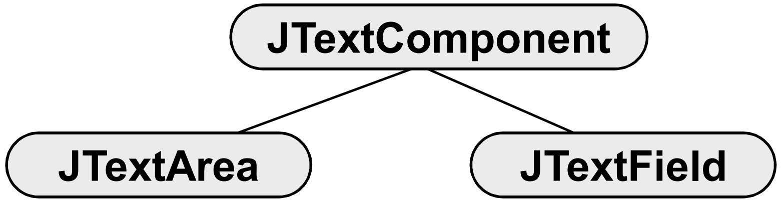
13.5 Text I/O for GUIs



Text fields and text areas

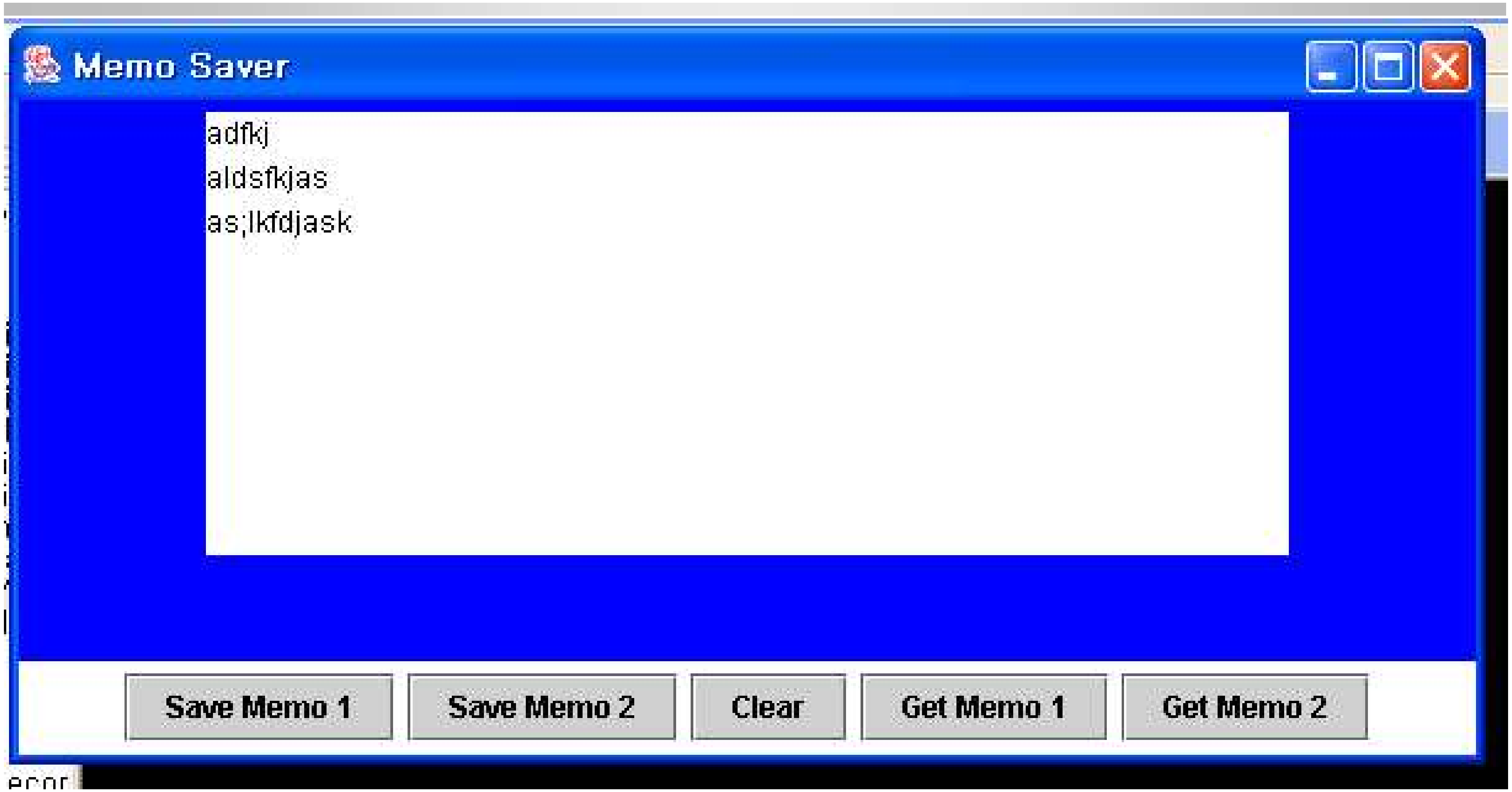
» **`getText`** method retrieves text in component

» **`setText`** changes text in component

If `memo1` is a `String` and `theText` is either a `JTextField` or a `JTextArea`, then you could write:

```
memo1 = theText.getText();
theText.setText("Hi Mom");
```

