**Non-container Components**

**Buttons**

Constructors:

public Button ()

This constructor creates an empty Button. You can set the label later with setLabel().

public Button (String label)

This constructor creates a Button whose initial text is label.

Button Labels

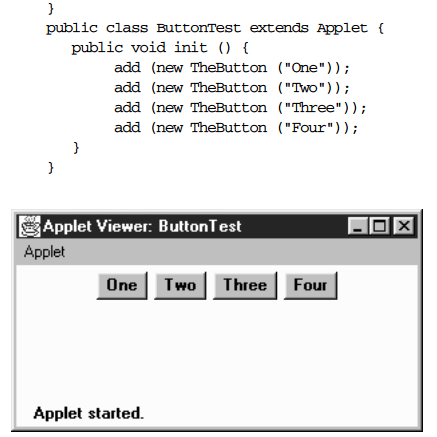
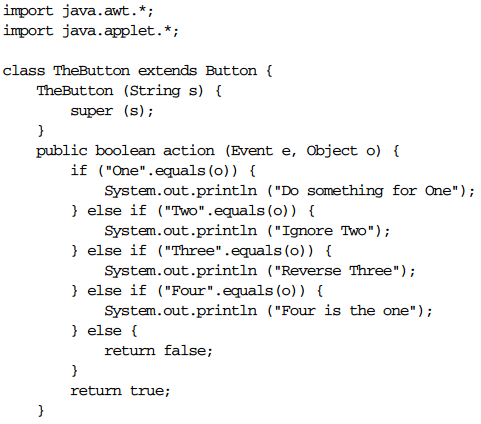
public String getLabel ()

The getLabel() method retrieves the current text of the label on the Button and returns it as a String.

public synchronized void setLabel (String label)

The setLabel() method changes the text of the label on the Button to label. If the new text is a different size from the old, it is necessary to revalidate the screen to ensure that the button size is correct.

Example Code and Output:



**Canvas**

## Class declaration

Following is the declaration for **java.awt.Canvas** class:

public class Canvas

extends Component

implements Accessible

## Class constructors

|  |  |
| --- | --- |
| **S.N.** | **Constructor & Description** |
| 1 | **Canvas()** Constructs a new Canvas. |
| 2 | **Canvas(GraphicsConfiguration config)** Constructs a new Canvas given a GraphicsConfiguration object. |

## Class methods

|  |  |
| --- | --- |
| **S.N.** | **Method & Description** |
| 1 | **void addNotify()** Creates the peer of the canvas. |
| 2 | **void createBufferStrategy(int numBuffers)** Creates a new strategy for multi-buffering on this component. |
| 3 | **void createBufferStrategy(int numBuffers, BufferCapabilities caps)** Creates a new strategy for multi-buffering on this component with the required buffer capabilities. |
| 4 | **AccessibleContext getAccessibleContext()** Gets the AccessibleContext associated with this Canvas. |
| 5 | **BufferStrategy getBufferStrategy()** Returns the BufferStrategy used by this component. |
| 6 | **void paint(Graphics g)** Paints this canvas. |
| 7 | **void pdate(Graphics g)** Updates this canvas. |

## Methods inherited

This class inherits methods from the following classes:

* java.awt.Component
* java.lang.Object

## Canvas Example

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

*AwtControlDemo.java*

package com.tutorialspoint.gui;

import java.awt.\*;

import java.awt.event.\*;

public class AwtControlDemo {

private Frame mainFrame;

private Label headerLabel;

private Label statusLabel;

private Panel controlPanel;

public AwtControlDemo(){

prepareGUI();

}

public static void main(String[] args){

AwtControlDemo awtControlDemo = new AwtControlDemo();

awtControlDemo.showCanvasDemo();

}

private void prepareGUI(){

mainFrame = new Frame("Java AWT 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 Label();

headerLabel.setAlignment(Label.CENTER);

statusLabel = new Label();

statusLabel.setAlignment(Label.CENTER);

statusLabel.setSize(350,100);

controlPanel = new Panel();

controlPanel.setLayout(new FlowLayout());

mainFrame.add(headerLabel);

mainFrame.add(controlPanel);

mainFrame.add(statusLabel);

mainFrame.setVisible(true);

}

private void showCanvasDemo(){

headerLabel.setText("Control in action: Canvas");

controlPanel.add(new MyCanvas());

mainFrame.setVisible(true);

}

class MyCanvas extends Canvas {

public MyCanvas () {

setBackground (Color.GRAY);

setSize(300, 300);

}

public void paint (Graphics g) {

Graphics2D g2;

g2 = (Graphics2D) g;

g2.drawString ("It is a custom canvas area", 70, 70);

}

}

}

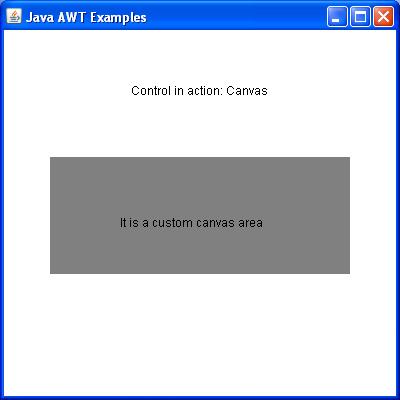
Compile the program using command prompt. Go to **D:/ > AWT**and type the following command.

D:\AWT>javac com\tutorialspoint\gui\AwtControlDemo.java

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

D:\AWT>java com.tutorialspoint.gui.AwtControlDemo

Verify the following output



**Checkbox**

Constructors

public Checkbox ()

This constructor for Checkbox creates a new instance with no label or grouping. The initial state of the item is false. A checkbox doesn’t necessarily need a label; however, a checkbox without a label might be confusing, unless it is being used as a column in a table or a spreadsheet.

public Checkbox (String label)

The second constructor creates a new Checkbox with a label of label and no grouping. The initial state of the item is false. If you want a simple yes/no choice and plan to make no the default, use this constructor. If the Checkbox will be in a group or you want its initial value to be true, use the next constructor.

public Checkbox (String label, boolean state) ★

This constructor allows you to specify the Checkbox’s initial state. With it you create a Checkbox with a label of label and an initial state of state.

public Checkbox (String label, boolean state, CheckboxGroup group) ★

public Checkbox (String label, CheckboxGroup group, boolean state)

The final constructor for Checkbox is the most flexible. With this constructor you create a Checkbox with a label of label, a CheckboxGroup of group, and an initial state of state. If group is null, the Checkbox is independent.

In Java 1.0, you created an independent Checkbox with an initial value of true by using null as the group:

Checkbox cb = new Checkbox ("Help", null, true)

The shape of the Checkbox reflects whether it’s in a CheckboxGroup or independent. On Microsoft Windows, grouped checkboxes are represented as circles. On a UNIX system, they are diamonds. On both systems, independent checkboxes are squares.

