Distributed Programming in Java

Session: 5

Lists and Panes

Objectives

- Explain JColorChooser, its use, and how to create it
- List and explain methods and event handling for JColorChooser
- Explain JList, its use, and how to create it
- List and explain methods and event handling for JList
- Explain JComboBox, its use, and how to create it
- List and explain methods and event handing of JComboBox
- Explain JSplitPane, its use, and how to create it
- Explain how to add, configure, and display a JSplitPane
- Explain JTabbedPane, its use and how to create it
- Explain how to add and display a JTabbedPane
- Explain event handling for JSplitPane and JTabbedPane

Introduction

- The `JColorChooser` has a control panel which allows the user to manipulate color.
- There are three levels of API in the `JColorChooser` class.
  - A static convenience method which displays a modal dialog and returns the color selected.
  - A static method for creating a dialog with that is invoked to set the color.
  - Directly creating instances of `JColorChooser` class within a container.

JColorChooser

- `JColorChooser` is a class which creates a component allowing you choose a color or a shade dynamically.
- The `JColorChooser` provides three tabs to choose a color:

1. Swatches: To select a color from a collection of swatches.
2. HSB: By specifying the Hue, Saturation, and Brightness.
3. RGB: By specifying the `Red`, `Green`, and `Blue` values.

- A `JColorChooser` can be created using any one of the following constructors:
    - `public JColorChooser()`
    - `public JColorChooser(Color initialColor)`
- The method `showDialog(Component parent, String title, Color initialColor)` displays the standard color dialog box.

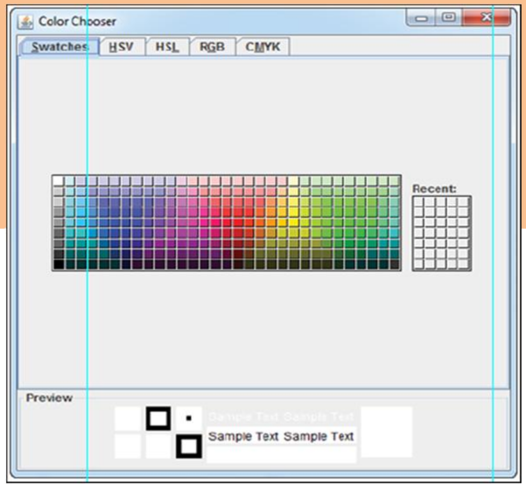
Adding JColorChooser

- A `JColorChooser` can also be added as a component. It is added to the container using `add()` method of the container.
- The `add()` method takes an object of the `JColorChooser` class as an argument.
- However, adding `JColorChooser` as a component occupies more space.
- Code Snippet shows how to add the `JColorChooser` as a component to the frame.

Code Snippet

```
JFrame frmColorChooser;
JColorChooser ccrChoser;
// Creates a frame with the title "Color Chooser"
    frmColorChooser = new JFrame("Color Chooser");
// Creates a color chooser
    ccrChoser = new JColorChooser();
// Adds the color chooser to the frame
   frmColorChooser.getContentPane().add(ccrChoser);
```

- Output:



Event Handling of JColorChooser [1-2]

- To handle the events of JColorChooser, perform the following steps:
    - **Retrieve the ColorSelectionModel**
        - The JColorChooser has a method getSelectionModel() which returns an object of the ColorSelectionModel interface. The ColorSelectionModel represents a model which supports selection of a color.
    - **Register a ChangeListener object**
        - The addChangeListener() method is used to register a javax.swing.event.ChangeListener interface object with the ColorSelectionModel to handle the event of selecting a color.
    - Code Snippet shows how to retrieve the ColorSelectionModel of a JColorChooser and register a ChangeListener with the ColorSelectionModel.

Code Snippet

```
// Declare the chooser and model
   JColorChooser ccrChoser;
   ColorSelectionModel colorSelectionModel;
// Retrieve the color selection model
   colorSelectionModel = ccrChoser.getSelectionModel();
// Register a change listener with the selection model
   colorSelectionModel.addChangeListener(new ChangeListener() { . .
. );
```

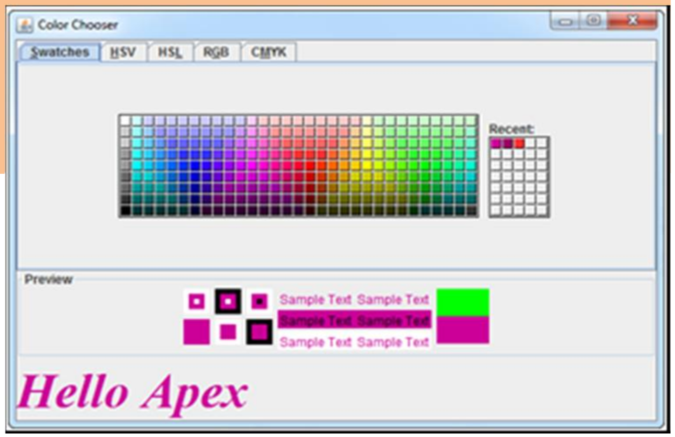
Event Handling of JColorChooser [2-2]

- **Handle the event in `stateChanged()` method**
  - The `ChangeListener` interface has a method `stateChanged()` which is invoked when you select a color. You provide the action code in this method.
  - Code Snippet how to handle the color selection event and change the background color of a label.