Listing 13.10 A GUI with a Text Area - MemoSaver.java

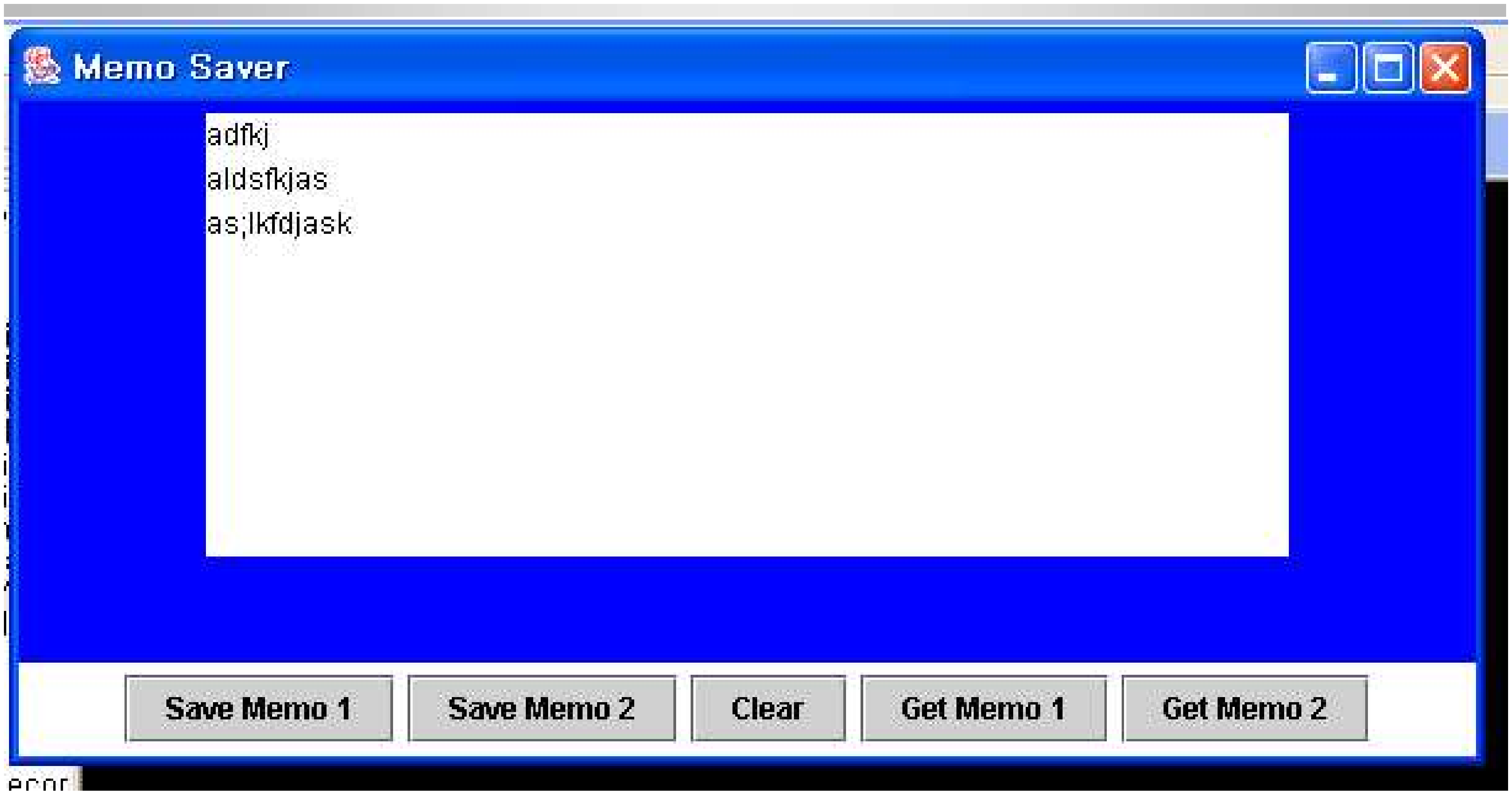


```
adfkj
aldsfkjas
as;lkfdjask
```

Listing 13.10 A GUI with a Text Area - MemoSaver.java

```
 //Listing 13.10  A GUI with a Text Area

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

// There is a demonstration main in part 2 of this display.

public class MemoSaver extends JFrame implements ActionListener
{
    public static final int WIDTH = 600;
    public static final int HEIGHT = 300;
    public static final int LINES = 10;
    public static final int CHAR_PER_LINE = 40;

    private JTextArea theText; //
    private String memo1 = "No Memo 1.";
    private String memo2 = "No Memo 2.";
    // If you get memo 1 before you set memo 1, you get the message
    //          "No Memo 1".
```

```
public MemoSaver( )
 {
    setSize(WIDTH, HEIGHT);
    addWindowListener(new WindowDestroyer( ));
    setTitle("Memo Saver");
    Container contentPane = getContentPane( );
    contentPane.setLayout(new BorderLayout( ));

    JPanel buttonPanel = new JPanel( );
    buttonPanel.setBackground(Color.WHITE);
    buttonPanel.setLayout(new FlowLayout( ));

    JButton memo1Button = new JButton("Save Memo 1");
    memo1Button.addActionListener(this);
    buttonPanel.add(memo1Button);

    JButton memo2Button = new JButton("Save Memo 2");
    memo2Button.addActionListener(this);
    buttonPanel.add(memo2Button);

    JButton clearButton = new JButton("Clear");
    clearButton.addActionListener(this);
    buttonPanel.add(clearButton);
    JButton get1Button = new JButton("Get Memo 1");
    get1Button.addActionListener(this);
    buttonPanel.add(get1Button);

    JButton get2Button = new JButton("Get Memo 2");
    get2Button.addActionListener(this);
    buttonPanel.add(get2Button);

    contentPane.add(buttonPanel, BorderLayout.SOUTH);

    JPanel textPanel = new JPanel( );
    textPanel.setBackground(Color.BLUE);
    theText = new JTextArea(LINES, CHAR_PER_LINE);

    theText.setBackground(Color.WHITE);
    theText.setLineWrap(true);
    textPanel.add(theText);
    contentPane.add(textPanel, BorderLayout.CENTER);
}
```



```
adfkj
aldsfkjas
as;lkfdjask
```

```
// theText is an instance variable
   public void actionPerformed(ActionEvent e)
   {
      String actionCommand = e.getActionCommand( );
      if (actionCommand.equals("Save Memo 1"))
         memo1 = theText.getText( );
      else if (actionCommand.equals("Save Memo 2"))
         memo2 = theText.getText( );
      else if (actionCommand.equals("Clear"))
         theText.setText("");
      else if (actionCommand.equals("Get Memo 1"))
         theText.setText(memo1);
      else if (actionCommand.equals("Get Memo 2"))
         theText.setText(memo2);
      else
         theText.setText("Error in memo interface");
    }

   public static void main(String[] args)
   {
      MemoSaver guiMemo = new MemoSaver( );
      guiMemo.setVisible(true);
   }
}
```