Example code and output:

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class JCheckBoxDemo extends JPanel {

//Four accessory choices provide for 16 different combinations

JCheckBox jcbChin;

JCheckBox jcbGlasses;

JCheckBox jcbHair;

JCheckBox jcbTeeth;

/\* The image for each combination is contained in a

separate image file whose name indicates the accessories.

The filenames are "geek-XXXX.gif" where XXXX can be one

\* of the following 16 choices.

\*/

StringBuffer choices;

JLabel jlbPicture;

CheckBoxListener myListener = null;

public JCheckBoxDemo() {

// Add an item listener for each of the check boxes.

// This is the listener class which contains business logic

myListener = new CheckBoxListener();

// Create check boxes with default selection true

jcbChin = new JCheckBox("Chin");

jcbChin.setMnemonic(KeyEvent.VK\_C);

//Alt+C Checks/Unchecks the check Box

jcbChin.setSelected(true);

jcbChin.addItemListener(myListener);

jcbGlasses = new JCheckBox("Glasses");

jcbGlasses.setMnemonic(KeyEvent.VK\_G);

//Alt+G Checks/Unchecks the check Box

jcbGlasses.setSelected(true);

jcbGlasses.addItemListener(myListener);

jcbHair = new JCheckBox("Hair");

jcbHair.setMnemonic(KeyEvent.VK\_H);

//Alt+H Checks/Unchecks the check Box

jcbHair.setSelected(true);

jcbHair.addItemListener(myListener);

jcbTeeth = new JCheckBox("Teeth");

jcbTeeth.setMnemonic(KeyEvent.VK\_T);

//Alt+T Checks/Unchecks the check Box

jcbTeeth.setSelected(true);

jcbTeeth.addItemListener(myListener);

// Indicates what's on the geek.

choices = new StringBuffer("cght");//Default Image has all the parts.

// Set up the picture label

jlbPicture = new JLabel(new ImageIcon("geek-" +

choices.toString().trim() + ".gif"));

jlbPicture.setToolTipText(choices.toString().trim());

// Put the check boxes in a column in a panel

JPanel jplCheckBox = new JPanel();

jplCheckBox.setLayout(new GridLayout(0, 1)); //0 rows, 1 Column

jplCheckBox.add(jcbChin);

jplCheckBox.add(jcbGlasses);

jplCheckBox.add(jcbHair);

jplCheckBox.add(jcbTeeth);

setLayout(new BorderLayout());

add(jplCheckBox, BorderLayout.WEST);

add(jlbPicture, BorderLayout.CENTER);

setBorder(BorderFactory.createEmptyBorder(20,20,20,20));

}

//Listens to the check boxes events

class CheckBoxListener implements ItemListener {

public void itemStateChanged(ItemEvent e) {

int index = 0;

char c = '-';

Object source = e.getSource();

if (source == jcbChin) {

index = 0;

c = 'c';

} else if (source == jcbGlasses) {

index = 1;

c = 'g';

} else if (source == jcbHair) {

index = 2;

c = 'h';

} else if (source == jcbTeeth) {

index = 3;

c = 't';

}

if (e.getStateChange() == ItemEvent.DESELECTED)

c = '-';

choices.setCharAt(index, c);

jlbPicture.setIcon(new ImageIcon("geek-"

+ choices.toString().trim() + ".gif"));

jlbPicture.setToolTipText(choices.toString());

}

}

public static void main(String s[]) {

JFrame frame = new JFrame("JCheckBox Usage Demo");

frame.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

System.exit(0);

}

});

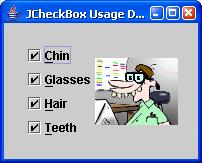
frame.setContentPane(new JCheckBoxDemo());

frame.pack();

frame.setVisible(true);

}

}



**Choice**

Class declaration

Following is the declaration for **java.awt.Choice** class:

public class Choice

extends Component

implements ItemSelectable, Accessible

Class constructors

|  |  |
| --- | --- |
| **S.N.** | **Constructor & Description** |
| 1 | **Choice() ()** Creates a new choice menu. |

Class methods

|  |  |
| --- | --- |
| **S.N.** | **Method & Description** |
| 1 | **void add(String item)** Adds an item to this Choice menu. |
| 2 | **void addItem(String item)** Obsolete as of Java 2 platform v1.1. |
| 3 | **void addItemListener(ItemListener l)** Adds the specified item listener to receive item events from this Choice menu. |
| 4 | **void addNotify()** Creates the Choice's peer. |
| 5 | **int countItems()** Deprecated. As of JDK version 1.1, replaced by getItemCount(). |
| 6 | **AccessibleContext getAccessibleContext()** Gets the AccessibleContext associated with this Choice. |
| 7 | **String getItem(int index)** Gets the string at the specified index in this Choice menu. |
| 8 | **int getItemCount()** Returns the number of items in this Choice menu. |
| 9 | **ItemListener[] getItemListeners()** Returns an array of all the item listeners registered on this choice. |
| 10 | **<T extends EventListener> T[] getListeners(Class<T> listenerType)** Returns an array of all the objects currently registered as FooListeners upon this Choice. |
| 11 | **int getSelectedIndex()** Returns the index of the currently selected item. |
| 12 | **String getSelectedItem()** Gets a representation of the current choice as a string. |
| 13 | **Object[] getSelectedObjects()** Returns an array (length 1) containing the currently selected item. |
| 14 | **void insert(String item, int index)** Inserts the item into this choice at the specified position. |
| 15 | **protected String paramString()** Returns a string representing the state of this Choice menu. |
| 16 | **protected void processEvent(AWTEvent e)** Processes events on this choice. |
| 17 | **protected void processItemEvent(ItemEvent e)** Processes item events occurring on this Choice menu by dispatching them to any registered ItemListener objects. |
| 18 | **void remove(int position)** Removes an item from the choice menu at the specified position. |
| 19 | **void remove(String item)** Removes the first occurrence of item from the Choice menu. |
| 20 | **void removeAll()** Removes all items from the choice menu. |
| 21 | **void removeItemListener(ItemListener l)** Removes the specified item listener so that it no longer receives item events from this Choice menu. |
| 22 | **void select(int pos)** Sets the selected item in this Choice menu to be the item at the specified position. |
| 23 | **void select(String str)** Sets the selected item in this Choice menu to be the item whose name is equal to the specified string. |

Methods inherited

This class inherits methods from the following classes:

* java.awt.Component
* java.lang.Object

Choice Example

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

*AwtControlDemo.java*

package com.tutorialspoint.gui;

import java.awt.\*;

import java.awt.event.\*;

public class AwtControlDemo {

private Frame mainFrame;

private Label headerLabel;

private Label statusLabel;

private Panel controlPanel;

public AwtControlDemo(){

prepareGUI();

}

public static void main(String[] args){

AwtControlDemo awtControlDemo = new AwtControlDemo();

awtControlDemo.showChoiceDemo();

