SWING - JCOMPONENT CLASS

Introduction

The class **JComponent** is the base class for all Swing components except top-level containers. To use a component that inherits from JComponent, you must place the component in a containment hierarchy whose root is a top-level Swing container.

Class declaration

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

```
public abstract class JComponent
   extends Container
      implements Serializable
```

Field

Following are the fields for **java.awt.Component** class:

- **protected AccessibleContext accessibleContext** -- The AccessibleContext associated with this JComponent.
- **protected EventListenerList listenerList** -- A list of event listeners for this component.
- **static String TOOL_TIP_TEXT_KEY** -- The comment to display when the cursor is over the component, also known as a "value tip", "flyover help", or "flyover label".
- **protected ComponentUI ui** -- The look and feel delegate for this component.
- **static int UNDEFINED_CONDITION** -- Constant used by some of the APIs to mean that no condition is defined.
- **static int WHEN_ANCESTOR_OF_FOCUSED_COMPONENT** -- Constant used for registerKeyboardAction that means that the command should be invoked when the receiving component is an ancestor of the focused component or is itself the focused component.
- **static int WHEN_FOCUSED** -- Constant used for registerKeyboardAction that means that the command should be invoked when the component has the focus.
- **static int WHEN_IN_FOCUSED_WINDOW** -- Constant used for registerKeyboardAction that means that the command should be invoked when the receiving component is in the window that has the focus or is itself the focused component.

Class constructors

S.N.	Constructor & Description
1	**JComponent** Default JComponent constructor.

Class methods

S.N.	Method & Description
1	**void addAncestorListener***AncestorListenerlistener*

Registers listener so that it will receive AncestorEvents when it or any of its ancestors move or are made visible or invisible.

2 **void addNotify**

Notifies this component that it now has a parent component.

3 **void addVetoableChangeListener***VetoableChangeListenerlistener*

Adds a VetoableChangeListener to the listener list.

4 **void computeVisibleRect***RectanglevisibleRect*

Returns the Component's "visible rect rectangle" - the intersection of the visible rectangles for this component and all of its ancestors.

5 **boolean contains***intx, inty*

Gives the UI delegate an opportunity to define the precise shape of this component for the sake of mouse processing.

6 **JToolTip createToolTip**

Returns the instance of JToolTip that should be used to display the tooltip.

7 **void disable**

Deprecated.As of JDK version 1.1, replaced by java.awt.Component.setEnabled*boolean*.

8 **void enable**

Deprecated. As of JDK version 1.1, replaced by java.awt.Component.setEnabled*boolean*.

9 **void firePropertyChange***StringpropertyName, booleanoldValue, booleannewValue*

Support for reporting bound property changes for boolean properties.

10 **void firePropertyChange***StringpropertyName, charoldValue, charnewValue*

Reports a bound property change.

11 **void firePropertyChange***StringpropertyName, intoldValue, intnewValue*

Support for reporting bound property changes for integer properties.

12 **protected void fireVetoableChange***StringpropertyName, ObjectoldValue, ObjectnewValue*

Supports reporting constrained property changes.

13 **AccessibleContext getAccessibleContext**

Returns the AccessibleContext associated with this JComponent.

14 **ActionListener getActionForKeyStroke***KeyStrokeaKeyStroke*

Returns the object that will perform the action registered for a given keystroke.

15 **ActionMap getActionMap**

Returns the ActionMap used to determine what Action to fire for particular KeyStroke binding.

16 **float getAlignmentX**

Overrides Container.getAlignmentX to return the vertical alignment.

17 **float getAlignmentY**

Overrides Container.getAlignmentY to return the horizontal alignment.

18 **AncestorListener[]getAncestorListeners**

Returns an array of all the ancestor listeners registered on this component.

19 **boolean getAutoscrolls**

Gets the autoscrolls property.

20 **int getBaseline***intwidth, intheight*

Returns the baseline.

21 **Component.BaselineResizeBehavior getBaselineResizeBehavior**

Returns an enum indicating how the baseline of the component changes as the size changes.

22 **Border getBorder**

Returns the border of this component or null if no border is currently set.

