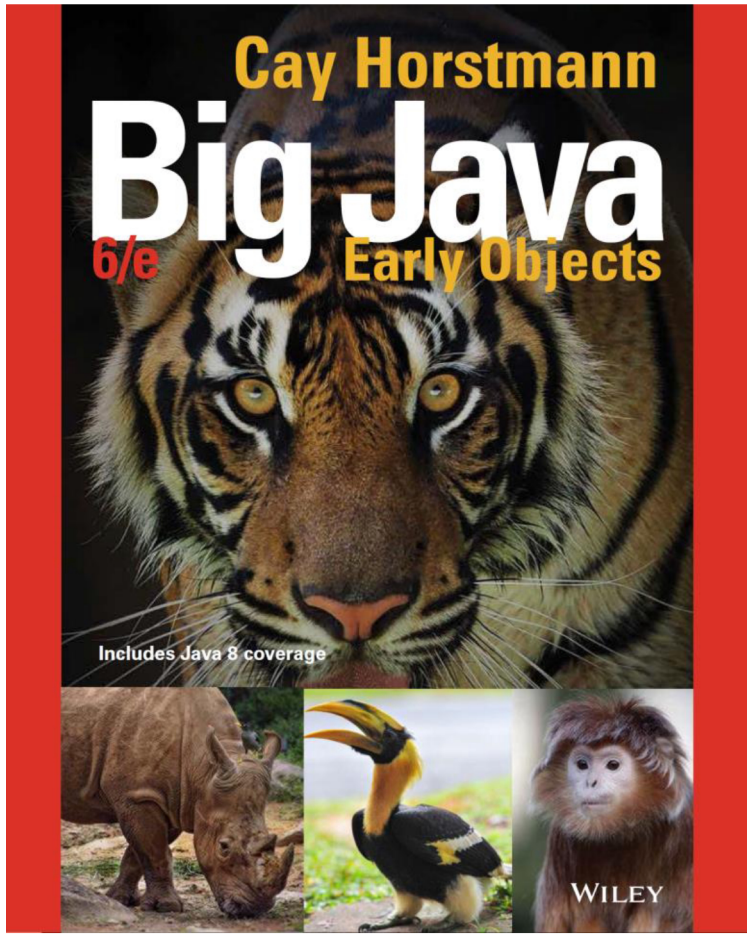

Chapter 9 : Inheritance

Part II (Optional)

Using Inheritance to Customize Frames

- Use inheritance for complex frames to make programs easier to understand
- Design a subclass of `JFrame`
- Store the components as instance fields
- Initialize them in the constructor of your subclass
- If initialization code gets complex, simply add some helper methods

Example: Investment Viewer Program

Lauren – I'm not sure what is supposed to go here.

Example: Investment Viewer Program

Of course, we still need a class with a `main` method:

Lauren – I'm not sure what is supposed to go here.

Example: Investment Viewer Program (cont.)

Lauren – I'm not sure what is supposed to go here.

Self Check 10.18

How many Java source files are required by the investment viewer application when we use inheritance to define the frame class?

Answer: Three: `InvestmentFrameViewer`, `InvestmentFrame`, and `BankAccount`.

Self Check 10.19

Why does the `InvestmentFrame` constructor call `setSize(FRAME_WIDTH, FRAME_HEIGHT)`, whereas the `main` method of the investment viewer class in Chapter 9 called `frame.setSize(FRAME_WIDTH, FRAME_HEIGHT)`?

Answer: The `InvestmentFrame` constructor adds the panel to *itself*.