}

private void prepareGUI(){

mainFrame = new Frame("Java AWT 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 Label();

headerLabel.setAlignment(Label.CENTER);

statusLabel = new Label();

statusLabel.setAlignment(Label.CENTER);

statusLabel.setSize(350,100);

controlPanel = new Panel();

controlPanel.setLayout(new FlowLayout());

mainFrame.add(headerLabel);

mainFrame.add(controlPanel);

mainFrame.add(statusLabel);

mainFrame.setVisible(true);

}

private void showChoiceDemo(){

headerLabel.setText("Control in action: Choice");

final Choice fruitChoice = new Choice();

fruitChoice.add("Apple");

fruitChoice.add("Grapes");

fruitChoice.add("Mango");

fruitChoice.add("Peer");

Button showButton = new Button("Show");

showButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String data = "Fruit Selected: "

+ fruitChoice.getItem(fruitChoice.getSelectedIndex());

statusLabel.setText(data);

}

});

controlPanel.add(fruitChoice);

controlPanel.add(showButton);

mainFrame.setVisible(true);

}

}

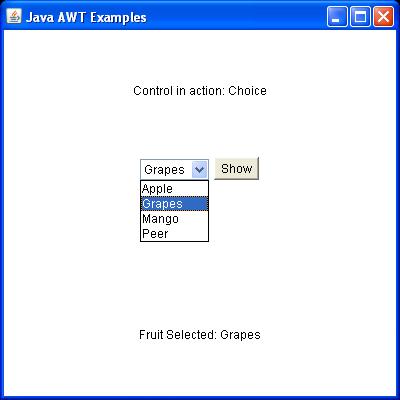
Compile the program using command prompt. Go to **D:/ > AWT**and type the following command.

D:\AWT>javac com\tutorialspoint\gui\AwtControlDemo.java

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

D:\AWT>java com.tutorialspoint.gui.AwtControlDemo

Verify the following output



**Label**

|  |
| --- |
| import java.awt.GridLayout;  import java.awt.event.WindowAdapter;  import java.awt.event.WindowEvent;  import javax.swing.JLabel;  import javax.swing.JPanel;  import javax.swing.JFrame;  import javax.swing.ImageIcon;  public class JlabelDemo extends JPanel {  JLabel jlbLabel1, jlbLabel2, jlbLabel3;  public JlabelDemo() {  ImageIcon icon = new ImageIcon("java-swing-tutorial.JPG",  "My Website");  // Creating an Icon  setLayout(new GridLayout(3, 1));  // 3 rows, 1 column Panel having Grid Layout  jlbLabel1 = new JLabel("Image with Text", icon, JLabel.CENTER);  // We can position of the text, relative to the icon:  jlbLabel1.setVerticalTextPosition(JLabel.BOTTOM);  jlbLabel1.setHorizontalTextPosition(JLabel.CENTER);  jlbLabel2 = new JLabel("Text Only Label");  jlbLabel3 = new JLabel(icon); // Label of Icon Only  // Add labels to the Panel  add(jlbLabel1);  add(jlbLabel2);  add(jlbLabel3);  }  public static void main(String[] args) {  JFrame frame = new JFrame("jLabel Usage Demo");  frame.addWindowListener(new WindowAdapter() {  // Shows code to Add Window Listener  public void windowClosing(WindowEvent e) {  System.exit(0);  }  });  frame.setContentPane(new JlabelDemo());  frame.pack();  frame.setVisible(true);  }  } |

**Output**



[**Download**](http://www.javabeginner.com/jlabel.zip)**jLabel Source Code**

### Java JLabel Hierarchy

javax.swing  
**Class JLabel**  
java.lang.Object  
java.awt.Component  
java.awt.Container  
javax.swing.JComponent  
javax.swing.JLabel

**All Implemented Interfaces:**  
Accessible, ImageObserver, MenuContainer, Serializable, SwingConstants  
**Direct Known Subclasses:**  
BasicComboBoxRenderer, DefaultListCellRenderer, DefaultTableCellRenderer, DefaultTreeCellRenderer

## JLabel Constructor

**JLabel()**  
Creates a JLabel instance with no image and with an empty string for the title.

**JLabel(Icon image)**  
Creates a JLabel instance with the specified image.

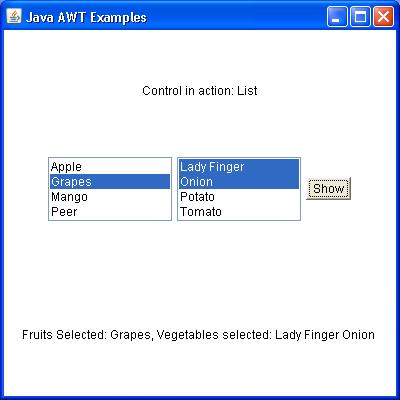
**JLabel(Icon image, int horizontalAlignment)**  
Creates a JLabel instance with the specified image and horizontal alignment.

**JLabel(String text)**  
Creates a JLabel instance with the specified text.

**JLabel(String text, Icon icon, int horizontalAlignment)**  
Creates a JLabel instance with the specified text, image, and horizontal alignment.

**JLabel(String text, int horizontalAlignment)**  
Creates a JLabel instance with the specified text and horizontal alignment.

**List**



**Scrollbar**

## Class declaration

Following is the declaration for **java.awt.Scrollbar** class:

public class Scrollbar

extends Component

implements Adjustable, Accessible

## Field

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

* **static int HORIZONTAL**--A constant that indicates a horizontal scroll bar.
* **static int VERTICAL**--A constant that indicates a vertical scroll bar.

## Class constructors

|  |  |
| --- | --- |
| **S.N.** | **Constructor & Description** |
| 1 | **Scrollbar()** Constructs a new vertical scroll bar. |
| 2 | **Scrollbar(int orientation)** Constructs a new scroll bar with the specified orientation. |
| 3 | **Scrollbar(int orientation, int value, int visible, int minimum, int maximum)** Constructs a new scroll bar with the specified orientation, initial value, visible amount, and minimum and maximum values. |