Code Snippet

```
// Declare a chooser and a label
   JLabel lblMessage;
// Create and add the ccrChoser and lblMessage
   . . .
// Retrieve the color selection model
   colorSelectionModel = ccrChoser.getSelectionModel();
// Add a change listener to the color selection model
   colorSelectionModel.addChangeListener(new ChangeListener() {
   public void stateChanged(ChangeEvent e) {
     Color clr = ccrChoser.getColor();
     lblMessage.setBackground(clr);
  }
});
```

  - Output:



Methods of JColorChooser

The important methods of `JColorChooser` are as follows:

- `Color getColor():` The method returns the currently selected color.

Code Snippet

```
JColorChooser ccrChooser;
Color clrColor;
// Creates a color chooser
ccrChooser = new JColorChooser();
// Retrieves the current color selected
clrColor = ccrChooser.getColor();
```

- `void setColor(Color color):` The method sets the specified color to be the current color.

Code Snippet

```
JColorChooser ccrChooser;
// Creates a color chooser
ccrChooser = new JColorChooser();
// Sets blue as the current color
ccrChooser.setColor(Color.blue);
```

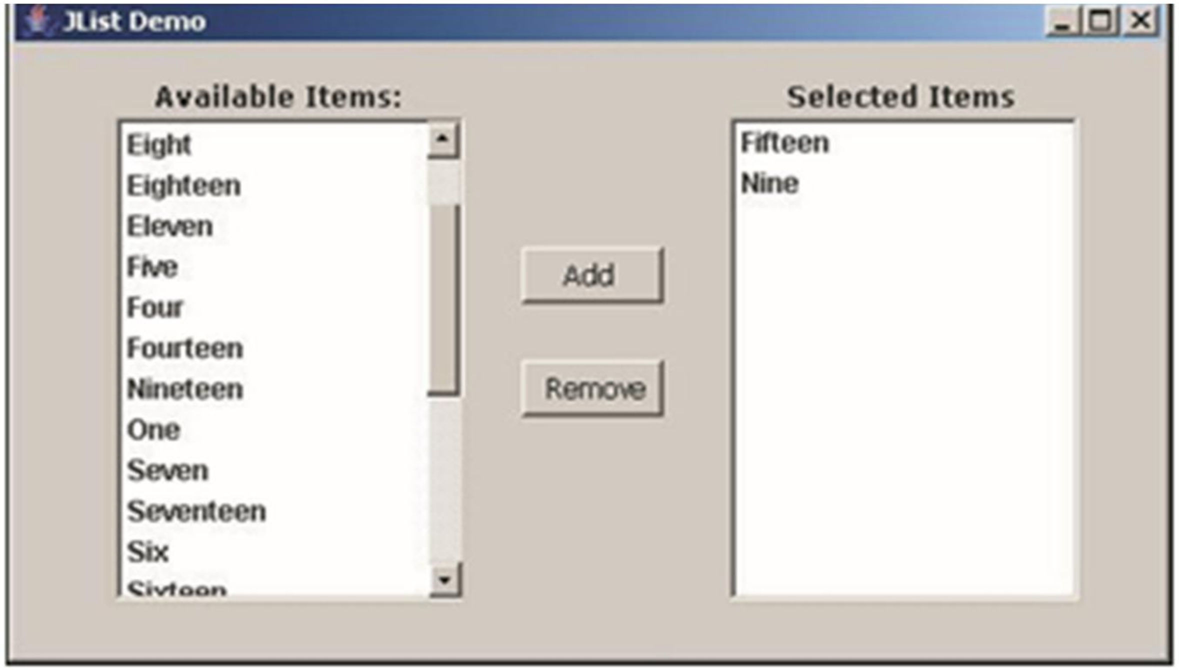
JList Class [1-3]

- `JList` is a class used to displays a group of items.
- It allows you to choose one or more items.
- The items in a `JList` can be displayed in one or more columns.
- A `JList` does not support scrolling inherently, a `JScrollPane` is used to provide the scrolling functionality.
- A `JList` can be created using any one of the following constructors:
  - `public JList()`
  - `public JList(Object[] items)`
  - `public JList(Vector vecItems)`

JList Class [2-3]

- To add a `JList` perform the following steps:

1. Create an instance of `JList`.
2. Create an instance of `JScrollPane` and pass the list instance as an argument.
3. Add the `JScrollPane` instance to a container.



Available Items: Eight, Eighteen, Eleven, Five, Four, Fourteen, Nineteen, One, Seven, Seventeen, Six, Sixteen

Selected Items: Fifteen, Nine

`JList`

JList Class [3-3]

- Code Snippet shows how to create and populate the `JList`.