Processing Text Input

- Use `JTextField` components to provide space for user input

```
final int FIELD_WIDTH = 10; // In characters
      final JTextField rateField = new
         JTextField(FIELD_WIDTH);
```

- Place a `JLabel` next to each text field

```
JLabel rateLabel = new JLabel("Interest Rate: ");
```

Continued

Processing Text Input

- Supply a button that the user can press to indicate that the input is ready for processing

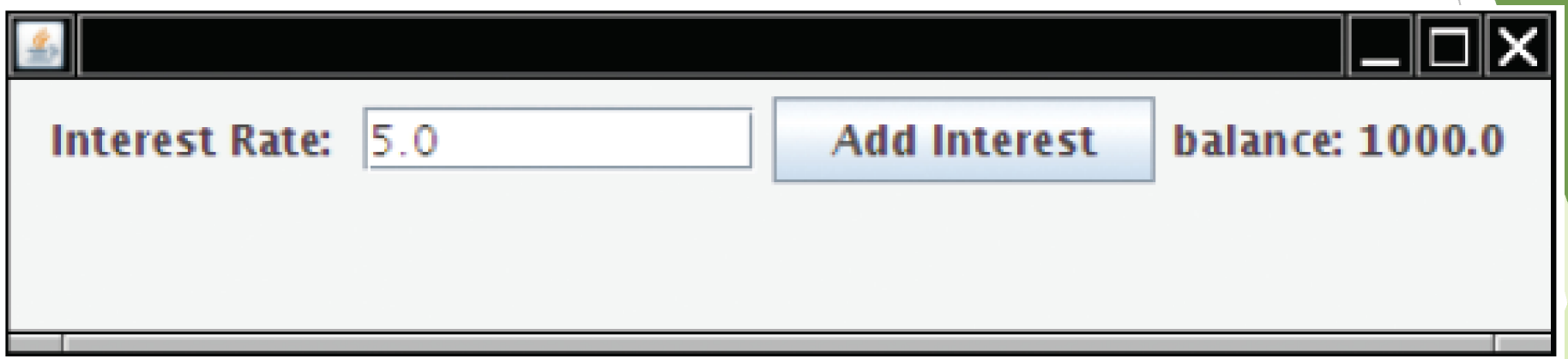

Figure 12 An Application with a Text Field

Continued

Processing Text Input (cont.)

- The button's `actionPerformed` method reads the user input from the text fields (use `getText`)

```
Class AddInterestListener implements ActionListener
{
   public void actionPerformed(ActionEvent event)
   {
      double rate =
         Double.parseDouble(rateField.getText());
      . . .
   }
}
```

ch10/textfield/InvestmentViewer3.java

```
01: import javax.swing.JFrame;
02:
03: /**
04:    This program displays the growth of an investment.
05: */
06: public class InvestmentViewer3
07: {
08:    public static void main(String[] args)
09:    {
10:       JFrame frame = new InvestmentFrame();
11:       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12:       frame.setVisible(true);
13:    }
14: }
```

ch10/textfield/InvestmentFrame.java

```
01: import java.awt.event.ActionEvent;
02: import java.awt.event.ActionListener;
03: import javax.swing.JButton;
04: import javax.swing.JFrame;
05: import javax.swing.JLabel;
06: import javax.swing.JPanel;
07: import javax.swing.JTextField;
08:
09: /**
10:    A frame that shows the growth of an investment with variable
interest.
11: */
12: public class InvestmentFrame extends JFrame
13: {
14:    public InvestmentFrame()
15:    {
16:       account = new BankAccount(INITIAL_BALANCE);
17:
18:       // Use instance fields for components
19:       resultLabel = new JLabel("balance: " + account.getBalance());
20:
```

Continued

ch10/textfield/InvestmentFrame.java (cont.)

```
21:         // Use helper methods
22:         createTextField();
23:         createButton();
24:         createPanel();
25:
26:         setSize(FRAME_WIDTH, FRAME_HEIGHT);
27:     }
28:
29:     private void createTextField()
30:     {
31:         rateLabel = new JLabel("Interest Rate: ");
32:
33:         final int FIELD_WIDTH = 10;
34:         rateField = new JTextField(FIELD_WIDTH);
35:         rateField.setText("" + DEFAULT_RATE);
36:     }
37:
38:     private void createButton()
39:     {
40:         button = new JButton("Add Interest");
41:
```