## Class methods

|  |  |
| --- | --- |
| **S.N.** | **Method & Description** |
| 1 | **void addAdjustmentListener(AdjustmentListener l)** Adds the specified adjustment listener to receive instances of AdjustmentEvent from this scroll bar. |
| 2 | **void addNotify()** Creates the Scrollbar's peer. |
| 3 | **int getBlockIncrement()** Gets the block increment of this scroll bar. |
| 4 | **int getLineIncrement()** Deprecated. As of JDK version 1.1, replaced by getUnitIncrement(). |
| 5 | **int getMaximum()** Gets the maximum value of this scroll bar. |
| 6 | **int getMinimum()** Gets the minimum value of this scroll bar. |
| 7 | **int getOrientation()** Returns the orientation of this scroll bar. |
| 8 | **int getPageIncrement()** Deprecated. As of JDK version 1.1, replaced by getBlockIncrement(). |
| 9 | **int getUnitIncrement()** Gets the unit increment for this scrollbar. |
| 10 | **int getValue()** Gets the current value of this scroll bar. |
| 11 | **boolean** getValueIsAdjusting()  Returns true if the value is in the process of changing as a result of actions being taken by the user. |
| 12 | **int getVisible()** Deprecated. As of JDK version 1.1, replaced by getVisibleAmount(). |
| 13 | **int getVisibleAmount()** Gets the visible amount of this scroll bar. |
| 14 | **protected String paramString()** Returns a string representing the state of this Scrollbar. |
| 15 | **protected void processAdjustmentEvent(AdjustmentEvent e)** Processes adjustment events occurring on this scrollbar by dispatching them to any registered AdjustmentListener objects. |
| 16 | **protected void processEvent(AWTEvent e)** Processes events on this scroll bar. |
| 17 | **void removeAdjustmentListener(AdjustmentListener l)** Removes the specified adjustment listener so that it no longer receives instances of AdjustmentEvent from this scroll bar. |
| 18 | **void setBlockIncrement(int v)** Sets the block increment for this scroll bar. |
| 19 | **void setLineIncrement(int v)** Deprecated. As of JDK version 1.1, replaced by setUnitIncrement(int). |
| 20 | **void setMaximum(int newMaximum)** Sets the maximum value of this scroll bar. |
| 21 | **void setMinimum(int newMinimum)** Sets the minimum value of this scroll bar. |
| 22 | **void setOrientation(int orientation)** Sets the orientation for this scroll bar. |
| 23 | **void setPageIncrement(int v)** Deprecated. As of JDK version 1.1, replaced by setBlockIncrement(). |
| 24 | **void setUnitIncrement(int v)** Sets the unit increment for this scroll bar. |
| 25 | **void setValue(int newValue)** Sets the value of this scroll bar to the specified value. |
| 26 | **void setValueIsAdjusting(boolean b)** Sets the valueIsAdjusting property. |
| 27 | **void setValues(int value, int visible, int minimum, int maximum)** Sets the values of four properties for this scroll bar: value, visibleAmount, minimum, and maximum. |
| 28 | **void setVisibleAmount(int newAmount)** Sets the visible amount of this scroll bar. |
| 29 | **AccessibleContext getAccessibleContext()** Gets the AccessibleContext associated with this Scrollbar. |
| 30 | **AdjustmentListener[] getAdjustmentListeners()** Returns an array of all the adjustment listeners registered on this scrollbar. |
| 31 | **<T extends EventListener>T[] getListeners(Class<T> listenerType)** Returns an array of all the objects currently registered as FooListeners upon this Scrollbar. |

## Methods inherited

This class inherits methods from the following classes:

* java.awt.Component
* java.lang.Object

## Choice Example

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

*AwtControlDemo*

package com.tutorialspoint.gui;

import java.awt.\*;

import java.awt.event.\*;

public class AwtControlDemo {

private Frame mainFrame;

private Label headerLabel;

private Label statusLabel;

private Panel controlPanel;

public AwtControlDemo(){

prepareGUI();

}

public static void main(String[] args){

AwtControlDemo awtControlDemo = new AwtControlDemo();

awtControlDemo.showScrollbarDemo();

}

private void prepareGUI(){

mainFrame = new Frame("Java AWT 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 Label();

headerLabel.setAlignment(Label.CENTER);

statusLabel = new Label();

statusLabel.setAlignment(Label.CENTER);

statusLabel.setSize(350,100);

controlPanel = new Panel();

controlPanel.setLayout(new FlowLayout());

mainFrame.add(headerLabel);

mainFrame.add(controlPanel);

mainFrame.add(statusLabel);

mainFrame.setVisible(true);

}

private void showScrollbarDemo(){

headerLabel.setText("Control in action: Scrollbar");

final Scrollbar horizontalScroller = new Scrollbar(Scrollbar.HORIZONTAL);

final Scrollbar verticalScroller = new Scrollbar();

verticalScroller.setOrientation(Scrollbar.VERTICAL);

horizontalScroller.setMaximum (100);

horizontalScroller.setMinimum (1);

verticalScroller.setMaximum (100);

verticalScroller.setMinimum (1);

horizontalScroller.addAdjustmentListener(new AdjustmentListener() {

@Override

public void adjustmentValueChanged(AdjustmentEvent e) {

statusLabel.setText("Horozontal: "

+horizontalScroller.getValue()

+" ,Vertical: "

+ verticalScroller.getValue());

}

});

verticalScroller.addAdjustmentListener(new AdjustmentListener() {

@Override

public void adjustmentValueChanged(AdjustmentEvent e) {

statusLabel.setText("Horozontal: "

+horizontalScroller.getValue()

+" ,Vertical: "+ verticalScroller.getValue());

}

});

controlPanel.add(horizontalScroller);

controlPanel.add(verticalScroller);

mainFrame.setVisible(true);

}

}

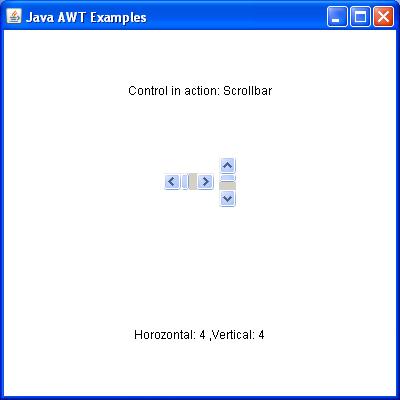
Compile the program using command prompt. Go to **D:/ > AWT**and type the following command.

D:\AWT>javac com\tutorialspoint\gui\AwtControlDemo.java

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

D:\AWT>java com.tutorialspoint.gui.AwtControlDemo

Verify the following output



**TextArea**

## Class declaration

Following is the declaration for **java.awt.TextArea** class:

public class TextArea

extends TextComponent

## Field

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

* **static int SCROLLBARS\_BOTH**-- Create and display both vertical and horizontal scrollbars.
* **static int SCROLLBARS\_HORIZONTAL\_ONLY**-- Create and display horizontal scrollbar only.
* **static int SCROLLBARS\_NONE**-- Do not create or display any scrollbars for the text area.
* **static int SCROLLBARS\_VERTICAL\_ONLY** -- Create and display vertical scrollbar only.

