Chapter 9: Creating User Interfaces

- What is JavaBean?
- JComponent
- JButton
- ImageIcon
- JLabel
- JTextField
- JTextArea
- JComboBox
- JList
- JCheckBox
- JRadioButton
- Menus
- Creating Multiple Windows
- JScrollBar
- JScrollPane

What is a JavaBean?

All Swing components are JavaBeans. JavaBeans are very useful either in GUI Java application or internet Web applications.

What is a JavaBean?

A JavaBean component is just a Java class that meets the following requirements.

Minimum requirements

1. Bean must be public
2. Must have either a public default constructor with the signature, or no constructor if its superclass has a default constructor
3. Must implement java.io.Serializble or java.io.Externalizable interface
4. Usually has properties with correctly constructed public accessor methods
5. With public registration methods

Optional requirements

Why JavaBeans?

Three common methods:

getPropertyNmae()

isPropertyName()

setPropertyName()

Why JavaBeans?

The JavaBeans technology was developed to enable the programmers to rapidly build applications by assembling objects and test them during design time, thus making reuse of the software more productive.

`JComponent` Properties

- toolTipText
- font
- background
- foreground
- doubleBuffered
- border
- preferredSize
- minimumSize
- maximumSize

JButton

A *button* is a component that triggers an action event when clicked. The following are `JButton` non-default constructors:

```
JButton(String text)

JButton(String text, Icon icon)

JButton(Icon icon)
```

Icon icon=new ImageIcon("photo.gif");

Icon icon=new ImageIcon("photo.jpg");

Fixed-size picture

Example 9.1: Using Buttons

JButton

JButton is a subclass of Jcomponent. Therefore all the properties in JComponent can be used in JButton.

- Text (Lable on the button)
- Icon (image icon on the button)
- Mnemonic (ALT and mnemonic key, as short key)
- horizontalAlignment method(SwingConstants.LEFT, SwingConstants.RIGHT, SwingConstants.CENTER)

JButton Properties

- verticalAlignment (method) (SwingConstants.TOP, SwingConstants.BUTTON, SwingConstants.CENTER)
- horizontalTextPosition (method) (SwingConstants.LEFT, SwingConstants.RIGHT, SwingConstants.CENTER)
- verticalTextPosition (method) (SwingConstants.TOP, SwingConstants.BUTTON, SwingConstants.CENTER)

Responding to `JButton` Events

```
public void actionPerformed(ActionEvent e)
{
  // Get the button label
  String actionCommand = e.getActionCommand();

  // Make sure the event source is a button
  if (e.getSource() instanceof JButton)
    // Make sure it is the right button
  if ("My Button".equals(actionCommand)
      System.out.println ("Button pressed!");
}
```

```
// ButtonDemo.java: Use buttons to move message in a panel
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.*;

public class ButtonDemo extends JFrame
  implements ActionListener
{
  // Declare a panel for displaying message
  private MessagePanel messagePanel;

  // Declare two buttons to move the message left and right
  private JButton jbtLeft, jbtRight;
```

```
// Main method
 public static void main(String[] args)
 {
   ButtonDemo frame = new ButtonDemo();
   // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.pack();
   frame.setVisible(true);
 }

 public ButtonDemo()
 {
   setTitle("Button Demo");

   // Create a MessagePanel instance and set colors
   messagePanel = new MessagePanel("Welcome to Java");
   messagePanel.setBackground(Color.yellow);
```

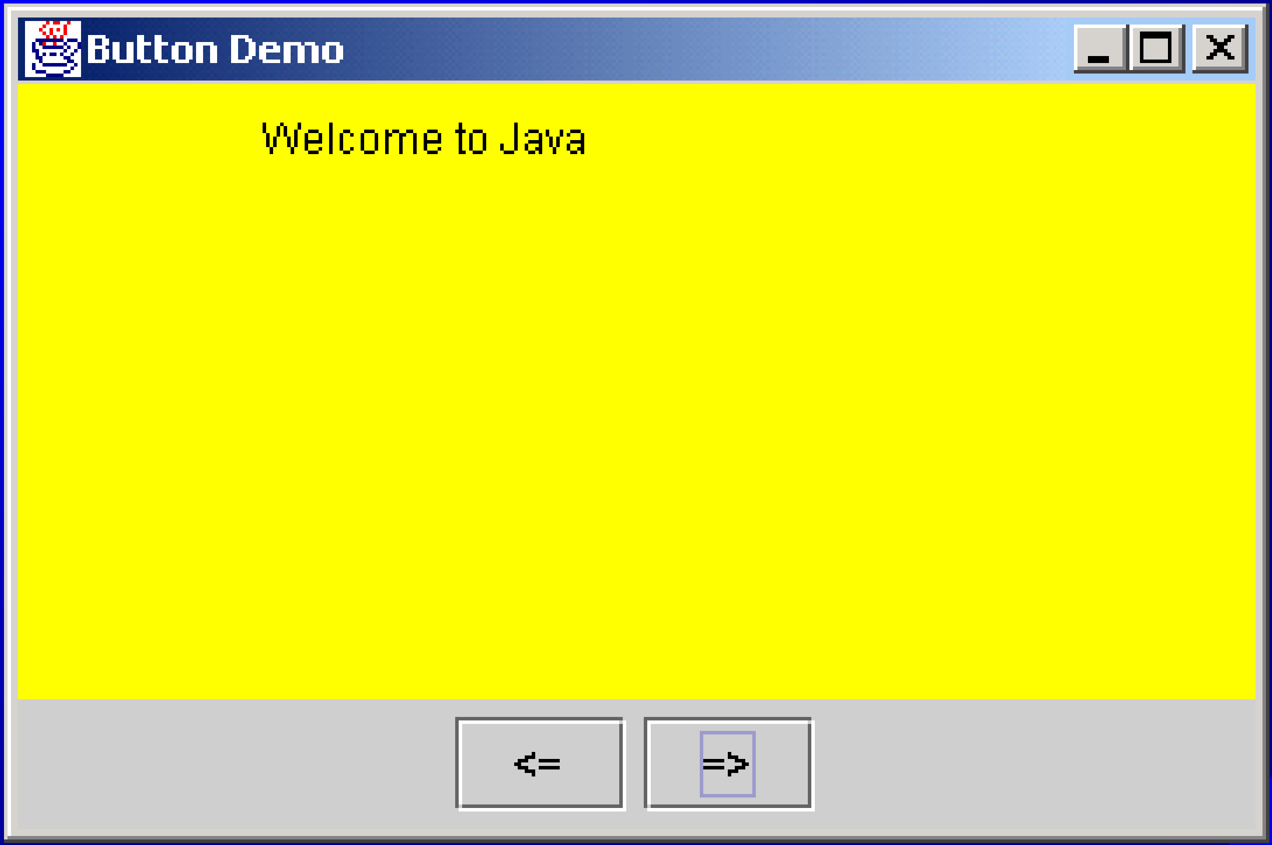
```
// Create Panel jpButtons to hold two Buttons "<=" and "right =>"
JPanel jpButtons = new JPanel();
jpButtons.setLayout(new FlowLayout());
jpButtons.add(jbtLeft = new JButton());
jpButtons.add(jbtRight = new JButton());

// Set button text
jbtLeft.setText("<=");
jbtRight.setText("=>");

// Set keyboard mnemonics
jbtLeft.setMnemonic('L');
jbtRight.setMnemonic('R');
```

```
// Set icons
  //jbtLeft.setIcon(new ImageIcon("images/left.gif"));
  //jbtRight.setIcon(new ImageIcon("images/right.gif"));

  // Set toolTipText on the "<=" and "=>" buttons
  jbtLeft.setToolTipText("Move message to left");
  jbtRight.setToolTipText("Move message to right");

  // Place panels in the frame
  getContentPane().setLayout(new BorderLayout());
  getContentPane().add(messagePanel, BorderLayout.CENTER);
  getContentPane().add(jpButtons, BorderLayout.SOUTH);

  // Register listeners with the buttons
  jbtLeft.addActionListener(this);
  jbtRight.addActionListener(this);
}
```

```
// Handle button events
 public void actionPerformed(ActionEvent e)
 {
  if (e.getSource() == jbtLeft)
  {
   left();
  }
  else if (e.getSource() == jbtRight)
  {
   right();
  }
 }
```

```
// Move the message in the panel left
  private void left()
  {
    int x = messagePanel.getXCoordinate();
    if (x > 10)
    {
      // Shift the message to the left
      messagePanel.setXCoordinate(x-10);
      messagePanel.repaint();
    }
  }
```

```
// Move the message in the panel right
private void right()
{
  int x = messagePanel.getXCoordinate();
  if (x < getSize().width - 20)
  {
    // Shift the message to the right
    messagePanel.setXCoordinate(x+10);
    messagePanel.repaint();
  }
}
}
```


Button Demo
Welcome to Java
<=
=>

JLabel

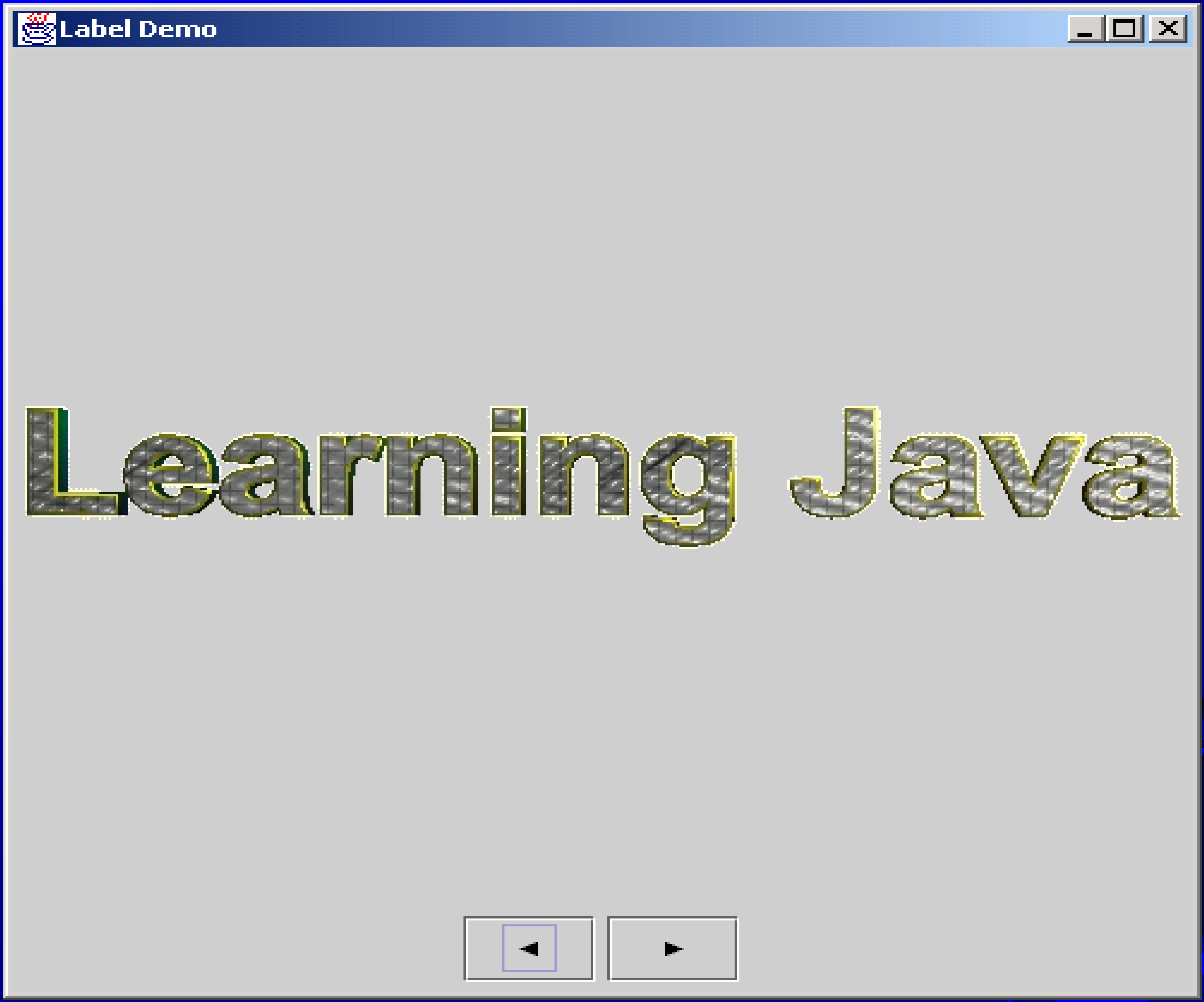
A *label* is a display area for a short text, an image, or both. The non-default constructors for labels are as follows:

```
JLabel(String text, int horizontalAlignment)

JLabel(String text)

JLabel(Icon icon)

JLabel(Icon icon, int horizontalAlignment)

Example:

JLabel myLabel = new JLabel("Calculate");

Jlabel myLabel =

      new Jlabel(new ImageIcon(images/map.gif");
```

`JLabel` Properties

- `text`
- `icon`
- `horizontalAlignment method`
- `verticalAlignment method`

Example 9.2: Using Labels


Label Demo
Learning Java
◄
►

JTextField

A *text field* is an input area where the user can type in characters. Text fields are useful in that they enable the user to enter in variable data (such as a name or a description).

After you input characters, you have to convert the characters to data, using Numerical Wrap classes.

Example 9.3: Using Text Fields

`JTextField` Constructors

- `JTextField(int columns)`

  Creates an empty text field with the specified number of columns.

- `JTextField(String text)`

  Creates a text field initialized with the specified text.

- `JTextField(String text, int columns)`

  Creates a text field initialized with the specified text and the column size.

`JTextField` Properties

- `text`
- `horizontalAlignment`
- `editable`
- `columns`

`JTextField` Methods

- `getText()`

  Returns the string from the text field.

- `setText(String text)`

  Puts the given string in the text field.

- `setEditable(boolean editable)`

  Enables or disables the text field to be edited. By default, `editable` is `true`.

- `setColumns(int)`

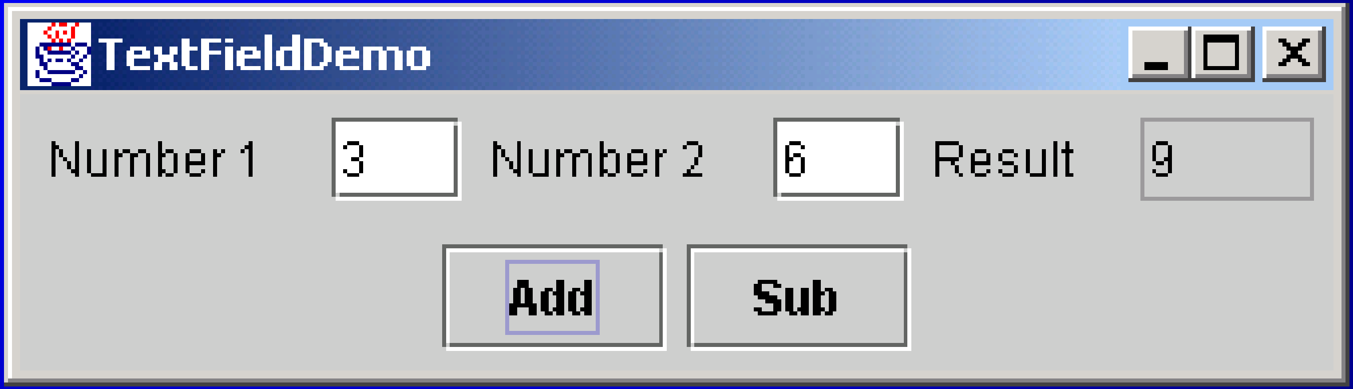
  Sets the number of columns in this text field. The length of the text field is changeable.

```
// TextFieldDemo.java: Add two numbers in the text fields
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TextFieldDemo extends JFrame implements ActionListener
{
  // Declare three text fields
  private JTextField jtfNum1, jtfNum2, jtfResult;
  private JButton jbtAdd; // Declare "Add" button
  private JButton jbtSub;

  // Main method
  public static void main(String[] args)
  {
    TextFieldDemo frame = new TextFieldDemo();
    frame.pack();
```

```
// frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setVisible(true);
 }

 // Constructor
 public TextFieldDemo()
 {
   setTitle("TextFieldDemo");
   setBackground(Color.yellow);
   setForeground(Color.black);

   // Use panel p1 to group text fields
   JPanel p1 = new JPanel();
   p1.setLayout(new FlowLayout());
   p1.add(new Label("Number 1"));
   p1.add(jtfNum1 = new JTextField(3));
   p1.add(new Label("Number 2"));
```

```
p1.add(jtfNum2 = new JTextField(3));
    p1.add(new Label("Result"));
    p1.add(jtfResult = new JTextField(4));
    jtfResult.setEditable(false);   // Set jtfResult noneditable

    // Use panel p2 for the button
    JPanel p2 = new JPanel();
    p2.setLayout(new FlowLayout());
    p2.add(jbtAdd = new JButton("Add"));
    p2.add(jbtSub = new JButton("Sub"));

    // Set FlowLayout for the frame and add panels to the frame
    getContentPane().setLayout(new BorderLayout());
    getContentPane().add(p1, BorderLayout.CENTER);
    getContentPane().add(p2, BorderLayout.SOUTH);
```

```
// Register listener
   jbtAdd.addActionListener(this);
   jbtSub.addActionListener(this);
 }

 // Handle the add operation
 public void actionPerformed(ActionEvent e)
 {
   if (e.getSource() == jbtAdd)
   {
     // Get int values from text fields and use trim() to
     // trim extraneous space in the text field
     int num1 = (Integer.parseInt(jtfNum1.getText().trim()));
     int num2 = (Integer.parseInt(jtfNum2.getText().trim()));
     int result = num1 + num2;
// Set result in TextField jtfResult
    jtfResult.setText(String.valueOf(result));
  }

      if (e.getSource() == jbtSub)
          {
           // Get int values from text fields and use trim() to
           // trim extraneous space in the text field
           int num1 = (Integer.parseInt(jtfNum1.getText().trim()));
           int num2 = (Integer.parseInt(jtfNum2.getText().trim()));
           int result = num1 - num2;

           // Set result in TextField jtfResult
           jtfResult.setText(String.valueOf(result));
  }
 }
}
```


TextFieldDemo
Number 1
3
Number 2
6
Result
9
Add
Sub

`JTextArea`

If you want to let the user enter multiple lines of text, you cannot use text fields unless you create several of them. The solution is to use `JTextArea`, which enables the user to enter multiple lines of text.

