Distributed Programming in Java

Session: 4

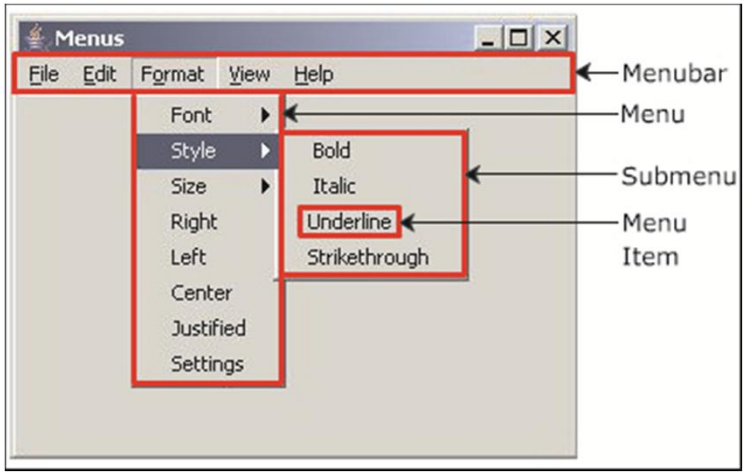
Menu Components

Objectives

- Describe JMenuBar and its methods
- Describe and explain JMenu and JMenuItem
- Describe and explain JCheckBoxMenuItem and JRadioButtonMenuItem
- Describe JPopupMenu and its methods
- Explain JFileChooser and its use
- Explain and state the syntax of the methods of JFileChooser
- Explain how to filter files in JFileChooser and also how to subclass it
- Explain JToolBar
- State the syntax of the methods of JToolBar

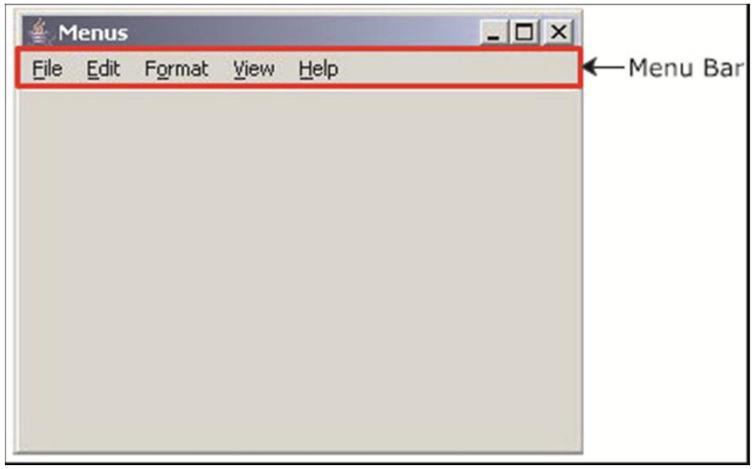
Introduction to Menu System

- A menu system provides a menu bar below the window's title bar.
- This bar contains items called a menu. When you click a menu it drops down and displays various choices called menu items.
- Each of these menu items is associated with actions to perform desired action. You can choose the task by clicking a menu item.
- Alternately, the GUI provides a pop-up menu, which is invisible at first sight.
- When you right-click the mouse on the screen, a menu containing menu items pops up besides where you clicked the mouse.
- It allows you to conveniently choose a desired task.



JMenuBar [1-2]

- A `JMenuBar` is a class which is used to create a menu bar.
- You typically see a menu bar in a window just below the title bar of a frame.
- A menu bar contains menu items, for example 'File', 'Edit', 'View', and so on.
- A menu bar can be created using the following constructor of the `JMenuBar` class:
  - `JMenuBar()`: The top level container has `setMenuBar()` method to add the `JMenuBar` to it. This method takes an object of the `JMenuBar` class as an argument.



JMenuBar [2-2]

