Introduction of AWT

* The Abstract Window Toolkit (AWT) is Java's original platform-independent windowing, graphics, and user-interface widget toolkit.

* The AWT is both powerful and flexible. Newcomers, however, often find that its power is veiled. The class and method descriptions found in the distributed documentation provide little guidance for the new programmer

* The AWT is now part of the Java Foundation Classes (JFC) - the standard API for providing a graphical user interface (GUI) for a Java program.

* Building a GUI involves creating "abstract" components such as buttons and windows, which are then mapped to "concrete" components for a specific platform.

* AWT is also the GUI toolkit for a number of Java ME profiles.

Example - Connected Device Configuration profiles require Java runtimes on mobile telephones to support AWT.

Creating a Graphical User Interface

* GUI programming in Java is based on three concepts:

* + Components- A component is an object that the user can see on the screen and-in most cases-interact with.
  + Containers- A container is a component that can hold other components.
  + Events- An event is an action triggered by the user, such as a key press or mouse click. Designing a graphical user interface involves creating components, putting them into containers, and arranging for the program to respond to events.
* Components are objects, so they're created by invoking a constructor.

* A button would be created by using a constructor belonging to the Button class.

* The most commonly used constructor has one argument (the button's label):

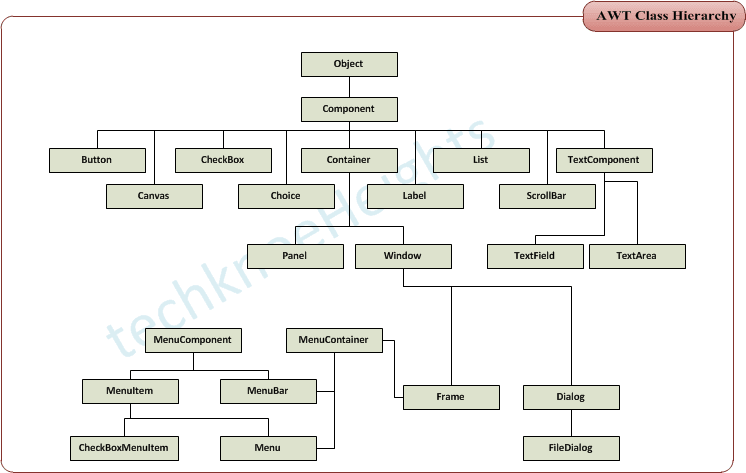
Button b = new Button("Testing");

* For a component to be visible, it must be added to a container (typically a frame) by the add method.

* To detect when an event occurs, a special " Listener " object can be attached to a component.

* When the user performs an action that involves the component, a method belonging to the listener object will be called automatically.

AWT Class Hierarchy



* **Java Applets**: import java.applet.Applet;
  + embedded in another application (not run on its own)
  + must be superclass of applet embedded in a Web page
  + subclass of Panel (which was a subclass of Container)

* **Container**: import java.awt.\*;

* + component that can contain other AWT components
* **Component**:import java.awt.\*;
  + interacts with the user, has graphical representation, displayed on the screen
  + abstract superclass of the nonmenu-related AWT components
  + Examples: buttons, checkboxes, text fields
  + Has methods for sizing and moving
  + public void paint(Graphics g) // Paints this component - don't call directly
  + void repaint(); // Requests this component be repainted
* **Graphics**:import java.awt.\*;
  + abstract base class for all graphics contexts
  + allow an application to draw onto components
  + encapsulates state information needed for the basic rendering color, font, object on which to draw, etc.
  + public void drawLine(int x1, int y1, int x2, int y2);
  + void drawString(String str, int x, int y);
  + void drawOval(int x, int y, int width, int height); // x,y: upper-left
  + void setColor(Color c); // set current drawing color
  + void setFont(Font font);
* **TextField**:import java.awt.\*;
  + component that allows for the editing of a single line of text
  + String getText(); // returns current text in the text field
  + void setText(String t); // set text for the text field

* **List**:import java.awt.\*;
  + List of String: General "container" but NOT an AWT Container
  + void add(String item) - adds an item to the end of the scrolling list
  + int getSelectedIndex() - gets the index of the selected item on the list
  + void remove(int position) - removes the item at the specified position index
* **ActionListener**:import java.awt.event.\*;
  + interface for receiving action events - used by Buttons, Timers
  + void actionPerformed(ActionEvent e); // must implement this
  + public class SimulationApplet extends java.applet.Applet
  + implements java.awt.event.ActionListener

* **ActionEvent**:import java.awt.event.\*;
  + event which indicates that a component-defined action occurred
  + Object getSource(); // Returns reference to initiating Object
  + String getActionCommand(); // returns the command string

Frames

* In Java terminology, a frame is a window with a title and a border.
* A frame may also have a menu bar.
* Frames play an important role in the AWT because a GUI program normally displays a frame when it's executed.
* The DrawableFrame objects used in previous chapters are examples of frames.
* Frames are created using one of the constructors in the Frame class.
* One constructor takes a single argument (the title to be displayed at the top of the frame):

Frame f = new Frame("Title goes here");

* Although the Frame object now exists, it's not visible on the screen.
* Before making the frame visible, a method should be called to set the size of the frame.
* If desired, the frame's location can also be specified.

Frame Methods

* Many methods used with Frame objects are inherited from Window (Frame's superclass) or from Component (Window's superclass).
* The setSize method sets the width and height of a frame:

f.setSize(width, height);

* If a program fails to call setSize or pack before displaying a frame, it will assume a default size.
* The size of a frame can change during the execution of a program.
* The getSize method returns a frame's current width and height:

Dimension frameSize = f.getSize();

* frameSize.width will contain f's width. frameSize.height will contain f's height.
* The setVisible method controls whether or not a frame is currently visible on the screen.
* Calling setVisible with true as the argument makes a frame visible:

f.setVisible(true);

* Calling it with false as the argument makes the frame disappear from the screen:

f.setVisible(false);

* The Frame object still exists; it can be made to reappear later by calling setVisible again.