## Class constructors

|  |  |
| --- | --- |
| **S.N.** | **Constructor & Description** |
| 1 | **TextArea()** Constructs a new text area with the empty string as text. |
| 2 | **TextArea(int rows, int columns)** Constructs a new text area with the specified number of rows and columns and the empty string as text. |
| 3 | **TextArea(String text)** Constructs a new text area with the specified text. |
| 4 | **TextArea(String text, int rows, int columns)** Constructs a new text area with the specified text, and with the specified number of rows and columns. |
| 5 | **TextArea(String text, int rows, int columns, int scrollbars)** Constructs a new text area with the specified text, and with the rows, columns, and scroll bar visibility as specified. |

## Class methods

|  |  |
| --- | --- |
| **S.N.** | **Method & Description** |
| 1 | **void addNotify()** Creates the TextArea's peer. |
| 2 | **void append(String str)** Appends the given text to the text area's current text. |
| 3 | **void appendText(String str)** Deprecated. As of JDK version 1.1, replaced by append(String). |
| 4 | **AccessibleContext getAccessibleContext()** Returns the AccessibleContext associated with this TextArea. |
| 5 | **int getColumns()** Returns the number of columns in this text area. |
| 6 | **Dimension getMinimumSize()** Determines the minimum size of this text area. |
| 7 | **Dimension getMinimumSize(int rows, int columns)** Determines the minimum size of a text area with the specified number of rows and columns. |
| 8 | **Dimension getPreferredSize()** Determines the preferred size of this text area. |
| 9 | **Dimension getPreferredSize(int rows, int columns)** Determines the preferred size of a text area with the specified number of rows and columns. |
| 10 | **int getRows()** Returns the number of rows in the text area. |
| 11 | **int getScrollbarVisibility()** Returns an enumerated value that indicates which scroll bars the text area uses. |
| 12 | **void insert(String str, int pos)** Inserts the specified text at the specified position in this text area. |
| 13 | **void insertText(String str, int pos)** Deprecated. As of JDK version 1.1, replaced by insert(String, int). |
| 14 | **Dimension minimumSize()** Deprecated. As of JDK version 1.1, replaced by getMinimumSize(). |
| 15 | **Dimension minimumSize(int rows, int columns)** Deprecated. As of JDK version 1.1, replaced by getMinimumSize(int, int). |
| 16 | **protected String paramString()** Returns a string representing the state of this TextArea. |
| 17 | **Dimension preferredSize()** Deprecated. As of JDK version 1.1, replaced by getPreferredSize(). |
| 18 | **Dimension preferredSize(int rows, int columns)** Deprecated. As of JDK version 1.1, replaced by getPreferredSize(int, int). |
| 19 | **void replaceRange(String str, int start, int end)** Replaces text between the indicated start and end positions with the specified replacement text. |
| 20 | **void replaceText(String str, int start, int end)** Deprecated. As of JDK version 1.1, replaced by replaceRange(String, int, int). |
| 21 | **void setColumns(int columns)** Sets the number of columns for this text area. |
| 22 | **void setRows(int rows)** Sets the number of rows for this text area. |

## Methods inherited

This class inherits methods from the following classes:

* java.awt.TextComponent
* java.awt.Component
* java.lang.Object

## TextArea Example

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

*AwtControlDemo.java*

package com.tutorialspoint.gui;

import java.awt.\*;

import java.awt.event.\*;

public class AwtControlDemo {

private Frame mainFrame;

private Label headerLabel;

private Label statusLabel;

private Panel controlPanel;

public AwtControlDemo(){

prepareGUI();

}

public static void main(String[] args){

AwtControlDemo awtControlDemo = new AwtControlDemo();

awtControlDemo.showTextAreaDemo();

}

private void prepareGUI(){

mainFrame = new Frame("Java AWT 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 Label();

headerLabel.setAlignment(Label.CENTER);

statusLabel = new Label();

statusLabel.setAlignment(Label.CENTER);

statusLabel.setSize(350,100);

controlPanel = new Panel();

controlPanel.setLayout(new FlowLayout());

mainFrame.add(headerLabel);

mainFrame.add(controlPanel);

mainFrame.add(statusLabel);

mainFrame.setVisible(true);

}

private void showTextAreaDemo(){

headerLabel.setText("Control in action: TextArea");

Label commentlabel= new Label("Comments: ", Label.RIGHT);

final TextArea commentTextArea = new TextArea("This is a AWT tutorial "

+"to make GUI application in Java.",5,30);

Button showButton = new Button("Show");

showButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

statusLabel.setText( commentTextArea.getText());

}

});

controlPanel.add(commentlabel);

controlPanel.add(commentTextArea);

controlPanel.add(showButton);

mainFrame.setVisible(true);

}

}

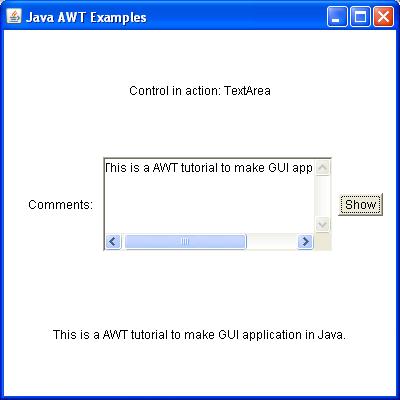
Compile the program using command prompt. Go to **D:/ > AWT**and type the following command.

D:\AWT>javac com\tutorialspoint\gui\AwtControlDemo.java

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

D:\AWT>java com.tutorialspoint.gui.AwtControlDemo

Verify the following output



**TextField**

# JTextField Source Code

|  |
| --- |
| // A program to demonstrate the use of JTextFields's  //Import Statements  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  public class JTextFieldDemo extends JFrame {  //Class Declarations  JTextField jtfText1, jtfUneditableText;  String disp = "";  TextHandler handler = null;  //Constructor  public JTextFieldDemo() {  super("TextField Test Demo");  Container container = getContentPane();  container.setLayout(new FlowLayout());  jtfText1 = new JTextField(10);  jtfUneditableText = new JTextField("Uneditable text field", 20);  jtfUneditableText.setEditable(false);  container.add(jtfText1);  container.add(jtfUneditableText);  handler = new TextHandler();  jtfText1.addActionListener(handler);  jtfUneditableText.addActionListener(handler);  setSize(325, 100);  setVisible(true);  }  //Inner Class TextHandler  private class TextHandler implements ActionListener {  public void actionPerformed(ActionEvent e) {  if (e.getSource() == jtfText1) {  disp = "text1 : " + e.getActionCommand();  } else if (e.getSource() == jtfUneditableText) {  disp = "text3 : " + e.getActionCommand();  }  JOptionPane.showMessageDialog(null, disp);  }  }  //Main Program that starts Execution  public static void main(String args[]) {  JTextFieldDemo test = new JTextFieldDemo();  test.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  }  }// End of class TextFieldTest  **Output**  http://www.javabeginner.com/images/jtextfield.JPG  [**Download**](http://www.javabeginner.com/images/jtextfield.zip)**jTextField Source Code** Another Example: JTextField Source Code public class JTextFieldDemo2 extends JFrame implements ActionListener {  JTextField jtfInput;  JTextArea jtAreaOutput;  String newline = "\n";  public JTextFieldDemo2() {  createGui();  }  public void createGui() {  jtfInput = new JTextField(20);  jtfInput.addActionListener(this);  jtAreaOutput = new JTextArea(5, 20);  jtAreaOutput.setEditable(false);  JScrollPane scrollPane = new JScrollPane(jtAreaOutput,  JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS,  JScrollPane.HORIZONTAL\_SCROLLBAR\_ALWAYS);  GridBagLayout gridBag = new GridBagLayout();  Container contentPane = getContentPane();  contentPane.setLayout(gridBag);  GridBagConstraints gridCons1 = new GridBagConstraints();  gridCons1.gridwidth = GridBagConstraints.REMAINDER;  gridCons1.fill = GridBagConstraints.HORIZONTAL;  contentPane.add(jtfInput, gridCons1);  GridBagConstraints gridCons2 = new GridBagConstraints();  gridCons2.weightx = 1.0;  gridCons2.weighty = 1.0;  contentPane.add(scrollPane, gridCons2);  }  public void actionPerformed(ActionEvent evt) {  String text = jtfInput.getText();  jtAreaOutput.append(text + newline);  jtfInput.selectAll();  }  public static void main(String[] args) {  JTextFieldDemo2 jtfTfDemo = new JTextFieldDemo2();  jtfTfDemo.pack();  jtfTfDemo.addWindowListener(new WindowAdapter() {  public void windowClosing(WindowEvent e) {  System.exit(0);  }  });  jtfTfDemo.setVisible(true);  }  } |