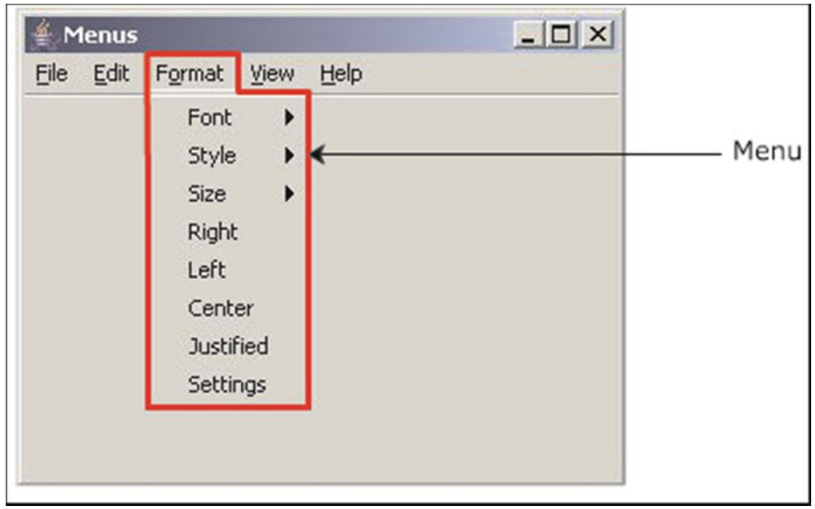
- Code Snippet shows how to create a menu bar and add it to the frame.

Code Snippet

```
JMenuBar mbrMenuBar;
JFrame frmMenu;
frmMenu = new JFrame("Menu bar");
// Creates a menu bar
mbrMenuBar = new JMenuBar();
. . .
// Adds the JMenuBar to the top level container JFrame
frmMenu.setMenuBar( mbrMenuBar);
```

JMenu [1-2]

- A `JMenu` is a class used to represent a menu on a menu bar.
- When you click `Jmenu` it drops down and displays one or more items.
- A menu can be created using any one of the following constructors of the `JMenu` class:
  - `JMenu()` - Constructs a `JMenu` without a label.
  - `JMenu(String label)` - Constructs a `JMenu` with the specified label.
- A JMenu is added to the menu bar using `add()` method of the `JMenuBar`.
- The `add()` method takes an object of `JMenu` class as an argument.



JMenu [2-2]

The `JMenu` class has some important methods as:

- `public void addSeparator()`
- `public void setMnemonic(int mnemonic)`

- Code Snippet shows how to add the 'File' menu and 'Edit' menu to the menu bar.

Code Snippet

```
JMenuBar mbrMenuBar;
JMenu mnuFile, mnuEdit;
// Creates a menu bar
mbrMenuBar = new JMenuBar();
. . .
// Creates a menu with the specified label
mnuFile = new JMenu("File");
mnuEdit = new JMenu("Edit");
// Adds the menu to the menu bar
mbrMenubar.add(mnuFile);
mbrMenubar.add(mnuEdit);
```

JMenuItem [1-2]

- A `JMenuItem` is a class which creates an item provided by a menu. You typically add a `JMenuItem` to a JMenu.
- A menu item has an action associated with it. When you click this item it generates an event and the action is performed.
- A menu item is created using any one of the following constructors of the `JMenuItem` class:
  - `JMenuItem(String label)`
  - `JMenuItem(String label, Icon icon)`
- The `JMenuItem` class has the following important methods:
  - **`public void setEnabled(boolean enable)`:** The method is used to enable or disable the menu item.
  - **`public void setMnemonic (int mnemonic)`:** The method is used to set the `mnemonic character`.
  - **`public void setAccelerator(KeyStroke keystroke)`:** The method is used to set the key combination which invokes the menu item's action listeners without navigating the menu hierarchy.

JMenuItem [2-2]

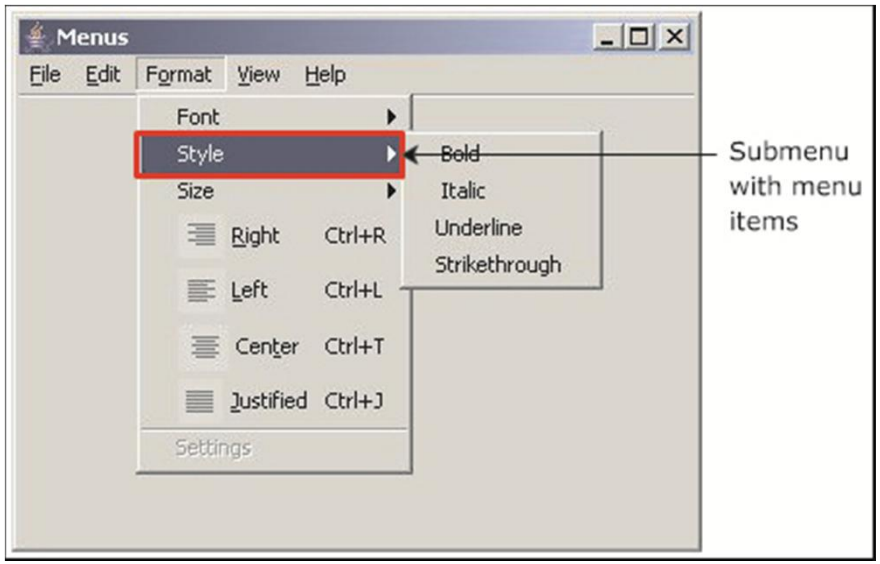
- Code Snippet shows how to set an accelerator key for the menu item 'Open'.