`JTextField` and `JTextArea`

- Both inherit from `JTextComponent`
- Both have `setText` and `getText` methods
- Both can have initializing text as parameter to constructor

- `JTextField` can only have one line of text
- `JTextArea` can have many lines of text
- `JTextArea` can have scroll bars

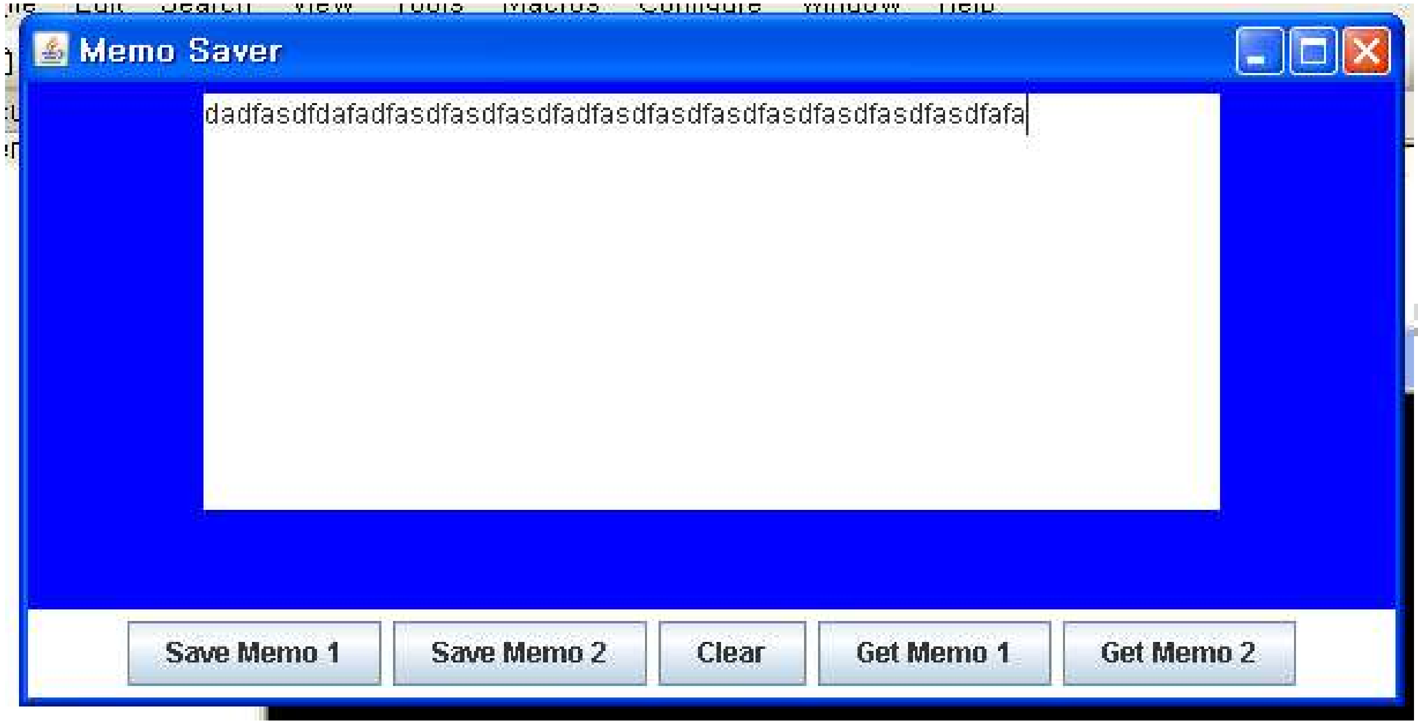
Big enough to hold 40 **m** characters

```
JTextField someText = new JTextField(40);
JTextArea someMoreText = new JTextArea(10, 40);
```

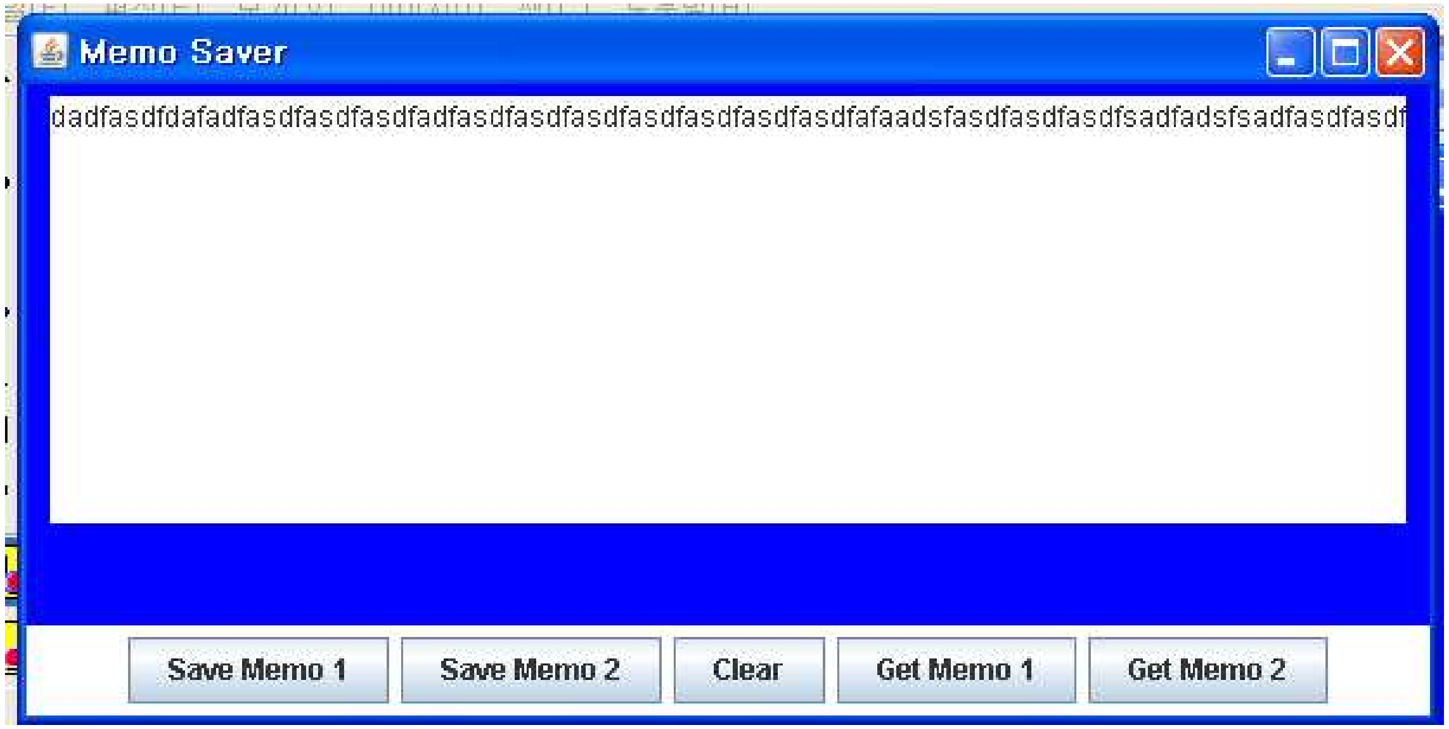
Big enough to hold 10 lines where each line can hold 40 **m** characters