23 **Rectangle getBounds***Rectanglerv*

Stores the bounds of this component into "return value" rv and returns rv.

24 **Object getClientProperty***Objectkey*

Returns the value of the property with the specified key.

25 **protected Graphics getComponentGraphics***Graphicsg*

Returns the graphics object used to paint this component.

26 **JPopupMenu getComponentPopupMenu**

Returns JPopupMenu that assigned for this component.

27 **int getConditionForKeyStroke***KeyStrokeaKeyStroke*

Returns the condition that determines whether a registered action occurs in response to the specified keystroke.

28 **int getDebugGraphicsOptions**

Returns the state of graphics debugging.

29 **static Locale getDefaultLocale**

Returns the default locale used to initialize each JComponent's locale property upon creation.

30 **FontMetrics getFontMetrics***Fontfont*

Gets the FontMetrics for the specified Font.

31 **Graphics getGraphics**

Returns this component's graphics context, which lets you draw on a component.

32 **int getHeight**

Returns the current height of this component.

33 **boolean getInheritsPopupMenu**

Returns true if the JPopupMenu should be inherited from the parent.

34 **InputMap getInputMap**

Returns the InputMap that is used when the component has focus.

35 **InputMap getInputMap***intcondition*

Returns the InputMap that is used during condition.

36 **InputVerifier getInputVerifier**

Returns the input verifier for this component.

37 **Insets getInsets**

If a border has been set on this component, returns the border's insets; otherwise calls super.getInsets.

38 **Insets getInsets***Insetsinsets*

Returns an Insets object containing this component's inset values.

39 **<T extends EventListener> T[] getListeners***Class < T > listenerType*

Returns an array of all the objects currently registered as FooListeners upon this JComponent.

40 **Point getLocation***Pointrv*

Stores the x,y origin of this component into "return value" rv and returns rv.

41 **Dimension getMaximumSize**

If the maximum size has been set to a non-null value just returns it.

42 **Dimension getMinimumSize**

If the minimum size has been set to a non-null value just returns it.

43 **Component getNextFocusableComponent**

Deprecated. As of 1.4, replaced by FocusTraversalPolicy.

44 **Point getPopupLocation***MouseEventevent*

Returns the preferred location to display the popup menu in this component's coordinate system.

45 **Dimension getPreferredSize**

If the preferredSize has been set to a non-null value just returns it.

46 **KeyStroke[] getRegisteredKeyStrokes**

Returns the KeyStrokes that will initiate registered actions.

47 **JRootPane getRootPane**

Returns the JRootPane ancestor for this component.

48 **Dimension getSize***Dimensionrv*

Stores the width/height of this component into "return value" rv and returns rv.

49 **Point getToolTipLocation***MouseEventevent*

Returns the tooltip location in this component's coordinate system.

50 **String getToolTipText**

Returns the tooltip string that has been set with setToolTipText.

51 **String getToolTipText***MouseEventevent*

Returns the string to be used as the tooltip for event.

52 **Container getTopLevelAncestor**

Returns the top-level ancestor of this component *eitherthecontainingWindoworApplet*, or null if this component has not been added to any container.

53 **TransferHandler getTransferHandler**

Gets the transferHandler property.

54 **String getUIClassID**

Returns the UIDefaults key used to look up the name of the swing.plaf.ComponentUI class that defines the look and feel for this component.

55 **boolean getVerifyInputWhenFocusTarget**

Returns the value that indicates whether the input verifier for the current focus owner will be called before this component requests focus.

56 **VetoableChangeListener[] getVetoableChangeListeners**

Returns an array of all the vetoable change listeners registered on this component.

57 **Rectangle getVisibleRect**

Returns the Component's "visible rectangle" - the intersection of this component's visible rectangle, new Rectangle0, 0, *getWidth*(, getHeight) , and all of its ancestors' visible rectangles.

58 **int getWidth**

Returns the current width of this component.

59 **int getX**

Returns the current x coordinate of the component's origin.

60 **int getY**

Returns the current y coordinate of the component's origin.

