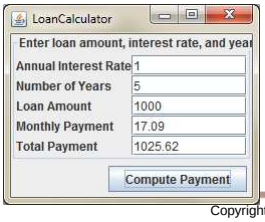
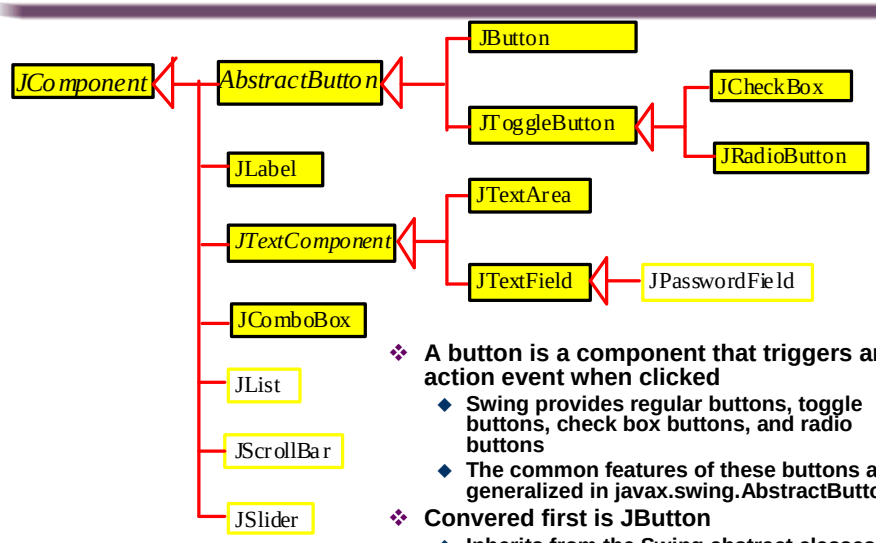
Chapter 17: Creating Graphical User Interfaces

- A graphical user interface (GUI) makes a system user-friendly and easy to use
  - Creating a GUI requires creativity
  - Knowledge of how GUI components work
  - GUI components in Java are very flexible and versatile
- Sections 1 – 8: Swing JComponent coverage:
  - JButton, JCheckBox, JRadioButton
  - JLabel
  - JTextField, JTextArea,
  - JComboBox



 1

Frequently Used GUI Components



- A button is a component that triggers an action event when clicked
  - Swing provides regular buttons, toggle buttons, check box buttons, and radio buttons
  - The common features of these buttons are generalized in javax.swing.AbstractButton
- Convered first is JButton
  - Inherits from the Swing abstract classes AbstractButton and JComponent

 2

AbstractButton UML Diagram

javax.swing.JComponent

javax.swing.AbstractButton

The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

-actionCommand: String	The action command of this button.
-text: String	The button's text (i.e., the text label on the button).
-icon: javax.swing.Icon	The button's default icon. This icon is also used as the "pressed" and "disabled" icon if there is no explicitly set pressed icon.
-pressedIcon: javax.swing.Icon	The pressed icon (displayed when the button is pressed).
-rolloverIcon: javax.swing.Icon	The rollover icon (displayed when the mouse is over the button).
-mnemonic: int	The mnemonic key value of this button. You can select the button by pressing the ALT key and the mnemonic key at the same time.
-horizontalAlignment: int	The horizontal alignment of the icon and text (default: CENTER).
-horizontalTextPosition: int	The horizontal text position relative to the icon (default: RIGHT).
-verticalAlignment: int	The vertical alignment of the icon and text (default: CENTER).
-verticalTextPosition: int	The vertical text position relative to the icon (default: CENTER).
-borderPainted: boolean	Indicates whether the border of the button is painted. By default, a regular button's border is painted, but the borders for a check box and a radio button is not painted.
-iconTextGap: int	The gap between the text and the icon on the button (JDK 1.4).
-selected(): boolean	The state of the button. True if the check box or radio button is selected, false if it's not.