Creating a Frame

* The FrameTest program creates a Frame object and displays it on the screen.
* This program illustrates three key steps:
  1. Using the Frame constructor to create a frame.
  2. Setting the size of the frame.
  3. Displaying the frame on the screen.

// Displays a frame on the screen.

// WARNING: Frame cannot be closed.

import java.awt.\*;

public class FrameTest

{

public static void main(String[] args)

{

Frame f = new Frame("Frame Test");

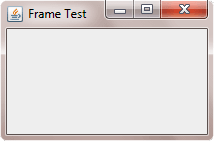
f.setSize(214,141);

f.setVisible(true);

}

}

Output :



Note: As with the other AWT components, the appearance of a frame depends on the platform.

Setting the Location of a Frame

* By default, all windows (including frames) are displayed in the upper-left corner of the screen, which has coordinates (0, 0).
* The setLocation method can be used to specify a different location:

f.setLocation(50, 75);

* To find the current location of a frame, call getLocation:

Point frameLocation = f.getLocation();

* The coordinates of f's upper-left corner will be stored in frameLocation.x and frameLocation.y.

Adding Components to a Frame

* The Frame class is rarely used to create objects directly.
* Instead, it's customary to define a subclass of Frame and then create an instance of the subclass.
* This strategy makes it possible to tailor the subclass.
* In particular, the constructor for the subclass can put components into the frame.

### Label

**Package :** Java.awt   
  
**Class:** java.awt.Label   
  
**Note :**A Label displays a single line of read-only text

##### **Constructors of Label Class**

| **Name** | **Description** |
| --- | --- |
| Label() | This constructor creates instance of a Label. |
| Label(String text) | This constructor creates instance of a Label with the given label of string |
| Label(String text, int alignment) | This constructor creates instance of a Label with the given label of string and also sets the alignment |

##### **Methods of Label Class**

| **Method** | **Description** |
| --- | --- |
| getAlignment() | This method gets the text alignment of Label. |
| getText() | This method gets the Text of Label. |
| setAlignment(int alignment) | This method sets the Text alignment of label. |
| setText(String text) | This method sets the Text on the Label. |

##### **Example :**

1. package com.techknow.lebel;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Label;
8. import java.awt.Rectangle;
9. import java.awt.SystemColor;
10. import java.awt.**event**.ActionEvent;
11. import java.awt.**event**.ActionListener;
13. **public** **class** LabelDemo extends Frame
14. {
15. **private** Label label1 = **new** Label();
16. **private** Label label2 = **new** Label();
17. **private** Button button1 = **new** Button();
18. **private** Button button2 = **new** Button();
19. **private** Button button3 = **new** Button();
21. **public** LabelDemo()
22. {
23. **try**
24. {
25. jbInit();
26. }
27. **catch** (Exception e)
28. {
29. e.printStackTrace();
30. }
31. }
32. **private** **void** jbInit() throws Exception
33. {
34. **this**.setLayout( **null** );
35. **this**.setSize(**new** Dimension(451, 300));
36. **this**.setBackground( SystemColor.control );
37. label1.setText("Label for Demonstration");
38. label1.setBounds(**new** Rectangle
39. (25, 45, 380, 35));
40. label1.setFont(**new** Font("Tahoma", 1, 14));
41. label1.setBackground(**new** java.awt.Color
42. (247, 255, 214));
43. label1.setForeground(java.awt.Color.red);
44. label2.setText("Set the alignment of Label");
45. label2.setBounds(**new** Rectangle
46. (65, 135, 280, 15));
47. label2.setFont(**new** Font("Tahoma", 1, 12));
48. button1.setLabel("Left");
49. button1.setBounds(**new** Rectangle
50. (60, 170, 70, 25));
51. button1.setFont(**new** Font("Tahoma", 1, 12));
52. button1.addActionListener(**new** ActionListener()
53. {
54. **public** **void** actionPerformed(ActionEvent e)
55. {
56. button1\_actionPerformed(e);
57. }
58. });
59. button2.setLabel("Center");
60. button2.setBounds(**new** Rectangle
61. (150, 170, 80, 25));
62. button2.setFont(**new** Font("Tahoma", 1, 12));
63. button2.addActionListener(**new** ActionListener()
64. {
65. **public** **void** actionPerformed(ActionEvent e)
66. {
67. button2\_actionPerformed(e);
68. }
69. });
70. button3.setLabel("Right");
71. button3.setBounds(**new** Rectangle
72. (255, 170, 70, 25));
73. button3.setFont(**new** Font("Tahoma", 1, 12));
74. button3.addActionListener
75. (**new** ActionListener()
76. {
77. **public** **void** actionPerformed
78. (ActionEvent e)
79. {
80. button3\_actionPerformed(e);
81. }
82. });
83. **this**.add(button3, **null**);
84. **this**.add(button2, **null**);
85. **this**.add(button1, **null**);
86. **this**.add(label2, **null**);
87. **this**.add(label1, **null**);
88. }
89. **private** **void** button1\_actionPerformed(ActionEvent e)
90. {
91. label1.setAlignment(0);
92. }
93. **private** **void** button2\_actionPerformed(ActionEvent e)
94. {
95. label1.setAlignment(1);
96. }
97. **private** **void** button3\_actionPerformed(ActionEvent e)
98. {
99. label1.setAlignment(2);
100. }
101. **public** **static** **void** main(String args[])
102. {
103. LabelDemo obj=**new** LabelDemo();
104. obj.setVisible(**true**);
105. obj.setLocation(150,150);
106. }
107. }