Line wrapping in Text Areas

- setLineWrap method
  - » set the line-wrapping policy for a JTextArea
  - » Takes one argument of type boolean
  - » if the argument is true, then at the end of a line, any additional characters for the line will appear on the following line of the text area.
  - » If the argument is false, the extra characters will be on the same line and will not be visible
  - » ➔ Test… memosaver.java



dadfasdfdafadfasdfasdfasdfadfasdfasdfasdfasdfasdfasdfasdfafa



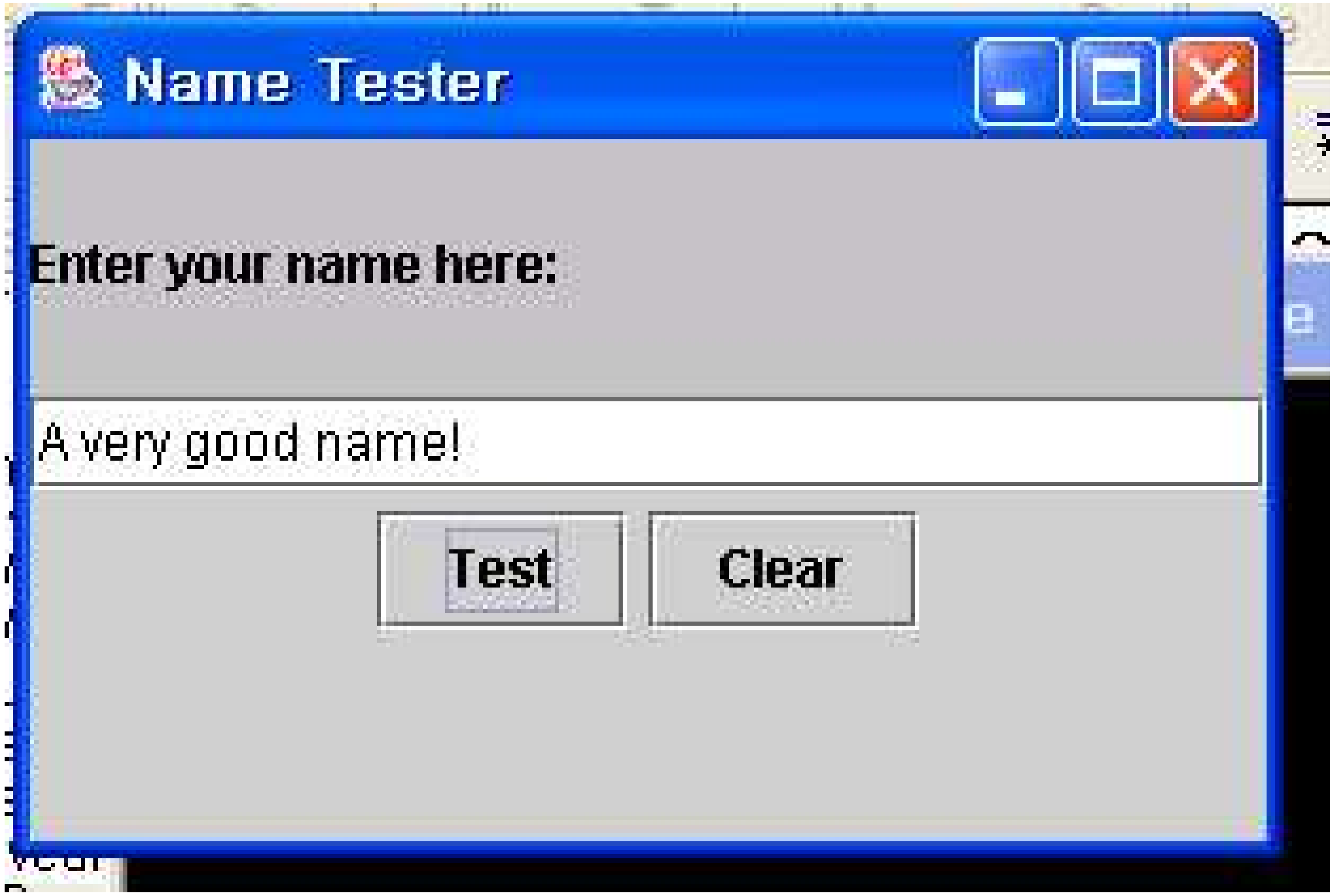
dadfasdfdafadfasdfasdfasdfadfasdfasdfasdfasdfasdfasdfasdfafaadsfasdfasdfasdfsadfadsfsadfasdfasdf

Read-Only Text Components

- Specify that a `JTextField` or `JTextArea` cannot be changed by the user.
  - » use method `setEditable` with argument `false`

    ```
    theText.setEditable(false);
    ```
  - » Only the GUI program can change the text in the component.
- Use the argument true to allow the user to edit.
  - » `theText.setEditable(true);`
- If `setEditable` is not called at all, the user *can* change the text.