3

JButton Properties and Methods

- JButton Properties inherited from AbstractButton
  - text
  - icon
  - mnemonic
  - horizontalAlignment
  - verticalAlignment
  - horizontalTextPosition
  - verticalTextPosition
  - iconTextGap
- JButton Methods
  - JButton()
  - JButton(String text)
  - JButton(String text, Icon icon)
  - JButton(Icon icon)
  - addActionListener(listener: ActionListener) : void
  - getter and setter Methods inherited from AbstractButton

 4

JButton supports 3 different Icon states

Default icon

Rollover icon

Pressed icon


```
1.   import javax.swing.*;
2.    public class IconSwapper extends JFrame  {
3.      public static void main(String[] args) {
4.        // Create a frame and set its properties
5.        JFrame fraMainWin = new IconSwapper();
6.        fraMainWin.setTitle("ButtonIcons");
7.        fraMainWin.setSize(170, 100);
8.        fraMainWin.setLocationRelativeTo(null); // Center the frame
9.        fraMainWin.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
10.       fraMainWin.setVisible(true);
11.     }
12.     public IconSwapper() {
13.       ImageIcon imgHomeDefault  = new ImageIcon("image/btnDefault.gif");
14.       ImageIcon imgHomeRollover = new ImageIcon("image/btnRollover.gif");
15.       ImageIcon imgHomePressed  = new ImageIcon("image/btnPressed.gif");
16.       JButton btnHome = new JButton("Go Home", imgHomeDefault);
17.       btnHome.setPressedIcon(imgHomePressed);
18.       btnHome.setRolloverIcon(imgHomeRollover);
19.       add(btnHome);
20.     }
21.   }
```

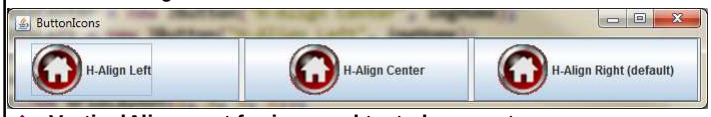
 5

JButton TextPostion

- HorizontalTextPosition for icon and text placement
  - `btnLeft.setHorizontalTextPosition(SwingConstants.LEFT);`
  - setSwingConstants.RIGHT is default but can set CENTER or RIGHT

- VerticalTextPosition for icon and text placement
  - `btnTop.setVerticalTextPosition(SwingConstants.TOP);`
  - setSwingConstants.CENTER is default but can set TOP or BOTTOM

- HorizontalAlignment for icon and text placement
  - `btnLeft.setHorizontalTextPosition(SwingConstants.LEFT);`
  - setSwingConstants. is CENTER default but can set LEFT or RIGHT



- VerticalAlignment for icon and text placement
  - `btnTop.setVerticalTextPosition(SwingConstants.TOP);`
  - setSwingConstants.CENTER is default but can set TOP or BOTTOM

- Tooltips for buttons with `btnHome.setToolTipText("Go Home");`

 6

JCheckBox

JCheckBox inherits all the properties such as text, icon, mnemonic, verticalAlignment, horizontalAlignment, horizontalTextPosition, verticalTextPosition, and selected from AbstractButton, and provides several constructors to create check boxes.

JCheckBox and JRadioButton inherit from abstract class javax.swing.JToggleButton

javax.swing.AbstractButton

↑

javax.swing.JToggleButton

↑

javax.swing.JCheckBox

+JCheckBox()	Creates a default check box button with no text and icon.
+JCheckBox(text: String)	Creates a check box with text.
+JCheckBox(text: String, selected: boolean)	Creates a check box with text and specifies whether the check box is initially selected.
+JCheckBox(icon: Icon)	Creates a checkbox with an icon.
+JCheckBox(text: String, icon: Icon)	Creates a checkbox with text and an icon.
+JCheckBox(text: String, icon: Icon, selected: boolean)	Creates a check box with text and an icon, and specifies whether the check box is initially selected.

7

JRadioButton

- **Radio buttons are variations of check boxes**
  - **Only one button in a group of radiobuttons can be checked at a time**
  - **Grouping of radiobuttons is accomplished using javax.swing.ButtonGroup()**

```
ButtonGroup btg = new ButtonGroup();
btg.add(jrb1);
btg.add(jrb2);
```

javax.swing.AbstractButton

↑

javax.swing.JToggleButton

↑

javax.swing.JRadioButton

+JRadioButton()	Creates a default radio button with no text and icon.
+JRadioButton(text: String)	Creates a radio button with text.
+JRadioButton(text: String, selected: boolean)	Creates a radio button with text and specifies whether the radio button is initially selected.
+JRadioButton(icon: Icon)	Creates a radio button with an icon.
+JRadioButton(text: String, icon: Icon)	Creates a radio button with text and an icon.
+JRadioButton(text: String, icon: Icon, selected: boolean)	Creates a radio button with text and an icon, and specifies whether the radio button is initially selected.