##### **Output :**

#### Textfield

**Package :** Java.awt   
  
**Class:** java.awt.Textfield   
  
**Note :**A Textfield used for taking input from user.

##### **Constructors of Textfield Class**

| **Name** | **Description** |
| --- | --- |
| TextField() | This constructor creates instance of a Textfield. |
| TextField(int columns) | This constructor creates instance of a Textfield with the given number of Columns |
| TextField(String text) | This constructor creates instance of a Textfield with the given text. |
| TextField(String text, int columns) | This constructor creates instance of a Textfield with the given text and number of columns. |

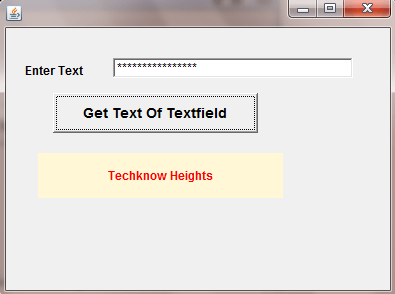
##### **Methods of Textfield Class**

| **Method** | **Description** |
| --- | --- |
| echoCharIsSet() | This method check the Textfield has character for echoing. |
| addActionListener(ActionListener l) | This method sets the actionListener on the Textfield. |
| getActionListeners() | This method gets the actionListener from Textfield. |
| getColumns() | This method gets the columns from Textfield. |
| getEchoChar() | This method gets the echoChar from Textfield. |
| removeActionListener(ActionListener l) | This method sets the actionListener on Textfield. |
| setColumns(int columns) | This method sets the columns on Textfield. |
| setEchoChar(char c) | This method sets the echoChar on Textfield. |
| setEchoCharacter(char c) | This method is the replacement of setEchoChar(char c) |
| setText(String t) | This method sets Text of Textfield. |

##### **Example :**

1. package com.techknow.textfields;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Label;
8. import java.awt.Rectangle;
9. import java.awt.SystemColor;
10. import java.awt.TextField;
11. import java.awt.**event**.ActionEvent;
12. import java.awt.**event**.ActionListener;
14. **public** **class** TextfieldDemo extends Frame
15. {
16. **private** TextField textField1 = **new** TextField();
17. **private** Button button1 = **new** Button();
18. **private** Label label1 = **new** Label();
19. **private** Label label2 = **new** Label();
21. **public** TextfieldDemo()
22. {
23. **try**
24. {
25. jbInit();
26. }
27. **catch** (Exception e)
28. {
29. e.printStackTrace();
30. }
31. }
32. **private** **void** jbInit() throws Exception
33. {
34. **this**.setLayout( **null** );
35. **this**.setSize( **new** Dimension( 400, 300 ) );
36. **this**.setBackground( SystemColor.control );
37. textField1.setBounds(**new** Rectangle
38. (115, 60, 240, 20));
39. textField1.setEchoChar('\*');
40. button1.setLabel("Get Text Of Textfield");
41. button1.setBounds(**new** Rectangle
42. (55, 95, 205, 40));
43. button1.setFont(**new** Font("Tahoma", 1, 14));
44. button1.addActionListener
45. (**new** ActionListener()
46. {
47. **public** **void** actionPerformed
48. (ActionEvent e)
49. {
50. button1\_actionPerformed(e);
51. }
52. });
53. label1.setBounds(**new** Rectangle
54. (40, 155, 245, 45));
55. label1.setFont(**new** Font("Tahoma", 1, 12));
56. label1.setAlignment(1);
57. label1.setBackground(**new** java.awt.Color
58. (255, 247, 214));
59. label1.setForeground(java.awt.Color.red);
60. label2.setText("Enter Text");
61. label2.setBounds(**new** Rectangle
62. (25, 65, 75, 15));
63. label2.setFont(**new** Font
64. ("Tahoma", 1, 12));
65. **this**.add(label2, **null**);
66. **this**.add(label1, **null**);
67. **this**.add(button1, **null**);
68. **this**.add(textField1, **null**);
69. }
70. **public** **static** **void** main(String args[])
71. {
72. TextfieldDemo obj=**new** TextfieldDemo();
73. obj.setVisible(**true**);
74. obj.setLocation(150,150);
75. }
76. **private** **void** button1\_actionPerformed
77. (ActionEvent e)
78. {
79. String str=textField1.getText();
80. label1.setText(str);
81. }
82. }

##### **Output :**



#### TextArea

**Package :** Java.awt   
  
**Class:** java.awt.TextArea   
  
**Note :**A TextArea used for taking input from user.TextArea have multiple line region to get input from user.

##### **Constructors of TextArea Class**

| **Name** | **Description** |
| --- | --- |
| TextArea() | This constructor creates instance of a TextArea. |
| TextArea(int rows, int columns) | This constructor creates instance of a Textarea with the given number of Columns and row. |
| TextArea(String text) | Constructs a new text area with the specified text. |
| TextArea(String text,int rows, int columns) | Constructs a new text area with the specified text and also row and column. |
| TextArea(String text, int rows,int columns, int scrollbars) | Constructs a new text area with the specified text,row and column and scrollbar. |

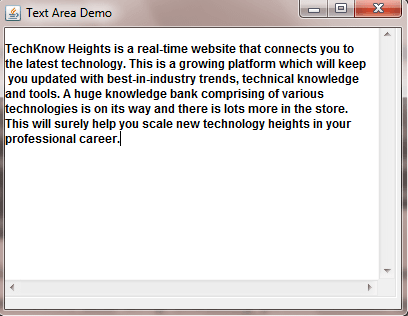
##### **Methods of TextArea Class**

| **Method** | **Description** |
| --- | --- |
| append(String str) | This method update the text in TextArea. |
| appendText(String str) | This method is replacement of append(String str). |
| getColumns() | This method gets the column count of TextArea. |
| getRows() | This method gets the rows from Textfield. |
| insert(String str, int pos) | This method insert the Text in TextArea at the specified position. |
| replaceRange(String str,int start, int end) | This method replace the String from start index to end index. |
| replaceText(String str,int start, int end) | This method replace the String from start index to end index. |
| setColumns(int columns) | This method sets the columns of Textarea. |
| setRows(int rows) | This method sets the rows of Textarea. |

