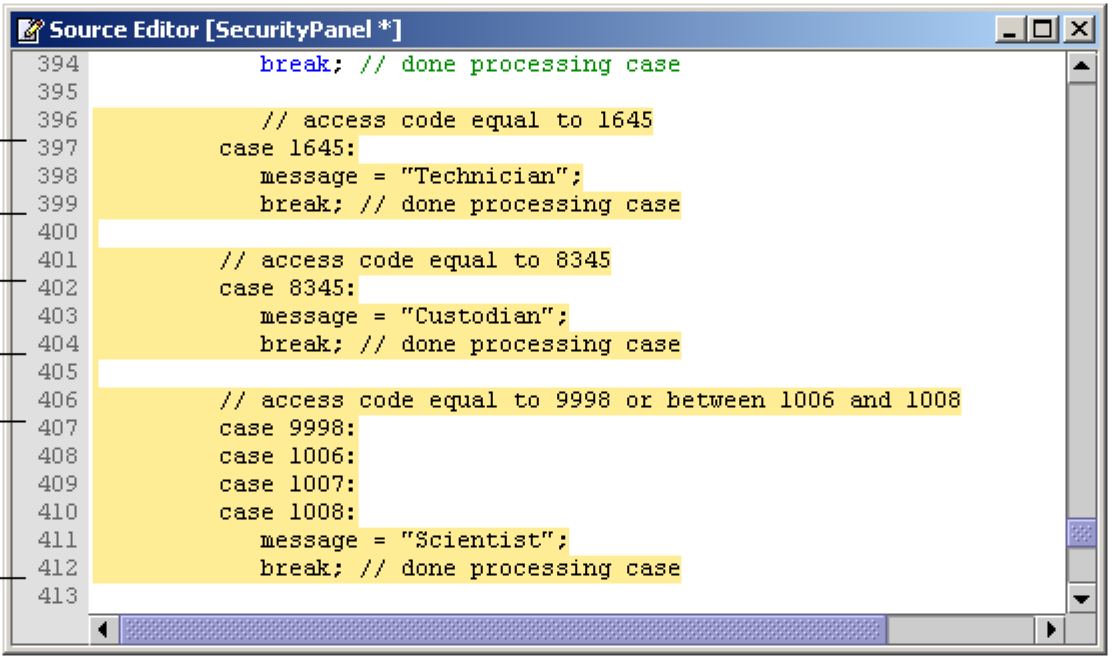


```
      switch ( accessCode ) // check access code input
      {
         // access code is 7, 8 or 9
         case 7:
         case 8:
         case 9:
            message = "Restricted Access";
            break; // done processing case
```

Multiple `case` labels result in same `message`

11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.12 Finishing the `switch` statement.



```
                break; // done processing case

                // access code equal to 1645
             case 1645:
                message = "Technician";
                break; // done processing case

             // access code equal to 8345
             case 8345:
                message = "Custodian";
                break; // done processing case

             // access code equal to 9998 or between 1006 and 1008
             case 9998:
             case 1006:
             case 1007:
             case 1008:
                message = "Scientist";
                break; // done processing case

```

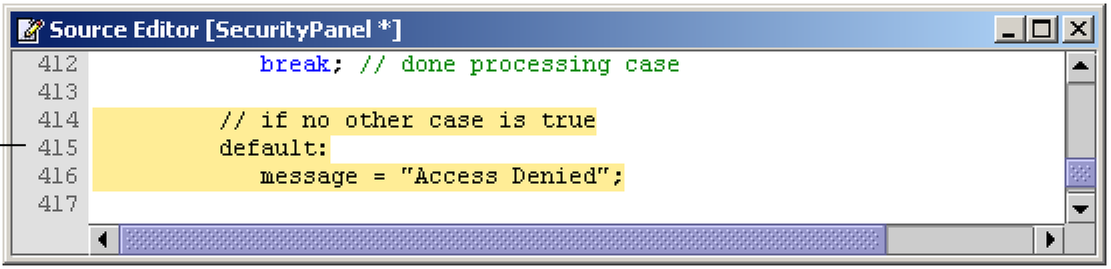
Access code 1645 for technicians

Access code 8345 for custodians

Access codes 9998, 1006, 1007 and 1008 for scientists

11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.13 Adding a `default` case to the `switch` statement.



