Class java.awt.Component

```
java.lang.Object
   |
   +----java.awt.Component
```

---

public class **Component**
extends Object
implements ImageObserver

A generic Abstract Window Toolkit component.

Version:
1.57, 10/17/95
Author:
Arthur van Hoff, Sami Shaio

Method Index

o **action**(Event, Object)
Called if an action occurs in the Component.
o **addNotify**()
Notifies the Component to create a peer.
o **bounds**()
Returns the current bounds of this component.
o **checkImage**(Image, ImageObserver)
Returns the status of the construction of a screen representation of the specified image.
o **checkImage**(Image, int, int, ImageObserver)
Returns the status of the construction of a scaled screen representation of the specified image.
o **createImage**(ImageProducer)
Creates an image from the specified image producer.
o **createImage**(int, int)
Creates an off−screen drawable Image to be used for double buffering.
o **deliverEvent**(Event)
Delivers an event to this component or one of its sub components.
o **disable**()
Disables a component.
o **enable**()

Enables a component.

- **enable**(boolean)
  Conditionally enables a component.
- **getBackground**()
  Gets the background color.
- **getColorModel**()
  Gets the ColorModel used to display the component on the output device.
- **getFont**()
  Gets the font of the component.
- **getFontMetrics**(Font)
  Gets the font metrics for this component.
- **getForeground**()
  Gets the foreground color.
- **getGraphics**()
  Gets a Graphics context for this component.
- **getParent**()
  Gets the parent of the component.
- **getPeer**()
  Gets the peer of the component.
- **getToolkit**()
  Gets the toolkit of the component.
- **gotFocus**(Event, Object)
  Indicates that this component has received the input focus.
- **handleEvent**(Event)
  Handles the event.
- **hide**()
  Hides the component.
- **imageUpdate**(Image, int, int, int, int, int)
  Repaints the component when the image has changed.
- **inside**(int, int)
  Checks whether a specified x,y location is "inside" this Component.
- **invalidate**()
  Invalidates a component.
- **isEnabled**()
  Checks if this Component is enabled.
- **isShowing**()
  Checks if this Component is showing on screen.
- **isValid**()
  Checks if this Component is valid.
- **isVisible**()
  Checks if this Component is visible.
- **keyDown**(Event, int)
  Called if a character is pressed.
- **keyUp**(Event, int)
  Called if a character is released.
- **layout**()
  Lays out the component.
- **list**()
  Prints a listing to a print stream.

- **list**(PrintStream)
  Prints a listing to the specified print out stream.
- **list**(PrintStream, int)
  Prints out a list, starting at the specified indention, to the specified print stream.
- **locate**(int, int)
  Returns the component or subcomponent that contains the x,y location.
- **location**()
  Returns the current location of this component.
- **lostFocus**(Event, Object)
  Indicates that this component has lost the input focus.
- **minimumSize**()
  Returns the minimum size of this component.
- **mouseDown**(Event, int, int)
  Called if the mouse is down.
- **mouseDrag**(Event, int, int)
  Called if the mouse is dragged (the mouse button is down).
- **mouseEnter**(Event, int, int)
  Called when the mouse enters the component.
- **mouseExit**(Event, int, int)
  Called when the mouse exits the component.
- **mouseMove**(Event, int, int)
  Called if the mouse moves (the mouse button is up).
- **mouseUp**(Event, int, int)
  Called if the mouse is up.
- **move**(int, int)
  Moves the Component to a new location.
- **nextFocus**()
  Moves the focus to the next component.
- **paint**(Graphics)
  Paints the component.
- **paintAll**(Graphics)
  Paints the component and its subcomponents.
- **paramString**()
  Returns the parameter String of this Component.
- **postEvent**(Event)
  Posts an event to this component.
- **preferredSize**()
  Returns the preferred size of this component.
- **prepareImage**(Image, ImageObserver)
  Prepares an image for rendering on this Component.
- **prepareImage**(Image, int, int, ImageObserver)
  Prepares an image for rendering on this Component at the specified width and height.
- **print**(Graphics)
  Prints this component.
- **printAll**(Graphics)
  Prints the component and its subcomponents.
- **removeNotify**()
  Notifies the Component to destroy the peer.
- **repaint**()

Repaints the component.

- **repaint**(long)
  Repaints the component.
- **repaint**(int, int, int, int)
  Repaints part of the component.
- **repaint**(long, int, int, int, int)
  Repaints part of the component.
- **requestFocus**()
  Requests the input focus.
- **reshape**(int, int, int, int)
  Reshapes the Component to the specified bounding box.
- **resize**(int, int)
  Resizes the Component to the specified width and height.
- **resize**(Dimension)
  Resizes the Component to the specified dimension.
- **setBackground**(Color)
  Sets the background color.
- **setFont**(Font)
  Sets the font of the component.
- **setForeground**(Color)
  Sets the foreground color.
- **show**()
  Shows the component.
- **show**(boolean)
  Conditionally shows the component.
- **size**()
  Returns the current size of this component.
- **toString**()
  Returns the String representation of this Component's values.
- **update**(Graphics)
  Updates the component.
- **validate**()
  Validates a component.