##### **Example :**

1. package com.techknow.textarea;
3. import java.awt.Dimension;
4. import java.awt.Font;
5. import java.awt.Frame;
6. import java.awt.Rectangle;
7. import java.awt.SystemColor;
8. import java.awt.TextArea;
10. **public** **class** TextAreaDemo extends Frame
11. {
12. **private** TextArea textArea1 = **new** TextArea();
13. **public** TextAreaDemo()
14. {
15. **try**
16. {
17. jbInit();
18. }
19. **catch** (Exception e)
20. {
21. e.printStackTrace();
22. }
23. }
24. **private** **void** jbInit() throws Exception
25. {
26. **this**.setLayout( **null** );
27. **this**.setSize(**new** Dimension(412, 319));
28. **this**.setBackground( SystemColor.control );
29. **this**.setTitle("Text Area Demo");
30. **this**.setFont(**new** Font("Dialog", 1, 12));
31. textArea1.setBounds(**new** Rectangle
32. (5, 25, 395, 275));
33. **this**.add(textArea1, **null**);
34. }
35. **public** **static** **void** main(String args[])
36. {
37. TextAreaDemo obj=**new** TextAreaDemo();
38. obj.setVisible(**true**);
39. obj.setLocation(150,150);
40. }
41. }

##### **Output :**



#### Button

**Package :** Java.awt   
  
**Class:** java.awt.Button   
  
**Note :** Button class use to perform action when the Button is clicked. Button has Text as well as an image.

##### **Constructors of Button Class**

| **Name** | **Description** |
| --- | --- |
| Button() | This constructor creates instance of a Button. |
| Button(String Text) | This constructor creates instance of a Button with the specified text. |

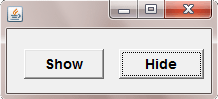
##### **Methods of Button Class**

| **Method** | **Description** |
| --- | --- |
| setLabel(String label) | This method sets the label at Button. |
| setActionCommand(String command) | This method perform a specific task which is set by this method. |
| removeActionListener(ActionListener l) | This method erase the Action from the Button. |
| getLabel() | This method gets the label from Button. |
| getActionListeners() | This method gets the actionListener from the Button. |
| getActionCommand() | This method gets the action command which set on the Button. |
| addActionListener(ActionListener l) | This method sets the actionListener on the Button. |
| addNotify() | This method creates the peer of Button. |

##### **Example :**

1. package com.techknow.buttonDemo;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.**event**.ActionEvent;
10. import java.awt.**event**.ActionListener;
12. **public** **class** ButtonDemo extends Frame
13. {
14. **private** Button button1 = **new** Button();
15. **private** Button button2 = **new** Button();
16. **public** ButtonDemo()
17. {
18. **try**
19. {
20. jbInit();
21. } **catch** (Exception e)
22. {
23. e.printStackTrace();
24. }
25. }
26. **private** **void** jbInit() throws Exception
27. {
28. **this**.setLayout( **null** );
29. **this**.setSize(**new** Dimension(247, 114));
30. **this**.setBackground( SystemColor.control );
31. **this**.setResizable(**false**);
32. button1.setLabel("Show");
33. button1.setBounds
34. (**new** Rectangle(25, 50, 80, 30));
35. button1.setFont(**new** Font("Tahoma", 1, 14));
36. button1.addActionListener(**new** ActionListener()
37. {
38. **public** **void** actionPerformed(ActionEvent e)
39. {
40. button1\_actionPerformed(e);
41. }
42. });
43. button2.setLabel("Hide");
44. button2.setBounds(**new** Rectangle
45. (120, 50, 85, 30));
46. button2.setFont(**new** Font("Tahoma", 1, 14));
47. button2.addActionListener(**new** ActionListener()
48. {
49. **public** **void** actionPerformed(ActionEvent e)
50. {
51. button2\_actionPerformed(e);
52. }
53. });
54. **this**.add(button2, **null**);
55. **this**.add(button1, **null**);
56. }
57. **public** **static** **void** main(String args[])
58. {
59. ButtonDemo obj=**new** ButtonDemo();
60. obj.setSize(220,102);
61. obj.setLocation(150,150);
62. obj.setVisible(**true**);
63. }
64. **private** **void** button1\_actionPerformed(ActionEvent e)
65. {
66. button1.setLabel("Hide");
67. button2.setLabel("Show");
68. }
69. **private** **void** button2\_actionPerformed(ActionEvent e)
70. {
71. button2.setLabel("Hide");
72. button1.setLabel("Show");
73. }
74. }

##### **Output :**



### Panel

**Package :** Java.awt   
  
**Class:** java.awt.Panel   
  
**Note :**Panel is the simplest container class. A panel provides space in which an application can attach any other component, including other panels. 