Continued

ch10/textfield/InvestmentFrame.java (cont.)

```
42:       class AddInterestListener implements ActionListener
43:       {
44:          public void actionPerformed(ActionEvent event)
45:          {
46:             double rate = Double.parseDouble(
47:                   rateField.getText());
48:             double interest = account.getBalance()
49:                   * rate / 100;
50:             account.deposit(interest);
51:             resultLabel.setText(
52:                   "balance: " + account.getBalance());
53:          }
54:       }
55:
56:       ActionListener listener = new AddInterestListener();
57:       button.addActionListener(listener);
58:    }
59:
60:    private void createPanel()
```

Continued

ch10/textfield/InvestmentFrame.java (cont.)

```
61:     {
62:        panel = new JPanel();
63:        panel.add(rateLabel);
64:        panel.add(rateField);
65:        panel.add(button);
66:        panel.add(resultLabel);
67:        add(panel);
68:     }
69:
70:     private JLabel rateLabel;
71:     private JTextField rateField;
72:     private JButton button;
73:     private JLabel resultLabel;
74:     private JPanel panel;
75:     private BankAccount account;
76:
77:     private static final int FRAME_WIDTH = 450;
78:     private static final int FRAME_HEIGHT = 100;
79:
80:     private static final double DEFAULT_RATE = 5;
81:     private static final double INITIAL_BALANCE = 1000;
82: }
```

Self Check 10.20

What happens if you omit the first `JLabel` object?

Answer: Then the text field is not labeled, and the user will not know its purpose.

Self Check 10.21

If a text field holds an integer, what expression do you use to read its contents?

Answer: `Integer.parseInt(textField.getText())`

Text Areas

- Use a `JTextArea` to show multiple lines of text
- You can specify the number of rows and columns:

```
final int ROWS = 10;
final int COLUMNS = 30;
JTextArea textArea = new JTextArea(ROWS, COLUMNS);
```

- `setText`: to set the text of a text field or text area
- `append`: to add text to the end of a text area
- Use `newline` characters to separate lines:

```
textArea.append(account.getBalance() + "\n");
```

- To use for display purposes only:

```
textArea.setEditable(false); // program can call setText
      and append to change it
```

Text Areas

- To add scroll bars to a text area:

```
JTextArea textArea = new JTextArea(ROWS, COLUMNS);
JScrollPane scrollPane = new JScrollPane(textArea);
```