`JTextArea` Constructors

- `JTextArea(int rows, int columns)`

  Creates a text area with the specified number of rows and columns.

- `JTextArea(String s, int rows, int columns)`

  Creates a text area with the initial text and the number of rows and columns specified.

`JTextArea` Properties

- `text`
- `editable`
- `columns`
- `lineWrap`
- `wrapStyleWord`
- `rows`
- `lineCount`
- `tabSize`

JTextArea mehtods

```
public void insert(String,int pos)

public void append(String s);

public void replaceRange
   (String s,int start, int end)
```

JTextArea mehtods

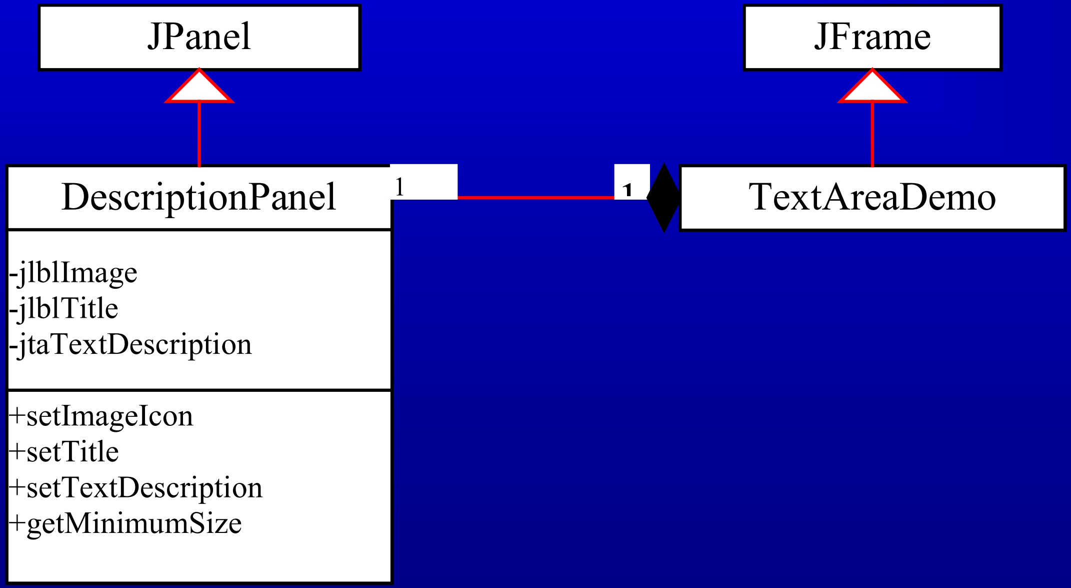
JTextArea does not handle scrolling;

Need JScrollPane object to hold an instant of JTextArea and let JScrollPane handle scrolling for JTextArea.

```
JScrollPane scrollPane=new JScrollPane(jta =new JTextArea());
getContentPane.add(scrollPane,BorderLayout.CENTER);
```

Example 9.4 Using Text Areas

- This example gives a program that displays an image in a label, a title in a label, and a text in a text area.



```
// Define a panel for displaying image and text
import javax.swing.*;
import java.awt.*;

public class DescriptionPanel extends JPanel
{
  // Label for displaying an image icon
  private JLabel jlblImage = new JLabel();
  // Label for displaying a title
  private JLabel jlblTitle = new JLabel();
  // Text area for displaying text
  private JTextArea jtaTextDescription  = new JTextArea();
```

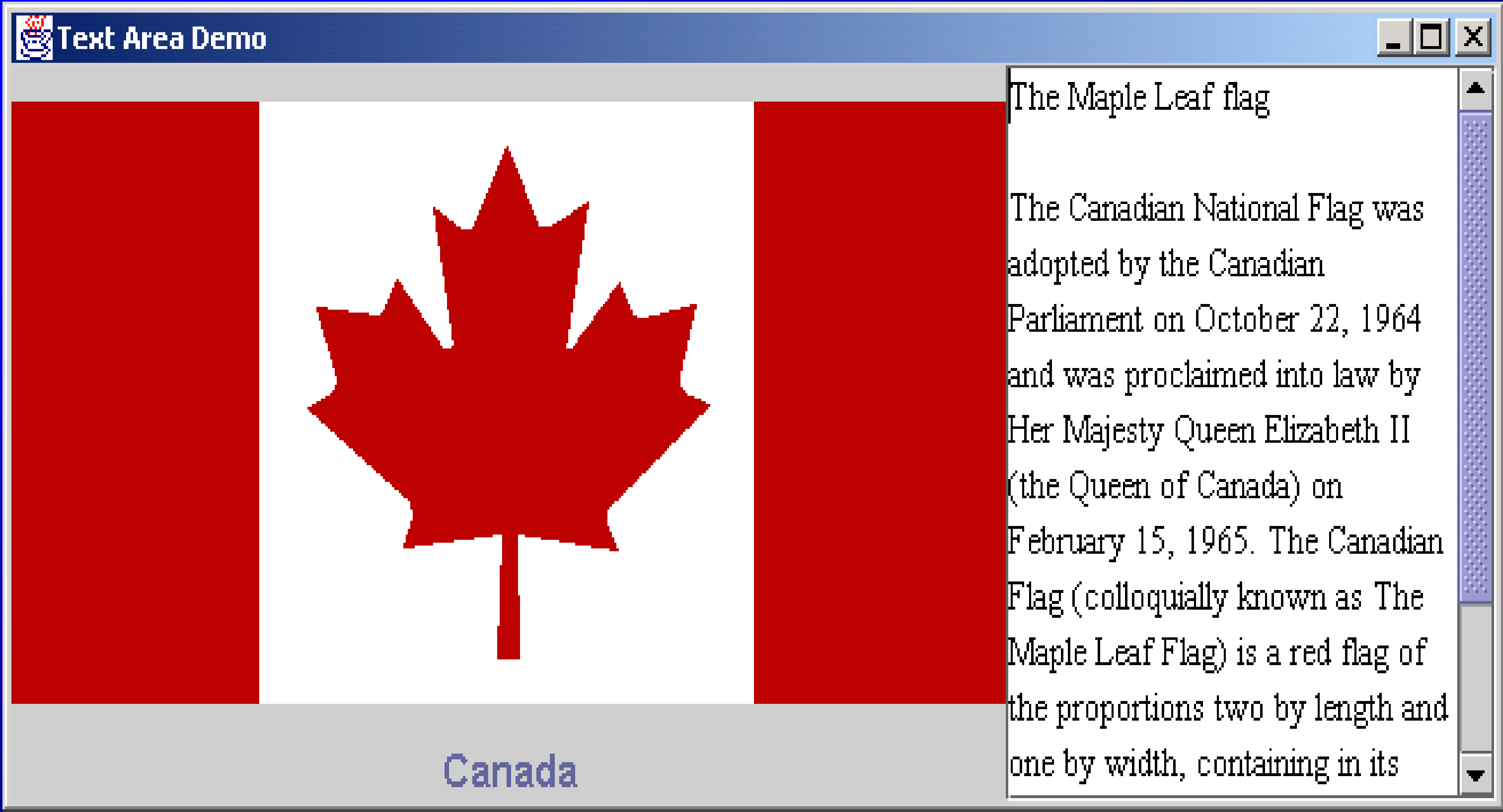
```
// Default constructor
 public DescriptionPanel()
  {
   // Group image label and title label in a panel
   JPanel panel = new JPanel();
   panel.setLayout(new BorderLayout());
   panel.add(jlblImage, BorderLayout.CENTER);
   panel.add(jlblTitle, BorderLayout.SOUTH);

   // Create a scroll pane to hold text area
   JScrollPane scrollPane = new JScrollPane
     (jtaTextDescription);

   // Center the title on the label
   jlblTitle.setHorizontalAlignment(JLabel.CENTER);
```

```
// Set the font for the title and text
  jlblTitle.setFont(new Font("SansSerif", Font.BOLD, 16));
  jtaTextDescription.setFont(new Font("Serif", Font.PLAIN, 14));

  // Set lineWrap and wrapStyleWord true for text area
  jtaTextDescription.setLineWrap(true);
  jtaTextDescription.setWrapStyleWord(true);

  // Set preferred size for the scroll pane
  scrollPane.setPreferredSize(new Dimension(200, 100));

  // Set BorderLayout for the whole panel, add panel and scrollpane
  setLayout(new BorderLayout());
  add(scrollPane, BorderLayout.CENTER);
  add(panel, BorderLayout.WEST);
 }
```

```
public void setTitle(String title) // Set the title
  {
    jlblTitle.setText(title);
  }
public void setImageIcon(ImageIcon icon) // Set the image icon
  {
    jlblImage.setIcon(icon);
    Dimension dimension = new Dimension(icon.getIconWidth(),
      icon.getIconHeight());
    jlblImage.setPreferredSize(dimension);
  }
  // Set the text description
  public void setTextDescription(String text)
  {
    jtaTextDescription.setText(text);
  }
}
```

```
// TextAreaDemo.java: Display an image in a label, the title for
// the image in a label, and the discription of the image in a
// text area
import java.awt.*;
import javax.swing.*;

public class TextAreaDemo extends JFrame
{
  // Declare and create a description panel
  private DescriptionPanel descriptionPanel = new DescriptionPanel();

  // Main method
  public static void main(String[] args)
  {
    TextAreaDemo frame = new TextAreaDemo();
    frame.pack();
```

```
// frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setTitle("Text Area Demo");
   frame.setVisible(true);
 }

 // Constructor
 public TextAreaDemo()
 {
   // Set title, text and image in the description panel
   descriptionPanel.setTitle("Canada");
   String description = "The Maple Leaf flag \n\n" +
     "The Canadian National Flag was adopted by the Canadian " +
     "Parliament on October 22, 1964 and was proclaimed into law " +
     "by Her Majesty Queen Elizabeth II (the Queen of Canada) on " +
     "February 15, 1965. The Canadian Flag (colloquially known " +
     "as The Maple Leaf Flag) is a red flag of the proportions " +
"two by length and one by width, containing in its centre a " +
     "white square, with a single red stylized eleven-point " +
     "mapleleaf centred in the white square.";
   descriptionPanel.setTextDescription(description);
   descriptionPanel.setImageIcon(new ImageIcon("images/ca.gif"));

   // Add the description panel to the frame
   getContentPane().setLayout(new BorderLayout());
   getContentPane().add(descriptionPanel, BorderLayout.CENTER);
 }
}
```



The Maple Leaf flag

The Canadian National Flag was adopted by the Canadian Parliament on October 22, 1964 and was proclaimed into law by Her Majesty Queen Elizabeth II (the Queen of Canada) on February 15, 1965. The Canadian Flag (colloquially known as The Maple Leaf Flag) is a red flag of the proportions two by length and one by width, containing in its

JComboBox

A *combo box* is a simple list of items from which the user can choose. It performs basically the same function as a list, but can get only one value. To create a choice, use its default constructor:

```
JComboBox()
```

Or use the following constructor to create a list with a set of string

```
JComboBox(Object[] stringItems)
```

Example 9.5: Using Combo Boxes

`JComboBox` properties

selectedIndex: int value indicating the index of the selected item in the combo box

selectedItem: selected item whose type ise Object

`JComboBox` Methods

To add an item to a `JComboBox jcbo`, use

`jcbo.addItem(Object item)`

To get an item from `JComboBox jcbo`, use

`jcbo.getItem()`

To remove an item from `JComboBox jcbo`, use

`jcbo.removeItem()`

To remove all items from `JComboBox jcbo`, use `jcbo.removeAllItems()`

Using the `itemStateChanged` Handler

When a choice is checked or unchecked, `itemStateChanged()` for `ItemEvent` is invoked as well as the `actionPerformed()` handler for `ActionEvent`.

```
public void itemStateChanged(ItemEvent e)
{
  // Make sure the source is a combo box
  if (e.getSource() instanceof JComboBox)
    String s = (String)e.getItem();
}
```

```
// ComboBoxDemo.java: Use a combo box to select a country and
// display the selected country's flag information
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ComboBoxDemo extends JFrame implements ItemListener
{
  // Declare an array of Strings for flag titles
  private String[] flagTitle = {"Canada", "China", "Denmark",
    "France", "Germany", "India", "Norway", "United Kingdom",
    "United States of America"};

  // Declare an ImageIcon array for the national flags of 9 countries
  private ImageIcon[] flagImage = new ImageIcon[9];
```

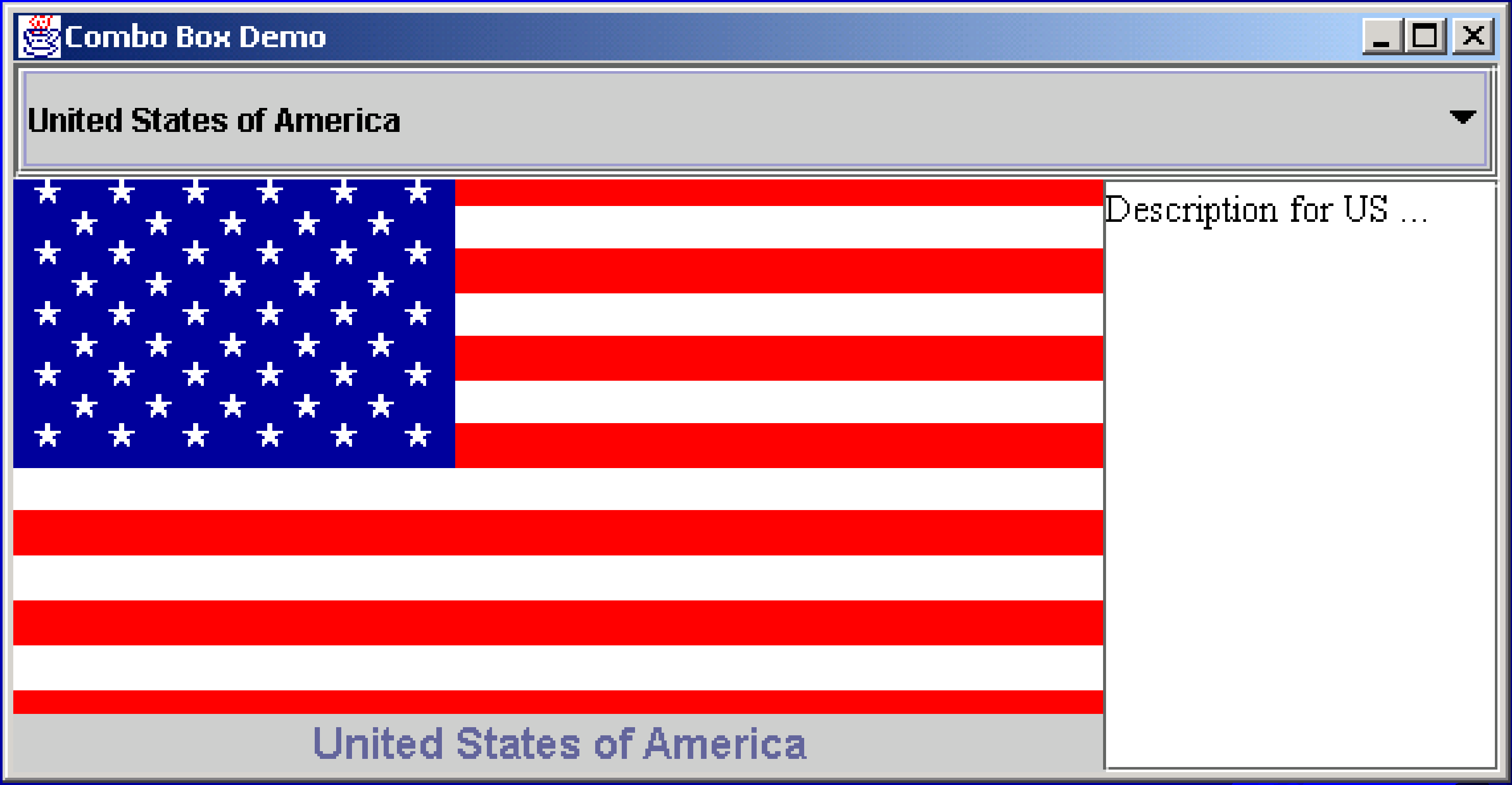
```
// Declare an array of strings for flag descriptions
 private String[] flagDescription = new String[9];

 // Declare and create a description panel
 private DescriptionPanel descriptionPanel = new DescriptionPanel();

 // The combo list for selecting countries
 private JComboBox jcbo;

 // Main Method
 public static void main(String[] args)
 {
   ComboBoxDemo frame = new ComboBoxDemo();
   frame.pack();
   frame.setTitle("Combo Box Demo");
   // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setVisible(true);
 }
```

```
// Default Constructor
public ComboBoxDemo()
{
  // Load images info flagImage array
  flagImage[0] = new ImageIcon("images/ca.gif");
  flagImage[1] = new ImageIcon("images/china.gif");
  flagImage[2] = new ImageIcon("images/denmark.gif");
  flagImage[3] = new ImageIcon("images/fr.gif");
  flagImage[4] = new ImageIcon("images/germany.gif");
  flagImage[5] = new ImageIcon("images/india.gif");
  flagImage[6] = new ImageIcon("images/norway.gif");
  flagImage[7] = new ImageIcon("images/uk.gif");
  flagImage[8] = new ImageIcon("images/us.gif");
```

```
// Set text description
  flagDescription[0] = "The Maple Leaf flag \n\n" +
    "The Canadian National Flag was adopted by the Canadian " +
    "Parliament on October 22, 1964 and was proclaimed into law " +
    "by Her Majesty Queen Elizabeth II (the Queen of Canada) on " +
    "February 15, 1965. The Canadian Flag (colloquially known " +
    "as The Maple Leaf Flag) is a red flag of the proportions " +
    "two by length and one by width, containing in its centre a " +
    "white square, with a single red stylized eleven-point " +
    "mapleleaf centred in the white square.";
  flagDescription[1] = "Description for China ... ";
  flagDescription[2] = "Description for Denmark ... ";
  flagDescription[3] = "Description for France ... ";
  flagDescription[4] = "Description for Germany ... ";
  flagDescription[5] = "Description for India ... ";
  flagDescription[6] = "Description for Norway ... ";
  flagDescription[7] = "Description for UK ... ";
  flagDescription[8] = "Description for US ... ";
```

```
  // Create items into the combo box
  jcbo = new JComboBox(flagTitle);

  // Set the first country (Canada) for display
  setDisplay(8);

  // Add combo box and description panel to the list
  getContentPane().add(new JScrollPane(jcbo),
                                          BorderLayout.NORTH);
  getContentPane().add(descriptionPanel, BorderLayout.CENTER);

  // Register listener
  jcbo.addItemListener(this);
}
```

```
// Handle item selection
public void itemStateChanged(ItemEvent e)
{
  setDisplay(jcbo.getSelectedIndex());
}

// Set display information on the description panel
public void setDisplay(int index)
{
  descriptionPanel.setTitle(flagTitle[index]);
  descriptionPanel.setImageIcon(flagImage[index]);
  descriptionPanel.setTextDescription(flagDescription[index]);
}
}
```