**Output**



[**Download**](http://www.javabeginner.com/jtextfield-2.zip)jTextField Source Code

### Java JTextField Hierarchy

javax.swing  
java.lang.Object  
java.awt.Component  
java.awt.Container  
javax.swing.JComponent  
javax.swing.text.JTextComponent  
javax.swing.JTextField  
**All Implemented Interfaces:**  
Accessible, ImageObserver, MenuContainer, Scrollable, Serializable, SwingConstants  
**Direct Known Subclasses:**  
DefaultTreeCellEditor.DefaultTextField, JFormattedTextField, JPasswordField

## JTextField Constructor

**JTextField()**  
Constructs a new TextField.

**JTextField(Document doc, String text, int columns)**  
Constructs a new JTextField that uses the given text storage model and the given number of columns.

**JTextField(int columns)**  
Constructs a new empty TextField with the specified n  
umber of columns.

**JTextField(String text)**  
Constructs a new TextField initialized with the specified text.

**JTextField(String text, int columns)**  
Constructs a new TextField initialized with the specified text and columns.

**Container Components**

**Window (Frame and Dialog)**

## Introduction

The class **Frame** is a top level window with border and title. It uses BorderLayout as default layout manager.

## Class declaration

Following is the declaration for **java.awt.Frame** class:

public class Frame

extends Window

implements MenuContainer

## Field

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

* **static float BOTTOM\_ALIGNMENT** -- Ease-of-use constant for getAlignmentY.
* **static int CROSSHAIR\_CURSOR**-- Deprecated. replaced by Cursor.CROSSHAIR\_CURSOR.
* **static int DEFAULT\_CURSOR**-- Deprecated. replaced by Cursor.DEFAULT\_CURSOR.
* **static int E\_RESIZE\_CURSOR**-- Deprecated. replaced by Cursor.E\_RESIZE\_CURSOR.
* **static int HAND\_CURSOR**-- Deprecated. replaced by Cursor.HAND\_CURSOR.
* **static int ICONIFIED**-- This state bit indicates that frame is iconified.
* **static int MAXIMIZED\_BOTH**-- This state bit mask indicates that frame is fully maximized (that is both horizontally and vertically).
* **static int MAXIMIZED\_HORIZ**-- This state bit indicates that frame is maximized in the horizontal direction.
* **static int MAXIMIZED\_VERT**-- This state bit indicates that frame is maximized in the vertical direction.
* **static int MOVE\_CURSOR**-- Deprecated. replaced by Cursor.MOVE\_CURSOR.
* **static int N\_RESIZE\_CURSOR**-- Deprecated. replaced by Cursor.N\_RESIZE\_CURSOR.
* **static int NE\_RESIZE\_CURSOR**-- Deprecated. replaced by Cursor.NE\_RESIZE\_CURSOR.
* **static int NORMAL**-- Frame is in the "normal" state.
* **static int NW\_RESIZE\_CURSOR**-- Deprecated. replaced by Cursor.NW\_RESIZE\_CURSOR.
* **static int S\_RESIZE\_CURSOR**-- Deprecated. replaced by Cursor.S\_RESIZE\_CURSOR.
* **static int SE\_RESIZE\_CURSOR**-- Deprecated. replaced by Cursor.SE\_RESIZE\_CURSOR.
* **static int SW\_RESIZE\_CURSOR**-- Deprecated. replaced by Cursor.SW\_RESIZE\_CURSOR.
* **static int TEXT\_CURSOR**-- Deprecated. replaced by Cursor.TEXT\_CURSOR.
* **static int W\_RESIZE\_CURSOR**-- Deprecated. replaced by Cursor.W\_RESIZE\_CURSOR.
* **static int WAIT\_CURSOR**-- Deprecated. replaced by Cursor.WAIT\_CURSOR.

## Class constructors

|  |  |
| --- | --- |
| **S.N.** | **Constructor & Description** |
| 1 | **Frame()** Constructs a new instance of Frame that is initially invisible. |
| 2 | **Frame(GraphicsConfiguration gc)** Constructs a new, initially invisible Frame with the specified GraphicsConfiguration. |
| 3 | **Frame(String title)** Constructs a new, initially invisible Frame object with the specified title. |
| 4 | **Frame(String title, GraphicsConfiguration gc)** Constructs a new, initially invisible Frame object with the specified title and a GraphicsConfiguration. |