Code Snippet

```
//  An int specifying the numeric code for a keyboard key
int key = KeyEvent.VK_O;
// An integer specifying the modifier
int modifier = KeyEvent.ALT_MASK;
KeyStroke keystroke;
// Returns instance of a KeyStroke, given a numeric key code and
// a set of modifiers.
keystroke =  KeyStroke.getKeyStroke(key, modifier);
// Sets the key combination which invokes the menu item's action
// listeners
mniOpen.setAccelerator(keystroke);
```

Sub-Menus [1-2]

- A menu item is added to the menu using `add()` method of the `JMenu` class.
- The `add()` method takes an object of the `JMenuItem` class as a parameter.
- A `JMenuItem` can be a sub-menu. This sub-menu is called a `cascaded menu`. They have a triangular icon pointing rightwards.
- To create a sub-menu simply add `JMenu` in place of a `JMenuItem`.
- Adding a `JMenu` to another `JMenu` creates a cascaded menu.
- A sub-menu (`cascaded menu`) when clicked displays its set of items.



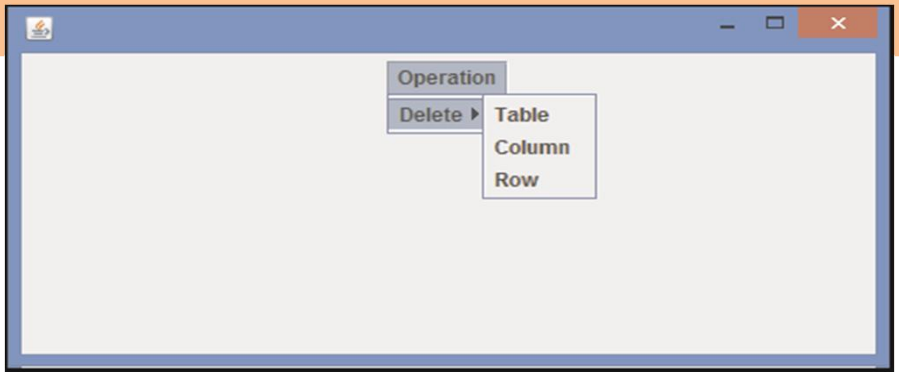
Sub-Menus [2-2]

- Code Snippet shows how to add the sub menu 'Delete' with the menu items 'Table', 'Columns', and 'Rows' to the 'Operation' menu.

Code Snippet

```
. . .
JMenu mnuTable, mnuDelete;
JMenuItem mniTable,mniColumns,mniRows;
. . .
// Adds the menu item to the "Delete" menu.
mnuDelete.add(mniTable);
mnuDelete.add(mniColumns);
mnuDelete.add(mniRows);
. . .
// Creates sub-menu(cascaded) "Delete" in the "Table" menu.
mnuTable.add(mnuDelete);
```

- Output:



Event Handling

- A menu item component listens using the `java.awt.event.ActionListener` interface.
- This interface has one method:
  - **`void actionPerformed(ActionEvent e):`** The code for the `action handler` of the clicked menu item is specified in the `actionPerformed()` method. The `JMenuItem` class has a method `addActionListener()`, which is used to register the `listener object`.
  - Code Snippet shows how to close the application when the 'Exit' menu item is clicked.

Code Snippet

```
// Event handling of the "Exit "
menuitem mniExit.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
          // Closes the application
          System.exit(0);
 } });
```

JCheckBoxMenuItem [1-3]

- A `JCheckBoxMenuItem` is similar to a `JMenuItem` with the additional feature of selecting and deselecting them.
- If `JCheckBoxMenuItem` is selected, it typically shows a checkmark, and if deselected the checkmark disappears.
- A `JCheckBoxMenuItem` like a `JMenuItem` can have a text label, an icon, or both.
- The `JCheckBoxMenuItem` can be created using any one of the constructors as mentioned:
  - `JCheckBoxMenuItem(String label)`
  - `JCheckBoxMenuItem(String label, boolean select)`

JCheckBoxMenuItem [2-3]

- Code Snippet shows how to create the check box menu item 'Standard', 'Formatting', and 'Auto Text' and add it to the 'Toolbars' menu item.