##### **Constructors of Panel Class**

| **Name** | **Description** |
| --- | --- |
| Panel() | This constructor creates panel . |
| Panel(LayoutManager layout) | This constructor creates panel with specified Layout. |

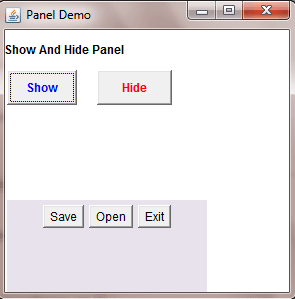
##### **Methods of Panel Class**

| **Method** | **Description** |
| --- | --- |
| getAccessibleContext() | This method Gets the AccessibleContext associated  with this Panel. |

##### **Example :**

1. package com.techknow.panel;
3. import java.awt.BorderLayout;
4. import java.awt.Button;
5. import java.awt.Color;
6. import java.awt.Font;
7. import java.awt.Frame;
8. import java.awt.Label;
9. import java.awt.Panel;
10. import java.awt.Rectangle;
11. import java.awt.**event**.ActionEvent;
12. import java.awt.**event**.ActionListener;
14. **public** **class** PanelDemo extends Frame
15. implements ActionListener
16. {
17. **private** Panel panel1 = **new** Panel();
18. **private** Panel panel2 = **new** Panel();
19. Button ok = **new** Button("Show");
20. Button cancel = **new** Button("Hide");
21. Button save = **new** Button("Save");
22. Button open = **new** Button("Open");
23. Button exit = **new** Button("Exit");
24. **private** Label label1 = **new** Label();
25. **public** PanelDemo()
26. {
27. **try**
28. {
29. jbInit();
30. }
31. **catch** (Exception e)
32. {
33. e.printStackTrace();
34. }
35. }
36. **public** **static** **void** main(String[] args)
37. {
38. PanelDemo panelDemo = **new** PanelDemo();
39. panelDemo.setSize(300,300);
40. panelDemo.setVisible(**true**);
41. }
42. **private** **void** jbInit() throws Exception
43. {
44. **this**.setLayout(**null**);
45. **this**.setTitle("Panel Demo");
46. panel1.setBounds(**new** Rectangle
47. (0, 0, 395, 120));
48. panel1.setLayout(**null**);
49. panel1.add(label1, **null**);
50. panel1.add(ok);
51. panel1.add(cancel, **null**);
52. **this**.add(panel1);
53. panel1.setSize(200, 200);
54. ok.setBounds(10, 20, 50, 50);
55. cancel.setBounds(90, 20, 50, 50);
56. ok.setBounds(**new** Rectangle
57. (10, 70, 70, 35));
58. ok.setFont(**new** Font("Tahoma", 1, 12));
59. ok.setForeground(Color.blue);
60. ok.addActionListener(**this**);
61. cancel.setBounds(**new** Rectangle
62. (100, 70, 75, 35));
63. cancel.setFont(**new** Font("Tahoma", 1, 12));
64. cancel.setForeground(Color.red);
65. cancel.addActionListener
66. (**new** PanelDemo\_cancel\_actionAdapter(**this**));
67. exit.addActionListener(**this**);
68. label1.setText("Show And Hide Panel");
69. label1.setBounds(**new** Rectangle
70. (5, 35, 240, 30));
71. label1.setFont(**new** Font("Tahoma", 1, 12));
72. panel1.setLayout(**null**);
73. panel2.setBounds(**new** Rectangle
74. (10, 200, 395, 265));
75. panel2.setSize(200, 200);
76. panel2.setBackground(**new** Color
77. (232, 226, 237));
78. save.setBounds(10, 220, 50, 50);
79. open.setBounds(90, 220, 50, 50);
80. exit.setBounds(140, 220, 50, 50);
81. panel2.add(save);
82. panel2.add(open);
83. panel2.add(exit);
84. **this**.add(panel2);
85. panel2.setVisible(**false**);
86. }
87. **public** **void** actionPerformed(ActionEvent e)
88. {
89. panel2.setVisible(**true**);
90. }
91. **void** cancel\_actionPerformed(ActionEvent e)
92. {
93. panel2.setVisible(**false**);
94. }
95. **void** exit\_actionPerformed(ActionEvent e)
96. {
97. System.exit(0);
98. }
99. }
100. final **class** PanelDemo\_cancel\_actionAdapter
101. implements ActionListener
102. {
103. **private** PanelDemo adaptee;
104. PanelDemo\_cancel\_actionAdapter
105. (PanelDemo adaptee)
106. {
107. **this**.adaptee = adaptee;
108. }
109. **public** **void** actionPerformed(ActionEvent e)
110. {
111. adaptee.cancel\_actionPerformed(e);
112. }
113. }

##### **Output :**



#### Checkbox

**Package :** Java.awt   
  
**Class:** java.awt.Checkbox   
  
**Note :** A check box is a graphical component that can be in either an "on" (true) or "off" (false) state.

##### **Constructors of CheckBox Class**

| **Name** | **Description** |
| --- | --- |
| Checkbox() | This constructor creates instance of a CheckBox. |
| Checkbox(String label) | This constructor creates instance of a CheckBox with the particular Label. |
| Checkbox(String label, boolean state) | This constructor creates instance of a CheckBox with particular label of srting and sets the particular state. |
| Checkbox(String label, boolean state,CheckboxGroup group) | This constructor creates instance of a CheckBox with particular label of srting,sets the particular state and add it in a checkboxGroup. |

##### **Methods of CheckBox Class**

| **Method** | **Description** |
| --- | --- |
| addItemListener(ItemListener l) | This method sets the actionListener on the Checkbox. |
| getCheckboxGroup() | This method gets the CheckboxGroup of Checkbox. |
| getItemListeners() | This method gets the actionListener. |
| getLabel() | This method gets the Label of Checkbox. |
| getState() | This method gets the state of the Checkbox. |
| paramString() | Returns a string representing the state of this Checkbox. |
| removeItemListener(ItemListener l) | This method removes the specific itemListener. |
| setLabel(String label) | This method sets the label of Checkbox. |
| setState(boolean state) | This method sets the state of Checkbox. |