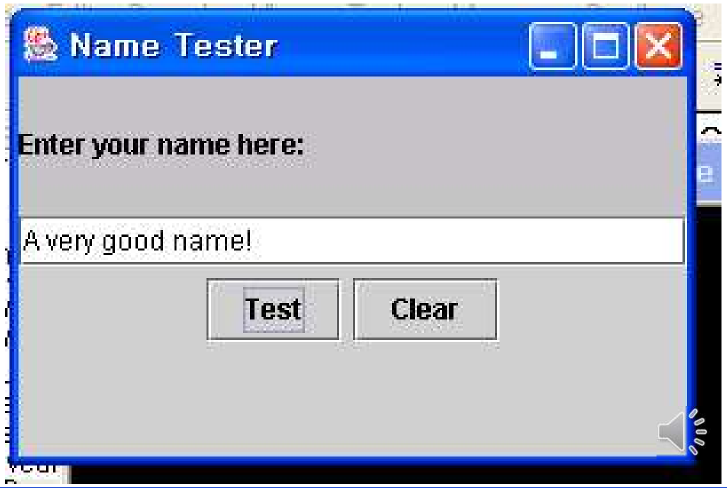
Listing 13.11 Labeling a Text Field - LabelDemo.java



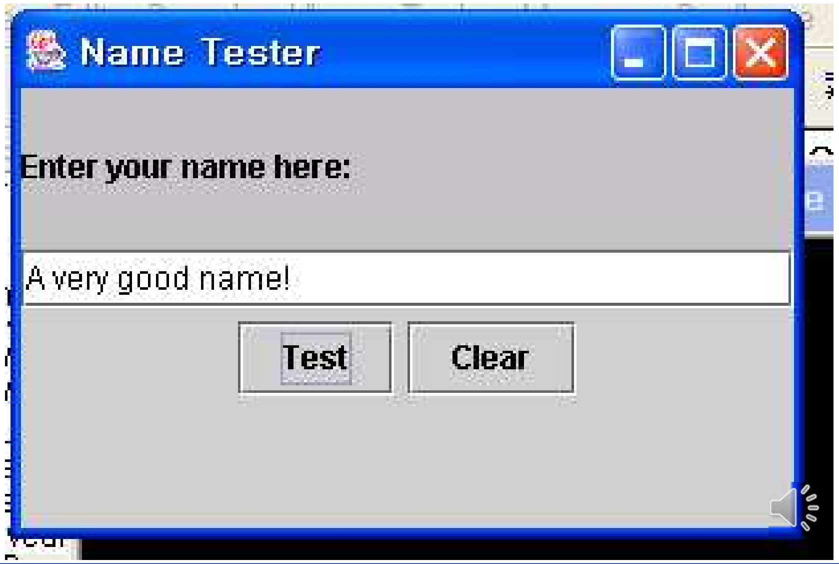
Listing 13.11 Labeling a Text Field - LabelDemo.java

```
 // Listing 13.11 Labeling a Text Field

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

/**
 Class to demonstrate placing a label on a text field.
*/
public class LabelDemo extends JFrame implements ActionListener
{
   public static final int WIDTH = 300;
   public static final int HEIGHT = 200;

   private JTextField name;
   public LabelDemo( )
   {

      setTitle("Name Tester");
      setSize(WIDTH, HEIGHT);
      addWindowListener(new WindowDestroyer( ));
```

```
    Container content = getContentPane( );
    content.setLayout(new GridLayout(2, 1));

    JPanel namePanel = new JPanel( );
    namePanel.setLayout(new BorderLayout( ));
    namePanel.setBackground(Color.LIGHT_GRAY);

            ////
    name = new JTextField(20);
    namePanel.add(name, BorderLayout.SOUTH);
    JLabel nameLabel = new JLabel("Enter your name here:");
    namePanel.add(nameLabel, BorderLayout.CENTER);

    content.add(namePanel);
    //////////

    JPanel buttonPanel = new JPanel( );
    buttonPanel.setLayout(new FlowLayout( ));
    JButton b = new JButton("Test");
    b.addActionListener(this);
    buttonPanel.add(b);
    b = new JButton("Clear");
    b.addActionListener(this);
    buttonPanel.add(b);

    content.add(buttonPanel);
}
```



```
public void actionPerformed(ActionEvent e)
{
    if (e.getActionCommand( ).equals("Test"))
        name.setText("A very good name!");
    else if (e.getActionCommand( ).equals("Clear"))
        name.setText("");
    else
        name.setText("Error in window interface.");
}


public static void main(String[] args)
{
    LabelDemo w = new LabelDemo( );
    w.setVisible(true);
}


}
```


Name Tester

Enter your name here:

A very good name!