Methods

- **getParent**

```
public Container getParent()
```

  Gets the parent of the component.

- **getPeer**

```
public ComponentPeer getPeer()
```

  Gets the peer of the component.

o **getToolkit**

```
public Toolkit getToolkit()
```

Gets the toolkit of the component. This toolkit is used to create the peer for this component.

o **isValid**

```
public boolean isValid()
```

Checks if this Component is valid. Components are invalidated when the are first shown on the screen.
See Also:
validate, invalidate

o **isVisible**

```
public boolean isVisible()
```

Checks if this Component is visible. Components are initially visible (with the exception of top level components such as Frame).
See Also:
show, hide

o **isShowing**

```
public boolean isShowing()
```

Checks if this Component is showing on screen. This means that the component must be visible, and it must be in a container that is visible and showing.
See Also:
show, hide

o **isEnabled**

```
public boolean isEnabled()
```

Checks if this Component is enabled. Components are initially enabled.
See Also:
enable, disable

o **location**

```
public Point location()
```

Returns the current location of this component. The location will be in the parent's coordinate space.
See Also:
move

o **size**

```
public Dimension size()
```

Returns the current size of this component.
See Also:
resize

o **bounds**

```
public Rectangle bounds()
```

Returns the current bounds of this component.
See Also:
reshape

o **enable**

```
public synchronized void enable()
```

Enables a component.
See Also:
isEnabled, disable

o **enable**

```
public void enable(boolean cond)
```

Conditionally enables a component.
Parameters:
cond – if true, enables component; disables otherwise.
See Also:
enable, disable

o **disable**

```
public synchronized void disable()
```

Disables a component.
See Also:
isEnabled, enable

o **show**

```
public synchronized void show()
```

Shows the component.
See Also:
isVisible, hide

o **show**

```
public void show(boolean cond)
```

Conditionally shows the component.
Parameters:
cond – if true, it shows the component; hides otherwise.
See Also:
show, hide

o **hide**

```
public synchronized void hide()
```

Hides the component.
See Also:
isVisible, hide

o **getForeground**

```
public Color getForeground()
```

Gets the foreground color. If the component does not have a foreground color, the foreground color of its parent is returned.
See Also:
setForeground

o **setForeground**

```
public synchronized void setForeground(Color c)
```

Sets the foreground color.
Parameters:
c – the Color
See Also:
getForeground

o **getBackground**

```
public Color getBackground()
```

Gets the background color. If the component does not have a background color, the background color of its parent is returned.
See Also:
setBackground

o **setBackground**

```
public synchronized void setBackground(Color c)
```

Sets the background color.

Parameters:

c – the Color

See Also:

getBackground

o **getFont**

```
public Font getFont()
```

Gets the font of the component. If the component does not have a font, the font of its parent is returned.

See Also:

setFont

o **setFont**

```
public synchronized void setFont(Font f)
```

Sets the font of the component.

Parameters:

f – the font

See Also:

getFont

o **getColorModel**

```
public synchronized ColorModel getColorModel()
```

Gets the ColorModel used to display the component on the output device.

See Also:

ColorModel

o **move**

```
public void move(int x,
                 int y)
```

Moves the Component to a new location. The x and y coordinates are in the parent's coordinate space.

Parameters:

x – the x coordinate

y – the y coordinate

See Also:

location, reshape

o **resize**

```
public void resize(int width,
```

```
                    int height)
```

Resizes the Component to the specified width and height.

Parameters:

width – the width of the component
height – the height of the component

See Also:

size, reshape

o **resize**

```
public void resize(Dimension d)
```

Resizes the Component to the specified dimension.

Parameters:

d – the component dimension

See Also:

size, reshape

o **reshape**

```
public synchronized void reshape(int x,
                                 int y,
                                 int width,
                                 int height)
```

Reshapes the Component to the specified bounding box.

Parameters:

x – the x coordinate
y – the y coordinate
width – the width of the component
height – the height of the component

See Also:

bounds, move, resize

o **preferredSize**

```
public Dimension preferredSize()
```

Returns the preferred size of this component.

See Also:

minimumSize, LayoutManager

o **minimumSize**

```
public Dimension minimumSize()
```

Returns the minimum size of this component.