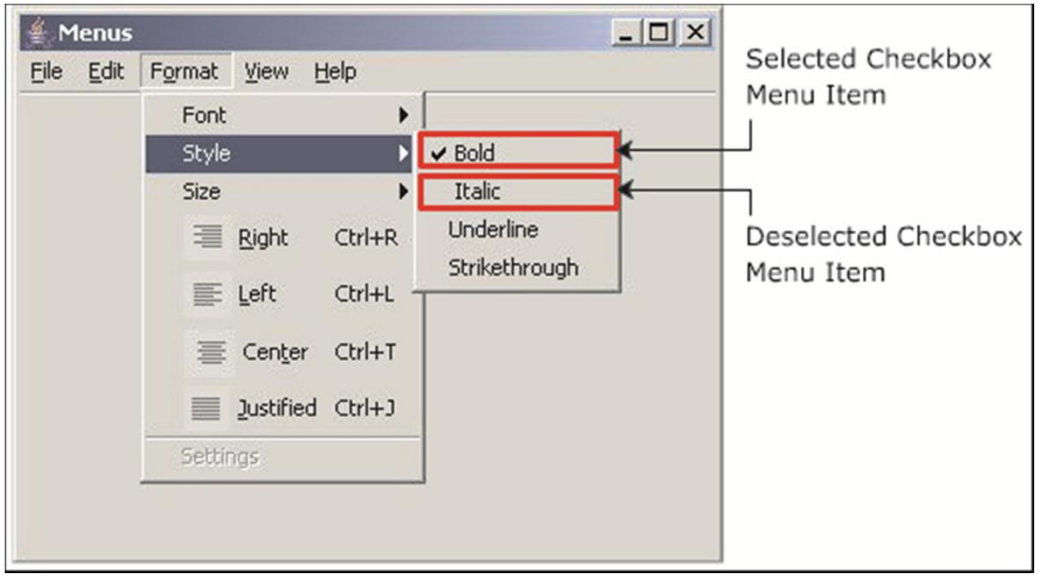
Code Snippet

```
   JMenuItem mniToolbars;
   JCheckBoxMenuItem mncStandard, mncFormatting, mncAutoText;
// Creates the checkbox menuitem with the label "Standard" and
// initially selected
   mncStandard = new JCheckBoxMenuItem("Standard", true);
// Creates the checkbox menuitem with the label "Formatting" and
// initially selected
    mncFormatting = new JCheckBoxMenuItem("Formatting", true);
// Creates the checkbox menuitem with the label "Auto Text" and
 // initially selected
    mncAutoText = new JCheckBoxMenuItem("Auto Text");
// Adds the checkbox menuitem to the "Toolbars"
   menu item mniToolbars.add(mncStandard);
   mniToolbars.add(mncFormatting);
   mniToolbars.add(mncAutoText);
```

JCheckBoxMenuItem [3-3]

The `JCheckBoxMenuItem` has the following methods:

- **`public boolean isSelected()`:** The method returns true if the `JCheckBoxMenuItem` is in selected state, else returns false. A `check box menu` item is said to be selected if the tick mark is visible on it.
- **`public void setSelected(boolean select)`:** The method allows you to programmatically set the state of `JCheckBoxMenuItem`.



JRadioButtonMenuItem [1-3]

- A `JRadioButtonMenuItem` is similar to a `JRadioButton` in appearance and functionality.
- The only difference is that a `JRadioButtonMenuItem` is a menu item and is added to a menu not container.
- A `JRadioButtonMenuItem`, like a `JRadioButton` can also be added to a `ButtonGroup` to be mutually exclusive.
- If one amongst the group is selected, all others are deselected automatically.
- A `JRadioButtonMenuItem` like a `JMenuItem` can have a text label, an icon, or both.
- The `JRadioButtonMenuItem` can be created using any one of the constructors:
  - `JRadioButtonMenuItem(String label)`
  - `JRadioButtonMenuItem(String label, boolean select)`

JRadioButtonMenuItem [2-3]

- Code Snippet shows how to create `JRadioButtonMenuItem` 'Normal', 'Web', 'Print' with the image icon and add it the 'View' menu