##### **Example :**

1. package com.techknow.checkbox;
3. import java.awt.Button;
4. import java.awt.Checkbox;
5. import java.awt.Dimension;
6. import java.awt.Font;
7. import java.awt.Frame;
8. import java.awt.Label;
9. import java.awt.Rectangle;
10. import java.awt.SystemColor;
11. import java.awt.**event**.ActionEvent;
12. import java.awt.**event**.ActionListener;
13. import java.awt.**event**.MouseAdapter;
14. import java.awt.**event**.MouseEvent;
15. import java.beans.PropertyChangeEvent;
16. import java.beans.PropertyChangeListener;
18. **public** **class** CheckBox extends Frame
19. {
20. **private** Checkbox checkbox1 = **new** Checkbox();
21. **private** Label label1 = **new** Label();
22. **private** Label label2 = **new** Label();
23. **private** boolean counter=**true**;
24. **public** CheckBox()
25. {
26. **try**
27. {
28. jbInit();
29. }
30. **catch** (Exception e)
31. {
32. e.printStackTrace();
33. }
34. }
35. **private** **void** jbInit() throws Exception
36. {
37. **this**.setLayout( **null** );
38. **this**.setSize(**new** Dimension(270, 202));
39. **this**.setBackground( SystemColor.control );
40. checkbox1.setLabel("Hello");
41. checkbox1.setBounds(**new** Rectangle
42. (85, 40, 105, 40));
43. checkbox1.setFont(**new** Font("Tahoma", 1, 14));
44. checkbox1.addMouseListener(**new** MouseAdapter()
45. {
46. **public** **void** mouseClicked(MouseEvent e)
47. {
48. checkbox1\_mouseClicked(e);
49. }
50. });
51. label1.setBounds(**new** Rectangle
52. (0, 100, 230, 70));
53. label1.setAlignment(1);
54. label1.setFont(**new** Font("Tahoma", 1, 14));
55. label2.setText("Check any Checkbox");
56. label2.setBounds(**new** Rectangle(35, 5, 170, 25));
57. label2.setFont(**new** Font("Tahoma", 1, 14));
58. label2.setAlignment(1);
59. **this**.add(label2, **null**);
60. **this**.add(label1, **null**);
61. **this**.add(checkbox1, **null**);
62. }
63. **public** **static** **void** main(String args[])
64. {
65. CheckBox obj=**new** CheckBox();
66. obj.setSize(200,200);
67. obj.setVisible(**true**);
68. obj.setLocation(150,150);
69. }
70. **private** **void** checkbox1\_mouseClicked(MouseEvent e)
71. {
72. **if**(counter)
73. {
74. String str=checkbox1.getLabel();
75. label1.setText(" "+str);
76. counter=**false**;
77. }
78. **else**
79. {
80. String str=checkbox1.getLabel();
81. label1.setText(" ");
82. counter=**true**;
83. }
84. }
85. }

##### **Output :**



#### FileDialog

**Package :** Java.awt   
  
**Class:** java.awt.FileDialog   
  
**Note :**The FileDialog class displays a dialog window from which the user can select a file.

##### **Constructors of FileDialog Class**

| **Name** | **Description** |
| --- | --- |
| FileDialog(Dialog parent) | This constructor creates instance of a FileDialog to load File. |
| FileDialog(Dialog parent, String title) | This constructor creates instance of a FileDialog to load file with a appropriate Label. |
| FileDialog(Dialog parent,String title, int mode) | This constructor creates instance of a FileDialog in a specific Parent Container with a appropriate Label and the specific mode to load or save file. |
| FileDialog(Frame parent) | This constructor creates instance of a FileDialog in a specific Parent Container. |
| FileDialog(Frame parent,String title) | This constructor creates instance of a FileDialog in a specific Parent Container with the particular label. |
| FileDialog(Frame parent,String title, int mode ) | This constructor creates instance of a FileDialog in a specific Parent Container, with label and mode to load or save file. |

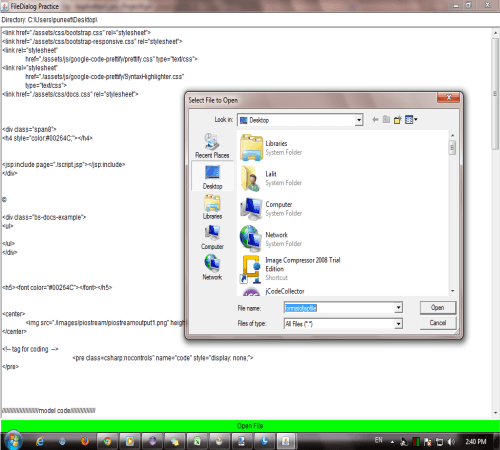
##### **Methods of FileDialog Class**

| **Method** | **Description** |
| --- | --- |
| getDirectory() | This method gets the Directory of File Dialog. |
| getFile() | This method gets the File. |
| getFilenameFilter() | This method shows the file dialog filename is filter. |
| getFiles() | This method returns the files which selected by user. |
| getMode() | This method returns the mode of File Dialog load or save. |
| isMultipleMode() | This method allow to select multiple files at a time. |
| setDirectory(String dir) | This method sets the directory of File Dialog. |
| setFile(String file) | This method sets the file. |
| setFilenameFilter(FilenameFilter filter) | This method sets the filter. |
| setMode(int mode) | This method sets the mode File Dialog. |