Code Snippet

```
JPanel pnlCountries;
JList lstCountries;
JScrollPane scpScroller;
String[] strCountries;
. . .
// Create a string array and initialize it the names of the
countries
    String[] strCountries = {"Australia", "Belgium", "Canada",.
. . "India"};
// Create a list with the string array
    lstCountries = new JList(strCountries);
// Provide the list to the scrollpane to facilitate scrolling
    scpScroller = new JScrollPane(lstCountries);
// Add the JScrollPane to the panel
    pnlCountries.add(scpScroller);
```

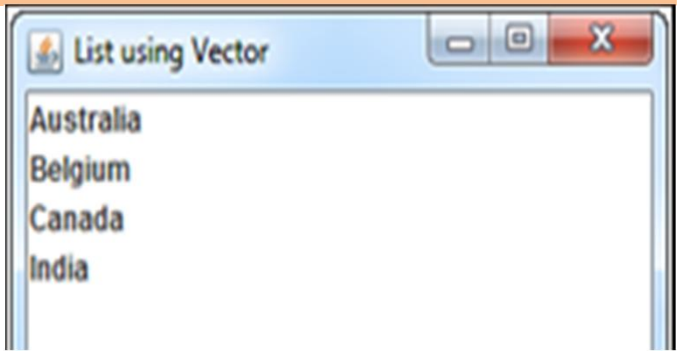
JList with Vector

- Code Snippet shows how to create a list with a `Vector` representing names of the countries.

Code Snippet

```
// Creates a vector to store the names of the countries Vector
   vecCountries = new vector();
// Adds the names of the countries to the vector
   vecCountries.addElement("Australia");
   vecCountries.addElement("Belgium");
   vecCountries.addElement("Canada");
    . . .
   vecCountries.addElement("India");
// Creates a list with a Vector
   JList lstCountries = new JList(vecCountries);
```

- Output:



```
Australia
Belgium
Canada
India
```

Configuring and Displaying JList

- To add a `JList`, perform the following steps:
  - Create an instance of `JList`.
  - Create an instance of `JScrollPane` and pass the list instance as an argument.
  - Add the `JScrollPane` instance to a container.
  - Code Snippet shows how to add the list to the panel.

Code Snippet

```
JPanel pnlCountries;
JList lstCountries;
JScrollPane scpScroller;
String[] strCountries;
. . .
// Creates string array and initialize it with countries name
String[] strCountries = {"Australia", "Belgium", "Canada",. . .
"India"};
// Creates a list with the string array
lstCountries = new JList(strCountries);
// Provides the list to the scrollpane to facilitate scrolling
scpScroller = new ScrollPane(lstCountries);
// Adds the JScrollPane to the panel pnlCountries.add(scpScroller);
```

Methods of JList [1-3]

- To display a `JList` no special step is required, when a `JScrollPane` is initialized with a `JList` instance, it is displayed by default.
- To configure a `JList`, invoke the following methods:
  - `public void setSelectionMode(int selectMode):` This method sets the selection mode for the list. The selection mode determines how the items can be selected. Items are selected as single, multiple selection contiguously, or in any order.
  - `public void setLayoutOrientation(int layoutOrient):` This method sets the layout of the cells in the list.
  - `public void setVisibleRowCount(int count):` This method sets the number of rows to be displayed in the list without scrollbars.

Methods of JList [2-3]

- `public void clearSelection():` This method is used to clear the selection of the items from the list.
- `public int getSelectedIndex():` This method returns index of the selected item. The index is zero-based. If the first item is selected then the index is 0.
- `public int[] getSelectedIndices():` This method returns an array of indices of selected items. The array containing the indices of multiple items selected.
- `public boolean isSelectionEmpty():` This method returns true if no item is selected.
- `public Color getSelectionBackground():` The `getSelectionBackground()` method returns the background color of the selected item.
- `public void setListData(Object[] items):` This method is used to provide the `JList` with items. The items can be provided as an array of Object or Vector. For an Object array a String array is typically used.

Methods of JList [3-3]

- `public void setSelectedIndex(int index)`: This method is used to select a single item programmatically. A zero-based integer index is passed as an argument.
- `public void setSelectedIndices(int[] indices)`: This method is used to select multiple items programmatically. An array of integers containing the indices of items is passed as an argument. The indices are zero-based.
- `public void setSelectionBackground(Color c)`: The `setSelectionBackground()` method sets the background color of the selected item.
- `public boolean isSelected(int index)`: The `isSelectedIndex()` method is used to check if the item at the specified index is selected.
- `public void setSelectionForeground(Color c)`: The `setSelectionForeground()` method is used to set the foreground color of the selected item from the list.

Event Handling of JList

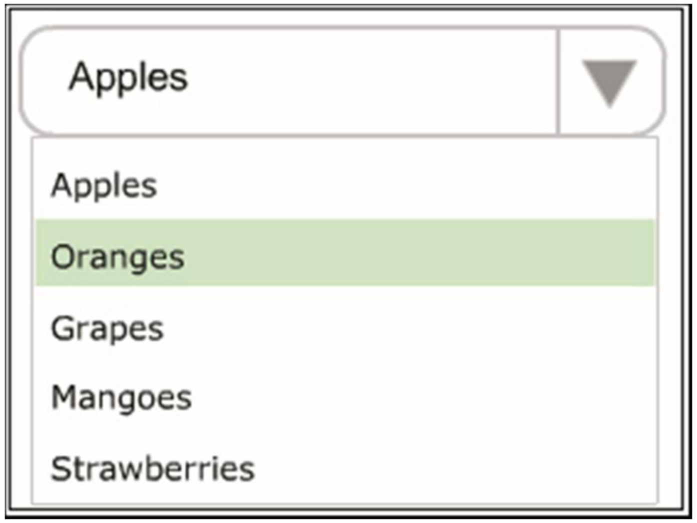
- To handle the events of a `JList`, you perform the steps:
  - Register a `ListSelectionListener`.
  - Add the action code in `valueChanged()` method.
  - Code Snippet demonstrates the code to register the list selection listener and add the `valueChanged ()` method.