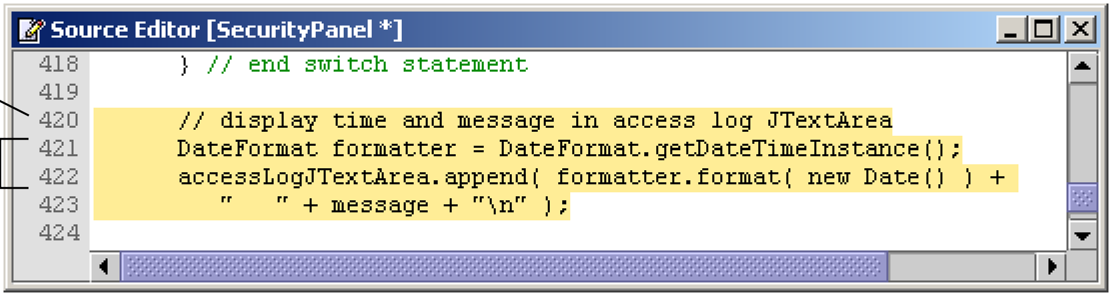
```
            break; // done processing case

         // if no other case is true
         default:
            message = "Access Denied";

```

`default` case at the end of the `switch` statement

- `switch` statements can have at most one `default` statement

11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.14 Outputting the current date, the time and the message.

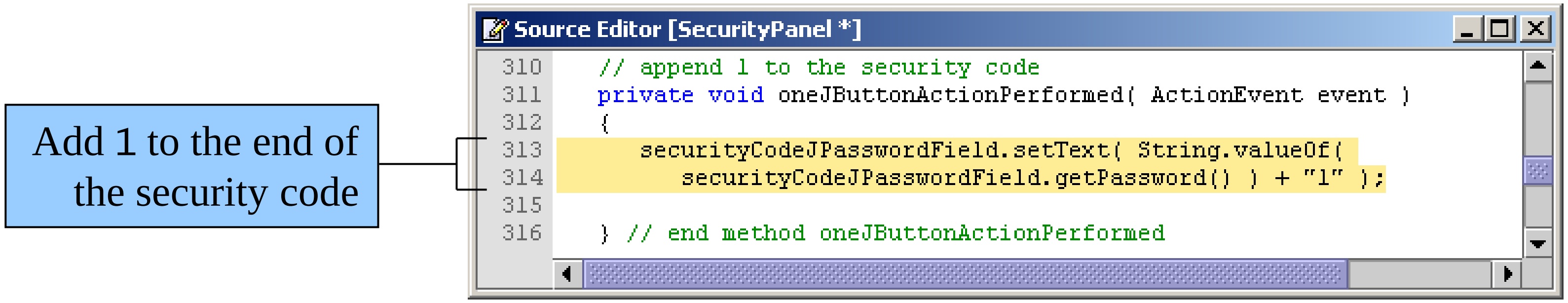

Format for the date and time
Add the date and time to the message
Source Editor [SecurityPanel *]

```
      } // end switch statement

      // display time and message in access log JTextArea
      DateFormat formatter = DateFormat.getDateTimeInstance();
      accessLogJTextArea.append( formatter.format( new Date() ) +
         "   " + message + "\n" );
```

- The `DateFormat` object is similar to a `DecimalFormat` object
  - Allows you to output a date and time

11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.15 Appending a one to the end of the security code.

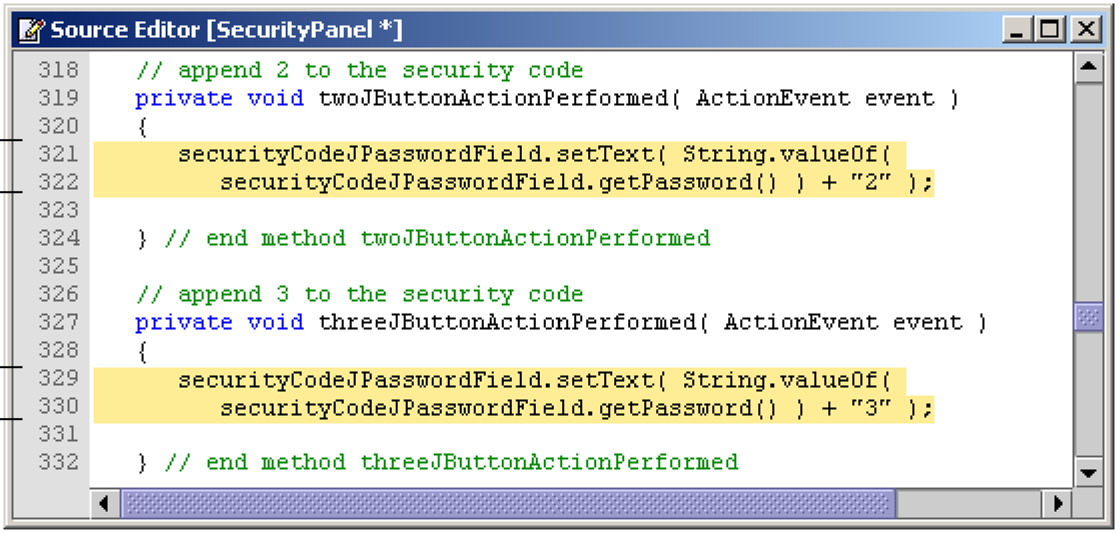

Add 1 to the end of the security code
Source Editor [SecurityPanel *]

```
    // append 1 to the security code
    private void oneJButtonActionPerformed( ActionEvent event )
    {
       securityCodeJPasswordField.setText( String.valueOf(
          securityCodeJPasswordField.getPassword() ) + "1" );

    } // end method oneJButtonActionPerformed
```

- Use the + operator to append (concatenate) `String`s to other `String`s

11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.16 Coding event handlers for **2** `JButton` and **3** `JButton`; other `JButtons` would be similar.



