SWING - JFILECHOOSER CLASS

Introduction

The class **JFileChooser** is a component which provides a simple mechanism for the user to choose a file.

Class declaration

Following is the declaration for **javax.swing.JFileChooser** class:

```
public class JFileChooser
   extends JComponent
      implements Accessible
```

Field

Following are the fields for **javax.swing.JFileChooser** class:

- **static String ACCEPT_ALL_FILE_FILTER_USED_CHANGED_PROPERTY** --Identifies whether a the AcceptAllFileFilter is used or not.
- **protected AccessibleContext accessibleContext**
- **static String ACCESSORY_CHANGED_PROPERTY** --Says that a different accessory component is in use *forexample, topreviewfiles*.
- **static String APPROVE_BUTTON_MNEMONIC_CHANGED_PROPERTY** --Identifies change in the mnemonic for the approve *yes, ok* button.
- **static String APPROVE_BUTTON_TEXT_CHANGED_PROPERTY** --Identifies change in the text on the approve *yes, ok* button.
- **static String APPROVE_BUTTON_TOOL_TIP_TEXT_CHANGED_PROPERTY** --Identifies change in the tooltip text for the approve *yes, ok* button.
- **static int APPROVE_OPTION** --Return value if approve *yes, ok* is chosen.
- **static String APPROVE_SELECTION** --Instruction to approve the current selection *sameaspressingyesorok*.
- **static int CANCEL_OPTION** --Return value if cancel is chosen.
- **static String CANCEL_SELECTION** --Instruction to cancel the current selection.
- **static String CHOOSABLE_FILE_FILTER_CHANGED_PROPERTY** --Identifies a change in the list of predefined file filters the user can choose from.
- **static String CONTROL_BUTTONS_ARE_SHOWN_CHANGED_PROPERTY** --Instruction to display the control buttons.
- **static int CUSTOM_DIALOG** --Type value indicating that the JFileChooser supports a developer-specified file operation.
- **static String DIALOG_TITLE_CHANGED_PROPERTY** --Identifies a change in the dialog title.
- **static String DIALOG_TYPE_CHANGED_PROPERTY** --Identifies a change in the type of files displayed *filesonly, directoriesonly, orbothfilesanddirectories*.
- **static int DIRECTORIES_ONLY** --Instruction to display only directories.
- **static String DIRECTORY_CHANGED_PROPERTY** --Identifies user's directory change.

- **static int ERROR_OPTION** --Return value if an error occured.
- **static String FILE_FILTER_CHANGED_PROPERTY** --User changed the kind of files to display.
- **static String FILE_HIDING_CHANGED_PROPERTY** --Identifies a change in the display-hidden-files property.
- **static String FILE_SELECTION_MODE_CHANGED_PROPERTY** --Identifies a change in the kind of selection *single, multiple, etc.* .
- **static String FILE_SYSTEM_VIEW_CHANGED_PROPERTY** --Says that a different object is being used to find available drives on the system.
- **static String FILE_VIEW_CHANGED_PROPERTY** --Says that a different object is being used to retrieve file information.
- **static int FILES_AND_DIRECTORIES** --Instruction to display both files and directories.
- **static int FILES_ONLY** --Instruction to display only files.
- **static String MULTI_SELECTION_ENABLED_CHANGED_PROPERTY** --Enables multiple-file selections.
- **static int OPEN_DIALOG** --Type value indicating that the JFileChooser supports an "Open" file operation.
- **static int SAVE_DIALOG** --Type value indicating that the JFileChooser supports a "Save" file operation.
- **static String SELECTED_FILE_CHANGED_PROPERTY** --Identifies change in user's single-file selection.
- **static String SELECTED_FILES_CHANGED_PROPERTY** --Identifies change in user's multiple-file selection.

Class constructors

S.N.	Constructor & Description
1	**JFileChooser** Constructs a JFileChooser pointing to the user's default directory.
2	**JFileChooser***FilecurrentDirectory* Constructs a JFileChooser using the given File as the path.
3	**JFileChooser***FilecurrentDirectory, FileSystemViewfsv* Constructs a JFileChooser using the given current directory and FileSystemView.
4	**JFileChooser***FileSystemViewfsv* Constructs a JFileChooser using the given FileSystemView.
5	**JFileChooser***StringcurrentDirectoryPath* Constructs a JFileChooser using the given path.
6	**JFileChooser***StringcurrentDirectoryPath, FileSystemViewfsv*

Constructs a JFileChooser using the given current directory path and FileSystemView.