Code Snippet

```
JList lstCountries;
. . .
lstCountries.addListSelectionListener(new
ListSelectionListener() {
        public void valueChanged(ListSelectionEvent e) {
         // Add the action code
                . . .
                . . .
        }
});
```

JComboBox Class

- A `JComboBox` is a combination of a drop-down and a textfield. The drop-down displays one or more items to choose from.
- The textfield allows you to type a new item not available in the drop-down.
- A `JComboBox` occupies less space and yet allows you to choose one of several items from a list.
- A `JComboBox` allows you to select only one item unlike a `JList` which allows multiple selections.
- A `JComboBox` can be created using any one of the following constructors:
  - `public JComboBox()`
  - `public JComboBox(Object[] items)`
  - `public JComboBox(Vector items)`



Configuring and Displaying JComboBox [1-2]

- To add a `JComboBox` perform the following steps:

1
- Create an instance of `JComboBox`.

2
- Add it to the container using `add`() method of the container.

- The `JComboBox` component is displayed when it is added to a container.
- The `JComboBox` is configured using `setEditable(boolean editable).`
- This method sets the editable property to allow inputting of text.
- If editable is set to true, you can input text in the combo box.
- By default, it is set to false.

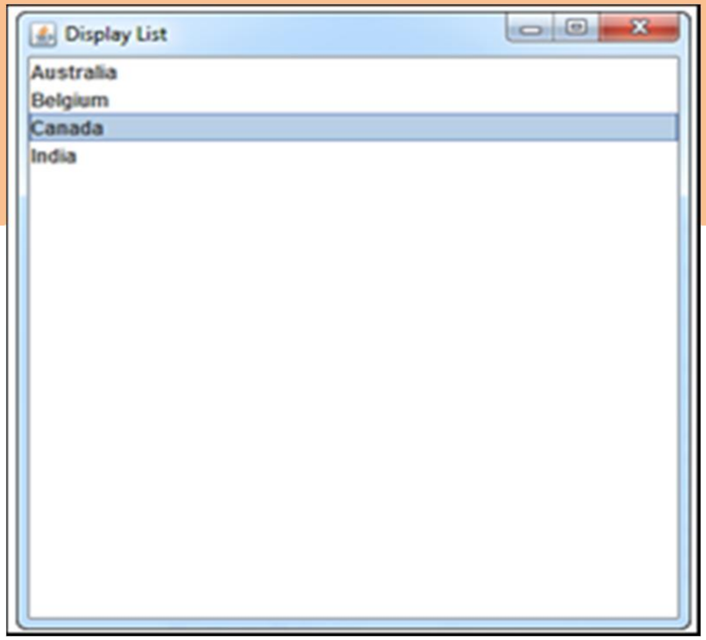
Configuring and Displaying JComboBox [2-2]

- Code Snippet shows how to add the combo box to the panel and enable the input of the text.

Code Snippet

```
// Creates a panel
   JPanel pnlFonts = new JPanel();
// Creates a string array with 10 elements
   String[] strFonts = new String[10];
// Initialize the array with font names
     . . .
// Creates a combo box with the items from string array
   JComboBox cboFonts  = new JComboBox(strFonts);
// Enables the inputting of text
   cboFonts.setEditable(true);
// Adds the combo box to the panel
   pnlFonts.add(cboFonts);
```

- Output:



Event Handling Using ActionListener

- The events of a `JComboBox` can be handled by `ActionListener` or `ItemListener`.
- To handle the events by a `ActionListener` you perform the following steps:
  - Register `ActionListener`
  - Add the action code in `actionPerformed()` method

Code Snippet

```
JComboBox cboFonts;
 . . .
 // Register the ActionListener
cboFonts.addActionListener(new ActionListener()
{
        // Handle the action performed event
        public void actionPerformed(ActionEvent e)
        {
                // Add the code to handle the event
        }
});
```

Event Handling Using ItemListener

To handle the events by a ItemListener, perform the following steps:

- Register a `ItemListener`
- Add the action code in `itemStateChanged()` method

Code Snippet

```
JComboBox cboFonts  = new JComboBox(strFonts);
// Register the ItemListener
cboFonts.addItemListener(new ItemListener() {  // Handle the
item state changed event
    public void itemStateChanged(ItemEvent e)    {
         // Check if it a Select or Deselect event
         if (e.getStateChange() == ItemEvent.SELECTED)
        {
               // SELECT event
                . . .
        } else {
                // DESELECT event
         . . . }    } });
```

Methods of JComboBox [1-3]

- `public void addItem(Object item)`: The method is used to add an item to the `JComboBox`.
- `public int getItemCount()`: The method returns the number of items in the `JComboBox`.
- `public int getSelectedIndex()`: The method returns an integer representing the index of the item selected. If none of the items is selected in the `JComboBox` the `getSelectedIndex()` method returns the value 1.
- `public Object getSelectedItem()`: The method returns an object representing the item selected. The returned object is typically cast to the appropriate class. If none of the items is selected, null is returned.
- `public void setEditable(boolean editable)`: This method is used to make the `JComboBox` editable. You can type a new item in a `JComboBox` only if it is editable.