61 **void grabFocus**

Requests that this Component get the input focus, and that this Component's top-level ancestor become the focused Window.

62 **boolean isDoubleBuffered**

Returns whether this component should use a buffer to paint.

63 **static boolean isLightweightComponent***Componentc*

Returns true if this component is lightweight, that is, if it doesn't have a native window system peer.

64 **boolean isManagingFocus**

Deprecated.As of 1.4, replaced by Component.setFocusTraversalKeys*int, Set* and Container.setFocusCycleRoot*boolean*.

65 **boolean isOpaque**

Returns true if this component is completely opaque.

66 **boolean isOptimizedDrawingEnabled**

Returns true if this component tiles its children -- that is, if it can guarantee that the children will not overlap.

67 **boolean isPaintingForPrint**

Returns true if the current painting operation on this component is part of a print operation.

68 **boolean isPaintingTile**

Returns true if the component is currently painting a tile.

69 **boolean isRequestFocusEnabled**

Returns true if this JComponent should get focus; otherwise returns false.

70 **boolean isValidateRoot**

If this method returns true, revalidate calls by descendants of this component will cause the entire tree beginning with this root to be validated.

71 **void paint***Graphicsg*

Invoked by Swing to draw components.

72 **protected void paintBorder***Graphicsg*

Paints the component's border.

73 **protected void paintChildren***Graphicsg*

Paints this component's children.

74 **protected void paintComponent***Graphicsg*

Calls the UI delegate's paint method, if the UI delegate is non-null.

75 **void paintImmediately***intx, inty, intw, inth*

Paints the specified region in this component and all of its descendants that overlap the region, immediately.

76 **void paintImmediately***Rectangler*

Paints the specified region now.

77 **protected String paramString**

Returns a string representation of this JComponent.

78 **void print***Graphicsg*

Invoke this method to print the component to the specified Graphics.

79 **void printAll***Graphicsg*

Invoke this method to print the component.

80 **protected void printBorder***Graphicsg*

Prints the component's border.

81 **protected void printChildren***Graphicsg*

Prints this component's children.

82 **protected void printComponent***Graphicsg*

This is invoked during a printing operation.

82 **protected void processComponentKeyEvent***KeyEvente*

Processes any key events that the component itself recognizes.

84 **protected boolean processKeyBinding***KeyStrokeks, KeyEvente, intcondition, booleanpressed*

Invoked to process the key bindings for ks as the result of the KeyEvent e.

85 **protected void processKeyEvent***KeyEvente*

Overrides processKeyEvent to process events.

86 **protected void processMouseEvent***MouseEvente*

Processes mouse events occurring on this component by dispatching them to any registered MouseListener objects, refer to Component.processMouseEvent*MouseEvent* for a complete description of this method.

87 **protected void processMouseMotionEvent***MouseEvente*

Processes mouse motion events, such as MouseEvent.MOUSE_DRAGGED.

88 **void putClientProperty***Objectkey, Objectvalue*

Adds an arbitrary key/value "client property" to this component.

89 **void registerKeyboardAction***ActionListeneranAction, KeyStrokeaKeyStroke, intaCondition*

This method is now obsolete, please use a combination of getActionMap

and getInputMap

for similiar behavior.

90 **void registerKeyboardAction**
ActionListeneranAction, StringaCommand, KeyStrokeaKeyStroke, intaCondition

This method is now obsolete, please use a combination of getActionMap

and getInputMap

for similiar behavior.

91 **void removeAncestorListener***AncestorListenerlistener*

Unregisters listener so that it will no longer receive AncestorEvents.

92 **void removeNotify**

Notifies this component that it no longer has a parent component.

93 **void removeVetoableChangeListener***VetoableChangeListenerlistener*

Removes a VetoableChangeListener from the listener list.

94 **void repaint***longtm, intx, inty, intwidth, intheight*

Adds the specified region to the dirty region list if the component is showing.

95 **void repaint***Rectangler*

Adds the specified region to the dirty region list if the component is showing.