Description for US ...

JList

A *list* is a component that performs basically the same function as a combo box, but it enables the user to choose a single value or multiple values.

Example 9.6: Using Lists

`JList` Constructors

- `JList()`

  Creates an empty list.

- `JList(Object[] stringItems)`

  Creates a new list initialized with items.

`JList` Properties

- `selectedIndexd`
- `selectedIndices`
- `selectedValue`
- `selectedValues`
- `selectionMode`
- `visibleRowCount`

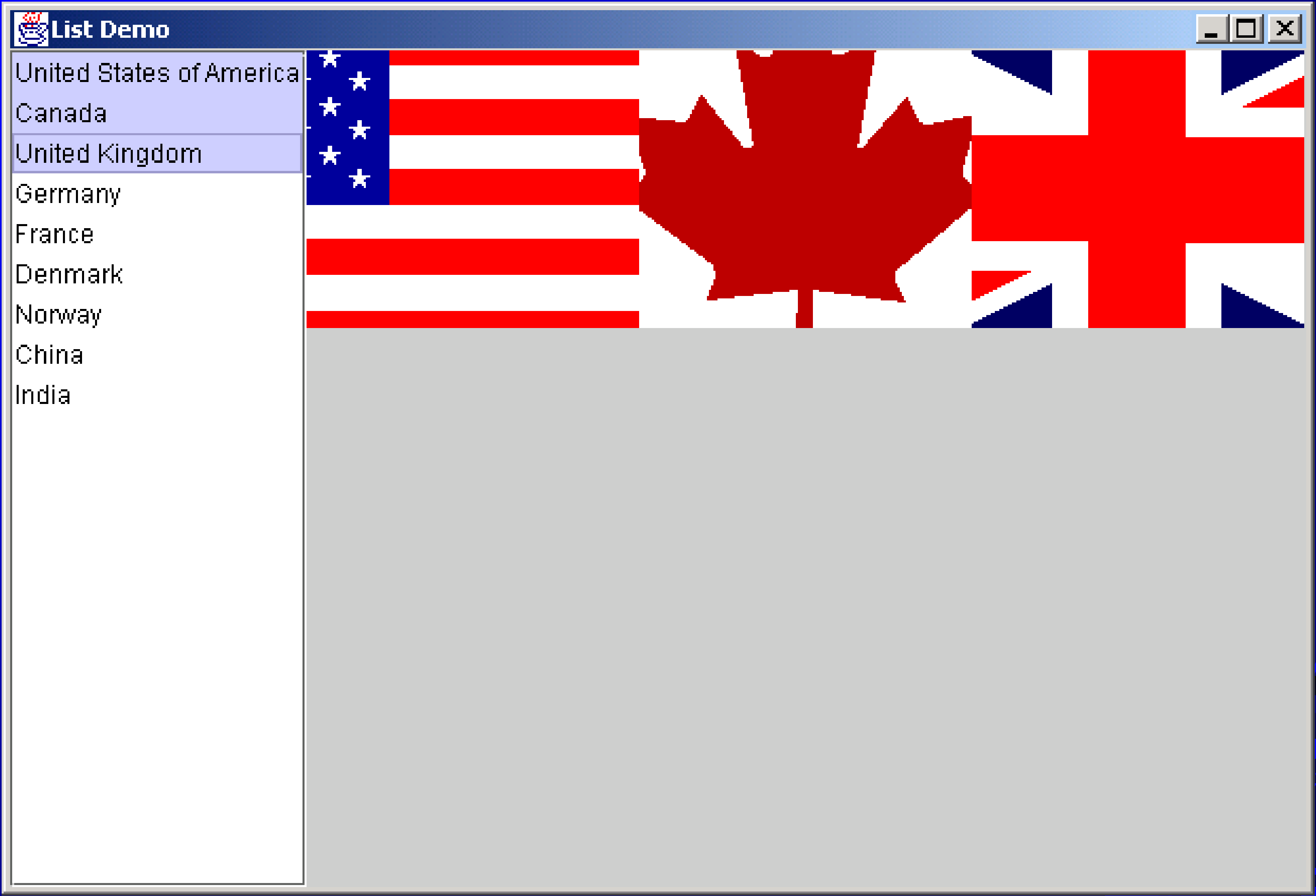
```
// ListDemo.java: Use list to select a country and display the
// selected country's flag
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
public class ListDemo extends JFrame
                                    implements ListSelectionListener
{
  // Declare an ImageIcon array for the national flags of 9 countries
  private ImageIcon[] imageIcon = new ImageIcon[9];

  // Arrays of labels for displaying images
  private JLabel[] jlblImageViewer = new JLabel[9];

  // The list for selecting countries
  JList jlst;
```

```
// Main Method
public static void main(String[] args)
{
  ListDemo frame = new ListDemo();
  frame.setSize(650, 500);
  frame.setTitle("List Demo");
  // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setVisible(true);
}

// Default Constructor
public ListDemo()
{
  // Load images into imageIcon array
  imageIcon[0] = new ImageIcon("images/us.gif");
```

```
imageIcon[1] = new ImageIcon("images/ca.gif");
  imageIcon[2] = new ImageIcon("images/uk.gif");
  imageIcon[3] = new ImageIcon("images/germany.gif");
  imageIcon[4] = new ImageIcon("images/fr.gif");
  imageIcon[5] = new ImageIcon("images/denmark.gif");
  imageIcon[6] = new ImageIcon("images/norway.gif");
  imageIcon[7] = new ImageIcon("images/china.gif");
  imageIcon[8] = new ImageIcon("images/india.gif");

  // Create a string of country names
  String[] countries = {"United States of America", "Canada",
    "United Kingdom", "Germany", "France", "Denmark", "Norway",
    "China", "India"};

  // Create a list with the country names
  jlst = new JList(countries);
```

```
// Create a panel to hold nine labels
   JPanel p = new JPanel();
   p.setLayout(new GridLayout(3, 3));

   for (int i=0; i<9; i++)
   {
     p.add(jlblImageViewer[i] = new JLabel());
     jlblImageViewer[i].setHorizontalAlignment
       (SwingConstants.CENTER);
   }

   // Add p and the list to the frame
   getContentPane().add(p, BorderLayout.CENTER);
   getContentPane().add(new JScrollPane(jlst), BorderLayout.WEST);
   // Register listeners
   jlst.addListSelectionListener(this);
 }
```

```
// Handle list selection
  public void valueChanged(ListSelectionEvent e)
  {
    int[] indices = jlst.getSelectedIndices(); // Get selected indices
    int i;
    // Set icons in the labels
    for (i=0; i<indices.length; i++)
    {
      jlblImageViewer[i].setIcon(imageIcon[indices[i]]);
    }
    // Remove icons from the rest of the labels
    for (; i<9; i++)
    {
      jlblImageViewer[i].setIcon(null);
    }
  }
}
```



List Demo
United States of America
Canada
United Kingdom
Germany
France
Denmark
Norway
China
India

JCheckBox

A *check box* is a component that enables the user to toggle a choice on or off, like a light switch.

Example 9.7: Using Check Boxes

`JCheckBox` Constructors

- `JCheckBox()`
- `JCheckBox(String text)`
- `JCheckBox(String text, boolean selected)`
- `JCheckBox(Icon icon)`
- `JCheckBox(String text, Icon icon)`
- `JCheckBox(String text, Icon icon, boolean selected)`

`JCheckBox` Properties

`JCheckBox` has all the properties in `JButton`. Additionally, `JButton` has the following property:

`selected`

JCheckBox has has a method called

isSelected() to verify the checking

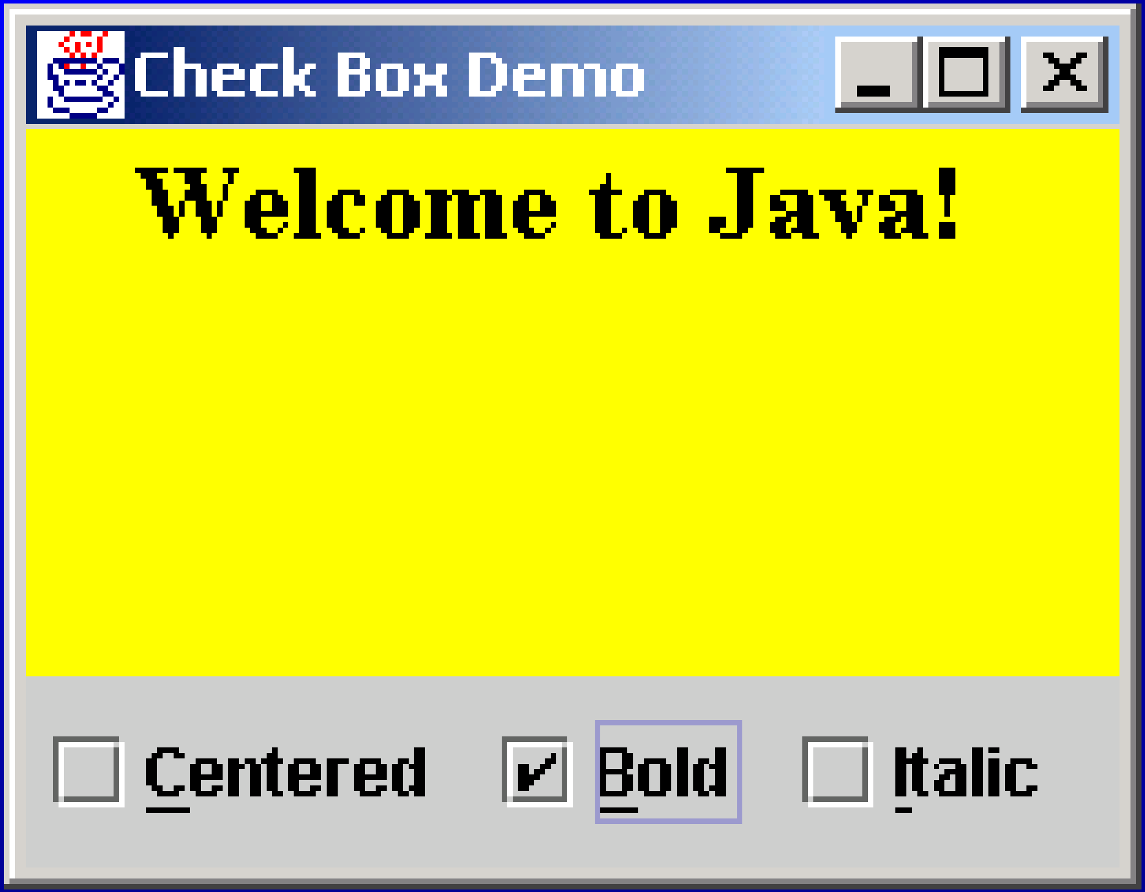
```
// CheckBoxDemo.java: Use check boxes to select one or more choices
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.event.*;
import javax.swing.*;

public class CheckBoxDemo extends JFrame implements ItemListener
{
  // Declare check boxes
  private JCheckBox jchkCentered, jchkBold, jchkItalic;

  // Declare a panel for displaying message
  private MessagePanel messagePanel;
```

```
// Main method
 public static void main(String[] args)
 {
   CheckBoxDemo frame = new CheckBoxDemo();
   // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.pack();
   frame.setVisible(true);
 }
 // Constructor
 public CheckBoxDemo()
 {
   setTitle("Check Box Demo");

   // Create the message panel
   messagePanel = new MessagePanel();
   messagePanel.setMessage("Welcome to Java!");
   messagePanel.setBackground(Color.yellow);
```

```
// Put three check boxes in panel p
   JPanel p = new JPanel();
   p.setLayout(new FlowLayout());
   p.add(jchkCentered = new JCheckBox("Centered"));
   p.add(jchkBold = new JCheckBox("Bold"));
   p.add(jchkItalic = new JCheckBox("Italic"));

   // Set keyboard mnemonics
   jchkCentered.setMnemonic('C');
   jchkBold.setMnemonic('B');
   jchkItalic.setMnemonic('I');

   // Place messagePanel and p in the frame
   getContentPane().setLayout(new BorderLayout());
   getContentPane().add(messagePanel, BorderLayout.CENTER);
   getContentPane().add(p, BorderLayout.SOUTH);
```

```
// Register listeners on jchkCentered, jchkBold, and jchkItalic
  jchkCentered.addItemListener(this);
  jchkBold.addItemListener(this);
  jchkItalic.addItemListener(this);
 }

 // Handle check box selection
 public void itemStateChanged(ItemEvent e)
 {
  if (e.getSource() instanceof JCheckBox)
  {
   // Determine a font style
   int selectedStyle = 0;
   if (jchkBold.isSelected())
    selectedStyle = selectedStyle+Font.BOLD;
   if (jchkItalic.isSelected())
    selectedStyle = selectedStyle+Font.ITALIC;
```

```
      // Set font for the message
      messagePanel.setFont(new Font("Serif", selectedStyle, 20));
      if (jchkCentered.isSelected())
        messagePanel.setCentered(true);
      else
        messagePanel.setCentered(false);

      // Make sure the message is repainted
      messagePanel.repaint();
    }
  }
}
```


Check Box Demo
Welcome to Java!
Centered
Bold
Italic

`JRadioButton`

Radio buttons are variations of check boxes. They are often used in the group, where only one button is checked at a time.