Code Snippet

```
JMenu mnuView;
JRadioButtonMenuItem mnrNormal, mnrWeb, mnrPrint;
ButtonGroup bgView;
// Creates the radio button menu item with the label "Normal
// Layout" and selected state.
    mnrNormal = new JRadioButtonMenuItem("Normal Layout",true);
// Creates the radio button menu item with the specified label
   mnrWeb = new JRadioButtonMenuItem("Web Layout");
   mnrPrint = new JRadioButtonMenuItem("Print Layout");
// Creating ButtonGroup instance
   bgView = new ButtonGroup();
// Adds the radio button menu item to the buttongroup
   bgView.add(mnrNormal);
   bgView.add(mnrWeb);
   bgView.add(mnrPrint);
// Adds the radio button menu item to the "View"
   mnuView.add(mnrNormal);
   mnuView.add(mnrWeb);
   mnuView.add(mnrPrint);
```

JRadioButtonMenuItem [3-3]

The `JRadioButtonMenuItem` has the following methods:

- **`public boolean isSelected()`:** The method returns true if the `JRadioButtonMenuItem` is in selected state, else returns false. A radio button menu item is selected if the dot is visible.
- **`public void setSelected(boolean select)`:** The method allows you to programmatically set the state of `JRadioButtonMenuItem`.

JPopupMenu

- A `JPopupMenu` is menu which pops up when you right-click a container.
- The pop up menu appears exactly at the location where you right-clicked.
- A `JPopupMenu` is more convenient than standard menus because it involves less mouse movements.
- When you click any where on the container, the menu pops up, allowing you to select an appropriate item.
- A `JPopupMenu` can have submenus.
- The `JPopupMenu` can be created using any one of constructors as mentioned:
  - `JPopupMenu()`
  - `JPopupMenu(String title)`
- The `JPopupMenu` has the following methods:
  - **`JMenuItem add(JMenuItem item)`:** The method adds a menu item at the end of the pop-up menu.
  - **`addSeparator()`:** The method adds a separator at the end of the pop-up menu.
  - **`setDefaultLightWeightPopupEnabled(boolean)`:** The method makes the pop-up a light weight menu if true is used as an argument. Lightweight pop-ups are more efficient than heavyweight menus.

Box Class [1-3]

- The `Box` class represents a lightweight container which uses a `BoxLayout` manager.
- The `Box` class provides several useful methods that are useful for containers which are governed by `BoxLayout` Manager.
- The `Box` class is used to create several types of invisible components such as `glue`, `strut`, and `rigid-areas`.
- This invisible components can be used to change the effect of the component layout.
- The `rigid-area` is used to specify fixed-size space between components in the layout.
- `Struts` have unlimited maximum height and width and hence, it is recommended to use rigid-area.
- The Box class has a static method `createVerticalGlue()` which creates a vertical glue between components.
- The Box class has a static method `createHorizontalGlue()` which creates a horizontal glue between components.

Box Class [2-3]

- Code Snippet shows how to use horizontal glue to provide a horizontal gap before the last menu.

Code Snippet

```
// Import the necessary classes
import javax.swing.*;
import java.beans.*;
// For property change listener interface and classes
import java.awt.*;
import java.io.File;
public class Demo2 {
        public static void main(String[] args){
                JFrame frm;
                JMenuBar mnbBar;
                JMenu mnuFile, mnuEdit, mnuView, mnuHelp;
                mnbBar = new JMenuBar();
                 // Creates a frame
                frm = new JFrame();
                frm.setSize(400,400);
                frm.add(mnbBar);
```

Box Class [3-3]

```
                mnuFile = new JMenu("File");
                mnuEdit = new JMenu("Edit");
                mnuView = new JMenu("View");
                mnuHelp = new JMenu("Help");
                 // Add the menus
                mnbBar.add(mnuFile);
                mnbBar.add(mnuEdit);
                mnbBar.add(mnuView);
                // Add a Horizontal Glue
                mnbBar.add(Box.createHorizontalGlue());
                // Add the last menu
                mnbBar.add(mnuHelp);
                frm.setJMenuBar(mnbBar);
                frm.setVisible(true);
        }
}
```

Introduction to JFileChooser

- A `JFileChooser` is a standard dialog box that is used to navigate the file system to choose a directory or a file.
- A `JFileChooser` is used to:

1. Accept a directory or file name to create a new one.
2. Perform operations like open and save, with different labels and buttons.
3. Provide previews and display appropriate icons.
4. Display the files with required criteria by applying filters.