8

JLabel

- A *label* is a display area for a short text, an image, or both
- Many of same format features as JButton, but no interactive events

javax.swing.JComponent

The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

javax.swing.JLabel	
-text: String	The label's text.
-icon: javax.swing.Icon	The label's image icon.
-horizontalAlignment: int	The horizontal alignment of the text and icon on the label.
-horizontalTextPosition: int	The horizontal text position relative to the icon on the label.
-verticalAlignment: int	The vertical alignment of the text and icon on the label.
-verticalTextPosition: int	The vertical text position relative to the icon on the label.
-iconTextGap: int	The gap between the text and the icon on the label (JDK 1.4).
+JLabel()	Creates a default label with no text and icon.
+JLabel(icon: javax.swing.Icon)	Creates a label with an icon.
+JLabel(icon: Icon, hAlignment: int)	Creates a label with an icon and the specified horizontal alignment.
+JLabel(text: String)	Creates a label with text.
+JLabel(text: String, icon: Icon, hAlignment: int)	Creates a label with text, an icon, and the specified horizontal alignment.
+JLabel(text: String, hAlignment: int)	Creates a label with text and the specified horizontal alignment.

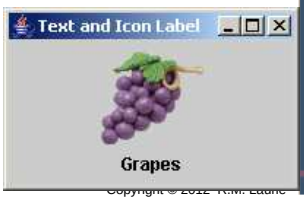
9

Using Labels

```
// Create an image icon from image file
ImageIcon icon = new ImageIcon("image/grapes.gif");

// Create a label with text, an icon,
// with centered horizontal alignment
JLabel jlbl = new JLabel("Grapes", icon,
SwingConstants.CENTER);

// Set label's text alignment and gap between text and icon
jlbl.setHorizontalTextPosition(SwingConstants.CENTER);
jlbl.setVerticalTextPosition(SwingConstants.BOTTOM);
jlbl.setIconTextGap(5);
```



10

JTextField

- A text field is a single line text input area
  - Useful for user to enter in variable data (such as a name)
  - Numbers may be entered but they must be parsed to extract the numerical value
    - `int Integer.parseInt(String);`
    - `double Double.parseDouble(String);`

The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

javax.swing.text.JTextComponent	
-text: String	The text contained in this text component.
-editable: boolean	Indicates whether this text component is editable (default: true).

javax.swing.JTextField	
-columns: int	The number of columns in this text field.
-horizontalAlignment: int	The horizontal alignment of this text field (default: LEFT).
+JTextField()	Creates a default empty text field with number of columns set to 0.
+JTextField(column: int)	Creates an empty text field with specified number of columns.
+JTextField(text: String)	Creates a text field initialized with the specified text.
+JTextField(text: String, columns: int)	Creates a text field initialized with the specified text and columns.

JTextField Properties and Methods

- JTextField Properties inherited from JTextComponent
  - text
  - horizontalAlignment
  - editable
  - columns
- TextField Methods
  - JTextField(int columns) Creates empty text field with fieldwidth
  - JTextField(String text) Creates a text field initialized with text
  - JTextField(String text, int columns)
  - getText() Returns the string from the text field.
  - setText(String text) Puts the given string in the text field.
  - setEditable(boolean editable) By default, editable is true.
  - setColumns(int) Sets the number of columns in this text field

Example: Using Text Fields

Add a text field to the preceding example to let the user set a new message.

JFrame ← ButtonDemo ← CheckBoxDemo ← RadioButtonDemo ← TextFieldDemo

The textbook works though several JComonent demos subclassing previous demos with the new demo. Each inherits the properties and methods Add a text field to the preceding example to let the user set a new message.