Inputting and Outputting Numbers

To get an `int` from a `TextArea` or `TextField`:

- Get a string using `getText from field(TextArea` or `TextField`)
- Trim extra white space using `trim`
- Convert the String to an int using parseInt

```
int n =                       t(                     .        );
```

Inputting and Outputting Numbers

To get an `int` from a `TextArea` or `TextField`:

- Get a `String` using `getText`
- Trim extra white space using `trim`
- Convert the `String` to an `int` using `parseInt`

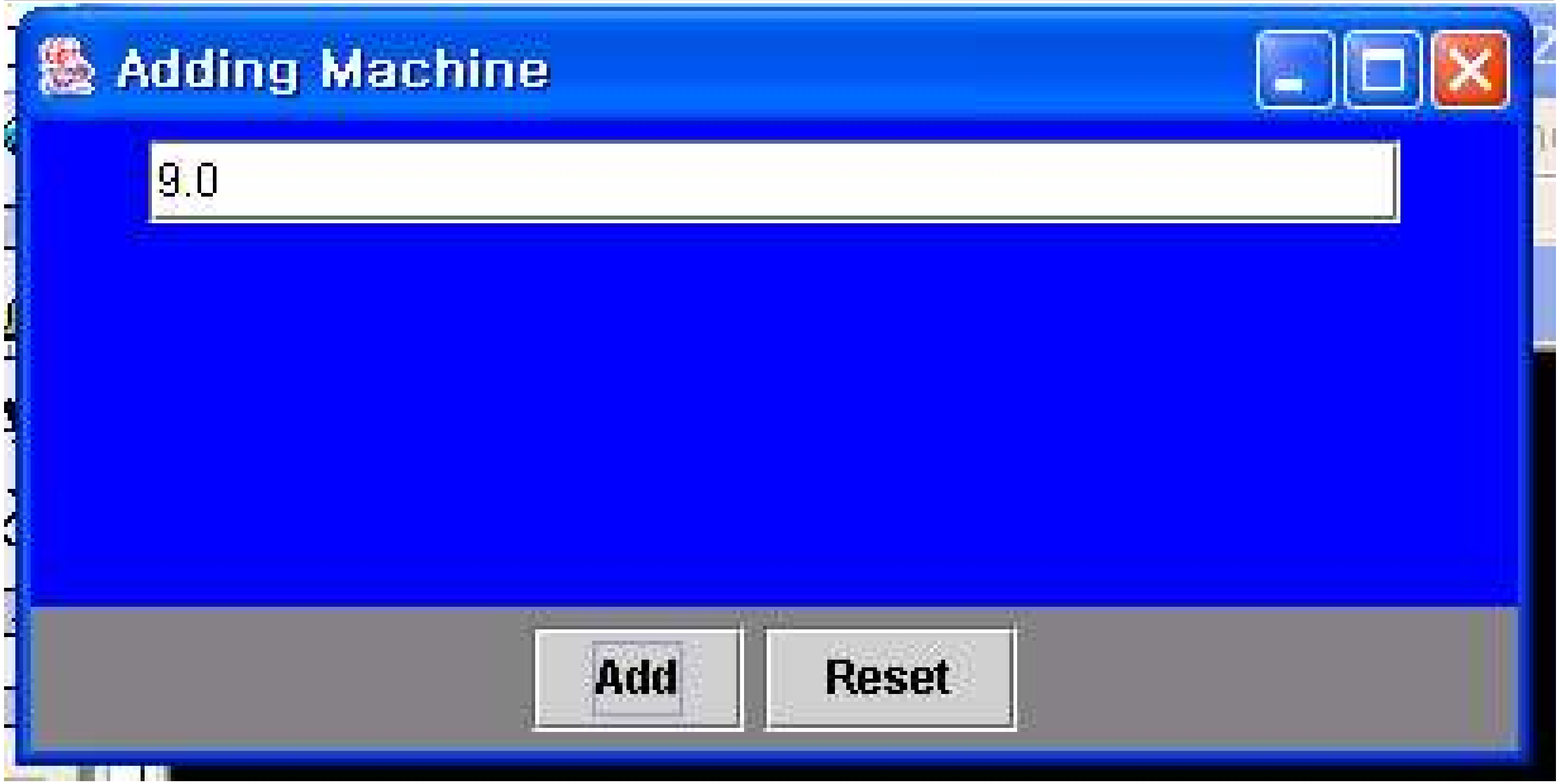
```
int n = Integer.parseInt(field.getText().trim());
```

To put an `int` into a `TextArea` or `TextField`:

- Convert the `int` to a `String` using `toString`
- Put the `String` in the text component using `setText`

```
field.setText(Integer.toString(total));
```