Example 9.8: Using Radio Buttons

JRadioButton Constructors

- ☞ `JRadioButton()`
- ☞ `JRadioButton(String text)`
- ☞ `JRadioButton(String text, boolean selected)`
- ☞ `JRadioButton(Icon icon)`
- ☞ `JRadioButton(String text, Icon icon)`
- ☞ `JRadioButton(String text, Icon icon, boolean selected)`

`JRadioButton` Properties

`JRadioButton` has all the properties in `JButton`. Additionally, `JButton` has the following property:

`selected`

Grouping Radio Buttons

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

```
// RadioButtonDemo.java: Use radio buttons to select a choice
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class RadioButtonDemo extends JFrame implements ItemListen
{
  // Declare radio buttons
  private JRadioButton jrbRed, jrbYellow, jrbGreen;

  // Declare a radio button group
  private ButtonGroup btg = new ButtonGroup();

  // Declare a traffic light display panel
  private Light light;
```

```
// Main method
public static void main(String[] args)
{
  RadioButtonDemo frame = new RadioButtonDemo();
  // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setSize(250, 170);
  frame.setVisible(true);
}

// Constructor
public RadioButtonDemo()
{
  setTitle("RadioButton Demo");
```

```
// Add traffic light panel to panel p1
   JPanel p1 = new JPanel();
   p1.setSize(200, 200);
   p1.setLayout(new FlowLayout(FlowLayout.CENTER));
   light = new Light();
   light.setSize(40, 90);
   p1.add(light);

   // Put the radio button in Panel p2
   JPanel p2 = new JPanel();
   p2.setLayout(new FlowLayout());
   p2.add(jrbRed = new JRadioButton("Red", false));
   p2.add(jrbYellow = new JRadioButton("Yellow", true));
   p2.add(jrbGreen = new JRadioButton("Green", false));
```

```
// Set keyboard mnemonics
   jrbRed.setMnemonic('R');
   jrbYellow.setMnemonic('Y');
   jrbGreen.setMnemonic('G');

   // Group radio buttons
   btg.add(jrbRed);
   btg.add(jrbYellow);
   btg.add(jrbGreen);

   // Place p1 and p2 in the frame
   getContentPane().setLayout(new BorderLayout());
   getContentPane().add(p1, BorderLayout.CENTER);
   getContentPane().add(p2, BorderLayout.SOUTH);
```

```
// Register listeners for check boxes
   jrbRed.addItemListener(this);
   jrbYellow.addItemListener(this);
   jrbGreen.addItemListener(this);
  }

  // Handle checkbox events
  public void itemStateChanged(ItemEvent e)
  {
   if (jrbRed.isSelected())
     light.turnOnRed(); // Set red light
   if (jrbYellow.isSelected())
     light.turnOnYellow(); // Set yellow light
   if (jrbGreen.isSelected())
     light.turnOnGreen(); // Set green light
  }
}
```

```
//Three traffic lights shown in a panel
class Light extends JPanel
{
  private boolean red;
  private boolean yellow;
  private boolean green;

  public Light()
  {
    red = false;
    yellow = true;
    green = false;
  }
```

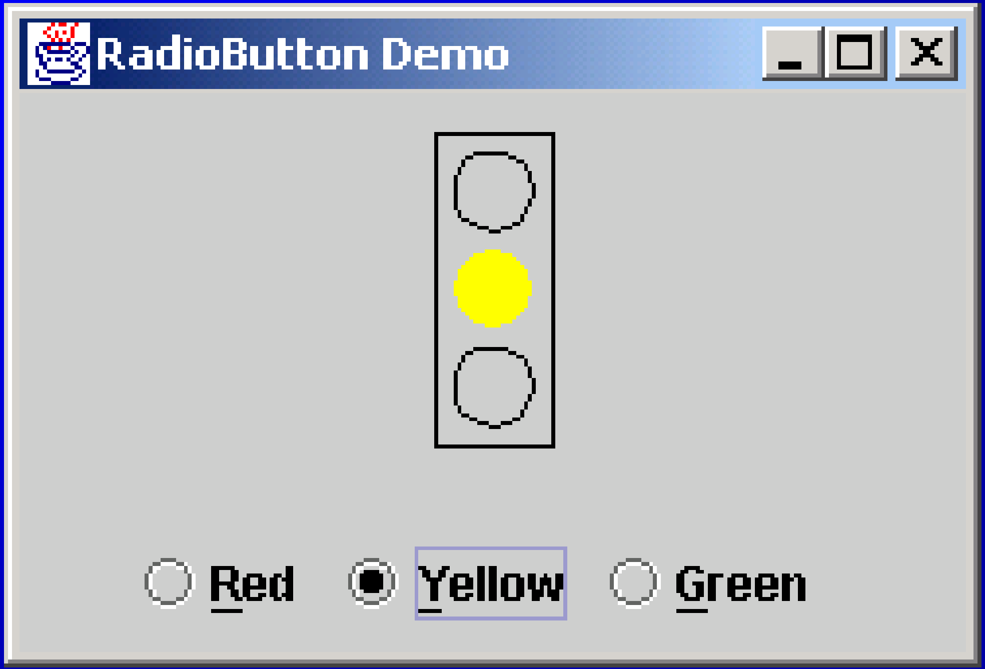
```
// Set red light on
  public void turnOnRed()
  {
    red = true;
    yellow = false;
    green = false;
    repaint();
  }

  // Set yellow light on
  public void turnOnYellow()
  {
    red = false;
    yellow = true;
    green = false;
    repaint();
  }
```

```
// Set green light on
public void turnOnGreen()
{
  red = false;
  yellow = false;
  green = true;
  repaint();
}

// Display lights
public void paintComponent(Graphics g)
{
  super.paintComponent(g);
```

```
if (red)
   {
     g.setColor(Color.red);
     g.fillOval(10, 10, 20, 20);
     g.setColor(Color.black);
     g.drawOval(10, 35, 20, 20);
     g.drawOval(10, 60, 20, 20);
     g.drawRect(5, 5, 30, 80);
   }
   else if (yellow)
   {
     g.setColor(Color.yellow);
     g.fillOval(10, 35, 20, 20);
     g.setColor(Color.black);
     g.drawRect(5, 5, 30, 80);
```

```
g.drawOval(10, 10, 20, 20);
    g.drawOval(10, 60, 20, 20);
  }
  else if (green)
  {
    g.setColor(Color.green);
    g.fillOval(10, 60, 20, 20);
    g.setColor(Color.black);
    g.drawRect(5, 5, 30, 80);
    g.drawOval(10, 10, 20, 20);
    g.drawOval(10, 35, 20, 20);
  }
  else
  {
    g.setColor(Color.black);
    g.drawRect(5, 5, 30, 80);
```

```
g.drawOval(10, 10, 20, 20);
    g.drawOval(10, 35, 20, 20);
    g.drawOval(10, 60, 20, 20);
  }
 }

 // Set preferred size
 public Dimension getPreferredSize()
 {
   return new Dimension(40, 90);
 }
}
```


RadioButton Demo
Red
Yellow
Green

Borders

You can set a border on any object of the `JComponent` class, but often it is useful to set a titled border on a `JPanel` that groups a set of related user interface components.

It can be used to set title for a group of desired components.

Properties, on Page 371

Border titleBorder=new TitleBorder("A Title");

Example 9.9: Using Borders

Static Method for Creating Borders

- createTitledBorder(String title)
- createLoweredBevelBorder()
- createRaisedBevelBorder()
- createLineBorder(Color color)
- createLineBorder(Color color, int thickness)
- createEtchedBorder()
- createEtchedBorder(Color highlight, Color shadow, boolean selected)
- createEmptyBorder()
- createMatteBorder(int top, int left, int bottom, int right, Icon tileIcon)
- createCompoundBorder(Border outsideBorder, Border insideBorder)

```
// BorderDemo.java: Use borders for JComponent components
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.*;
import javax.swing.border.*;

public class BorderDemo extends JFrame implements ActionListener
{
  // Declare a panel for displaying message
  private MessagePanel messagePanel;

  // A check box for selecting a border with or without a title
  private JCheckBox jchkTitled;
```

```
// Radio buttons for border styles
 private JRadioButton jrbLoweredBevel, jrbRaisedBevel,
   jrbEtched, jrbLine, jrbMatte, jrbEmpty;
 // Radio buttons for titled border options
 private JRadioButton jrbAboveBottom, jrbBottom,
   jrbBelowBottom, jrbAboveTop, jrbTop, jrbBelowTop,
   jrbLeft, jrbCenter, jrbRight;
 // TitledBorder for the message panel
 private TitledBorder messagePanelBorder = new TitledBorder("");
 // Main method
 public static void main(String[] args)
 {
   BorderDemo frame = new BorderDemo();
   // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.pack();
   frame.setVisible(true);
 }
```

```
// Constructor
public BorderDemo()
{
  setTitle("Border Demo");

  // Create a MessagePanel instance and set colors
  messagePanel = new MessagePanel("Display the border type");
  messagePanel.setCentered(true);
  messagePanel.setBackground(Color.yellow);
  messagePanel.setBorder(messagePanelBorder);

  // Place title position radio buttons
  JPanel jpPosition = new JPanel();
  jpPosition.setLayout(new GridLayout(3, 2));
  jpPosition.add(
    jrbAboveBottom = new JRadioButton("ABOVE_BOTTOM"));
```

```
jpPosition.add(jrbAboveTop = new JRadioButton("ABOVE_TOP"));
  jpPosition.add(jrbBottom = new JRadioButton("BOTTOM"));
  jpPosition.add(jrbTop = new JRadioButton("TOP"));
  jpPosition.add(
    jrbBelowBottom = new JRadioButton("BELOW_BOTTOM"));
  jpPosition.add(jrbBelowTop = new JRadioButton("BELOW_TOP"
  jpPosition.setBorder(new TitledBorder("Position"));

  // Place title justification radio buttons
  JPanel jpJustification = new JPanel();
  jpJustification.setLayout(new GridLayout(3,1));
  jpJustification.add(jrbLeft = new JRadioButton("LEFT"));
  jpJustification.add(jrbCenter = new JRadioButton("CENTER"));
  jpJustification.add(jrbRight = new JRadioButton("RIGHT"));
  jpJustification.setBorder(new TitledBorder("Justification"));
```

```
// Create panel jpTitleOptions to hold jpPosition and jpJustification
   JPanel jpTitleOptions = new JPanel();
   jpTitleOptions.setLayout(new BorderLayout());
   jpTitleOptions.add(jpPosition, BorderLayout.CENTER);
   jpTitleOptions.add(jpJustification, BorderLayout.EAST);

   // Create Panel jpTitle to hold a check box and title position
   // radio buttons, and title justification radio buttons
   JPanel jpTitle = new JPanel();
   jpTitle.setBorder(new TitledBorder("Border Title"));
   jpTitle.setLayout(new BorderLayout());
   jpTitle.add(jchkTitled = new JCheckBox("Titled"),
     BorderLayout.NORTH);
   jpTitle.add(jpTitleOptions, BorderLayout.CENTER);
```

```
// Group radio buttons for title position
   ButtonGroup btgTitlePosition = new ButtonGroup();
   btgTitlePosition.add(jrbAboveBottom);
   btgTitlePosition.add(jrbBottom);
   btgTitlePosition.add(jrbBelowBottom);
   btgTitlePosition.add(jrbAboveTop);
   btgTitlePosition.add(jrbTop);
   btgTitlePosition.add(jrbBelowTop);

   // Group radio buttons for title justification
   ButtonGroup btgTitleJustification = new ButtonGroup();
   btgTitleJustification.add(jrbLeft);
   btgTitleJustification.add(jrbCenter);
   btgTitleJustification.add(jrbRight);
```

```
// Create Panel jpBorderStyle to hold border style radio buttons
   JPanel jpBorderStyle = new JPanel();
   jpBorderStyle.setBorder(new TitledBorder("Border Style"));
   jpBorderStyle.setLayout(new GridLayout(6, 1));
   jpBorderStyle.add(jrbLoweredBevel =
     new JRadioButton("Lowered Bevel"));
   jpBorderStyle.add(jrbRaisedBevel =
     new JRadioButton("Raised Bevel"));
   jpBorderStyle.add(jrbEtched = new JRadioButton("Etched"));
   jpBorderStyle.add(jrbLine = new JRadioButton("Line"));
   jpBorderStyle.add(jrbMatte = new JRadioButton("Matte"));
   jpBorderStyle.add(jrbEmpty = new JRadioButton("Empty"));

   // Group radio buttons for border styles
   ButtonGroup btgBorderStyle = new ButtonGroup();
   btgBorderStyle.add(jrbLoweredBevel);
   btgBorderStyle.add(jrbRaisedBevel);
```

```
btgBorderStyle.add(jrbEtched);
  btgBorderStyle.add(jrbLine);
  btgBorderStyle.add(jrbMatte);
  btgBorderStyle.add(jrbEmpty);

  // Create Panel jpAllChoices to place jpTitle and jpBorderStyle
  JPanel jpAllChoices = new JPanel();
  jpAllChoices.setLayout(new BorderLayout());
  jpAllChoices.add(jpTitle, BorderLayout.CENTER);
  jpAllChoices.add(jpBorderStyle, BorderLayout.EAST);

  // Place panels in the frame
  getContentPane().setLayout(new BorderLayout());
  getContentPane().add(messagePanel, BorderLayout.CENTER);
  getContentPane().add(jpAllChoices, BorderLayout.SOUTH);
```

```
// Register listeners
   jchkTitled.addActionListener(this);
   jrbAboveBottom.addActionListener(this);
   jrbBottom.addActionListener(this);
   jrbBelowBottom.addActionListener(this);
   jrbAboveTop.addActionListener(this);
   jrbTop.addActionListener(this);
   jrbBelowTop.addActionListener(this);
   jrbLeft.addActionListener(this);
   jrbCenter.addActionListener(this);
   jrbRight.addActionListener(this);
   jrbLoweredBevel.addActionListener(this);
   jrbRaisedBevel.addActionListener(this);
   jrbLine.addActionListener(this);
   jrbEtched.addActionListener(this);
   jrbMatte.addActionListener(this);
   jrbEmpty.addActionListener(this);
 }
```

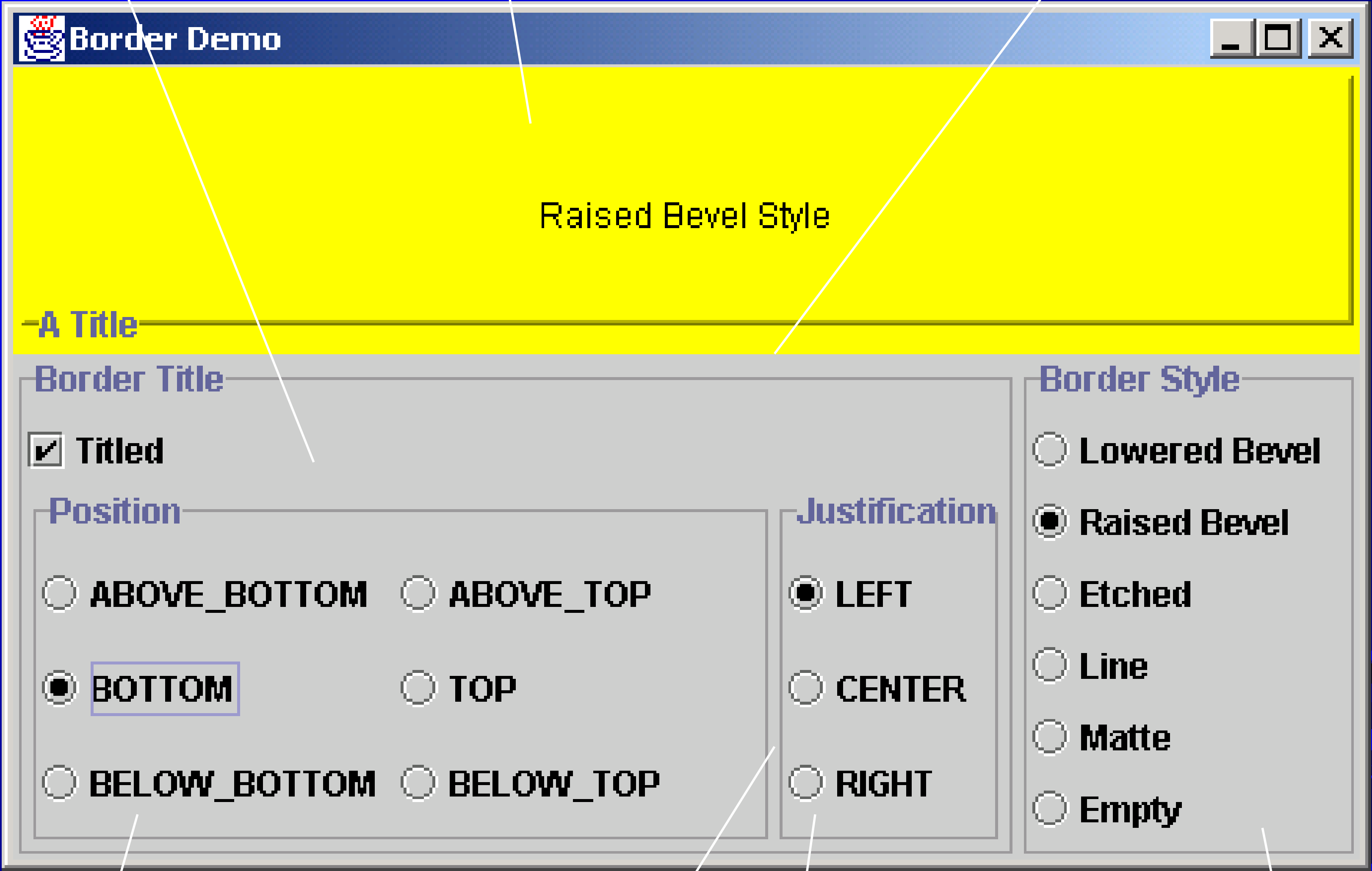
```
// Handle ActionEvents on check box and radio buttons
public void actionPerformed(ActionEvent e)
{
  // Get border style
  Border border = new EmptyBorder(2, 2, 2, 2);

  if (jrbLoweredBevel.isSelected())
  {
    border = new BevelBorder(BevelBorder.LOWERED);
    messagePanel.setMessage("Lowered Bevel Style");
  }
  else if (jrbRaisedBevel.isSelected())
  {
    border = new BevelBorder(BevelBorder.RAISED);
    messagePanel.setMessage("Raised Bevel Style");
  }
```

```
else if (jrbEtched.isSelected())
   {
    border = new EtchedBorder();
    messagePanel.setMessage("Etched Style");
   }
   else if (jrbLine.isSelected())
   {
    border = new LineBorder(Color.black, 5);
    messagePanel.setMessage("Line Style");
   }
   else if (jrbMatte.isSelected())
   {
    border = new MatteBorder(20, 20, 20, 20,
      new ImageIcon("images/swirl.gif"));
    messagePanel.setMessage("Matte Style");
   }
```

```
else if (jrbEmpty.isSelected())
   {
     border = new EmptyBorder(2, 2, 2, 2);
     messagePanel.setMessage("Empty Style");
   }

   if (jchkTitled.isSelected())
   {
     // Get the title position and justification
     int titlePosition = TitledBorder.DEFAULT_POSITION;
     int titleJustification = TitledBorder.DEFAULT_JUSTIFICATION;

     if (jrbAboveBottom.isSelected())
       titlePosition = TitledBorder.ABOVE_BOTTOM;
     else if (jrbBottom.isSelected())
       titlePosition = TitledBorder.BOTTOM;
     else if (jrbBelowBottom.isSelected())
```

```
titlePosition = TitledBorder.BELOW_BOTTOM;
    else if (jrbAboveTop.isSelected())
      titlePosition = TitledBorder.ABOVE_TOP;
    else if (jrbTop.isSelected())
      titlePosition = TitledBorder.TOP;
    else if (jrbBelowTop.isSelected())
      titlePosition = TitledBorder.BELOW_TOP;

    if (jrbLeft.isSelected())
      titleJustification = TitledBorder.LEFT;
    else if (jrbCenter.isSelected())
      titleJustification = TitledBorder.CENTER;
    else if (jrbRight.isSelected())
      titleJustification = TitledBorder.RIGHT;
```

```
messagePanelBorder = new TitledBorder("A Title");
     messagePanelBorder.setBorder(border);
     messagePanelBorder.setTitlePosition(titlePosition);
     messagePanelBorder.setTitleJustification(titleJustification);
     messagePanelBorder.setTitle("A Title");
     messagePanel.setBorder(messagePanelBorder);
   }
   else
   {
     messagePanel.setBorder(border);
   }
  }
}
```