If a text field is a password use JPasswordField

 13

JTextArea

- For multiple lines of text, use JTextArea
  - Lets user enter multiple lines of text
  - Can be used for displaying multiple lines of text

javax.swing.text.JTextComponent

The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

javax.swing.JTextArea	
-columns: int	The number of columns in this text area.
-rows: int	The number of rows in this text area.
-tabSize: int	The number of characters used to expand tabs (default: 8).
-lineWrap: boolean	Indicates whether the line in the text area is automatically wrapped (default: false).
-wrapStyleWord: boolean	Indicates whether the line is wrapped on words or characters (default: false).
+JTextArea()	Creates a default empty text area.
+JTextArea(rows: int, columns: int)	Creates an empty text area with the specified number of rows and columns.
+JTextArea(text: String)	Creates a new text area with the specified text displayed.
+JTextArea(text: String, rows: int, columns: int)	Creates a new text area with the specified text and number of rows and columns.
+append(s: String): void	Appends the string to text in the text area.
+insert(s: String, pos: int): void	Inserts string s in the specified position in the text area.
+replaceRange(s: String, start: int, end: int): void	Replaces partial text in the range from position start to end with string s.
+getLineCount(): int	Returns the actual number of lines contained in the text area.

14

JTextArea Properties and Methods

- JTextArea Properties inherited from JTextComponent
  - text
  - editable
  - columns
  - lineWrap
  - wrapStyleWord
  - rows
  - lineCount
  - tabSize

- JTextArea Methods
  - JTextArea(int rows, int columns)
  - JTextArea(String s, int rows, int columns)
  - See textbook 17.7 UML

 15

JComboBox

A *combo box* is a simple list of items from which the user can select one item.
Creates a simple drop down menu

javax.swing.JComponent

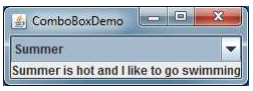
↑

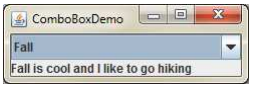
javax.swing.JComboBox	
+JComboBox()	Creates a default empty combo box.
+JComboBox(items: Object[])	Creates a combo box that contains the elements in the specified array.
+addItem(item: Object): void	Adds an item to the combo box.
+getItemAt(index: int): Object	Returns the item at the specified index.
+getItemCount(): int	Returns the number of items in the combo box.
+getSelectedIndex(): int	Returns the index of the selected item.
+setSelectedIndex(index: int): void	Sets the selected index in the combo box.
+getSelectedItem(): Object	Returns the selected item.
+setSelectedItem(item: Object): void	Sets the selected item in the combo box.
+removeItem(anObject: Object): void	Removes an item from the item list.
+removeItemAt(anIndex: int): void	Removes the item at the specified index in the combo box.
+removeAllItems(): void	Removes all items in the combo box.

16

Combo Box Demo

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ComboBoxDemo extends JFrame {
  private String[]  sSeasons = {"Summer", "Fall", "Winter","Spring"};
  private String[]  sDescription = new String[4];
  private JComboBox cboSeasons = new JComboBox(sSeasons);
  private JLabel    lblSports = new JLabel();
  public static void main(String[] args) {
    ComboBoxDemo fraMain = new ComboBoxDemo();
    fraMain.pack();
    fraMain.setTitle("ComboBoxDemo");
    fraMain.setLocationRelativeTo(null); // Center the frame
    fraMain.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fraMain.setVisible(true);
  }
  public ComboBoxDemo() {
    sDescription[0] = "Summer is hot and I like to go swimming";
    sDescription[1] = "Fall is cool and I like to go hiking";
    sDescription[2] = "Winter is freezing and I like to XC skiing";
    sDescription[3] = "Spring is the perfect time to bicycle";
    setDisplay(0);
    setLayout(new BorderLayout());
    add(cboSeasons, BorderLayout.NORTH);
    add(lblSports, BorderLayout.SOUTH);
    cboSeasons.addItemListener( new ItemListener() {
      public void itemStateChanged(ItemEvent e) {
        setDisplay(cboSeasons.getSelectedIndex());
      }
    });
  }
  public void setDisplay(int index) {
    lblSports.setText(sDescription[index]);
  }
}
```





Authors Example : Combo Boxes

This example lets users view an image and a description of a country's flag by selecting the country from a combo box.

 18