Methods of JComboBox [2-3]

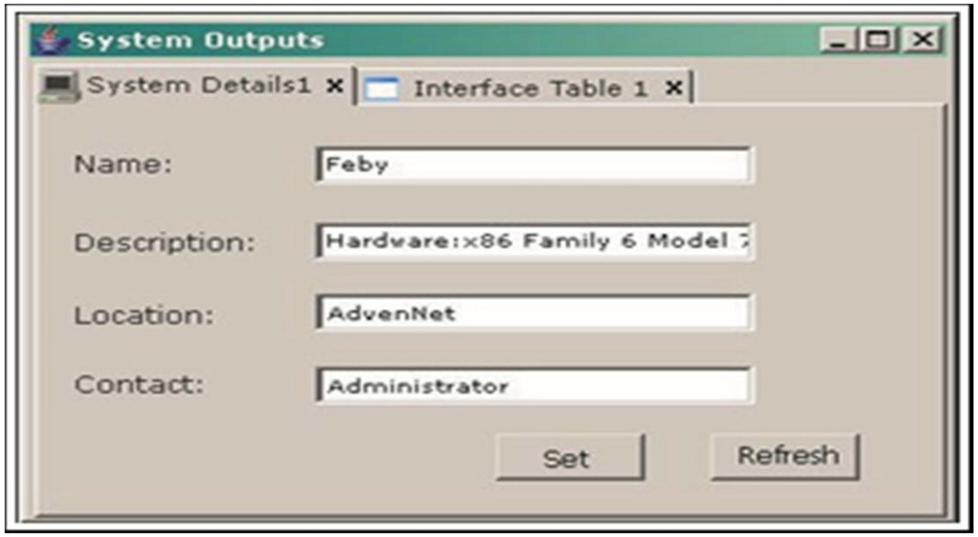
- `public void setSelectedIndex(int index)`: This method is used to select an item programmatically. You specify the zero-based index of the item to be selected. The index specified should be in the range of 0 to n-1, where n is the item count. If the index specified is not in the proper range this method throws `IllegalArgumentException`.
- `public void setSelectedItem(Object item)`: This method is used to select an item programmatically. You specify the object to be selected from the list of items available. If the object specified does not exist, no action is taken.
- `public Object getItemAt(int index)`: The `getItemAt()` method of `JComboBox` is used to retrieve the item at the specified index.
- `public int getMaximumRowCount()`: The `getMaximunRowCount()` method of `JComboBox` is used to retrieve the maximum number of items the combo box can display without a scrollbar.

Methods of JComboBox [3-3]

- `public Object[] getSelectedObjects()`: The `getSelectedObjects()` method of `JComboBox` is used to retrieve multiple items selected.
- `public void insertItemAt(Object item, int index)`: The `insertItemAt()` method of `JComboBox` is used to insert an item at the specified index.
- `public void removeAllItems()`: The `removeAllItems()` method of `JComboBox` is used to remove all items from the combobox.
- `public void removeItemAt(int index)`: The `removeItemAt()` method of `JComboBox` is used to remove the item at specified position from the combobox. This method works only if the `JComboBox` uses a data model.

JTabbedPane Class

- A `JTabbedPane` is a component which allows you to add several components usually panels, to share the same space.
- Each component added to `JTabbedPane` has a tab, which can have a textual label or an icon.
- When a tab is clicked its associated component becomes current and is visible.
- The main advantage of using a `JTabbedPane` is that several panels can be clubbed together to occupy the same space in the GUI.
- It is very convenient to choose the panel based on the tab and to work in a seamless manner.
- Figure shows the tabbed pane.



Name: Feby
Description: Hardware:x86 Family 6 Model 7
Location: AdvenNet
Contact: Administrator

Adding Tabs to JTabbedPane [1-2]

- To add tabs to a `JTabbedPane`, perform the following steps:

1. Create an instance of `JTabbedPane`.
2. Create an instance of a `JPanel` for each tab.
3. Add the components in the respective panels.
4. Use the `addTab()` method of `JTabbedPane` to add the panel.

The various `addTab()` methods of `JTabbedPane` are as follows:

- `void addTab(String, Component)`: The method adds a component with the specified title.
- `void addTab(String, Icon, Component)`: The method adds a component with the specified title and icon.
- `void addTab(String, Icon, Component, String)`: The method adds a component with the specified title, icon, and the tool tip.

Adding Tabs to JTabbedPane [2-2]

- Code Snippet shows how to add a panel with the title ‘English’, icon, and tooltip to the tabbed pane and add the tabbed pane to the frame.