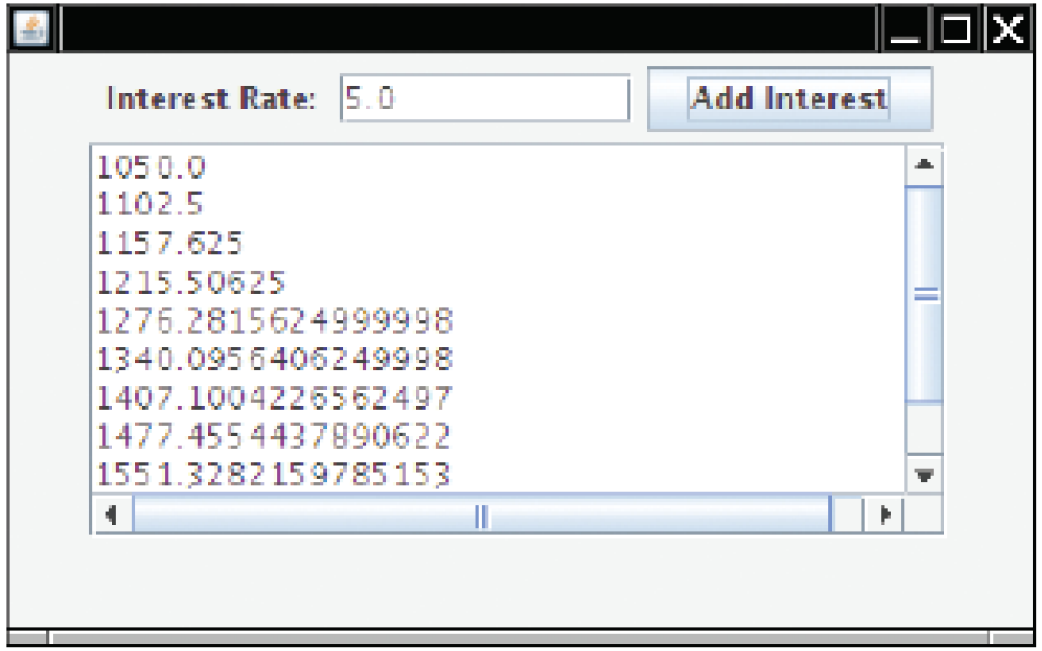

Figure 13 The Investment Application with a Text Area

ch10/textarea/InvestmentFrame.java

```
01: import java.awt.event.ActionEvent;
02: import java.awt.event.ActionListener;
03: import javax.swing.JButton;
04: import javax.swing.JFrame;
05: import javax.swing.JLabel;
06: import javax.swing.JPanel;
07: import javax.swing.JScrollPane;
08: import javax.swing.JTextArea;
09: import javax.swing.JTextField;
10:
11: /**
12:    A frame that shows the growth of an investment with variable
interest.
13: */
14: public class InvestmentFrame extends JFrame
15: {
16:    public InvestmentFrame()
17:    {
18:       account = new BankAccount(INITIAL_BALANCE);
19:       resultArea = new JTextArea(AREA_ROWS, AREA_COLUMNS);
20:       resultArea.setEditable(false);
21:
```

Continued

ch10/textarea/InvestmentFrame.java (cont.)

```
22:         // Use helper methods
23:         createTextField();
24:         createButton();
25:         createPanel();
26:
27:         setSize(FRAME_WIDTH, FRAME_HEIGHT);
28:     }
29:
30:     private void createTextField()
31:     {
32:         rateLabel = new JLabel("Interest Rate: ");
33:
34:         final int FIELD_WIDTH = 10;
35:         rateField = new JTextField(FIELD_WIDTH);
36:         rateField.setText("" + DEFAULT_RATE);
37:     }
38:
39:     private void createButton()
40:     {
41:         button = new JButton("Add Interest");
42:
```

Continued

ch10/textarea/InvestmentFrame.java (cont.)

```
43:       class AddInterestListener implements ActionListener
44:       {
45:          public void actionPerformed(ActionEvent event)
46:          {
47:             double rate = Double.parseDouble(
48:                   rateField.getText());
49:             double interest = account.getBalance()
50:                   * rate / 100;
51:             account.deposit(interest);
52:             resultArea.append(account.getBalance() + "\n");
53:          }
54:       }
55:
56:       ActionListener listener = new AddInterestListener();
57:       button.addActionListener(listener);
58:    }
59:
60:    private void createPanel()
61:    {
62:       panel = new JPanel();
63:       panel.add(rateLabel);
64:       panel.add(rateField);
65:       panel.add(button);
```

Continued

ch10/textarea/InvestmentFrame.java (cont.)

```
66:         JScrollPane scrollPane = new JScrollPane(resultArea);
67:         panel.add(scrollPane);
68:         add(panel);
69:     }
70:
71:     private JLabel rateLabel;
72:     private JTextField rateField;
73:     private JButton button;
74:     private JTextArea resultArea;
75:     private JPanel panel;
76:     private BankAccount account;
77:
78:     private static final int FRAME_WIDTH = 400;
79:     private static final int FRAME_HEIGHT = 250;
80:
81:     private static final int AREA_ROWS = 10;
82:     private static final int AREA_COLUMNS = 30;
83:
84:     private static final double DEFAULT_RATE = 5;
85:     private static final double INITIAL_BALANCE = 1000;
86: }
```

Self Check 10.22

What is the difference between a text field and a text area?

Answer: A text field holds a single line of text; a text area holds multiple lines.

Self Check 10.23

Why did the `InvestmentFrame` program call `resultArea.setEditable(false)`?

Answer: The text area is intended to display the program output. It does not collect user input.

Self Check 10.24

How would you modify the `InvestmentFrame` program if you didn't want to use scroll bars?

Answer: Don't construct a `JScrollPane` and add the `resultArea` object directly to the frame.