Border Demo

Raised Bevel Style

A Title

Border Title

[✓] Titled

Position

( ) ABOVE_BOTTOM ( ) ABOVE_TOP

(●) BOTTOM ( ) TOP

( ) BELOW_BOTTOM ( ) BELOW_TOP

Justification

(●) LEFT

( ) CENTER

( ) RIGHT

Border Style

( ) Lowered Bevel

(●) Raised Bevel

( ) Etched

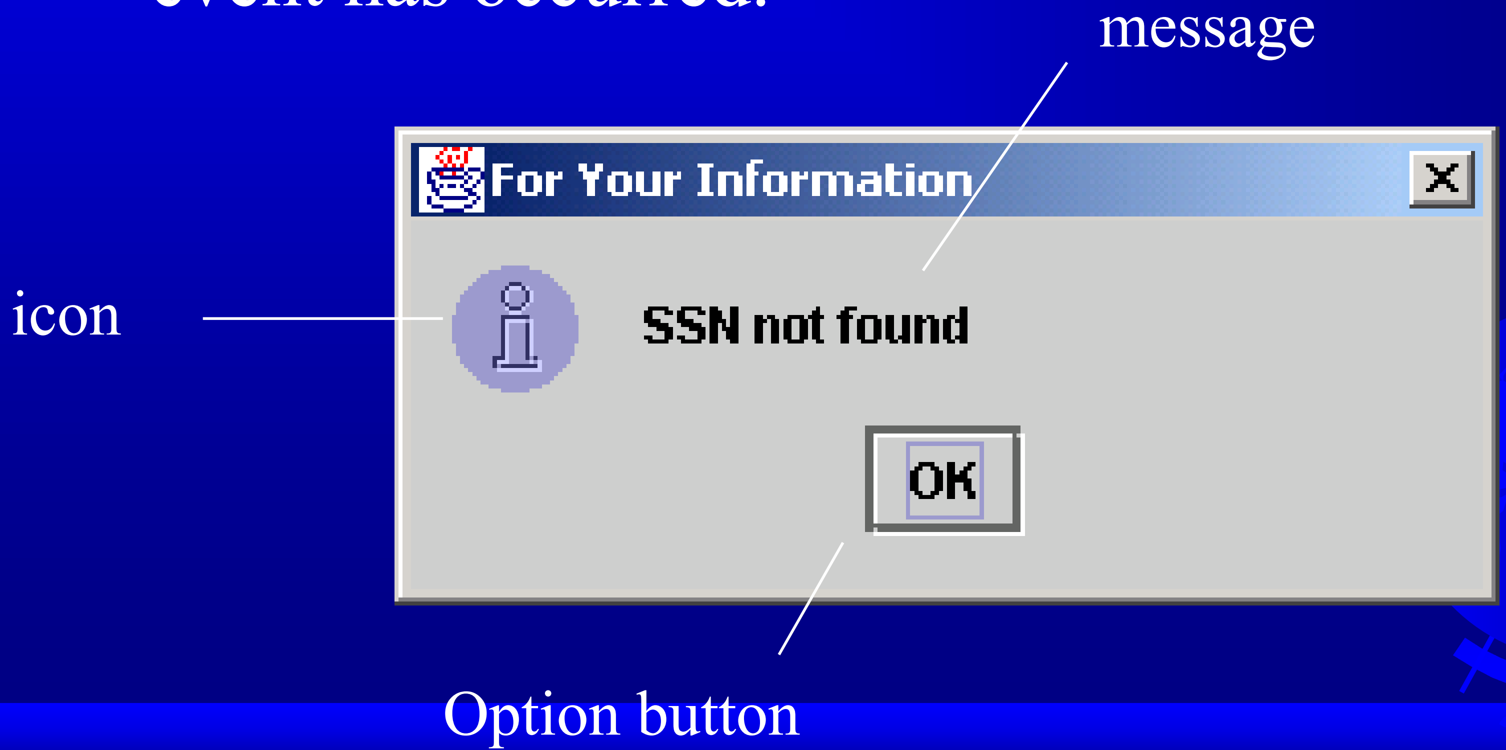
( ) Line

( ) Matte

( ) Empty

Message Dialogs

- A *dialog* is normally used as a temporary window to receive additional information from the user, or to provide notification that some event has occurred.



Creating Message Dialogs

Use static method in `JOptionPane` class.

```
showMessageDialog(Component
parentComponent, Object message,
String title, int messageType)
```

```
showMessageDialog(Component
parentComponent, Object message,
String title, int messageType, Icon
icon)
```

Creating Message Dialogs

Use static method in `JOptionPane` class.

```
messageType:
     ERROR
     INFORMATION_MESSAGE
     PLAIN_MESSAGE
     WARNING_MESSAGE
     QUESTION_MESSAGE
```

Example 9.10: Using Message Dialogs

- Objective: Display student exam scores. The program prompts the user to enter the user's last name and the password in a dialog box. Upon receiving the correct user name and password, the program displays the student's full name and the exam score.

```
// DialogDemo.java: Use message dialog box to select information
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class DialogDemo extends JFrame implements ActionListener
{
  // Create sample student information in arrays
  // with name, SSN, password, and grade
  private String[][] student =
    {
     {"John Willow", "111223333", "a450", "A"},
     {"Jim Brown", "111223334", "b344", "B"},
     {"Bill Beng", "111223335", "33342csa", "C"},
     {"George Wall", "111223336", "343rea2", "D"},
     {"Jill Jones", "111223337", "34g", "E"}
    };
```

```
 // Declare text fields for last name, password, full name and score
  private JTextField jtfSSN;
  private JPasswordField jpfPassword;
  private JTextField jtfName;
  private JTextField jtfGrade;
  private JButton jbtFind;

// Main method
  public static void main(String[] args)
  {
    DialogDemo frame = new DialogDemo();
    // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
  }
```

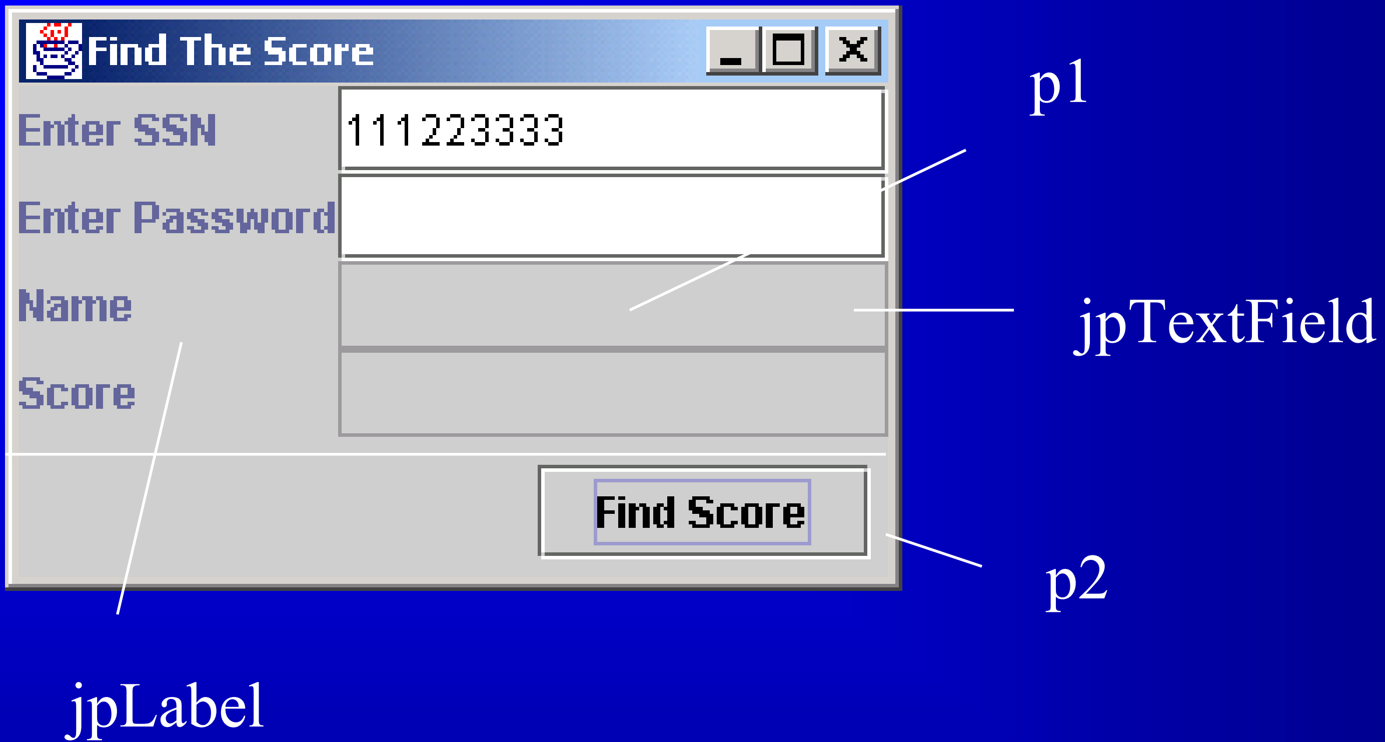
```
public DialogDemo()
{
  setTitle("Find The Score");

  // Panel jpLables to hold labels
  JPanel jpLabels = new JPanel();
  jpLabels.setLayout(new GridLayout(4, 1));
  jpLabels.add(new JLabel("Enter SSN"));
  jpLabels.add(new JLabel("Enter Password"));
  jpLabels.add(new JLabel("Name"));
  jpLabels.add(new JLabel("Score"));

  // Panel jpTextFields to hold text fields and password
  JPanel jpTextFields = new JPanel();
  jpTextFields.setLayout(new GridLayout(4, 1));
  jpTextFields.add(jtfSSN = new JTextField(10));
jpTextFields.add(jpfPassword = new JPasswordField(10));
  jpTextFields.add(jtfName = new JTextField(10));
  jpTextFields.add(jtfGrade = new JTextField(10));
  jtfName.setEditable(false);
  jtfGrade.setEditable(false);

  // Panel p1 for holding jpLables and jpTextFields
  JPanel p1 = new JPanel();
  p1.setLayout(new BorderLayout());
  p1.add(jpLabels, BorderLayout.WEST);
  p1.add(jpTextFields, BorderLayout.CENTER);

  // Panel p2 for holding the Find button
  JPanel p2 = new JPanel();
  p2.setLayout(new FlowLayout(FlowLayout.RIGHT));
  p2.add(jbtFind = new JButton("Find Score"));
```

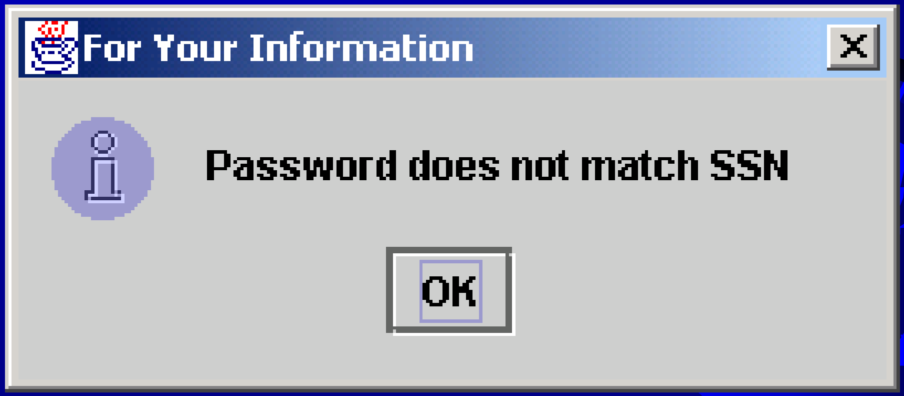
```
// Place panels into the frame
   getContentPane().setLayout(new BorderLayout());
   getContentPane().add(p1, BorderLayout.CENTER);
   getContentPane().add(p2, BorderLayout.SOUTH);

   // Register listener for jbtFind
   jbtFind.addActionListener(this);
 }
```

```
public void actionPerformed(ActionEvent e)
 {
   // Find the student in the database
   int index = find(jtfSSN.getText().trim(),
     new String(jpfPassword.getPassword()));
   if (index == -1)
   {
     JOptionPane.showMessageDialog(this, "SSN not found",
       "For Your Information",
               JOptionPane.INFORMATION_MESSAGE);
   }
   else if (index == -2)
   {
     JOptionPane.showMessageDialog(this,
       "Password does not match SSN",
       "For Your Information",
                      JOptionPane.INFORMATION_MESSAGE);
   }
```

```
else
  {
    // Display name and score
    jtfName.setText(student[index][0]);
    jtfGrade.setText(student[index][3]);
  }
 }

 // Find the student who matched user name and password
 // return the index if found; return -1 if SSN is not in
 // the database, and return -2 if password does not match SSN
 public int find(String SSN, String pw)
 {
   // Find a student who matches SSN and pw
   for (int i=0; i<student.length; i++)
     if (student[i][1].equals(SSN) && student[i][2].equals(pw))
       return i;
```

```
// Determine if the SSN is in the database
    for (int i=0; i<student.length; i++)
      if (student[i][1].equals(SSN))
        return -2;

    // Return -1 since the SSN and pw do not match
    return -1;
  }
}
```





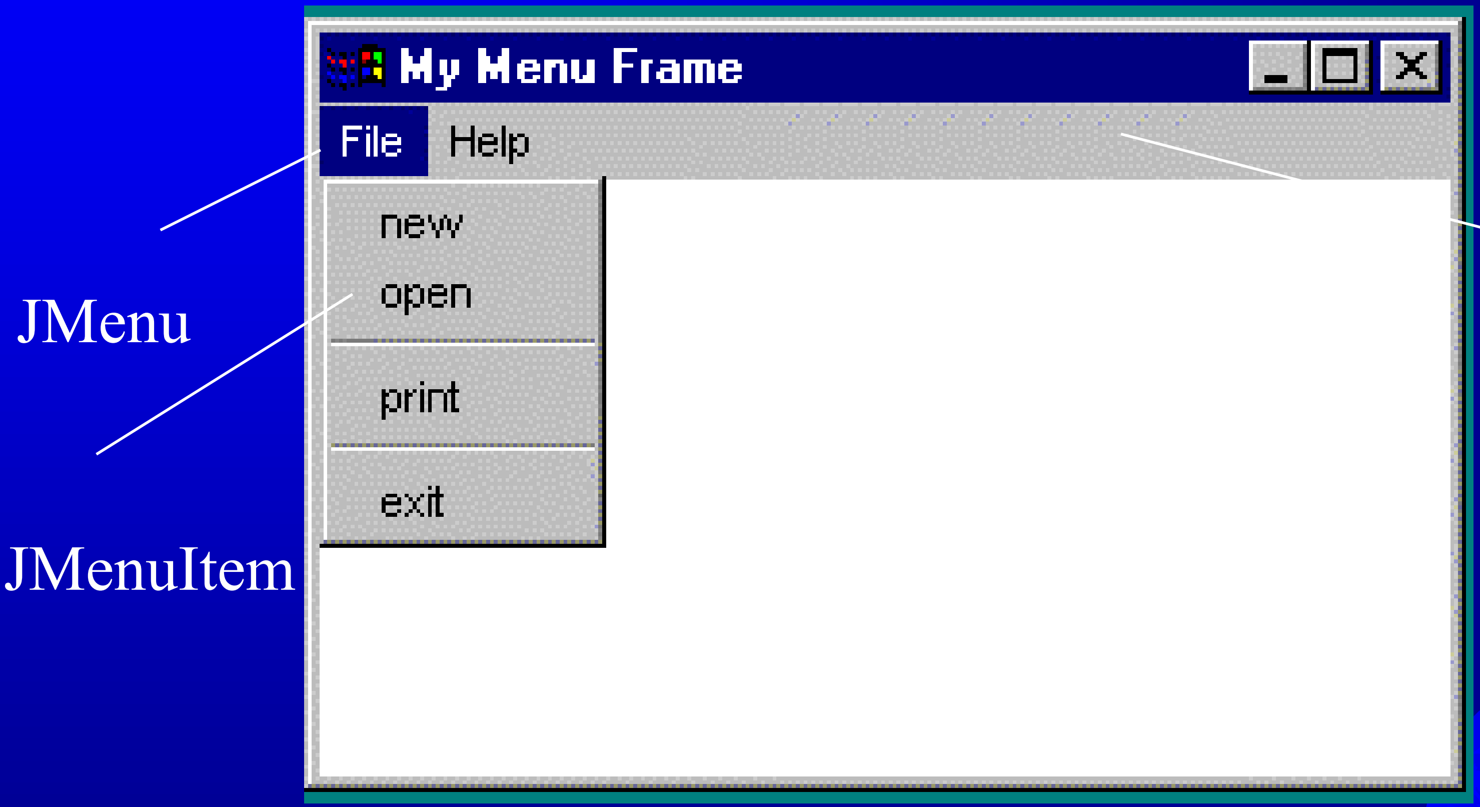
Password does not match SSN

Menus

- Menus make selection easier, and widely used in window applications.
- Java provides several classes—`JMenuBar`, `JMenu`, `JMenuItem`, `JCheckBoxMenuItem`, and `JRadioButtonMenuItem` —to implement menus in a frame.