Code Snippet

```
Jframe frmSubjects;
JPanel pnlEnglish;
JTabbedPane tpnSubjects;
  . . .
// Creates a frame with the title “Subjects”
    frmSubjects = new JFrame(“Subjects”);
// Create an image icon
   ImageIcon iconEnglish = new ImageIcon(“English.gif”);
// Adds panel having the title “English”, icon and tooltip to
// tabbedpane
   tpnSubjects.addTab(“English”,iconEnglish,pnlEnglish,
“English Subject”);


// Adds the tabbedpane to the frame
   frmSubjects.getContentPane().add(tpnSubjects);
```

Event Handling

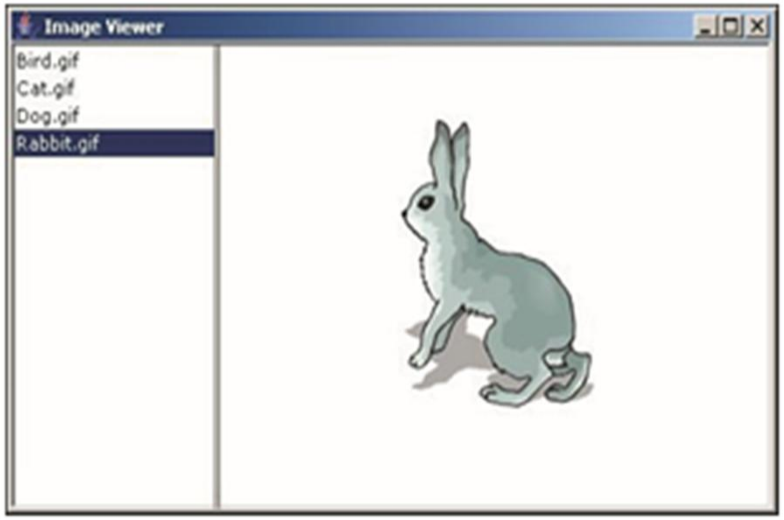
- To handle the change event perform the following steps:
  - Register a `ChangeListener`
  - Add a `stateChanged()` method
  - Code Snippet shows how to handle state change event of the tabbed pane.

Code Snippet

```
JTabbedPane tpnSubjects;
. . .
// Register the ChangeListener
   tpnSubjects.addChangeListener(new ChangeListener() {
   // Handle the state changed event
      public void stateChanged(ChangeEvent e) {
         // Add the action code
            . . .
        }
});
```

JSplitPane [1-3]

- A `JSplitPane` is a component which displays two components and a divider.
- These two components can be displayed horizontally side by side, or vertically one below the other.
- The divider can be dragged to specify how much of the total area is to be given to each component.
- Two small arrows appear at the top of the divider.
- These arrows allow you to collapse (and then expand) either of the components with a single click.
- The main advantage of `JSplitPane` is the ability to dynamically change the sizing requirements per component basis.
- Figure displays the `JsplitPane`.



JSplitPane [2-3]

- A `JSplitPane` is created using any one of the constructors:
  - `JSplitPane()`
  - `JSplitPane(int)`
  - `JSplitPane(int, boolean)`
  - `JSplitPane(int, boolean, Component, Component)`
- Code Snippet shows how to create a `JSplitPane` with components placed side by side and added it to the frame.

Code Snippet

```
JFrame frmExplorer;
JSplitPane spnExplorer;
// Creates a frame
   frmExplorer = new JFrame("Explorer");
// Creates a splitpane
   spnExplorer = new JSplitPane();
// Adds the splitpane to the frame
   frmExplorer.getContentPane().add(spnExplorer);
```

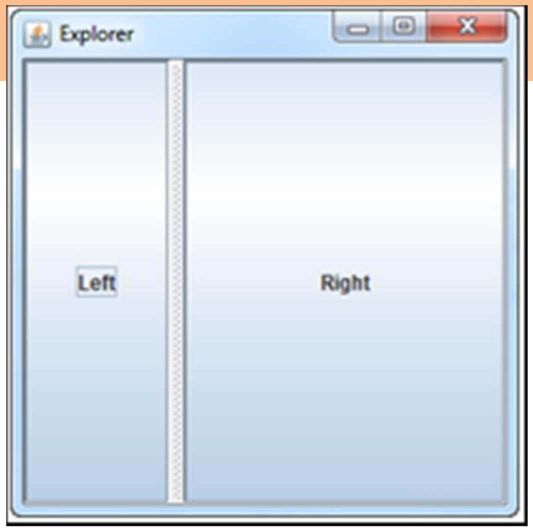
JSplitPane [3-3]

- Code Snippet shows how to create a `JSplitPane` with the horizontal orientation, continuous layout, and panel added to the right and left side.