Constructors and Methods of JFileChooser

- A `FileChooser` can be created using any one of the following constructors:
  - `JFileChooser()`
  - `JFileChooser(String directoryPath)`

- The `JFileChooser` has the following methods:
  - **`public int showDialog(Component parent, String approveButtonLabel)`:** The method displays a custom dialog box having an approve button with the specified label.
  - **`public int showSaveDialog(Component parent)`:** The method displays a standard 'Save File' dialog box.
  - **`public int showOpenDialog(Component parent)`:** The method displays a standard 'Open File' dialog box.

More Methods of JFileChooser [1-2]

- **`public File getSelectedFile()`:** The method returns an instance of the File representing the file selected from the dialog box.
- Code Snippet shows how to retrieve the file selected in the dialog box.

Code Snippet

```
JFileChooser fcrChooser;
// Creates a file chooser
fcrChooser = new JFileChooser();
// Opens the standard "Open File"
dialog box. int returnValue = fcrChooser.showOpenDialog(parent);
// Checks whether you have accepted the file
    if (returnValue == JFileChooser.APPROVE_OPTION) {
       // Retrieve the file name
       File file = fcrChooser.getSelectedFile();
     } else {
          System.out.println("File not selected.");
    }
```

More Methods of JFileChooser [2-2]

- **`public File getCurrentDirectory()`:** The method returns an instance of the File representing the current directory.
- Code Snippet shows how to retrieve the directory selected in the dialog box.

Code Snippet

```
JFileChooser fcrChooser;
File currentDir;
// Creates a file chooser
    fcrChooser = new JFileChooser();
// Retrieves the selected
   directory. currentDir = fcrChooser.getCurrentDirectory();
```

Creating Filter [1-2]

- The `JFileChooser` by default displays all the files in the current directory.
- However, at times it is required to display only those files which match certain criteria.
- To meet this requirement you have to apply a filter to the `JFileChooser` so that only those files which match the criteria are displayed.
- Swing provides the `javax.swing.filechooser.FileFilter` class to create filters.
- The `javax.swing.filechooser.FileFilter` is an abstract class with no default implementation.
- You have to sub-class this abstract class and provide your own implementation to specify the filter criteria.

Creating Filter [2-2]

- Code Snippet shows how to create a filter which will display only directories and image files with extension .gif, .jpg, and .png.

Code Snippet

```
public class ImageFilter extends FileFilter {
     // Method to decide the criteria
     public boolean accept(File file) {
          // If it is a directory it may contain image files so ignore it
             if (file.isDirectory())
             return true;
          // Retrieves the file name.
              String name = file.getName();
          // Only accept images with extension of .gif, .jpg and  .png
             if (name.endsWith(".gif") || name.endsWith(".jpg") ||
          name.endsWith(".png"))
             return true;
          else
             return false;
          }
          public String getDescription() {
              return "Image files";    }
}
```

Setting Filter

- Once you have an instance of the sub-class of `javax.swing.filechooser.FileFilter` class, you typically set the filter using the `setFileFilter()` method of `JFileChooser`.
- The method takes one argument, an object of any class which implements `javax.swing.filechooser.FileFilter` interface.

  Syntax: `public void setFileFilter(Filter filter)`
- Code Snippet how to set the filter.

Code Snippet

```
// Create a file chooser
   fcrChooser = new JFileChooser();
// Creates an object of class
   ImageFilter ImageFilter filter = new ImageFilter();
// Sets the filter
   fcrChooser.setFileFilter(filter);
```

Customizing the File View [1-4]

- To create a customized file view, create a subclass of the `FileView` class.
- The `javax.swing.filechooser.FileView` class is an abstract class with five methods which have to be implemented.
- The abstract methods are:
  - **`getName()`:** The `getName()` method returns the file-name.
  - **`getDescription()`:** The `getDescription()` method returns a human readable description of the file name.
  - **`getTypeDescription()`:** The `getTypeDescription()` method returns a human readable description of the file extension.
  - **`getIcon()`:** The `getIcon()` method returns an icon representing a file in the `JFileChooser`.
  - **`isTraversable()`:** The `isTraversable()` method decides whether a directory is `traversable`. Most implementations of this method should return null to indicate that the Look and Feel should decide it. It is normally used to prevent users from descending into a certain type of directory representing a compound document. The `isTraversable()` method should never return true for a non-directory.

Customizing the File View [2-4]

- Code Snippet shows how to subclass the `FileView` class and use its methods.