- A JFrame or JApplet can hold a *menu bar* to which the *pull-down menus* are attached. Menus consist of *menu items* that the user can select (or toggle on or off). Menu bars can be viewed as a structure to support menus.

Menu Demo


JMenu
JMenuItem
JMenuBar
My Menu Frame
File
Help
new
open
print
exit

The `JMenuBar` Class

A menu bar holds menus; the menu bar can only be added to a frame. Following are the steps to create and add a `JMenuBar` to a frame:

```
JFrame f = new JFrame();
f.setSize(300, 200);
f.setVisible(true);
JMenuBar mb = new JMenuBar();
f.setJMenuBar(mb);
```

The `Menu` Class

You attach menus onto a `JMenuBar`. The following code creates two menus, File and Help, and adds them to the `JMenuBar mb`:

```
JMenu fileMenu = new JMenu("File", false);
JMenu helpMenu = new JMenu("Help", true);
mb.add(fileMenu);
mb.add(helpMenu);
```

The `JMenuItem` Class

You add menu items on a menu. The following code adds menu items and item separators in menu `fileMenu`:

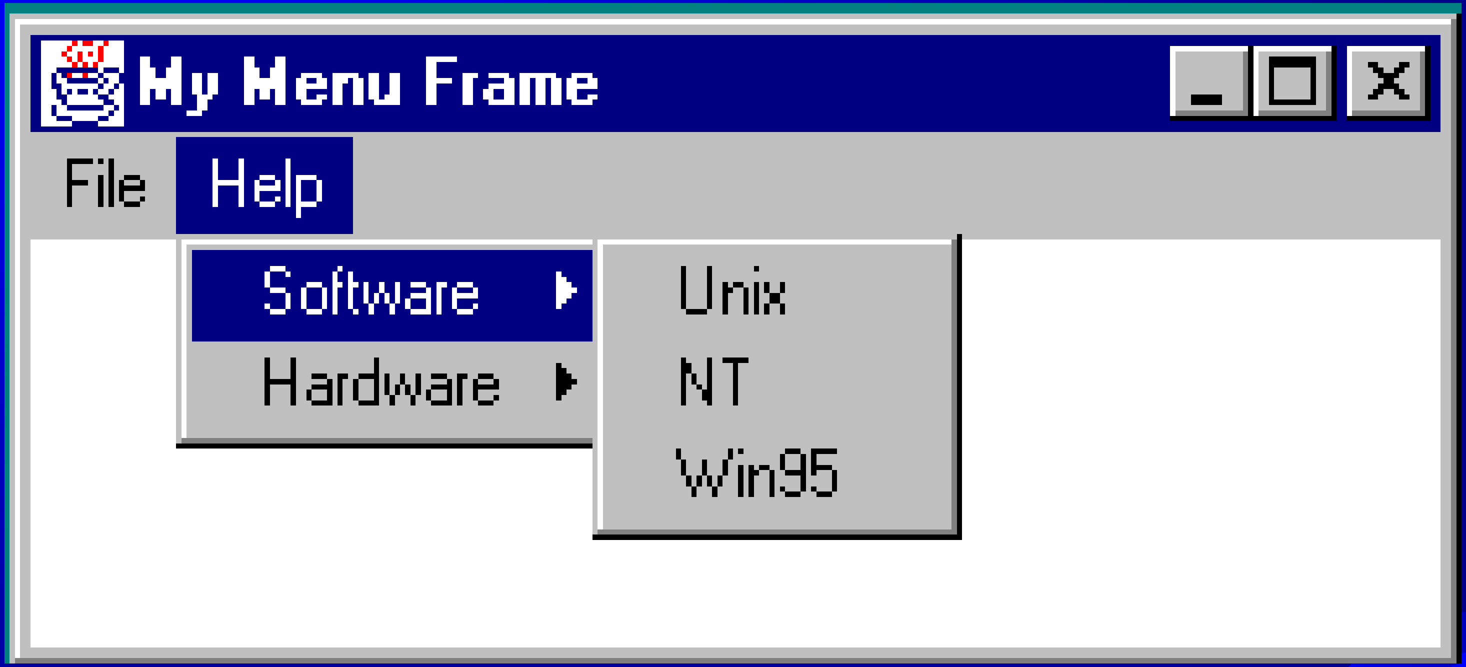
```
fileMenu.add(new JMenuItem("new"));
fileMenu.add(new JMenuItem("open"));
fileMenu.add(new JMenuItem("-"));
fileMenu.add(new JMenuItem("print"));
fileMenu.add(new JMenuItem("exit"));
fileMenu.add(new JMenuItem("-"));
```

Submenus

You can add submenus into menu items. The following code adds the submenus “Unix,” “NT,” and “Win95” into the menu item “Software.”

```
JMenu softwareHelpSubMenu = new JMenu("Software");
JMenu hardwareHelpSubMenu = new JMenu("Hardware");
helpMenu.add(softwareHelpSubMenu);
helpMenu.add(hardwareHelpSubMenu);
softwareHelpSubMenu.add(new JMenuItem("Unix"));
softwareHelpSubMenu.add(new JMenuItem("NT"));
softwareHelpSubMenu.add(new JMenuItem("Win95"));
```

Submenu Demo



Check box and radio button menus

- Checkbox menu:

```
helpMenu.add(new JCheckBoxMenuItem("Check it"));
```

- Radio button menu items

```
Jmenu colorHelpSubMenu=new Jmenu("Color");
helpMenu.add(colorHelpSubMenu);
colorHelpSubMenu.add(jrbBlue=
          new JRadioButtonMenuItem("Blue"));
colorHelpSubMenu.add(jrbYellow=
          new JRadioButtonMenuItem("Yellow"));
```

Check box and radio button menus

☞ Radio button menu items

```
colorHelpSubMenu.add(jrbRed=
            new JRadioButtonMenuItem("Red"));
ButtonGroup btg=new ButtonGroup();
btg.add(jrbmiBlue);
btg.add(jrbmiRed);
btg.add(jrbmiYellow);
```

Example 9.11: Using Menus

- Objective: Create a user interface that performs arithmetic. The interface contains labels and text fields for Number 1, Number 2, and Result. The Result box displays the result of the arithmetic operation between Number 1 and Number 2.

```
// MenuDemo.java: Use menus to move message in a panel
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MenuDemo extends JFrame implements ActionListener
{
  // Text fields for Number 1, Number 2, and Result
  private JTextField jtfNum1, jtfNum2, jtfResult;

  // Buttons "Add", "Subtract", "Multiply" and "Divide"
  private JButton jbtAdd, jbtSub, jbtMul, jbtDiv;

  // Menu items "Add", "Subtract", "Multiply","Divide" and "Close"
  private JMenuItem jmiAdd, jmiSub, jmiMul, jmiDiv, jmiClose;
```

```
// Main Method
 public static void main(String[] args)
 {
  MenuDemo frame = new MenuDemo();
  // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.pack();
  frame.setVisible(true);
 }

 // Default Constructor
 public MenuDemo()
 {
  setTitle("Menu Demo");

  // Create menu bar
  JMenuBar jmb = new JMenuBar();
```

```
// Set menu bar to the frame
  setJMenuBar(jmb);

  // Add menu "Operation" to menu bar
  JMenu operationMenu = new JMenu("Operation");
  operationMenu.setMnemonic('O');
  jmb.add(operationMenu);

  // Add menu "Exit" in menu bar
  JMenu exitMenu = new JMenu("Exit");
  exitMenu.setMnemonic('E');
  jmb.add(exitMenu);

  // Add menu items with mnemonics to menu "Operation"
  operationMenu.add(jmiAdd= new JMenuItem("Add", 'H'));
  operationMenu.add(jmiSub = new JMenuItem("Subtract", 'S'));
operationMenu.add(jmiMul = new JMenuItem("Multiply", 'M'));
  operationMenu.add(jmiDiv = new JMenuItem("Divide", 'D'));
  exitMenu.add(jmiClose = new JMenuItem("Close", 'C'));
  // Set keyboard accelerators
  jmiAdd.setAccelerator(
    KeyStroke.getKeyStroke(KeyEvent.VK_H,
                    ActionEvent.CTRL_MASK));
  jmiSub.setAccelerator(
    KeyStroke.getKeyStroke(KeyEvent.VK_S,
                                ActionEvent.CTRL_MASK));
  jmiMul.setAccelerator(
    KeyStroke.getKeyStroke(KeyEvent.VK_M,
                               ActionEvent.CTRL_MASK));
  jmiDiv.setAccelerator(
    KeyStroke.getKeyStroke(KeyEvent.VK_D,
                             ActionEvent.CTRL_MASK));
```

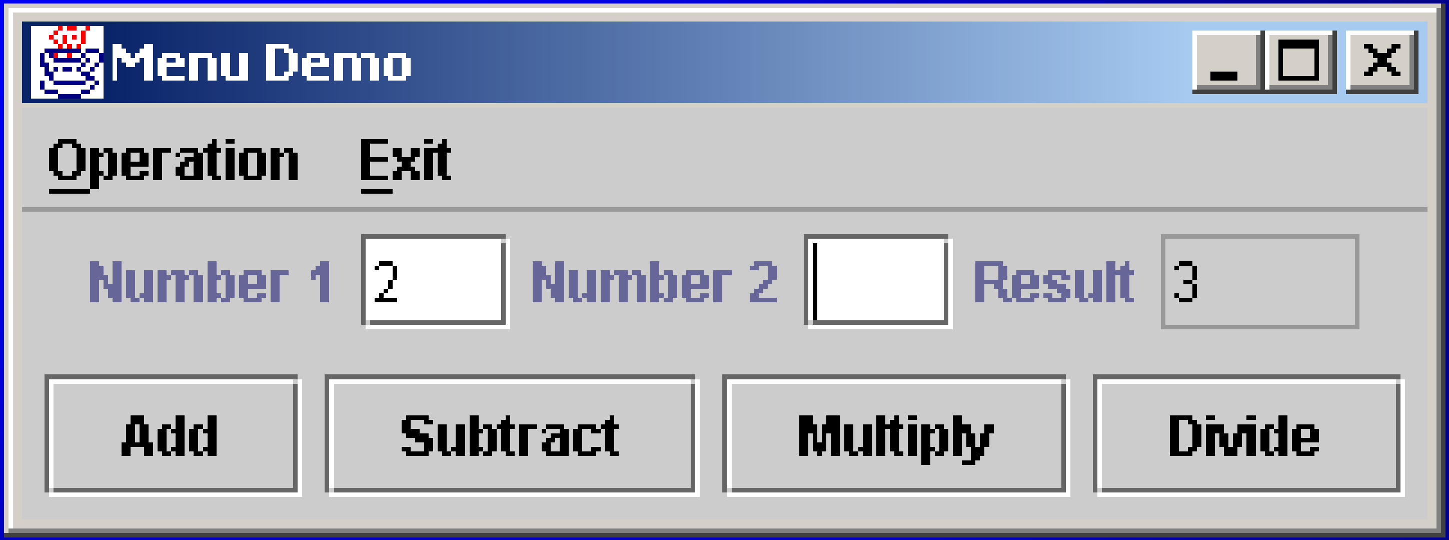
```
// Panel p1 to hold text fields and labels
 JPanel p1 = new JPanel();
 p1.setLayout(new FlowLayout());
p1.add(new JLabel("Number 1"));
 p1.add(jtfNum1 = new JTextField(3));
 p1.add(new JLabel("Number 2"));
 p1.add(jtfNum2 = new JTextField(3));
 p1.add(new JLabel("Result"));
 p1.add(jtfResult = new JTextField(4));
 jtfResult.setEditable(false);
 // Panel p2 to hold buttons
 JPanel p2 = new JPanel();
 p2.setLayout(new FlowLayout());
 p2.add(jbtAdd = new JButton("Add"));
 p2.add(jbtSub = new JButton("Subtract"));
 p2.add(jbtMul = new JButton("Multiply"));
 p2.add(jbtDiv = new JButton("Divide"));
```

```
// Add panels to the frame
   getContentPane().setLayout(new BorderLayout());
   getContentPane().add(p1, BorderLayout.CENTER);
   getContentPane().add(p2, BorderLayout.SOUTH);

   // Register listeners
   jbtAdd.addActionListener(this);
   jbtSub.addActionListener(this);
   jbtMul.addActionListener(this);
   jbtDiv.addActionListener(this);
   jmiAdd.addActionListener(this);
   jmiSub.addActionListener(this);
   jmiMul.addActionListener(this);
   jmiDiv.addActionListener(this);
   jmiClose.addActionListener(this);
 }
```

```
// Handle ActionEvent from buttons and menu items
 public void actionPerformed(ActionEvent e)
 {
   String actionCommand = e.getActionCommand();

   // Handle button events
   if (e.getSource() instanceof JButton)
   {
     if ("Add".equals(actionCommand))
       calculate('+');
     else if ("Subtract".equals(actionCommand))
       calculate('-');
     else if ("Multiply".equals(actionCommand))
       calculate('*');
     else if ("Divide".equals(actionCommand))
       calculate('/');
   }
```

```
  else if (e.getSource() instanceof JMenuItem)
  {
    // Handle menu item events
    if ("Add".equals(actionCommand))
      calculate('+');
    else if ("Subtract".equals(actionCommand))
      calculate('-');
    else if ("Multiply".equals(actionCommand))
      calculate('*');
    else if ("Divide".equals(actionCommand))
      calculate('/');
    else if ("Close".equals(actionCommand))
      System.exit(0);
  }
}
```

```
 // Calculate and show the result in jtfResult
  private void calculate(char operator)
  {
    // Obtain Number 1 and Number 2
    int num1 = (Integer.parseInt(jtfNum1.getText().trim()));
    int num2 = (Integer.parseInt(jtfNum2.getText().trim()));
    int result = 0;
try
{
    // Perform selected operation
    switch (operator)
    {
      case '+': result = num1 + num2;
                break;
      case '-': result = num1 - num2;
                break;
 case '*': result = num1  * num2;
               break;
      case '/': result = num1 / num2;
    }
}
catch (Exception e)
{
            System.out.println(e);

}

    // Set result in jtfResult
    jtfResult.setText(String.valueOf(result));
  }
}
```


Menu Demo
Operation
Exit
Number 1
2
Number 2
Result
3
Add
Subtract
Multiply
Divide

Creating Multiple Windows

The following slides show step-by-step how to create an additional window from an application or applet.

Main frame and sub-frame

Creating Additional Windows, Step 1

Step 1: Create a subclass of `JFrame` (called a `SubFrame`) that tells the new window what to do. For example, all the GUI application programs extend `JFrame` and are subclasses of `JFrame`.

Creating Additional Windows, Step 2

Step 2: Create an instance of `SubFrame` in the application or applet.