##### **Example :**

1. import java.io.\*;
2. import java.awt.\*;
3. import java.awt.**event**.\*;
5. **public** **class** FileDialogDemo extends Frame
6. implements ActionListener
7. {
8. FileDialog fd1;
9. Button openPlease;
10. Label lab1;
11. TextArea ta1;
12. **public** FileDialogDemo()
13. {
14. fd1 = **new** FileDialog
15. (**this**, "Select File to Open");
16. openPlease = **new** Button("Open File");
17. openPlease.setBackground(Color.green);
18. lab1 = **new** Label
19. ("Complete path of the selected file");
20. ta1 = **new** TextArea(40, 20);
21. add(openPlease, "South");
22. add(ta1, "Center");
23. add(lab1, "North");
24. openPlease.addActionListener(**this**);
25. setTitle("FileDialog Practice");
26. setSize(525, 325);
27. setVisible(**true**);
28. // a shortcut to close the frame
29. addWindowListener(**new** WindowAdapter()
30. {
31. **public** **void** windowClosing
32. (WindowEvent e)
33. {
34. System.exit(0);
35. }
36. });
37. }
38. **public** **void** actionPerformed(ActionEvent e)
39. {
40. fd1.setVisible(**true**);
41. lab1.setText("Directory: " +
42. fd1.getDirectory());
43. display(fd1.getDirectory() +
44. fd1.getFile());
45. }
46. **public** **void** display(String fname)
47. {
48. // this method is for reading a file
49. **try**
50. {
51. FileInputStream fis1 = **new**
52. FileInputStream(fname);
53. **int** fileSize = fis1.available();
54. **byte** buf1[] = **new** **byte**[fileSize];
55. fis1.read(buf1);
56. String str1 = **new** String(buf1);
57. ta1.setText(str1);
58. }
59. **catch**(IOException e)
60. {
61. System.exit(0);
62. }
63. }
64. **public** **static** **void** main(String args[])
65. {
66. **new** FileDialogDemo();
67. }
68. }

##### **Output :**



#### Print Dialog

**Package :** Java.awt   
  
**Class:** java.awt.PrintDialog   
  
**Note :**Class used to handle the "Print" command.

* The PrintDialog used to get print information rather than soft copy.
* Print Dialog is mostly used Dialog box in comparison others .
* Most AWT application have Print Dialog to make them Standard.

##### **Constructors of Textfield Class**

| **Name** | **Description** |
| --- | --- |
| PrintDialog(Printable) | This constructor creates instance of a PrintDialog. |

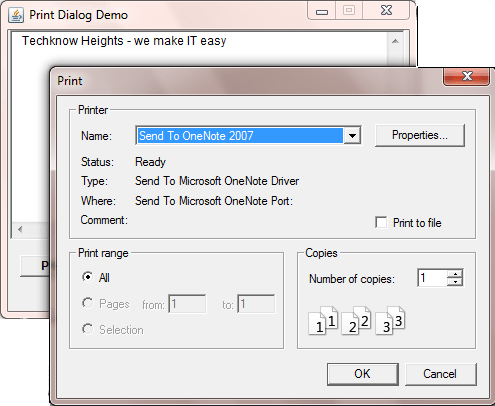
##### **Methods of Print Dialog Class**

| **Method** | **Description** |
| --- | --- |
| doPrint(String text, String  printCommand, String printFile) | This method executes the print command. |
| actionPerformed(ActionEvent e) | This method perform a specific event. |

##### **Example :**

1. package com.techknow.printdialog;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.TextArea;
10. import java.awt.**event**.ActionEvent;
11. import java.awt.**event**.ActionListener;
12. import java.awt.print.PrinterJob;
13. import javax.print.PrintService;
15. **public** **class** PrintDialogDemo extends Frame
16. {
17. **private** Button button1 = **new** Button();
18. **private** TextArea textArea1 = **new** TextArea();
19. **public** PrintDialogDemo()
20. {
21. **try**
22. {
23. jbInit();
24. }
25. **catch** (Exception e)
26. {
27. e.printStackTrace();
28. }
29. }
30. **private** **void** jbInit() throws Exception
31. {
32. **this**.setLayout( **null** );
33. **this**.setSize(**new** Dimension(407, 309));
34. **this**.setBackground( SystemColor.control );
35. **this**.setTitle("Print Dialog Demo");
36. **this**.setVisible(**true**);
37. **this**.setResizable(**false**);
38. **this**.setLocation(250,200);
39. button1.setLabel("Print");
40. button1.setBounds(**new** Rectangle
41. (15, 250, 70, 22));
42. button1.setFont(**new** Font("Tahoma", 1, 12));
43. button1.addActionListener(**new** ActionListener()
44. {
45. **public** **void** actionPerformed(ActionEvent e)
46. {
47. button1\_actionPerformed(e);
48. }
49. });
50. textArea1.setBounds(**new** Rectangle
51. (5, 25, 395, 210));
52. **this**.add(textArea1, **null**);
53. **this**.add(button1, **null**);
54. }
55. **public** **static** **void** main(String args[])
56. {
57. **new** PrintDialogDemo();
58. }
59. **private** **void** button1\_actionPerformed
60. (ActionEvent e)
61. {
62. // PrinterJob Class controls
63. // printing to a particular print
64. service (such **as** a printer or fax capability).
65. PrinterJob printJob = PrinterJob.getPrinterJob();
66. PrintService printer = printJob.getPrintService();
67. printJob.printDialog();
68. }
69. }

##### **Output :**



### Canvas

**Package :** Java.awt   
  
**Class:** java.awt.Canvas   
  
**Note :** A Canvas component represents a blank rectangular area of the screen onto which the application can draw or from which the application can trap input events from the user.