96 **boolean requestDefaultFocus**

Deprecated.As of 1.4, replaced by FocusTraversalPolicy.getDefaultComponent*Container* .requestFocus

97 **void requestFocus**

Requests that this Component gets the input focus.

98 **boolean requestFocus***booleantemporary*

Requests that this Component gets the input focus.

99 **boolean requestFocusInWindow**

Requests that this Component gets the input focus.

100 **protected boolean requestFocusInWindow***booleantemporary*

Requests that this Component gets the input focus.

101 **void resetKeyboardActions**

Unregisters all the bindings in the first tier InputMaps and ActionMap.

102 **void reshape***intx, inty, intw, inth*

Deprecated.As of JDK 5, replaced by Component.setBounds*int*, *int*, *int*, *int*.Moves and resizes this component.

103 **void revalidate**

Supports deferred automatic layout.

104 **void scrollRectToVisible***RectangleaRect*

Forwards the scrollRectToVisible

message to the JComponent's parent.

105 **void setActionMap***ActionMapam*

Sets the ActionMap to am.

106 **void setAlignmentX***floatalignmentX*

Sets the the vertical alignment.

107 **void setAlignmentY***floatalignmentY*

Sets the the horizontal alignment.

108 **void setAutoscrolls***booleanautoscrolls*

Sets the autoscrolls property.

109 **void setBackground***Colorbg*

Sets the background color of this component.

110 **void setBorder***Borderborder*

Sets the border of this component.

111 **void setComponentPopupMenu***JPopupMenupopup*

Sets the JPopupMenu for this JComponent.

112 **void setDebugGraphicsOptions***intdebugOptions*

Enables or disables diagnostic information about every graphics operation performed within the component or one of its children.

113 **static void setDefaultLocale***Localel*

Sets the default locale used to initialize each JComponent's locale property upon creation.

114 **void setDoubleBuffered***booleanaFlag*

Sets whether this component should use a buffer to paint.

115 **void setEnabled***booleanenabled*

Sets whether or not this component is enabled.

116 **void setFocusTraversalKeys***intid, Set < ?extendsAWTKeyStroke > keystrokes*

Sets the focus traversal keys for a given traversal operation for this Component.

117 **void setFont***Fontfont*

Sets the font for this component.

118 **void setForeground***Colorfg*

Sets the foreground color of this component.

119 **void setInheritsPopupMenu***booleanvalue*

Sets whether or not getComponentPopupMenu should delegate to the parent if this component does not have a JPopupMenu assigned to it.

120 **void setInputMap***intcondition, InputMapmap*

Sets the InputMap to use under the condition condition to map.

121 **void setInputVerifier***InputVerifierinputVerifier*

Sets the input verifier for this component.

122 **void setMaximumSize***DimensionmaximumSize*

Sets the maximum size of this component to a constant value.

123 **void setMinimumSize***DimensionminimumSize*

Sets the minimum size of this component to a constant value.

124 **void setNextFocusableComponent***ComponentaComponent*

Deprecated. As of 1.4, replaced by FocusTraversalPolicy

125 **void setOpaque***booleanisOpaque*

If true the component paints every pixel within its bounds.

126 **void setPreferredSize***DimensionpreferredSize*

Sets the preferred size of this component.

127 **void setRequestFocusEnabled***booleanrequestFocusEnabled*

Provides a hint as to whether or not this JComponent should get focus.

128 **void setToolTipText***Stringtext*

Registers the text to display in a tool tip.

129 **void setTransferHandler***TransferHandlernewHandler*

Sets the transferHandler property, which is null if the component does not support data transfer operations.

130 **protected void setUI***ComponentUInewUI*

Sets the look and feel delegate for this component.

131 **void setVerifyInputWhenFocusTarget***booleanverifyInputWhenFocusTarget*

Sets the value to indicate whether input verifier for the current focus owner will be called before this component requests focus.

132 **void setVisible***booleanaFlag*

Makes the component visible or invisible.

133 **void unregisterKeyboardAction***KeyStrokeaKeyStroke*

This method is now obsolete.

134 **void update*Graphicsg***

Calls paint.

135 **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:

- java.awt.Container
- java.awt.Component
- java.lang.Object