Example:

```
SubFrame subFrame = new
   SubFrame("SubFrame Title");
```

Creating Additional Windows, Step 3

Step 3: Create a `JButton` for activating the `subFrame`.

```
add(new JButton("Activate SubFrame"));
```

Creating Additional Windows, Step 4

Step 4: Override the `actionPerformed()` method as follows:

```
public actionPerformed(ActionEvent e)
{
  String actionCommand = e.getActionCommand();
  if (e.target instanceof Button)
  {
    if ("Activate SubFrame".equals(actionCommand))
    {
      subFrame.setVisible(true);
    }
  }
}
```

Example 9.12 Creating Multiple Windows

- This example creates a main window with a text area in the scroll pane, and a button named "Show Histogram." When the user clicks the button, a new window appears that displays a histogram to show the occurrence of the letters in the text area.

```
// Histogram.java: Display a histogram in a panel to show the
// occurence of the letters
import javax.swing.*;
import java.awt.*;

public class Histogram extends JPanel
{
  // Count the occurrence of 26 letters
  private int count[];

  // Set the count and display histogram
  public void showHistogram(int[] count)
  {
    this.count = count;
    repaint();
  }
```

```
// Paint the histogram
 public void paintComponent(Graphics g)
 {
   if (count == null) return; // No display if count is null

   super.paintComponent(g);

   // Find the panel size and bar width and interval dynamically
   int width = getSize().width;
   int height = getSize().height;
   int interval = (width-40)/count.length;
   int individualWidth = (int)(((width-40)/24)*0.60);

   // Find the maximum count. The maximum count has the highest bar
   int maxCount = 0;
```

```
for (int i=0; i<count.length; i++)
   {
     if (maxCount < count[i])
       maxCount = count[i];
   }

   // x is the start position for the first bar in the histogram
   int x = 30;

   // Draw a horizontal base line
   g.drawLine(10, height-45, width-10, height-45);
   for (int i=0; i<count.length; i++)
   {
     // Find the bar height
     int barHeight =
             (int)(((double)count[i]/(double)maxCount)*(height-55));
// Display a bar (i.e. rectangle)
    g.drawRect(x, height-45-barHeight, individualWidth,
     barHeight);

    // Display a letter under the base line
    g.drawString((char)(65+i)+"", x, height-30);

    // Move x for displaying the next character
    x += interval;
  }
 }

 public Dimension getPreferredSize()
 {
  return new Dimension(300, 300);
 }
}
```

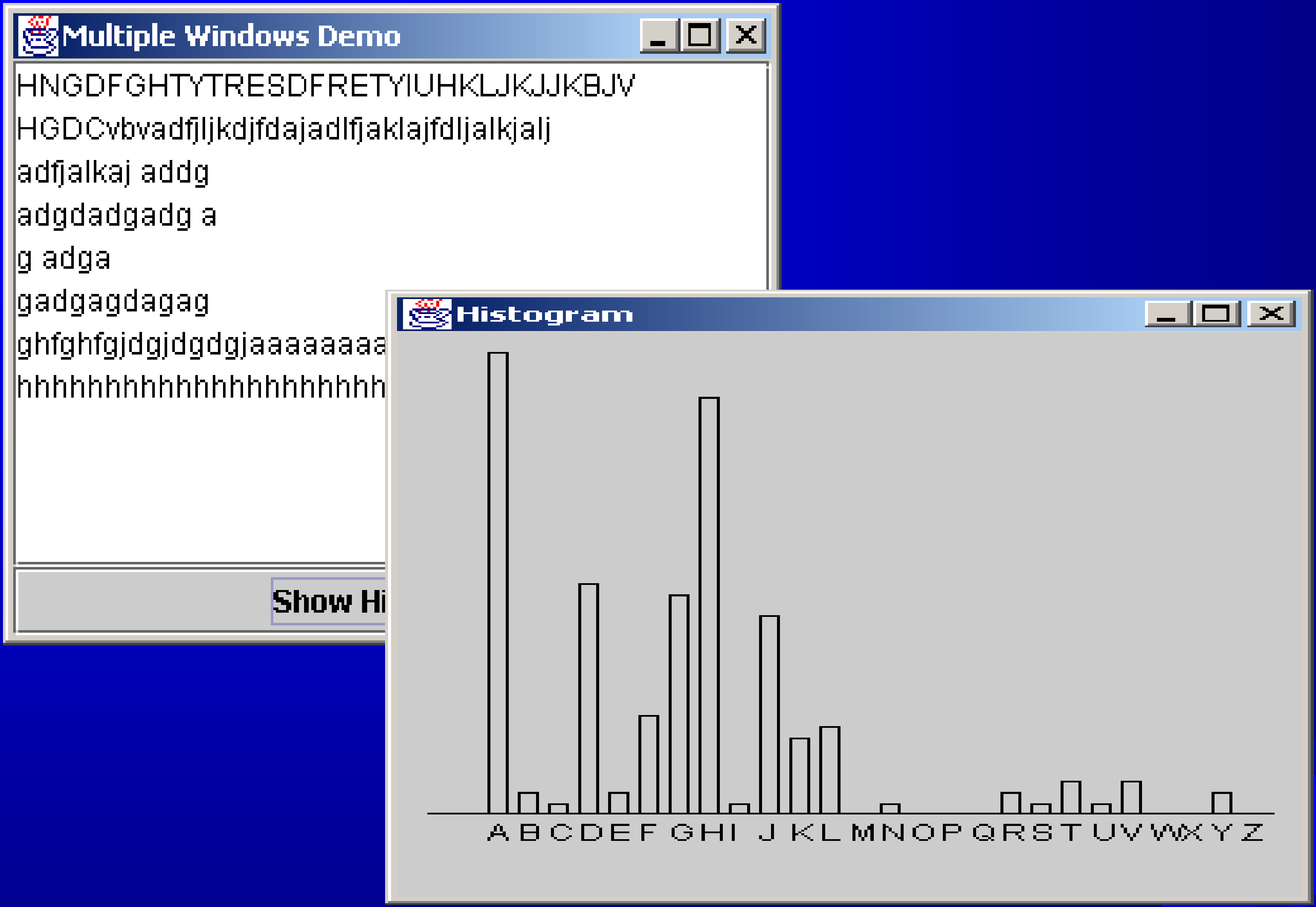
```
// MultipleWindowsDemo.java:
//        Display histogram in a seperate window
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MultipleWindowsDemo extends JFrame
                        implements ActionListener
{
  private JTextArea jta;
  private JButton jbtShowHistogram =
                              new JButton("Show Histogram");
  private Histogram histogram = new Histogram();

  // Create a new frame to hold the histogram panel
  private JFrame histogramFrame = new JFrame();
```

```
// Construct the frame
 public MultipleWindowsDemo()
 {
   // Store text area in a scroll pane
   JScrollPane scrollPane = new JScrollPane(jta = new JTextArea());
   scrollPane.setPreferredSize(new Dimension(300, 200));
   jta.setWrapStyleWord(true);
   jta.setLineWrap(true);

   // Place scroll pane and button in the frame
   getContentPane().add(scrollPane, BorderLayout.CENTER);
   getContentPane().add(jbtShowHistogram, BorderLayout.SOUTH);

   // Register listener
   jbtShowHistogram.addActionListener(this);
// Create a new frame to hold the histogram panel
  histogramFrame.getContentPane().add(histogram);
  histogramFrame.pack();
  histogramFrame.setTitle("Histogram");
 }

 // Handle the button action
 public void actionPerformed(ActionEvent e)
 {
  // Count the letters in the text area
  int[] count = countLetters();

  // Set the letter count to histogram for display
  histogram.showHistogram(count);

  // Show the frame
  histogramFrame.setVisible(true);
 }
```

```
// Count the letters in the text area
private int[] countLetters()
{
  // Count for 26 letters
  int[] count = new int[26];

  // Get contents from the text area
  String text = jta.getText();

  // Count occurence of each letter (case insensitive)
  for (int i=0; i<text.length(); i++)
  {
    char character = text.charAt(i);
```

```
if ((character >= 'A') && (character <= 'Z'))
    {
     count[(int)character-65]++; // The ASCII for 'A' is 65
    }
    else if ((character >= 'a') && (character <= 'z'))
    {
     count[(int)character-97]++; // The ASCII for 'a' is 97
    }
  }

  return count; // Return the count array
 }
```

```
// Main method
 public static void main(String[] args)
 {
   MultipleWindowsDemo frame = new MultipleWindowsDemo();
   // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setTitle("Multiple Windows Demo");
   frame.pack();
   frame.setVisible(true);
 }
}
```



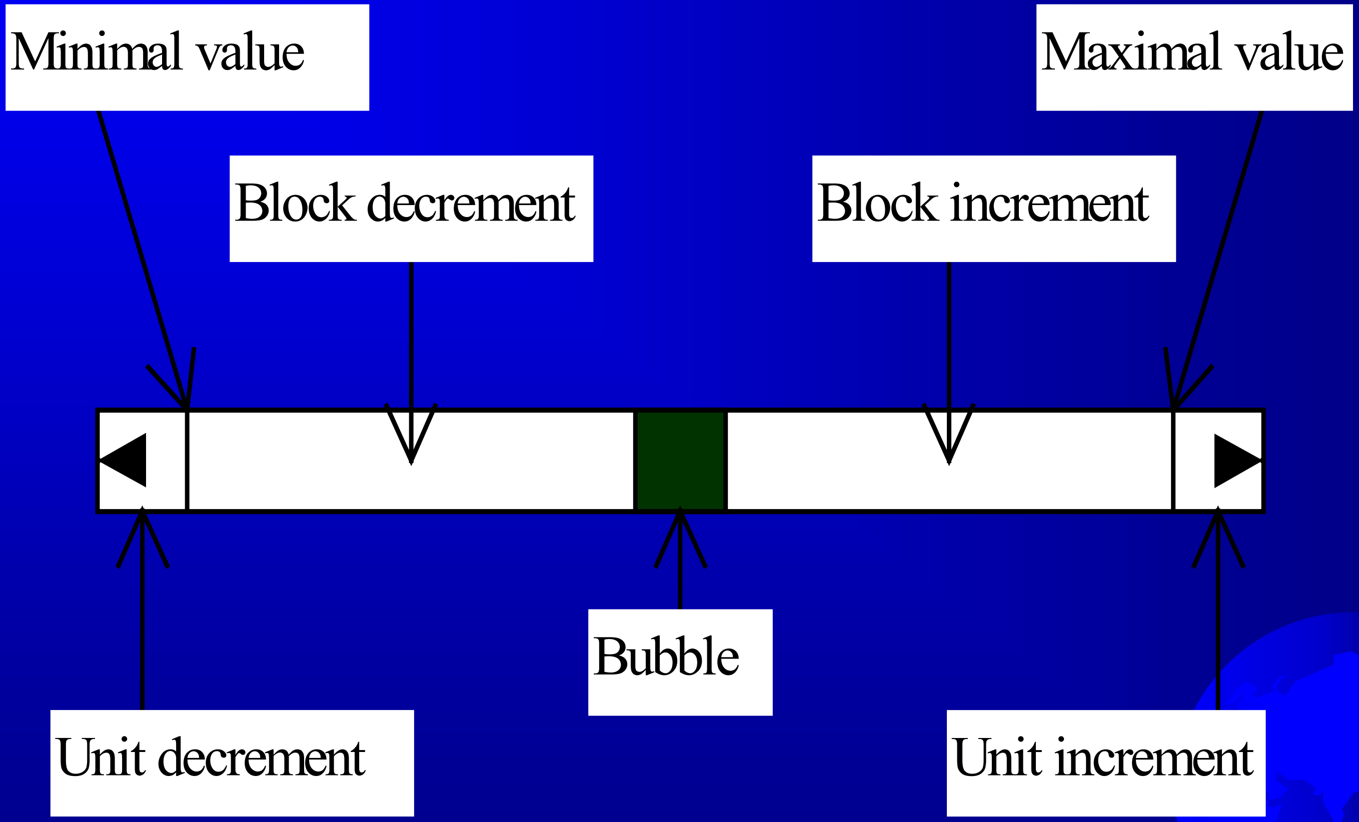
```
HNGDFGHTYTRESDFRETYIUHKLJKJJKBJV
HGDCvbvadfjljkdjfdajadlfjaklajfdljalkjalj
adfjalkaj addg
adgdadgadg a
g adga
gadgagdagag
ghfghfgjdgjdgdgjaaaaaaaa
hhhhhhhhhhhhhhhhhhhh
```

JScrollBar

- A *scroll bar* is a control that enables the user to select from a range of values. The scrollbar appears in two styles: *horizontal* and *vertical*.

Example 9.13: Using Scrollbars

Scroll Bar Properties



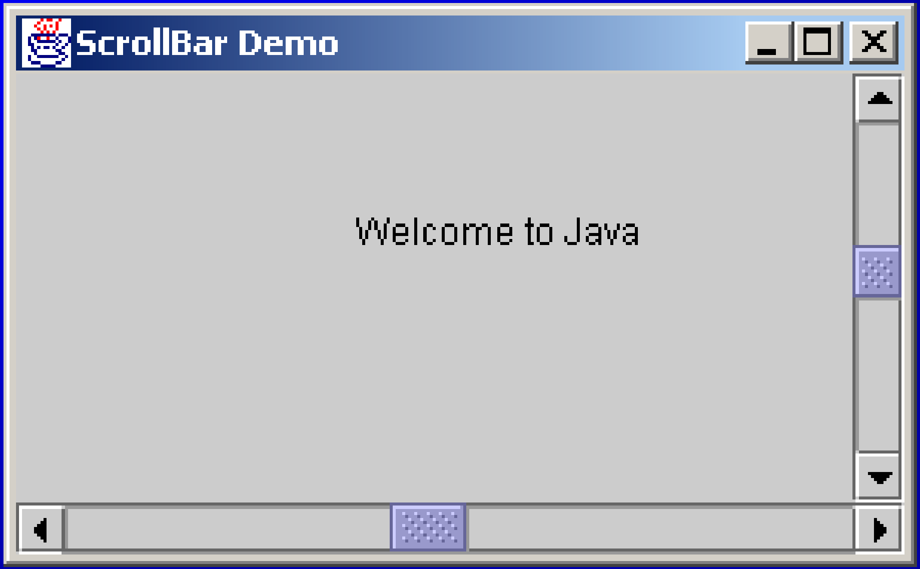
```
// ScrollBarDemo.java: Use scrollbars to move the message
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ScrollBarDemo extends JFrame
                    implements AdjustmentListener
{
  // Declare scrollbars
  JScrollBar jscbHort, jscbVert;

  // Declare a MessagePanel
  MessagePanel messagePanel;
```

```
// Main method
 public static void main(String[] args)
 {
   ScrollBarDemo frame = new ScrollBarDemo();
   // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.pack();
   frame.setVisible(true);
 }

 // Constructor
 public ScrollBarDemo()
 {
   setTitle("ScrollBar Demo");

   // Create a vertical scrollbar
   jscbVert = new JScrollBar();
   jscbVert.setOrientation(Adjustable.VERTICAL);
```

```
// Create a horizontal scrollbar
  jscbHort = new JScrollBar();
  jscbHort.setOrientation(Adjustable.HORIZONTAL);

  // Add scrollbars and message panel to the frame
  messagePanel = new MessagePanel("Welcome to Java");
  getContentPane().setLayout(new BorderLayout());
  getContentPane().add(messagePanel, BorderLayout.CENTER);
  getContentPane().add(jscbVert, BorderLayout.EAST);
  getContentPane().add(jscbHort, BorderLayout.SOUTH);

  // Register listener for the scrollbars
  jscbHort.addAdjustmentListener(this);
  jscbVert.addAdjustmentListener(this);
 }
```

```
// Handle scrollbar adjustment actions
  public void adjustmentValueChanged(AdjustmentEvent e)
  {
    if (e.getSource() == jscbHort)
    {
      // getValue() and getMaximumValue() return int, but for better
      // precision, use double
      double value = jscbHort.getValue();
      double maximumValue = jscbHort.getMaximum();
      double newX =
            (value*messagePanel.getSize().width/maximumValue);
      messagePanel.setXCoordinate((int)newX);
      messagePanel.repaint();
    }
```

```
else if (e.getSource() == jscbVert)
   {
    // getValue() and getMaximumValue() return int, but for better
    // precision, use double
    double value = jscbVert.getValue();
    double maximumValue = jscbVert.getMaximum();
    double newY =
           (value*messagePanel.getSize().height/maximumValue);
    messagePanel.setYCoordinate((int)newY);
    messagePanel.repaint();
   }
  }
}
```



Welcome to Java

JScrollPane

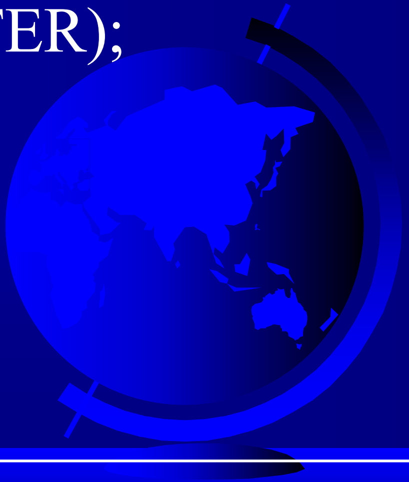
- *A scroll pane* is a component that supports automatically scrolling without coding.