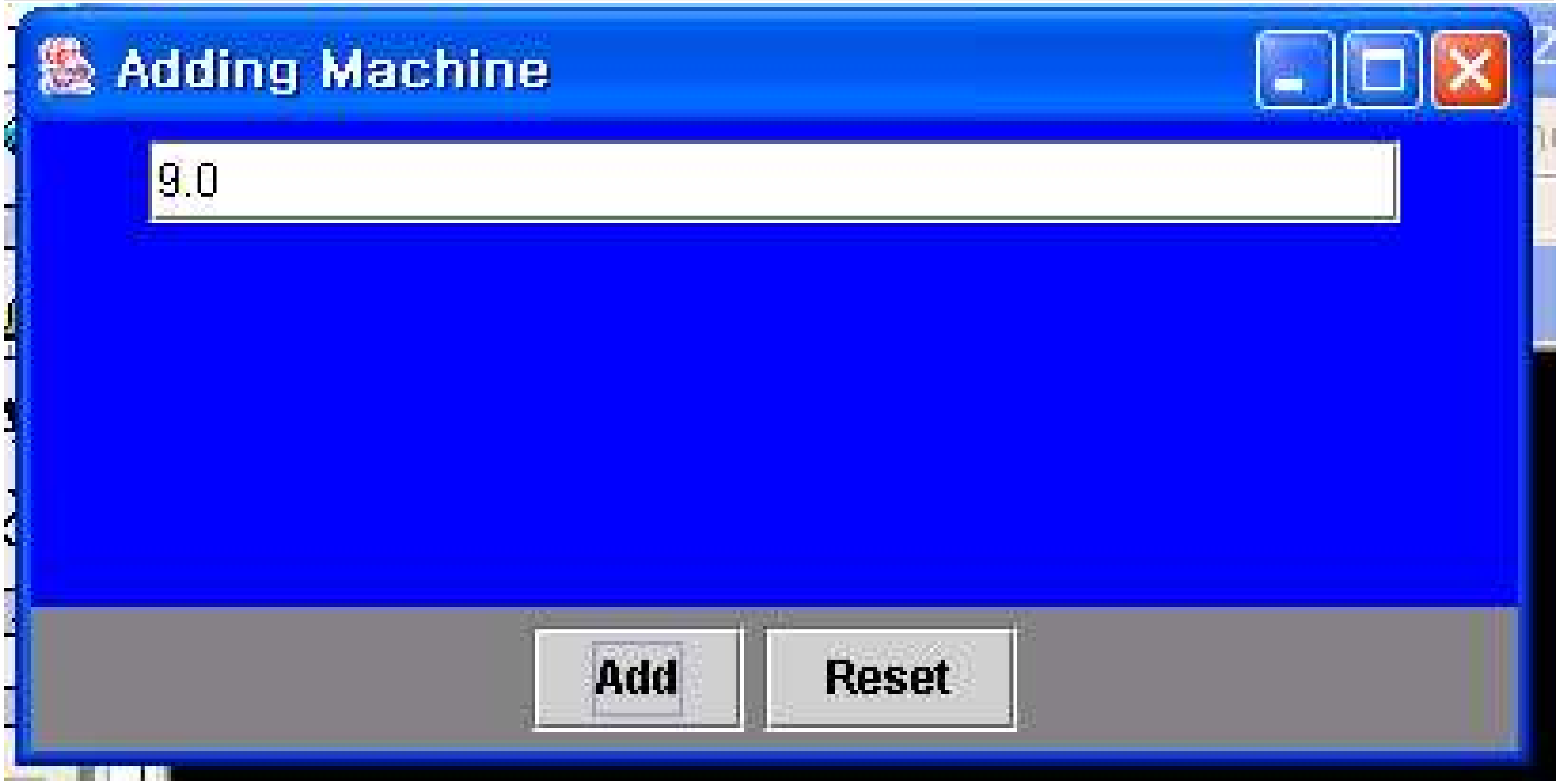
Listing 13.12 An Addition GUI - Adder.java



Listing 13.12 An Addition GUI - Adder.java

```
 // Listing 13.12 An Addition GUI

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

/**
 GUI for totaling a series of numbers.
*/
public class Adder extends JFrame implements ActionListener
{
    public static final int WIDTH = 400;  //-Applet
    public static final int HEIGHT = 200;    //-Applet

    private JTextField inputOutputField;
    private double sum = 0;

    public static void main(String[] args)  //-Applet
    {
        Adder guiAdder = new Adder( );
        guiAdder.setVisible(true);
    }                                         //-Applet
```

```
public Adder( )
 {
    setTitle("Adding Machine");                //-Applet
    addWindowListener(new WindowDestroyer( ));   //-Applet
    setSize(WIDTH, HEIGHT);                    //-Applet
    Container contentPane = getContentPane( );
    contentPane.setLayout(new BorderLayout( ));

    JPanel buttonPanel = new JPanel( );
    buttonPanel.setBackground(Color.GRAY);
    buttonPanel.setLayout(new FlowLayout( ));
    JButton addButton = new JButton("Add");
    addButton.addActionListener(this);
    buttonPanel.add(addButton);
    JButton resetButton = new JButton("Reset");
    resetButton.addActionListener(this);
    buttonPanel.add(resetButton);
    contentPane.add(buttonPanel, BorderLayout.SOUTH);

    JPanel textPanel = new JPanel( );
    textPanel.setBackground(Color.BLUE);
    textPanel.setLayout(new FlowLayout( ));

    inputOutputField = new JTextField("Numbers go here.", 30);
    inputOutputField.setBackground(Color.WHITE);
    textPanel.add(inputOutputField);
    contentPane.add(textPanel, BorderLayout.CENTER);
 }
```


Adding Machine
9.0
Add
Reset

```
 public void actionPerformed(ActionEvent e)
  {
     if (e.getActionCommand( ).equals("Add"))
     {
        sum = sum +
           stringToDouble(inputOutputField.getText( ));
        inputOutputField.setText(Double.toString(sum));
     }
     else if (e.getActionCommand( ).equals("Reset"))
     {
        sum = 0;
        inputOutputField.setText("0.0");
     }
     else
        inputOutputField.setText("Error in adder code.");
   }

  private static double stringToDouble(String stringObject)
  {
     return Double.parseDouble(stringObject.trim( ));
  }
}
```

Catching a NumberFormatException

```
Double.parseDouble(stringObject.trim())
```