Class methods

S.N.	Method & Description
1	**boolean accept*Filef*** Returns true if the file should be displayed.
2	**void addActionListener*ActionListenerl*** Adds an ActionListener to the file chooser.
3	**void addChoosableFileFilter*FileFilterfilter*** Adds a filter to the list of user choosable file filters.
4	**void approveSelection** Called by the UI when the user hits the Approve button *labeled* " *Open* " *or* " *Save* " , *bydefault*.
5	**void cancelSelection** Called by the UI when the user chooses the Cancel button.
6	**void changeToParentDirectory** Changes the directory to be set to the parent of the current directory.
7	**protected JDialog createDialog*Componentparent*** Creates and returns a new JDialog wrapping this centered on the parent in the parent's frame.
8	**void ensureFileIsVisible*Filef*** Makes sure that the specified file is viewable, and not hidden.
9	**protected void fireActionPerformed*Stringcommand*** Notifies all listeners that have registered interest for notification on this event type.
10	**FileFilter getAcceptAllFileFilter** Returns the AcceptAll file filter.
11	**AccessibleContext getAccessibleContext** Gets the AccessibleContext associated with this JFileChooser.
12	**JComponent getAccessory** Returns the accessory component.

13 **ActionListener[]getActionListeners**

Returns an array of all the action listeners registered on this file chooser.

14 **int getApproveButtonMnemonic**

Returns the approve button's mnemonic.

15 **String getApproveButtonText**

Returns the text used in the ApproveButton in the FileChooserUI.

16 **String getApproveButtonToolTipText**

Returns the tooltip text used in the ApproveButton.

17 **FileFilter[] getChoosableFileFilters**

Gets the list of user choosable file filters.

18 **boolean getControlButtonsAreShown**

Returns the value of the controlButtonsAreShown property.

19 **File getCurrentDirectory**

Returns the current directory.

20 **String getDescription***Filef*

Returns the file description.

21 **String getDialogTitle**

Gets the string that goes in the JFileChooser's titlebar.

22 **int getDialogType**

Returns the type of this dialog.

23 **boolean getDragEnabled**

Gets the value of the dragEnabled property.

24 **FileFilter getFileFilter**

Returns the currently selected file filter.

25 **int getFileSelectionMode**

Returns the current file-selection mode.

26 **FileSystemView getFileSystemView**

Returns the file system view.

27 **FileView getFileView**

Returns the current file view.

28 **Icon getIcon*Filef***

Returns the icon for this file or type of file, depending on the system.

29 **String getName*Filef***

Returns the filename.

30 **File getSelectedFile**

Returns the selected file.

31 **File[] getSelectedFiles**

Returns a list of selected files if the file chooser is set to allow multiple selection.

32 **String getTypeDescription*Filef***

Returns the file type.

33 **FileChooserUI getUI**

Gets the UI object which implements the L&F for this component.

34 **String getUIClassID**

Returns a string that specifies the name of the L&F class that renders this component.

35 **boolean isAcceptAllFileFilterUsed**

Returns whether the AcceptAll FileFilter is used.

36 **boolean isDirectorySelectionEnabled**

Convenience call that determines if directories are selectable based on the current file selection mode.

37 **boolean isFileHidingEnabled**

Returns true if hidden files are not shown in the file chooser; otherwise, returns false.

38 **boolean isFileSelectionEnabled**

Convenience call that determines if files are selectable based on the current file selection mode.

39 **boolean isMultiSelectionEnabled**

Returns true if multiple files can be selected.

40 **boolean isTraversable*Filef***

Returns true if the file *directory* can be visited.

41 **protected String paramString**

Returns a string representation of this JFileChooser.

42 **void removeActionListener***ActionListenerl*

Removes an ActionListener from the file chooser.

43 **boolean removeChoosableFileFilter***FileFilterf*

Removes a filter from the list of user choosable file filters.

44 **void rescanCurrentDirectory**

Tells the UI to rescan its files list from the current directory.

45 **void resetChoosableFileFilters**

Resets the choosable file filter list to its starting state.

46 **void setAcceptAllFileFilterUsed***booleanb*

Determines whether the AcceptAll FileFilter is used as an available choice in the choosable filter list.

47 **void setAccessory***JComponentnewAccessory*

Sets the accessory component.

48 **void setApproveButtonMnemonic***charmnemonic*

Sets the approve button's mnemonic using a character.

