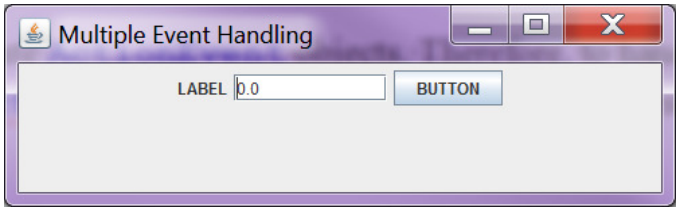
TEXT FIELD INPUT VALIDATION

The examples of the previous topic suffer in that the user is able to enter invalid input into the text field and cause runtime exceptions. To deal with this, we use the **`try`** and **`catch`** statements to detect, and ignore, any invalid input.

Example

We'll modify **`EventHandlingDemo2`** so that it validates the user's input. Specifically, it displays this GUI:



If the user enters a value in the text field then the user's entry replaces whatever value had been there. The button increments any value in the text field. In both cases, if the text field contains a non-number, then any user action is ignored.

The application is below. Code that significantly differs from **`EventHandlingDemo2`** is in black; code that is substantially the same is in gray.

The modified listener class is on lines 10-30. It uses the **`try`** and **`catch`** statements to detect that a **`NumberFormatException`** has been thrown by the **`parseDouble`** method.

```
 1  import javax.swing.*;
 2  import java.awt.FlowLayout;
 3  import java.awt.event.*;
 4
 5  public class EventHandlingDemo3
 6  {
 7     private JTextField textField; // must be accessed by MyListener
 8     private JButton button;       // same
 9
10     private class MyListener implements ActionListener
11     {  // implement the handler method to do what I need.
12        public void actionPerformed( ActionEvent e )
13        {
14           double x;
15           try // to get text from text field and parse into a double
16           {
17              // parseDouble can throw NumberFormatException
18              x = Double.parseDouble( textField.getText( ) );
19              // conversion worked, go on to handle event
```

```
20              if ( e.getSource( ) == textField ) // source is text field
21                 { } // don't do anything, we just want to put value back
22              else if ( e.getSource( ) == button ) // source is button
23                 x++;  // increment the text field
24              textField.setText( Double.toString( x ) );
25           }
26           catch ( NumberFormatException exc )
27           {  // do nothing if text field value is bad
28           }
29        }  // end actionPerformed
30     }  // end MyListener
31
32     public static void main ( String [] args )
33     {                              // avoid percolation of 'static'
34        new EventHandlingDemo3( ); // by using an app constructor
35     }
36
37     public EventHandlingDemo3( )  // the application constructor
38     {
39        JFrame win = new JFrame( "Multiple Event Handling" );
40        win.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
41        JLabel label = new JLabel( "LABEL" );
42        button = new JButton( "BUTTON" );
43        textField = new JTextField( 10 );
44        textField.setText( "0.0" ); // initialize the text field
45        win.setLayout( new FlowLayout( ) );
46        win.add( label );
47        win.add( textField );
48        win.add( button );
49        // create listener for the event
50        MyListener listener = new MyListener( );
51        // register the listener with the button and text field
52        button.addActionListener( listener );
53        textField.addActionListener( listener );
54        // normally these are last
55        win.setSize( 500, 150 );
56        win.setVisible( true );
57     }
58  }
```

Exercises

For each of the following write a Java program (application or applet) that adheres to the given specifications.

1. Modify your solution to exercise #1 of the topic *Event Handling*, adding these specifications.

   5. Input Validation
      5.1. The text field must accept only positive or negative integers; any other value is INVALID.
      5.2. If the user presses the Enter key, any valid value must be retained and any invalid value must be replaced by the previous text field contents.

2. Modify your solution to exercise #2 of the topic *Event Handling*, adding these specifications.

   4. Input Validation
      4.1. The text field must accept only a valid calendar date of the form mm/dd/yyyy; any other value is INVALID.
      4.2. If the user presses the Enter key or clicks the GO button, if the text field value is invalid then the program must display a modal error dialog.
         4.2.1. Once the user acknowledges the program must cancel any processing and reestablish the conditions in 1.2.2 and 1.2.3.

3. Modify your solution to exercise #3 of the topic *Event Handling*, adding these specifications.

   6. Input Validation
      6.1. The text fields must accept only numeric values; any other value is INVALID.
      6.2. Integer input must be taken as floating point.
      6.3. If the value in the Fahrenheit text field is invalid, replace the contents of both text fields with blanks and place the focus on, and select all contents of, the Fahrenheit text field.
      6.4. If the value in the Celsius text field is invalid, replace the contents of both text fields with blanks and place the focus on, and select all contents of, the Celsius text field.