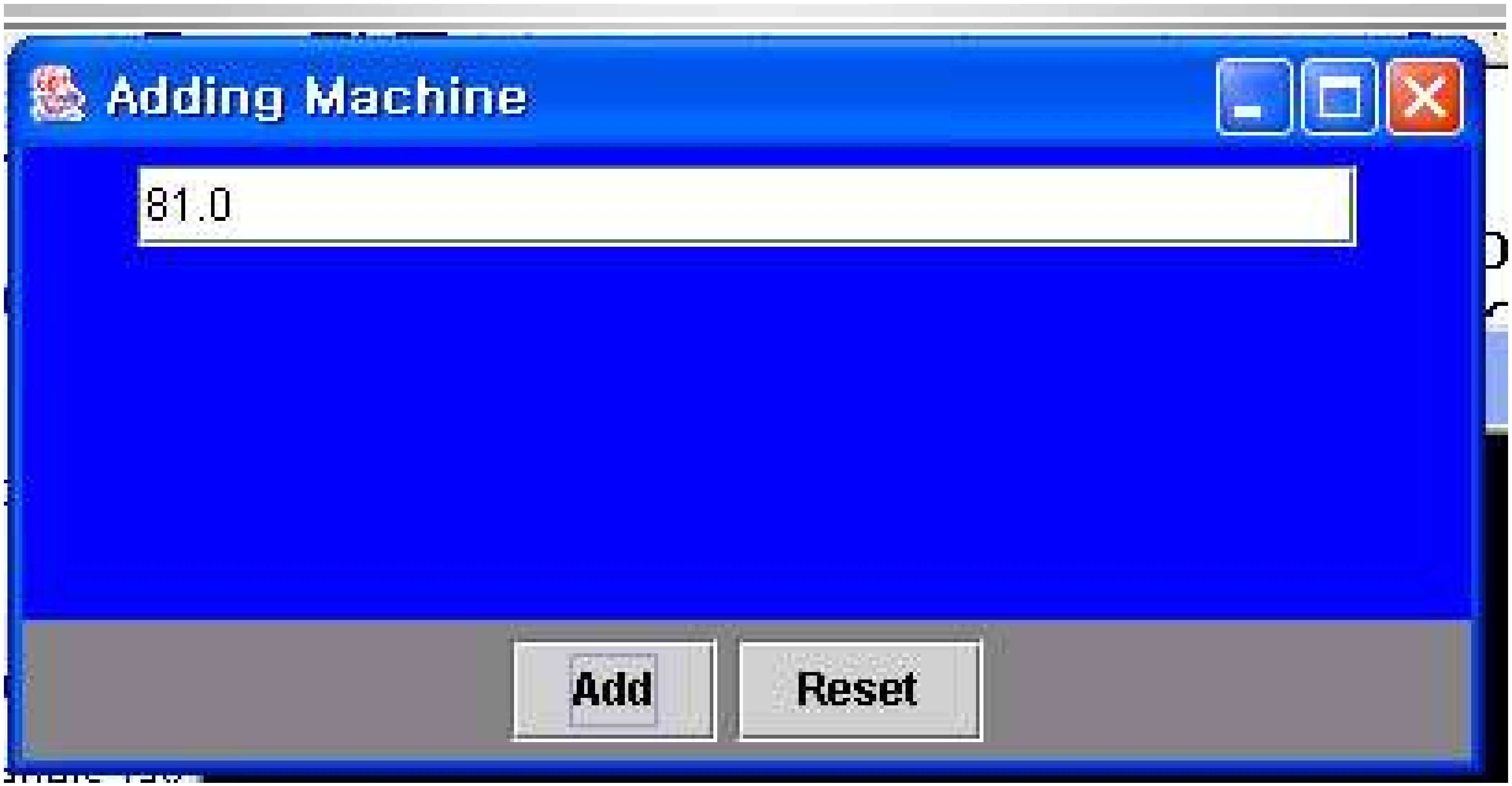
- `parseDouble` and similar methods will throw the `NumberFormatException` if the string is not the proper format for the numeric type
- Your program should catch the exception so that it can do something "graceful".
  - » display an error message rather than crashing
- Methods that throw the `NumberFormatException` do not have to have a `throws` clause.
  - » java does not require you to declare a run-time exception in a throws clause.

Listing 13.13A GUI with Exception Handling - ImprovedAdder.java

```
 // Listing 13.13 A GUI with Exception Handling
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/**
 GUI for totaling a series of numbers. If the user
 enters a number in an incorrect format, such as
 2,000 with a comma, then an error message is generated
 and the user can restart the computation.
*/
public class ImprovedAdder extends JFrame
                       implements ActionListener
{
   public static final int WIDTH = 400;
   public static final int HEIGHT = 200;

   private JTextField inputOutputField;
   private double sum = 0;

   public static void main(String[] args)
   {
      ImprovedAdder guiAdder = new ImprovedAdder( );
      guiAdder.setVisible(true);
   }
```

```
public ImprovedAdder( )
{
    setTitle("Adding Machine");
    addWindowListener(new WindowDestroyer( ));
    setSize(WIDTH, HEIGHT);
    Container contentPane = getContentPane( );
    contentPane.setLayout(new BorderLayout( ));

    JPanel buttonPanel = new JPanel( );
    buttonPanel.setBackground(Color.GRAY);
    buttonPanel.setLayout(new FlowLayout( ));
    JButton addButton = new JButton("Add");
    addButton.addActionListener(this);
    buttonPanel.add(addButton);
    JButton resetButton = new JButton("Reset");
    resetButton.addActionListener(this);
    buttonPanel.add(resetButton);
    contentPane.add(buttonPanel, BorderLayout.SOUTH);

    JPanel textPanel = new JPanel( );
    textPanel.setBackground(Color.BLUE);
    textPanel.setLayout(new FlowLayout( ));

    inputOutputField = new JTextField("Numbers go here.", 30);
    inputOutputField.setBackground(Color.WHITE);
    textPanel.add(inputOutputField);
    contentPane.add(textPanel, BorderLayout.CENTER);
}
```

```
 // this class is identical to the class Adder in display 12.21, except that
//    the name of the class is changed and the method actionPerformed is
changed.

    public void actionPerformed(ActionEvent e)
    {
        try
        {
            tryingCorrectNumberFormats(e);
        }
        catch (NumberFormatException e2)
        {
            inputOutputField.setText("Error: Reenter Number.");
        }
    }
    //This method can throw NumberFormatExceptions.
    //  NumberFormatExceptions do not need to be delcared in a throws
clause,
    //    but they can be caught like other exceptions

    public void tryingCorrectNumberFormats(ActionEvent e)
    {
       if (e.getActionCommand( ).equals("Add"))
       {
          sum = sum +
               stringToDouble(inputOutputField.getText( ));
          inputOutputField.setText(Double.toString(sum));
       }
       else if (e.getActionCommand( ).equals("Reset"))
       {
          sum = 0;
          inputOutputField.setText("0.0");
       }
       else
          inputOutputField.setText("Error in adder code.");
     }

    //This method can throw NumberFormatExceptions.
    private static double stringToDouble(String stringObject)
    {
       return Double.parseDouble(stringObject.trim( ));
    }
}
```


Adding Machine
81.0
Add
Reset

끝