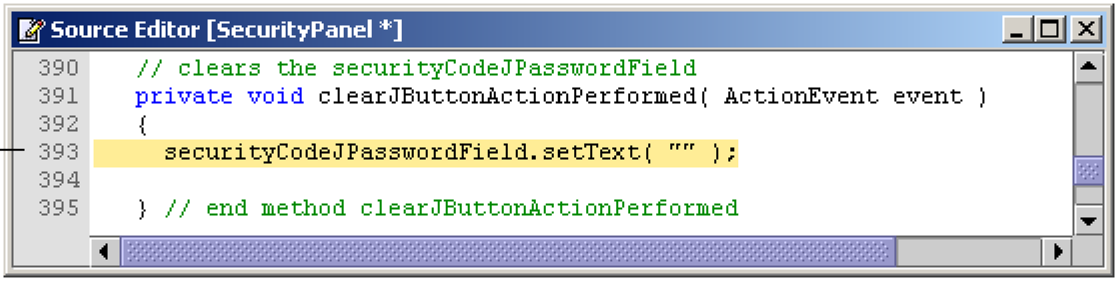
```
   // append 2 to the security code
   private void twoJButtonActionPerformed( ActionEvent event )
   {
      securityCodeJPasswordField.setText( String.valueOf(
         securityCodeJPasswordField.getPassword() ) + "2" );

   } // end method twoJButtonActionPerformed

   // append 3 to the security code
   private void threeJButtonActionPerformed( ActionEvent event )
   {
      securityCodeJPasswordField.setText( String.valueOf(
         securityCodeJPasswordField.getPassword() ) + "3" );

   } // end method threeJButtonActionPerformed
```

Add 2 to the end of the security code

Add 3 to the end of the security code

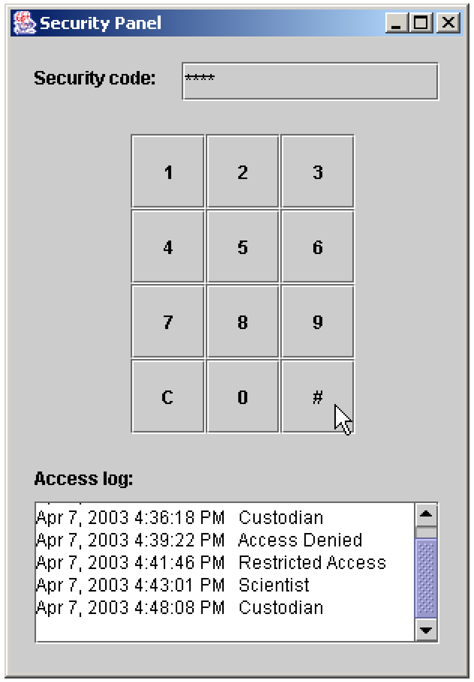
11.3 Constructing the **Security Panel** Application (Cont.)

Figure 11.17 Clearing the **Security Code:** `JPasswordField`.



```
// clears the securityCodeJPasswordField
private void clearJButtonActionPerformed( ActionEvent event )
{
   securityCodeJPasswordField.setText( "" );

} // end method clearJButtonActionPerformed
```

11.3 Constructing the **Security Panel** Application (Cont.)



```
Apr 7, 2003 4:36:18 PM  Custodian
Apr 7, 2003 4:39:22 PM  Access Denied
Apr 7, 2003 4:41:46 PM  Restricted Access
Apr 7, 2003 4:43:01 PM  Scientist
Apr 7, 2003 4:48:08 PM  Custodian
```

Figure 11.18 Completed **Security Panel** application.

Outline

SecurityPanel.java (19 of 19)

```
         // if no other case is true
         default:
            message = "Access Denied";

      } // end switch statement

      // display time and message in access log JTextArea
      DateFormat formatter = DateFormat.getDateTimeInstance();
      accessLogJTextArea.append( formatter.format( new Date() ) +
         "   " + message + "\n" );

   } // end method enterJButtonActionPerformed

   // main method
   public static void main( String[] args )
   {
      SecurityPanel application = new SecurityPanel();
      application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

   } // end method main

} // end class SecurityPanel
```

`Default` case executes if no other `cases` match

Right brace ends the `switch` statement

Append the current date and time to the `message`