See Also:

preferredSize, LayoutManager

o **layout**

```
public void layout()
```

Lays out the component. This is usually called when the component is validated.
See Also:
validate, LayoutManager

o **validate**

```
public void validate()
```

Validates a component.
See Also:
invalidate, layout, LayoutManager

o **invalidate**

```
public void invalidate()
```

Invalidates a component.
See Also:
validate, layout, LayoutManager

o **getGraphics**

```
public Graphics getGraphics()
```

Gets a Graphics context for this component. This method will return null if the component is currently not on the screen.
See Also:
paint

o **getFontMetrics**

```
public FontMetrics getFontMetrics(Font font)
```

Gets the font metrics for this component. This will return null if the component is currently not on the screen.
Parameters:
font – the font
See Also:
getFont

o **paint**

```
public void paint(Graphics g)
```

Paints the component.

Parameters:

g – the specified Graphics window

See Also:

update

o **update**

```
public void update(Graphics g)
```

Updates the component. This method is called in response to a call to repaint. You can assume that the background is not cleared.

Parameters:

g – the specified Graphics window

See Also:

paint, repaint

o **paintAll**

```
public void paintAll(Graphics g)
```

Paints the component and its subcomponents.

Parameters:

g – the specified Graphics window

See Also:

paint

o **repaint**

```
public void repaint()
```

Repaints the component. This will result in a call to update as soon as possible.

See Also:

paint

o **repaint**

```
public void repaint(long tm)
```

Repaints the component. This will result in a call to update within *tm* milliseconds.

Parameters:

tm – maximum time in milliseconds before update

See Also:

paint

o **repaint**

```
public void repaint(int x,
                    int y,
```

```
                    int width,
                    int height)
```

Repaints part of the component. This will result in a call to update as soon as possible.

Parameters:

x – the x coordinate
y – the y coordinate
width – the width
height – the height

See Also:

repaint

o **repaint**

```
public void repaint(long tm,
                    int x,
                    int y,
                    int width,
                    int height)
```

Repaints part of the component. This will result in a call to update width *tm* millseconds.

Parameters:

tm – maximum time in milliseconds before update
x – the x coordinate
y – the y coordinate
width – the width
height – the height

See Also:

repaint

o **print**

```
public void print(Graphics g)
```

Prints this component. The default implementation of this method calls paint.

Parameters:

g – the specified Graphics window

See Also:

paint

o **printAll**

```
public void printAll(Graphics g)
```

Prints the component and its subcomponents.

Parameters:

g – the specified Graphics window

See Also:

print

o **imageUpdate**

```
public boolean imageUpdate(Image img,
                           int flags,
                           int x,
                           int y,
                           int w,
                           int h)
```

Repaints the component when the image has changed.
Returns:
true if image has changed; false otherwise.

o **createImage**

```
public Image createImage(ImageProducer producer)
```

Creates an image from the specified image producer.
Parameters:
producer – the image producer

o **createImage**

```
public Image createImage(int width,
                         int height)
```

Creates an off–screen drawable Image to be used for double buffering.
Parameters:
width – the specified width
height – the specified height

o **prepareImage**

```
public boolean prepareImage(Image image,
                            ImageObserver observer)
```

Prepares an image for rendering on this Component. The image data is downloaded asynchronously in another thread and the appropriate screen representation of the image is generated.
Parameters:
image – the Image to prepare a screen representation for
observer – the ImageObserver object to be notified as the image is being prepared
Returns:
true if the image has already been fully prepared
See Also:
ImageObserver

o **prepareImage**

```
public boolean prepareImage(Image image,
```

```
                        int width,
                        int height,
                        ImageObserver observer)
```

Prepares an image for rendering on this Component at the specified width and height. The image data is downloaded asynchronously in another thread and an appropriately scaled screen representation of the image is generated.

Parameters:

image – the Image to prepare a screen representation for
width – the width of the desired screen representation
height – the height of the desired screen representation
observer – the ImageObserver object to be notified as the image is being prepared

Returns:

true if the image has already been fully prepared

See Also:

ImageObserver

o **checkImage**

```
public int checkImage(Image image,
                      ImageObserver observer)
```

Returns the status of the construction of a screen representation of the specified image. This method does not cause the image to begin loading, use the prepareImage method to force the loading of an image.

Parameters:

image – the Image to check the status of
observer – the ImageObserver object to be notified as the image is being prepared

Returns:

the boolean OR of the ImageObserver flags for the data that is currently available

See Also:

ImageObserver, prepareImage

o **checkImage**

```
public int checkImage(Image image,
                      int width,
                      int height,
                      ImageObserver observer)
```

Returns the status of the construction of a scaled screen representation of the specified image. This method does not cause the image to begin loading, use the prepareImage method to force the loading of an image.

Parameters:

image – the Image to check the status of
width – the width of the scaled version to check the status of
height – the height of the scaled version to check the status of
observer – the ImageObserver object to be notified as the image is being prepared

Returns:

the boolean OR of the ImageObserver flags for the data that is currently available

See Also:

ImageObserver, prepareImage

o **inside**

```
public synchronized boolean inside(int x,
                                   int y)
```

Checks whether a specified x,y location is "inside" this Component. By default, x and y are inside an Component if they fall within the bounding box of that Component.