Code Snippet

```
import java.io.File;
import javax.swing.*;
import javax.swing.filechooser.*;
public class ImageFileView extends FileView {
         ImageIcon jpgIcon = new ImageIcon("jpgIcon.gif");
         ImageIcon gifIcon = new  ImageIcon("gifIcon.gif");
         ImageIcon tiffIcon = new  ImageIcon("tiffIcon.gif");
         ImageIcon pngIcon = new ImageIcon("pngIcon.png");
         public String getName(File f) {
                  return null; //let the Look and Feel(L&F) decide
         }
          public String getDescription(File f) {
                   return null; //let the L&F of FileView figure this out
         }
```

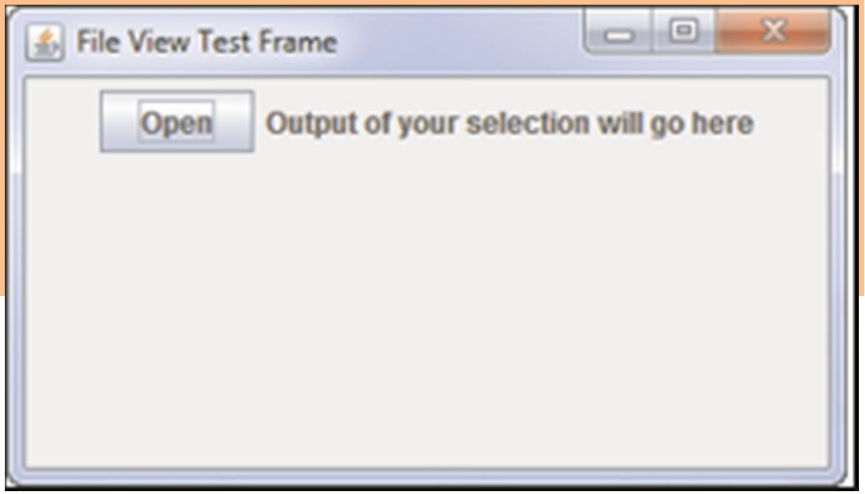
Customizing the File View [3-4]

```
public Boolean isTraversable(File f) {
   return null; //let the L&F FileView figure this out
}
public String getTypeDescription(File f) {
String extension;
String type = null;
int dotPosition = f.getName().indexOf(".");
extension = f.getName().substring(dotPosition);
if (extension != null) {
   if (extension.equals("jpeg") || extension.equals("jpg"))
   {
        type = "JPEG Image";
    } else if (extension.equals(".gif")){
       type = "GIF Image";
    } else if (extension.equals("tiff") || extension.equals("tif")) {
       type = "TIFF Image";
    } else if (extension.equals("png")){
       type = "PNG Image";
   }
}
   return type;
}
```

Customizing the File View [4-4]

```
    return type; }
public Icon getIcon(File f) {
   String extension;
   Icon icon = null;
   int dotPosition = f.getName().indexOf(".");
   extension = f.getName().substring(dotPosition);
    if (extension != null) {
     if (extension.equals("jpeg") || extension.equals("jpg"))
     {
          icon = jpgIcon;
     } else if (extension.equals("gif")) {
         icon = gifIcon;
     } else if (extension.equals("tiff") ||
                 extension.equals("tif")) {
                  icon = tiffIcon;
     } else if (extension.equals("png")) {
          icon = pngIcon;
     } }
        return icon;
} }
```

- Output:



JToolBar

- A `JToolBar` is similar to a menu bar and contains buttons with icons.
- These buttons generate events when clicked and can be associated with actions.
- A `JToolBar` serves the same purpose as a `JMenuBar`, except that a `JToolBar` takes up less space with an icon compared to a menu item on a menu bar.
- A `JToolBar` is `dockable`.
- You can click the `JToolBar` and drag it to another location or any side of its parent container.
- A `dockable` toolbar is also called as a floatable toolbar.
- A `JToolBar` can be created using any one of the following constructors:
  - `JToolBar()`
  - `JToolBar(int orientation)`
  - `JToolBar(String title)`
  - `JToolBar(String title, int orientation)`

Steps for Adding Buttons to JToolBar

1. Declare an object of `JButton` class.
2. Instantiate the object with an `ImageIcon`.
3. Use the method `setToolTipText()` of `JButton` class to set a tool tip.
4. Use the `add()` method of `JToolBar` and pass the button instance as an argument.

Adding Buttons to JToolBar

- Code Snippet shows how to add the 'Save' button with the image icon to the container and set its tool tip text.