Example 9.14: Using Scroll Panes

```
// ScrollPaneDemo.java: Use scroll pane to view large maps
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class ScrollPaneDemo extends JFrame implements ItemListener
{
  // Create images in labels
  private JLabel lblIndianaMap =
    new JLabel(new ImageIcon("images/indianaMap.gif"));
  private JLabel lblOhioMap =
    new JLabel(new ImageIcon("images/ohioMap.gif"));

  // Declare a scroll pane to scroll map in the labels
  private JScrollPane jspMap;
```

```
// Main method
public static void main(String[] args)
{
  ScrollPaneDemo frame = new ScrollPaneDemo();
  // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setSize(300, 300);
  frame.setVisible(true);
}

// Default constructor
public ScrollPaneDemo()
{
  setTitle("ScrollPane Demo");

  // Create a scroll pane with northern California map
  jspMap = new JScrollPane(lblIndianaMap);
```

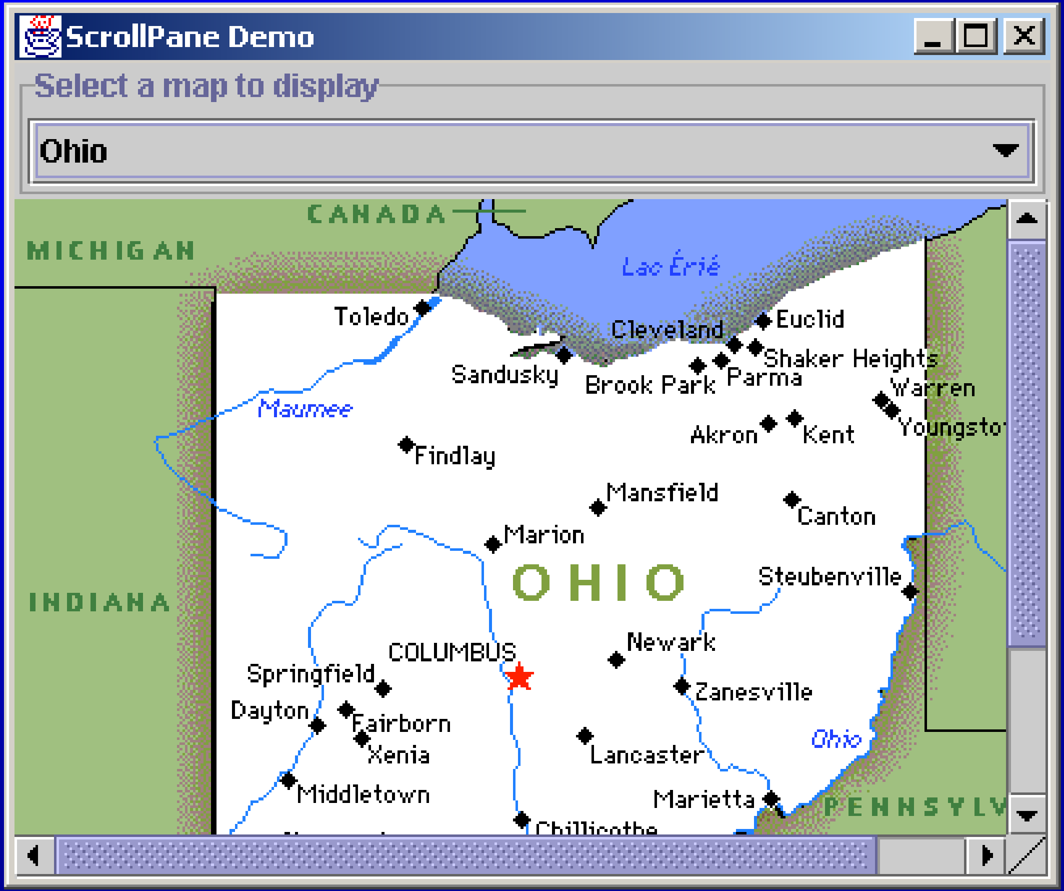
```
// Create a combo box for selecting maps
JComboBox jcboMap = new JComboBox();
jcboMap.addItem("Indiana");
jcboMap.addItem("Ohio");

// Panel p to hold combo box
JPanel p = new JPanel();
p.setLayout(new BorderLayout());
p.add(jcboMap);
p.setBorder(new TitledBorder("Select a map to display"));

// Set row header, column header and corner header
jspMap.setColumnHeaderView(
  new JLabel(new ImageIcon("images/horizontalRuler.gif")));
jspMap.setRowHeaderView(
new JLabel(new ImageIcon("images/verticalRuler.gif")));
   jspMap.setCorner(JScrollPane.UPPER_LEFT_CORNER,
     new CornerPanel(JScrollPane.UPPER_LEFT_CORNER));
   jspMap.setCorner(ScrollPaneConstants.UPPER_RIGHT_CORNER,
     new CornerPanel(JScrollPane.UPPER_RIGHT_CORNER));
   jspMap.setCorner(JScrollPane.LOWER_RIGHT_CORNER,
     new CornerPanel(JScrollPane.LOWER_RIGHT_CORNER));
   jspMap.setCorner(JScrollPane.LOWER_LEFT_CORNER,
     new CornerPanel(JScrollPane.LOWER_LEFT_CORNER));

   // Add the scroll pane and combo box panel to the frame
   getContentPane().add(jspMap, BorderLayout.CENTER);
   getContentPane().add(p, BorderLayout.NORTH);

   // Register listener
   jcboMap.addItemListener(this);
  }
```



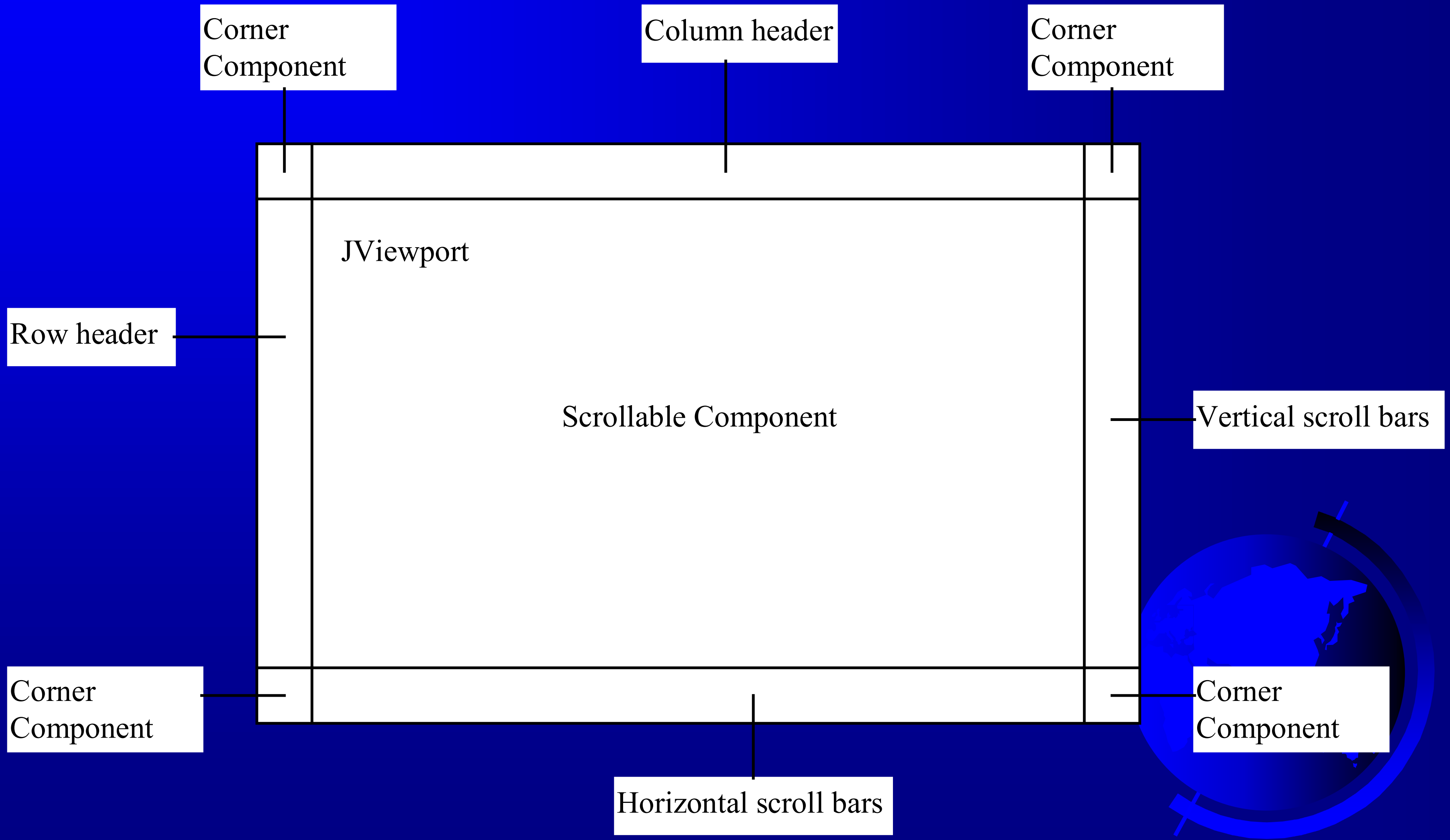
```
public void itemStateChanged(ItemEvent e)
 {
   String selectedItem = (String)e.getItem();
   if (selectedItem.equals("Indiana"))
   {
     // Set a new view in the view port
     jspMap.setViewportView(lblIndianaMap);
   }
   else if (selectedItem.equals("Ohio"))
   {
     // Set a new view in the view port
     jspMap.setViewportView(lblOhioMap);
   }
   // Revalidate the scroll pane
   jspMap.revalidate();
 }
}
```

```
// A panel displaying a line used for scroll pane corner
class CornerPanel extends JPanel implements ScrollPaneConstants
{
  // Line location
  private String location;

  // Constructor
  public CornerPanel(String location)
  {
    this.location = location;
  }
```

```
// Draw a line depending on the location
 public void paintComponent(Graphics g)
 {
   super.paintComponents(g);

   if (location == "UPPER_LEFT_CORNER")
     g.drawLine(0, getSize().height, getSize().width, 0);
   else if (location == "UPPER_RIGHT_CORNER")
     g.drawLine(0, 0, getSize().width, getSize().height);
   else if (location == "LOWER_RIGHT_CORNER")
     g.drawLine(0, getSize().height, getSize().width, 0);
   else if (location == "LOWER_LEFT_CORNER")
     g.drawLine(0, 0, getSize().width, getSize().height);
 }
}
```


ScrollPane Demo
Select a map to display
Ohio
CANADA
MICHIGAN
INDIANA
PENNSYLV
Lac Érié
Maumee
Ohio
OHIO
Toledo
Sandusky
Cleveland
Euclid
Shaker Heights
Brook Park
Parma
Warren
Youngsto
Akron
Kent
Findlay
Mansfield
Canton
Marion
Steubenville
COLUMBUS
Newark
Springfield
Zanesville
Dayton
Fairborn
Xenia
Lancaster
Middletown
Marietta
Chillicothe

Scroll Pane Structures



JTabbedPane

☞ *A tabbed pane* provides a set of mutually exclusive tabs for accessing multiple components.

Example 9.15: Using Tabbed Panes

```
// TabbedPaneDemo.java: Use tabbed pane to select figures
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.TitledBorder;

public class TabbedPaneDemo extends JFrame
                         implements ItemListener
{
  // Create a tabbed pane to hold figure panels
  private JTabbedPane jtpFigures = new JTabbedPane();

  // Radio buttons for specifying where tab is placed
  private JRadioButton jrbTop, jrbLeft, jrbRight, jrbBottom;
```

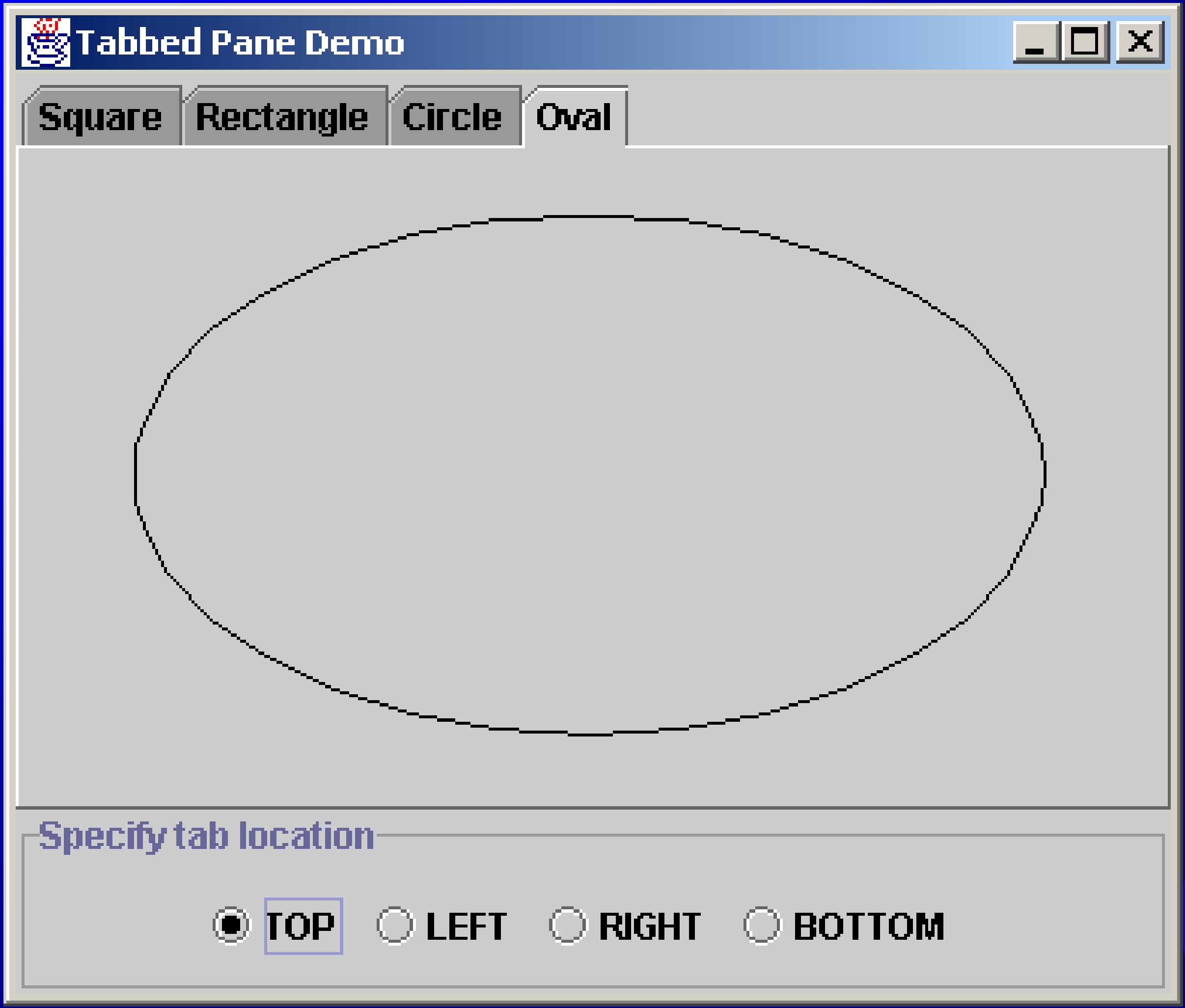
```
// Main method
 public static void main(String[] args)
 {
   TabbedPaneDemo frame = new TabbedPaneDemo();
   // frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setSize(200, 300);
   frame.setVisible(true);
 }

 // Constructor
 public TabbedPaneDemo()
 {
   setTitle("Tabbed Pane Demo");

   jtpFigures.add(new FigurePanel(FigurePanel.SQUARE), "Square");
   jtpFigures.add(
new FigurePanel(FigurePanel.RECTANGLE), "Rectangle");
  jtpFigures.add(new FigurePanel(FigurePanel.CIRCLE), "Circle");
  jtpFigures.add(new FigurePanel(FigurePanel.OVAL), "Oval");

  // Panel p to hold radio buttons for specifying tab location
  JPanel p = new JPanel();
  p.add(jrbTop = new JRadioButton("TOP"));
  p.add(jrbLeft = new JRadioButton("LEFT"));
  p.add(jrbRight = new JRadioButton("RIGHT"));
  p.add(jrbBottom = new JRadioButton("BOTTOM"));
  p.setBorder(new TitledBorder("Specify tab location"));

  // Group radio buttons
  ButtonGroup btg = new ButtonGroup();
  btg.add(jrbTop);
  btg.add(jrbLeft);
```

```
btg.add(jrbRight);
  btg.add(jrbBottom);

  // Place tabbed pane and panel p into the frame
  this.getContentPane().add(jtpFigures, BorderLayout.CENTER);
  this.getContentPane().add(p, BorderLayout.SOUTH);

  // Register listeners
  jrbTop.addItemListener(this);
  jrbLeft.addItemListener(this);
  jrbRight.addItemListener(this);
  jrbBottom.addItemListener(this);
}
```

```
// Handle radio button selection
 public void itemStateChanged(ItemEvent e)
 {
   if (jrbTop.isSelected())
     jtpFigures.setTabPlacement(SwingConstants.TOP);
   else if (jrbLeft.isSelected())
     jtpFigures.setTabPlacement(SwingConstants.LEFT);
   else if (jrbRight.isSelected())
     jtpFigures.setTabPlacement(SwingConstants.RIGHT);
   else if (jrbBottom.isSelected())
     jtpFigures.setTabPlacement(SwingConstants.BOTTOM);
 }
}
```

```
// The panel for displaying a figure
class FigurePanel extends JPanel
{
  final static int SQUARE = 1;
  final static int RECTANGLE = 2;
  final static int CIRCLE = 3;
  final static int OVAL = 4;
  private int figureType = 1;

  // Constructing a figure panel
  public FigurePanel(int figureType)
  {
    this.figureType = figureType;
  }
```

```
// Drawing a figure on the panel
 public void paintComponent(Graphics g)
 {
   super.paintComponent(g);

   // Get the appropriate size for the figure
   int width = getSize().width;
   int height = getSize().height;
   int side = (int)(0.80*Math.min(width, height));

   switch (figureType)
   {
     case 1:
       g.drawRect((width-side)/2, (height-side)/2, side, side);
       break;
case 2:
       g.drawRect((int)(0.1*width), (int)(0.1*height),
         (int)(0.8*width), (int)(0.8*height));
       break;
     case 3:
       g.drawOval((width-side)/2, (height-side)/2, side, side);
       break;
     case 4:
       g.drawOval((int)(0.1*width), (int)(0.1*height),
         (int)(0.8*width), (int)(0.8*height));
       break;
    }
  }
}
```



Specify tab location: [✓ TOP] LEFT RIGHT BOTTOM