Parameters:

x – the x coordinate

y – the y coordinate

See Also:

locate

o **locate**

```
public Component locate(int x,
                        int y)
```

Returns the component or subcomponent that contains the x,y location.

Parameters:

x – the x coordinate

y – the y coordinate

See Also:

inside

o **deliverEvent**

```
public void deliverEvent(Event e)
```

Delivers an event to this component or one of its sub components.

Parameters:

e – the event

See Also:

handleEvent, postEvent

o **postEvent**

```
public void postEvent(Event e)
```

Posts an event to this component. This will result in a call to handleEvent. If handleEvent returns false the event is passed on to the parent of this component.

Parameters:

e – the event

See Also:

handleEvent, deliverEvent

o **handleEvent**

```
public boolean handleEvent(Event evt)
```

Handles the event. Returns true if the event is handled and should not be passed to the parent of this component. The default event handler calls some helper methods to make life easier on the programmer.

Parameters:

evt – the event

See Also:

mouseEnter, mouseExit, mouseMove, mouseDown, mouseDrag, mouseUp, keyDown, action

o **mouseDown**

```
public boolean mouseDown(Event evt,
                         int x,
                         int y)
```

Called if the mouse is down.

Parameters:

evt – the event

x – the x coordinate

y – the y coordinate

See Also:

handleEvent

o **mouseDrag**

```
public boolean mouseDrag(Event evt,
                         int x,
                         int y)
```

Called if the mouse is dragged (the mouse button is down).

Parameters:

evt – the event

x – the x coordinate

y – the y coordinate

See Also:

handleEvent

o **mouseUp**

```
public boolean mouseUp(Event evt,
                       int x,
                       int y)
```

Called if the mouse is up.

Parameters:

evt – the event

x – the x coordinate
y – the y coordinate

See Also:

handleEvent

o **mouseMove**

```
public boolean mouseMove(Event evt,
                         int x,
                         int y)
```

Called if the mouse moves (the mouse button is up).

Parameters:

evt – the event
x – the x coordinate
y – the y coordinate

See Also:

handleEvent

o **mouseEnter**

```
public boolean mouseEnter(Event evt,
                          int x,
                          int y)
```

Called when the mouse enters the component.

Parameters:

evt – the event
x – the x coordinate
y – the y coordinate

See Also:

handleEvent

o **mouseExit**

```
public boolean mouseExit(Event evt,
                         int x,
                         int y)
```

Called when the mouse exits the component.

Parameters:

evt – the event
x – the x coordinate
y – the y coordinate

See Also:

handleEvent

o **keyDown**

```
public boolean keyDown(Event evt,
```

```
                        int key)
```

Called if a character is pressed.
Parameters:
evt – the event
key – the key that's pressed
See Also:
handleEvent

o **keyUp**

```
public boolean keyUp(Event evt,
                     int key)
```

Called if a character is released.
Parameters:
evt – the event
key – the key that's released
See Also:
handleEvent

o **action**

```
public boolean action(Event evt,
                      Object what)
```

Called if an action occurs in the Component.
Parameters:
evt – the event
what – the action that's occuring
See Also:
handleEvent

o **addNotify**

```
public void addNotify()
```

Notifies the Component to create a peer.
See Also:
getPeer, removeNotify

o **removeNotify**

```
public synchronized void removeNotify()
```

Notifies the Component to destroy the peer.
See Also:
getPeer, addNotify

o **gotFocus**

```
public boolean gotFocus(Event evt,
                        Object what)
```

Indicates that this component has received the input focus.
See Also:
requestFocus, lostFocus

o **lostFocus**

```
public boolean lostFocus(Event evt,
                         Object what)
```

Indicates that this component has lost the input focus.
See Also:
requestFocus, gotFocus

o **requestFocus**

```
public void requestFocus()
```

Requests the input focus. The gotFocus() method will be called if this method is successful.
See Also:
gotFocus

o **nextFocus**

```
public void nextFocus()
```

Moves the focus to the next component.
See Also:
requestFocus, gotFocus

o **paramString**

```
protected String paramString()
```

Returns the parameter String of this Component.

o **toString**

```
public String toString()
```

Returns the String representation of this Component's values.
Overrides:
toString in class Object

o **list**

```
public void list()
```

Prints a listing to a print stream.

o **list**

```
public void list(PrintStream out)
```

Prints a listing to the specified print out stream.

Parameters:

out – the Stream name

o **list**

```
public void list(PrintStream out,
                 int indent)
```

Prints out a list, starting at the specified indention, to the specified print stream.

Parameters:

out – the Stream name

indent – the start of the list