Code Snippet

```
JToolBar tbrTools;
JButton btnSave;
. . .
. . .
// Creates a toolbar
   tbrTools = new JToolBar();
// Adds the toolbar to the frame in the north direction
    getContentPane().add(tbrTools,BorderLayout.NORTH);
. . .
// Creates a button with the icon
   btnSave = new JButton(new ImageIcon("Save.gif"));
// Sets the tool tip of the button "Save"
   btnSave.setToolTipText("Save");
// Adds the button "Save" to the toolbar
    tbrTools.add(btnSave);
 . . .
```

Methods of JToolBar

The important methods of the `JToolBar` are as follows:

- **`public Component add(Component component)`:** The method is used to add a button with an icon to the toolbar. This method is actually derived from Container class, the base class of `JToolBar`.
- **`public void addSeparator()`:** The method adds a separator between buttons of a toolbar.
- **`public void setBorderPainted(boolean paintBorder)`:** The method is used to paint a border around the toolbar.
- **`public void setFloatable(boolean floatable)`:** The method is used to make the toolbar dockable or non dockable.
- **`public void setOrientation(int orientation)`:** The method is used to change the orientation of the toolbar. The orientation can be either horizontal or vertical.

Making the Edge of the Toolbar Buttons Invisible

- The `JToolBar` has a method `setRollover()` which can be used to make the edges of the toolbar buttons invisible, except for the toolbar button which is under the mouse pointer.

Code Snippet

```
    JToolBar tbrTool;
// Creates a toolbar tbrTool = new JToolBar();
// Adds the toolbar to the frame in the north direction
    getContentPane.add(tbrTool,BorderLayout.NORTH);
// Make the edge of buttons invisible, except the one
//under mouse pointer
    tbrTool. setRollover(true);
```

Adding Separator [1-2]

The `JToolBar` has a method `addSeparator()` which can be used to add a separator between buttons of the toolbar.

Code Snippet

```
public class Demo {
        public static void main(String[] args){
                JFrame frm;
                JToolBar tbrTool;
                JButton btnOpen, btnSave;
                // Creates a frame
                frm=new JFrame();
                frm.setSize(400,400);
                 frm.setVisible(true);
                // Creates a toolbar
                tbrTool = new JToolBar();
                // Adds the toolbar to the frame in the north
direction
                frm.add(tbrTool, BorderLayout.NORTH);
```

Adding Separator [2-2]

```
        // Create the Open button
        btnOpen = new JButton(new ImageIcon("Open.gif"));
        // Add the open button
        tbrTool.add(btnOpen);
        // Add a separator before the next button on the
toolbar
        tbrTool.addSeparator();
        // Create the Save button
        btnSave = new JButton(new ImageIcon("Save.gif"));
        // Add the open button
        tbrTool.add(btnSave);
        }
}
```

Adding Non-Button Component to a Toolbar [1-2]

- The `add()` method of the `JToolBar` also allows to add a non-button component like a textfield to the toolbar.
- Code Snippet shows how to add a non-button component to a toolbar.

```
// Import the necessary classes
import javax.swing.*;
import java.beans.*; // For property change listener interface and classes
import java.awt.*; `
import java.io.File;
public class Demo1 {
        public static void main(String[] args){
                JToolBar tbrTool;
                JButton btnOpen;
                JTextField txfOpen;
                JFrame frm;
```

Adding Non-Button Component to a Toolbar [2-2]

```
// Creates a frame
   frm=new JFrame();
   frm.setSize(400,400);
   frm.setVisible(true);
// Creates a toolbar
    tbrTool = new JToolBar();
// Adds the toolbar to the frame in the north direction
        frm.add(tbrTool, BorderLayout.NORTH);
// Create the Open button
        btnOpen = new JButton(new ImageIcon("Open.gif"));
// Add the open button
   tbrTool.add(btnOpen);
// Add a non-button component to the toolbar
   txfOpen = new JTextField();
   tbrTool.add(txfOpen);
}
```

Summary

- A GUI based application has to perform various tasks; all these tasks cannot be accommodated on a single screen.
- A menu system provides a convenient means to handle these tasks.
- A JFileChooser is a standard dialog box to navigate the file system to choose a Directory or a File.
- A JFileChooser also allows you to input a directory or file name to create a new one.
- A JFileChooser dialog box is customized to perform operations like file open and save, with different labels and buttons.
- A JToolBar is similar to a menu bar and contains buttons with icons.
- These buttons generate events when clicked and can be associated with actions.
- A JToolBar serves the same purpose as a JMenuBar.
- A JToolBar takes up less space with an icon compared to a menu item on a menu bar.