## Class methods

|  |  |
| --- | --- |
| **S.N.** | **Method & Description** |
| 1 | **void addNotify()** Makes this Frame displayable by connecting it to a native screen resource. |
| 2 | **AccessibleContext getAccessibleContext()** Gets the AccessibleContext associated with this Frame. |
| 3 | **int getCursorType()** Deprecated. As of JDK version 1.1, replaced by Component.getCursor(). |
| 4 | **int getExtendedState()** Gets the state of this frame. |
| 5 | **static Frame[] getFrames()** Returns an array of all Frames created by this application. |
| 6 | **Image getIconImage()** Returns the image to be displayed as the icon for this frame. |
| 7 | **Rectangle getMaximizedBounds()** Gets maximized bounds for this frame. |
| 8 | **MenuBar getMenuBar()** Gets the menu bar for this frame. |
| 9 | **int getState()** Gets the state of this frame (obsolete). |
| 10 | **String getTitle()** Gets the title of the frame. |
| 11 | **boolean isResizable()** Indicates whether this frame is resizable by the user. |
| 12 | **boolean isUndecorated()** Indicates whether this frame is undecorated. |
| 13 | **protected String paramString()** Returns a string representing the state of this Frame. |
| 14 | **void remove(MenuComponent m)** Removes the specified menu bar from this frame. |
| 15 | **void removeNotify()** Makes this Frame undisplayable by removing its connection to its native screen resource. |
| 16 | **void setCursor(int cursorType)** Deprecated. As of JDK version 1.1, replaced by Component.setCursor(Cursor). |
| 17 | **void setExtendedState(int state)** Sets the state of this frame. |
| 18 | **void setIconImage(Image image)** Sets the image to be displayed as the icon for this window. |
| 19 | **void setMaximizedBounds(Rectangle bounds)** Sets the maximized bounds for this frame. |
| 20 | **void setMenuBar(MenuBar mb)** Sets the menu bar for this frame to the specified menu bar. |
| 21 | **void setResizable(boolean resizable)** Sets whether this frame is resizable by the user. |
| 22 | **void setState(int state)** Sets the state of this frame (obsolete). |
| 23 | **void setTitle(String title)** Sets the title for this frame to the specified string. |
| 24 | **void setUndecorated(boolean undecorated)** Disables or enables decorations for this frame. |

## Methods inherited

This class inherits methods from the following classes:

* java.awt.Window
* java.awt.Container
* java.awt.Component
* java.lang.Object

## Frame Example

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

*AwtContainerDemo.java*

package com.tutorialspoint.gui;

import java.awt.\*;

import java.awt.event.\*;

public class AwtContainerDemo {

private Frame mainFrame;

private Label headerLabel;

private Label statusLabel;

private Panel controlPanel;

private Label msglabel;

public AwtContainerDemo(){

prepareGUI();

}

public static void main(String[] args){

AwtContainerDemo awtContainerDemo = new AwtContainerDemo();

awtContainerDemo.showFrameDemo();

}

private void prepareGUI(){

mainFrame = new Frame("Java AWT 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 Label();

headerLabel.setAlignment(Label.CENTER);

statusLabel = new Label();

statusLabel.setAlignment(Label.CENTER);

statusLabel.setSize(350,100);

msglabel = new Label();

msglabel.setAlignment(Label.CENTER);

msglabel.setText("Welcome to TutorialsPoint AWT Tutorial.");

controlPanel = new Panel();

controlPanel.setLayout(new FlowLayout());

mainFrame.add(headerLabel);

mainFrame.add(controlPanel);

mainFrame.add(statusLabel);

mainFrame.setVisible(true);

}

private void showFrameDemo(){

headerLabel.setText("Container in action: Frame");

final Frame frame = new Frame();

frame.setSize(300, 300);

frame.setLayout(new FlowLayout());

frame.add(msglabel);

frame.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent windowEvent){

frame.dispose();

}

});

Button okButton = new Button("Open a Frame");

okButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

statusLabel.setText("A Frame shown to the user.");

frame.setVisible(true);

}

});

controlPanel.add(okButton);

mainFrame.setVisible(true);

}

}

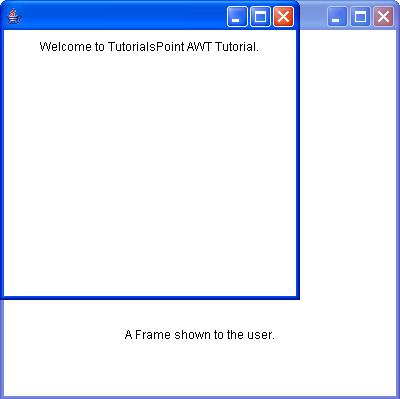
Compile the program using command prompt. Go to **D:/ > AWT**and type the following command.

D:\AWT>javac com\tutorialspoint\gui\AwtContainerDemo.java

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

D:\AWT>java com.tutorialspoint.gui.AwtContainerDemo

Verify the following output



## Introduction

Dialog control represents a top-level window with a title and a border used to take some form of input from the user.

## Class declaration

Following is the declaration for **java.awt.Dialog** class:

public class Dialog

extends Window

## Field

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

* **static Dialog.ModalityType DEFAULT\_MODALITY\_TYPE**-- Default modality type for modal dialogs.

## Class constructors

|  |  |
| --- | --- |
| **S.N.** | **Constructor & Description** |
| 1 | **Dialog(Dialog owner)** Constructs an initially invisible, modeless Dialog with the specified owner Dialog and an empty title. |
| 2 | **Dialog(Dialog owner, String title)** Constructs an initially invisible, modeless Dialog with the specified owner Dialog and title. |
| 3 | **Dialog(Dialog owner, String title, boolean modal)** Constructs an initially invisible Dialog with the specified owner Dialog, title, and modality. |
| 4 | **Dialog(Dialog owner, String title, boolean modal, GraphicsConfiguration gc)** Constructs an initially invisible Dialog with the specified owner Dialog, title, modality and GraphicsConfiguration. |
| 5 | **Dialog(Frame owner)** Constructs an initially invisible, modeless Dialog with the specified owner Frame and an empty title. |
| 6 | **Dialog(Frame owner, boolean modal)** Constructs an initially invisible Dialog with the specified owner Frame and modality and an empty title. |
| 7 | **Dialog(Frame owner, String title)** Constructs an initially invisible, modeless Dialog with the specified owner Frame and title. |
| 8 | **Dialog(Frame owner, String title, boolean modal)** Constructs an initially invisible Dialog with the specified owner Frame, title and modality. |
| 9 | **Dialog(Frame owner, String title, boolean modal, GraphicsConfiguration gc)** Constructs an initially invisible Dialog with the specified owner Frame, title, modality, and GraphicsConfiguration. |
| 10 | **Dialog(Window owner)** Constructs an initially invisible, modeless Dialog with the specified owner Window and an empty title. |
| 11 | **Dialog(Window owner, Dialog.ModalityType modalityType)** Constructs an initially invisible Dialog with the specified owner Window and modality and an empty title. |
| 12 | **Dialog(Window owner, String title)** Constructs an initially invisible, modeless Dialog with the specified owner Window and title. |
| 13 | **Dialog(Window owner, String title, Dialog.ModalityType modalityType)** Constructs an initially invisible Dialog with the specified owner Window, title and modality. |
| 14 | **Dialog(Window owner, String title, Dialog.ModalityType modalityType, GraphicsConfiguration gc)** Constructs an initially invisible Dialog with the specified owner Window, title, modality and GraphicsConfiguration |