Code Snippet

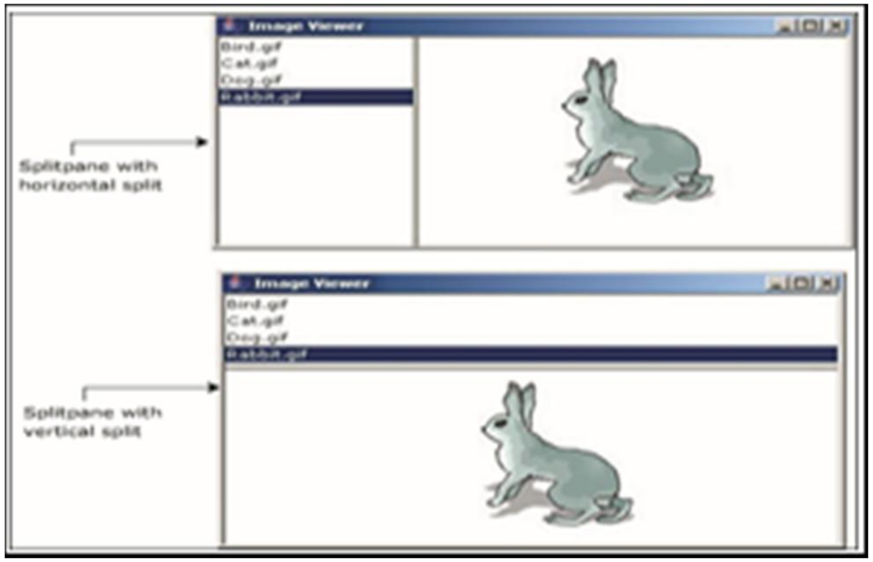
```
JSplitPane spnExplorer;
JPanel pnlLeft, pnlRight;
// Create a frame
   frmExplorer = new JFrame("Explorer");
// Create a splitpane with the given orientation and continuous
    layout. spnExplorer = new
JSplitPane(JSplitPane.HORIZONTAL_SPLIT,true, pnlLeft,pnlRight);
// Add the splitpane to the frame
   frmExplorer.getContentPane().add(spnExplorer);
```

- Output:



Configuring and Displaying JSplitPane

- To add two components with the desired orientation you use the constructor of the `JSplitPane`.
- The components are configured based on the orientation used.
- The orientation `JSplitPane.HORIZONTAL_SPLIT` places the two components side by side.
- The orientation `JSplitPane.VERTICAL_SPLIT` places the two components one below the other.
- Figure displays the configuring and displaying of `JSplitPane`.



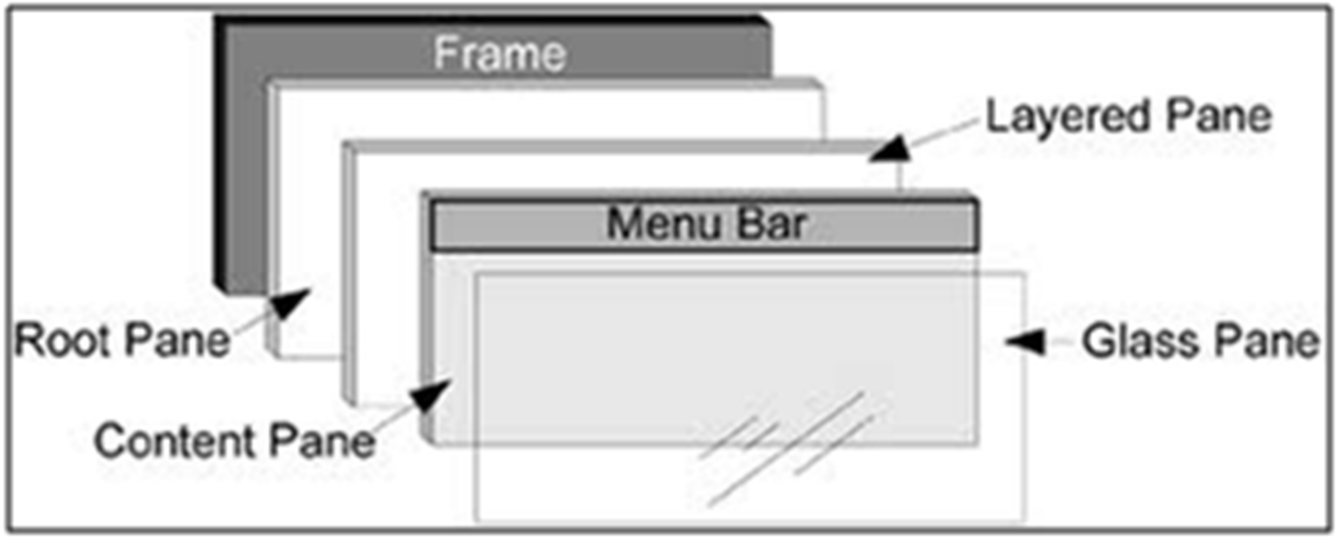
Methods of JSplitPane

- Table lists the methods that can be used to configure `JSplitPane`.

Method	Description
`void setOneTouchExpandable (boolean)`	The `setOneTouchExpandable()` method is used to activate the feature of one touch expansion of the split panes. This feature is Look-And-Feel dependent and not available by default. Set true to activate. Example: `spnExplorer.setOneTouchExpandable(true);`
`void setDividerLocation(int )`	The method is used to programmatically set the divider location of the split pane. The minimum and preferred size of the component is used to determine the initial location of the divider. Example: `spnExplorer. setDividerLocation(200);`