49 **void setApproveButtonMnemonic***intmnemonic*

Sets the approve button's mnemonic using a numeric keycode.

50 **void setApproveButtonText***StringapproveButtonText*

Sets the text used in the ApproveButton in the FileChooserUI.

51 **void setApproveButtonToolTipText***StringtoolTipText*

Sets the tooltip text used in the ApproveButton.

52 **void setControlButtonsAreShown***booleanb*

Sets the property that indicates whether the approve and cancel buttons are shown in the file chooser.

53 **void setCurrentDirectory***Filedir*

Sets the current directory.

54 **void setDialogTitle***StringdialogTitle*

Sets the string that goes in the JFileChooser window's title bar.

55 **void setDialogType***intdialogType*

Sets the type of this dialog.

56 **void setDragEnabled***booleanb*

Sets the dragEnabled property, which must be true to enable automatic drag handling *thefirstpartofdraganddrop* on this component.

57 **void setFileFilter***FileFilterfilter*

Sets the current file filter.

58 **void setFileHidingEnabled***booleanb*

Sets file hiding on or off.

59 **void setFileSelectionMode***intmode*

Sets the JFileChooser to allow the user to just select files, just select directories, or select both files and directories.

60 **void setFileSystemView***FileSystemViewfsv*

Sets the file system view that the JFileChooser uses for accessing and creating file system resources, such as finding the floppy drive and getting a list of root drives.

61 **void setFileView***FileViewfileView*

Sets the file view to used to retrieve UI information, such as the icon that represents a file or the type description of a file.

62 **void setMultiSelectionEnabled***booleanb*

Sets the file chooser to allow multiple file selections.

63 **void setSelectedFile***Filefile*

Sets the selected file.

64 **void setSelectedFiles***File***[]***selectedFiles*

Sets the list of selected files if the file chooser is set to allow multiple selection.

65 **protected void setup***FileSystemViewview*

Performs common constructor initialization and setup.

66 **int showDialog***Componentparent, StringapproveButtonText*

Pops a custom file chooser dialog with a custom approve button.

67 **int showOpenDialog***Componentparent*

Pops up an "Open File" file chooser dialog.

68 **int showSaveDialog*Componentparent***

Pops up a "Save File" file chooser dialog.

69 **void updateUI**

Resets the UI property to a value from the current look and feel.

Methods inherited

This class inherits methods from the following classes:

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

JFileChooser Example

Create the following java program using any editor of your choice in say **D:/ > SWING > com > tutorialspoint > gui >**

SwingControlDemo.java

```
package com.tutorialspoint.gui;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class SwingControlDemo {

   private JFrame mainFrame;
   private JLabel headerLabel;
   private JLabel statusLabel;
   private JPanel controlPanel;

   public SwingControlDemo(){
      prepareGUI();
   }

   public static void main(String[] args){
      SwingControlDemo  swingControlDemo = new SwingControlDemo();
      swingControlDemo.showFileChooserDemo();
   }

   private void prepareGUI(){
      mainFrame = new JFrame("Java Swing Examples");
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));
      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
      headerLabel = new JLabel("", JLabel.CENTER);
      statusLabel = new JLabel("",JLabel.CENTER);

      statusLabel.setSize(350,100);

      controlPanel = new JPanel();
      controlPanel.setLayout(new FlowLayout());
```

```
      mainFrame.add(headerLabel);
      mainFrame.add(controlPanel);
      mainFrame.add(statusLabel);
      mainFrame.setVisible(true);
   }

   private void showFileChooserDemo(){
      headerLabel.setText("Control in action: JFileChooser");

      final JFileChooser  fileDialog = new JFileChooser();
      JButton showFileDialogButton = new JButton("Open File");
      showFileDialogButton.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            int returnVal = fileDialog.showOpenDialog(mainFrame);
            if (returnVal == JFileChooser.APPROVE_OPTION) {
               java.io.File file = fileDialog.getSelectedFile();
               statusLabel.setText("File Selected :"
               + file.getName());
            }
            else{
               statusLabel.setText("Open command cancelled by user." );
            }
         }
      });
      controlPanel.add(showFileDialogButton);
      mainFrame.setVisible(true);
   }
}
```

Compile the program using command prompt. Go to **D:/ > SWING** and type the following command.

```
D:\SWING>javac com\tutorialspoint\gui\SwingControlDemo.java
```

If no error comes that means compilation is successful. Run the program using following command.

```
D:\SWING>java com.tutorialspoint.gui.SwingControlDemo
```

Verify the following output