## Class methods

|  |  |
| --- | --- |
| **S.N.** | **Method & Description** |
| 1 | **void addNotify()** Makes this Dialog displayable by connecting it to a native screen resource. |
| 2 | **AccessibleContext getAccessibleContext()** Gets the AccessibleContext associated with this Dialog. |
| 3 | **Dialog.ModalityType getModalityType()** Returns the modality type of this dialog. |
| 4 | **String getTitle()** Gets the title of the dialog. |
| 5 | **void hide()** Deprecated. As of JDK version 1.5, replaced by setVisible(boolean). |
| 6 | **boolean isModal()** Indicates whether the dialog is modal. |
| 7 | **boolean isResizable()** Indicates whether this dialog is resizable by the user. |
| 8 | **boolean isUndecorated()** Indicates whether this dialog is undecorated. |
| 9 | **protected String paramString()** Returns a string representing the state of this dialog. |
| 10 | **void setModal(boolean modal)** Specifies whether this dialog should be modal. |
| 11 | **void setModalityType(Dialog.ModalityType type)** Sets the modality type for this dialog. |
| 12 | **void setResizable(boolean resizable)** Sets whether this dialog is resizable by the user. |
| 13 | **void setTitle(String title)** Sets the title of the Dialog. |
| 14 | **void setUndecorated(boolean undecorated)** Disables or enables decorations for this dialog. |
| 15 | **void setVisible(boolean b)** Shows or hides this Dialog depending on the value of parameter b. |
| 16 | **void show()** Deprecated. As of JDK version 1.5, replaced by setVisible(boolean). |
| 17 | **void toBack()** If this Window is visible, sends this Window to the back and may cause it to lose focus or activation if it is the focused or active Window. |

## Methods inherited

This class inherits methods from the following classes:

* java.awt.Window
* java.awt.Component
* java.lang.Object

## Dialog Example

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

*AwtControlDemo.java*

package com.tutorialspoint.gui;

import java.awt.\*;

import java.awt.event.\*;

public class AwtControlDemo {

private Frame mainFrame;

private Label headerLabel;

private Label statusLabel;

private Panel controlPanel;

public AwtControlDemo(){

prepareGUI();

}

public static void main(String[] args){

AwtControlDemo awtControlDemo = new AwtControlDemo();

awtControlDemo.showDialogDemo();

}

private void prepareGUI(){

mainFrame = new Frame("Java AWT 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 Label();

headerLabel.setAlignment(Label.CENTER);

statusLabel = new Label();

statusLabel.setAlignment(Label.CENTER);

statusLabel.setSize(350,100);

controlPanel = new Panel();

controlPanel.setLayout(new FlowLayout());

mainFrame.add(headerLabel);

mainFrame.add(controlPanel);

mainFrame.add(statusLabel);

mainFrame.setVisible(true);

}

private void showDialogDemo(){

headerLabel.setText("Control in action: Dialog");

Button showAboutDialogButton = new Button("Show About Dialog");

showAboutDialogButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

AboutDialog aboutDialog = new AboutDialog(mainFrame);

aboutDialog.setVisible(true);

}

});

controlPanel.add(showAboutDialogButton);

mainFrame.setVisible(true);

}

class AboutDialog extends Dialog {

public AboutDialog(Frame parent){

super(parent, true);

setBackground(Color.gray);

setLayout(new BorderLayout());

Panel panel = new Panel();

panel.add(new Button("Close"));

add("South", panel);

setSize(200,200);

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent windowEvent){

dispose();

}

});

}

public boolean action(Event evt, Object arg){

if(arg.equals("Close")){

dispose();

return true;

}

return false;

}

public void paint(Graphics g){

g.setColor(Color.white);

g.drawString("TutorialsPoint.Com", 25,70 );

g.drawString("Version 1.0", 60, 90);

}

}

}

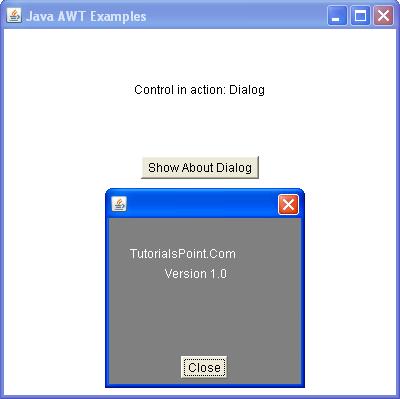
Compile the program using command prompt. Go to **D:/ > AWT**and type the following command.

D:\AWT>javac com\tutorialspoint\gui\AwtControlDemo.java

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

D:\AWT>java com.tutorialspoint.gui.AwtControlDemo

Verify the following output



**Panel (Applet)**

## Introduction

The class **Panel** is the simplest container class. It provides space in which an application can attach any other component, including other panels. It uses FlowLayout as default layout manager.

## Class declaration

Following is the declaration for **java.awt.Panel** class:

public class Panel

extends Container

implements Accessible

## Class constructors

|  |  |
| --- | --- |
| **S.N.** | **Constructor & Description** |
| 1 | **Panel()** Creates a new panel using the default layout manager. |
| 2 | **Panel(LayoutManager layout)** Creates a new panel with the specified layout manager. |

## Class methods

|  |  |
| --- | --- |
| **S.N.** | **Method & Description** |
| 1 | **void addNotify()** Creates the Panel's peer. |
| 2 | **AccessibleContext getAccessibleContext()** Gets the AccessibleContext associated with this Panel. |

## Methods inherited

This class inherits methods from the following classes:

* java.awt.Container
* java.awt.Component
* java.lang.Object

## Panel Example

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

*AwtContainerDemo.java*

package com.tutorialspoint.gui;

import java.awt.\*;

import java.awt.event.\*;

public class AwtContainerDemo {

private Frame mainFrame;

private Label headerLabel;

private Label statusLabel;

private Panel controlPanel;

private Label msglabel;

public AwtContainerDemo(){

prepareGUI();

}

public static void main(String[] args){

AwtContainerDemo awtContainerDemo = new AwtContainerDemo();

awtContainerDemo.showPanelDemo();

}

private void prepareGUI(){

mainFrame = new Frame("Java AWT 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 Label();

headerLabel.setAlignment(Label.CENTER);

statusLabel = new Label();

statusLabel.setAlignment(Label.CENTER);

statusLabel.setSize(350,100);

msglabel = new Label();

msglabel.setAlignment(Label.CENTER);

msglabel.setText("Welcome to TutorialsPoint AWT Tutorial.");

controlPanel = new Panel();

controlPanel.setLayout(new FlowLayout());

mainFrame.add(headerLabel);

mainFrame.add(controlPanel);

mainFrame.add(statusLabel);

mainFrame.setVisible(true);

}

private void showPanelDemo(){

headerLabel.setText("Container in action: Panel");

Panel panel = new Panel();

panel.setBackground(Color.magenta);

panel.setLayout(new FlowLayout());

panel.add(msglabel);

controlPanel.add(panel);

mainFrame.setVisible(true);

}

}

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

D:\AWT>javac com\tutorialspoint\gui\AwtContainerDemo.java

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

D:\AWT>java com.tutorialspoint.gui.AwtContainerDemo

Verify the following output