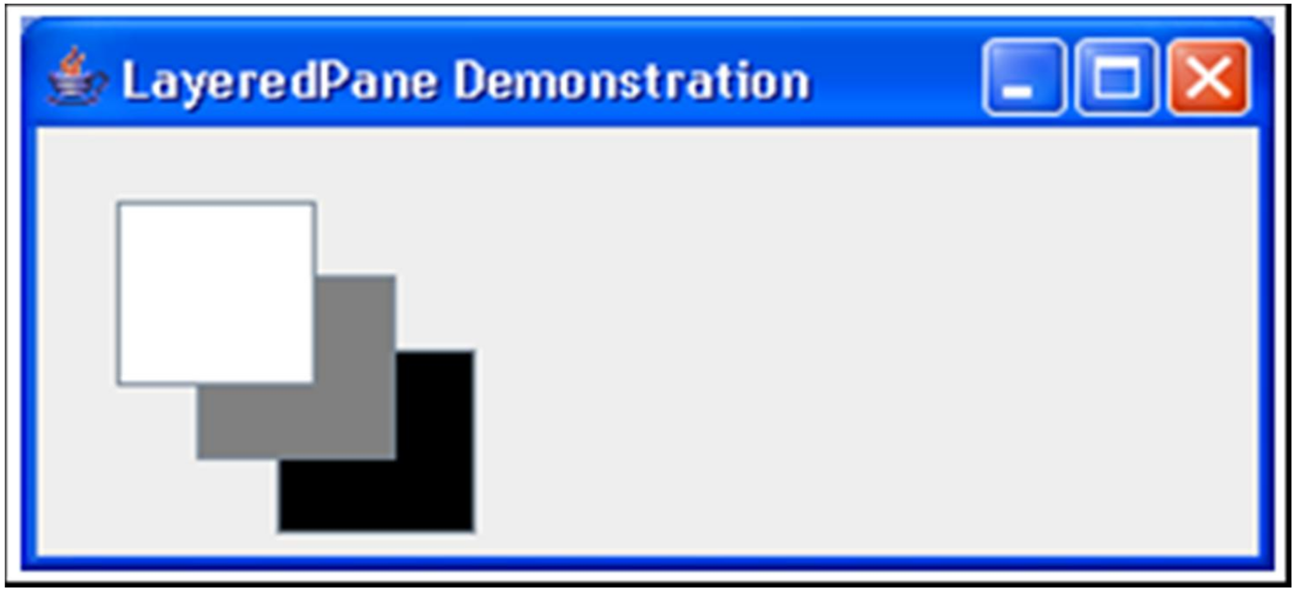
JLayeredPane Class [1-2]

- Each container component that acts as top-level container defines a set of panes.
- At the top of the hierarchy, there is a root pane which is an instance of `JRootPane`.
- The main purpose of the `JRootPane` object is to manage other panes which comprises glass pane, content pane, and layered pane.
- Figure shows the pane in a container.



JLayeredPane Class [2-2]

- The `JLayeredPane` API defines several layers that allow the components to overlap each other based on their depth value.
- Higher the depth value, closer the component occupying the top position in the container. The top position components are overlapped with the components with a lower depth value.
- Figure shows a `JLayeredPane`.



Configuring a JLayeredPane

- When a default `JRootPane` is created for a class that implements `RootPaneContainer`, the `JRootPane` creates a `JLayeredPane` for its component area.
- The `JLayeredPane` has only a single constructor.

  Syntax:

  ```
  public JLayeredPane()
  ```
- Code Snippet shows how to create a `JLayeredPane`.

Code Snippet

```
layeredPane = new JLayeredPane();
layeredPane.setPreferredSize(new Dimension(310, 320));
layeredPane.setBorder(BorderFactory.createTitledBorder("Move the
mouse"));
layeredPane.addMouseMotionListener(new MouseMotionAdapter()
 { . . . });
```

Methods of JLayeredPane [1-2]

- The layer can be set with layout manager constraints when a component is added to it.
- Table lists the predefined constants for special values of `JLayeredPane`.

Constant	Description
`FRAME_CONTENT_LAYER`	The name of the machine where the resource resides.
`PALETTE_LAYER`	The path to the file on the host.
`MODAL_LAYER`	The port number to which to connect (optional).
`POPUP_LAYER`	Level 300 for holding popup menus and tooltips.
`DRAG_LAYER`	Level 400 to ensure the dragged objects remain on top.

Methods of JLayeredPane [2-2]

- Table lists some of the methods of the `JLayeredPane`.

Method	Description
`addImpl()`	Adds the specified component to this container at the specified index. This also notifies the layout manager to add the component to this container's layout using the specified constraints object via the `addLayoutComponent` method.
`getComponentsInLayer()`	Return components of a specified layer in array format.
`getIndexOf()`	Returns the index of a specified component.
`moveToFront()`	Moves the specified component to the top in its current layer.
`moveToBack()`	Moves the specified component to bottom position in its current layer.
`setPosition()`	Moves the specified component to a specified position within its current layer. Zero is the topmost position and -1 is the bottom most position in a layer.

Summary

- The JColorChooser class creates a component that allows you to choose a color dynamically.
- The JList is a component that displays a group of objects in one or more columns for the user to choose from.
- The JComboBox combines a button or editable field and a drop-down list. This lets the user select a value from the drop-down list and also provide value through the editable field.
- The JTabbedPane class allows several components to share the same space. The user is given the freedom to choose which components to view by selecting the tab of the desired component.
- JSplitPane divides two components. The two components are graphically divided based on the look and feel implementation.
- A container gets its depth from the JLayeredPane. It allows components to overlap each other when needed.
- Components are placed at specified depth along their Z-axis. Higher numbered components are placed in top of lower numbered components.