##### **Constructors of Canvas Class**

| **Name** | **Description** |
| --- | --- |
| Canvas() | This constructor creates instance of a Canvas. |
| Canvas(GraphicsConfiguration config) | This constructor creates instance of a Canvas with the given object of GraphicsConfiguration. |

##### **Methods of Canvas Class**

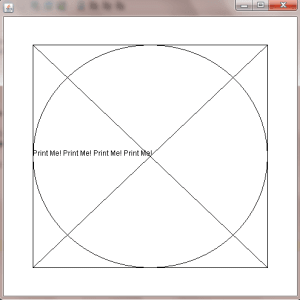
| **Method** | **Description** |
| --- | --- |
| getAccessibleContext() | This method gets the accessible context of Canvas. |
| getBufferStrategy() | This method gets the Buffer Strategy. |
| paint(Graphics g) | This method paint the Canvas |
| update(Graphics g) | update the Canvas. |
| createBufferStrategy(int numBuffers) | This method creates the BufferStrategy. |

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-components-canvas-menu-submenu-nestedmenu.jsp)

1. package com.techknow.canvas;
3. import java.awt.Canvas;
4. import java.awt.Graphics;
5. import java.awt.PrintJob;
6. import java.awt.Toolkit;
7. import java.util.Properties;
8. import java.awt.Dimension;
9. import java.awt.Frame;
10. import java.awt.SystemColor;
12. **public** **class** CanvasDemo extends Frame
13. {
14. MyCanvas canvas = **new** MyCanvas();
15. **public** CanvasDemo()
16. {
17. add("Center", canvas);
18. setSize(500, 500);
19. setVisible(**true**);
20. String name = "Test print job";
21. Properties properties = **new** Properties();
22. PrintJob pj = Toolkit.getDefaultToolkit()
23. .getPrintJob(CanvasDemo.**this**,
24. name, properties);
25. **if** (pj != **null**)
26. {
27. canvas.printAll(pj.getGraphics());
28. pj.end();
29. }
30. }
31. **private** **void** jbInit() throws Exception
32. {
33. **this**.setLayout( **null** );
34. **this**.setSize(**new** Dimension(390, 217));
35. **this**.setBackground( SystemColor.control );
36. }
37. **public** **static** **void** main(String args[])
38. {
39. CanvasDemo app = **new** CanvasDemo();
40. }
41. }
43. **class** MyCanvas extends Canvas
44. {
45. **public** **void** paint(Graphics g)
46. {
47. Dimension size = getSize();
48. **int** width = size.width;
49. **int** height = size.height;
50. **int** x1 = (**int**) (width \* 0.1);
51. **int** x2 = (**int**) (width \* 0.9);
52. **int** y1 = (**int**) (height \* 0.1);
53. **int** y2 = (**int**) (height \* 0.9);
54. g.drawRect(x1, y1, x2 - x1, y2 - y1);
55. g.drawOval(x1, y1, x2 - x1, y2 - y1);
56. g.drawLine(x1, y1, x2, y2);
57. g.drawLine(x2, y1, x1, y2);
58. String text = "Print Me! ";
59. text += text;
60. text += text;
61. g.drawString(text, x1, (**int**) ((y1 + y2) / 2));
62. g.dispose();
63. }
64. }

##### **Output :**



#### Menu

**Package :** Java.awt   
  
**Class:** java.awt.Menu   
  
**Note :**A Menu object is a drop-down menu component that is deployed from a menu bar.

##### **Constructors of Menu Class**

| **Name** | **Description** |
| --- | --- |
| Menu() | This constructor creates instance of a Menu. |
| Menu(String text) | This constructor creates instance of a Menu with the given label of string |
| Menu(String text, boolean tearOff) | This constructor creates instance of a Label with the given label of string and also check menu is turn off. |

##### **Methods of Menu Class**

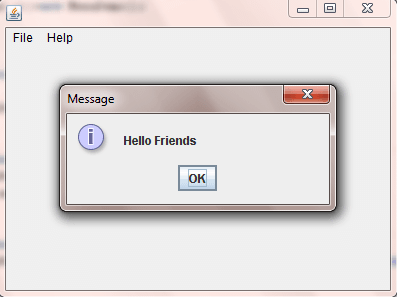
| **Method** | **Description** |
| --- | --- |
| add(MenuItem mi) | This method add menu item in Menu. |
| add(String label) | This method add string at menu. |
| addSeparator() | This method add separator between menu items. |
| getItem(int index) | This method gets the item from the given index. |
| getItemCount() | This method gets the number of item inside the menu. |
| insert(MenuItem menu item, int index) | This method inserts menu item inside menu. |
| insertSeparator(int index) | This method inserts separator between menu item. |
| remove(MenuComponent item) | This method remove menu item from menu. |
| remove(int index) | This method remove menu item from menu as per the index. |
| removeAll() | This method remove all menu item from menu. |

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-components-canvas-menu-submenu-nestedmenu.jsp)

1. package com.techknow.menu;
3. import java.awt.Dimension;
4. import java.awt.Frame;
5. import java.awt.Menu;
6. import java.awt.MenuBar;
7. import java.awt.MenuItem;
8. import java.awt.SystemColor;
9. import java.awt.**event**.ActionEvent;
10. import java.awt.**event**.ActionListener;
11. import javax.swing.JOptionPane;
13. **public** **class** MenuDemo extends Frame
14. {
15. **private** MenuBar menuBar1 = **new** MenuBar();
16. **private** Menu menu1 = **new** Menu("File");
17. **private** Menu menu2 = **new** Menu("Help");
18. **private** MenuItem hello=**new** MenuItem("Hello");
19. **private** MenuItem bye=**new** MenuItem("Bye");
20. **private** MenuItem aboutus=**new** MenuItem("About Us");
21. **public** MenuDemo()
22. {
23. **try**
24. {
25. jbInit();
26. }
27. **catch** (Exception e)
28. {
29. e.printStackTrace();
30. }
31. }
32. **private** **void** jbInit() throws Exception
33. {
34. **this**.setLayout( **null** );
35. **this**.setSize( **new** Dimension( 400, 300 ) );
36. **this**.setBackground( SystemColor.control );
37. **this**.setVisible(**true**);
38. **this**.setLocation(150,150);
39. **this**.setMenuBar(menuBar1);
40. menu1.setLabel("File");
41. hello.addActionListener(**new** ActionListener()
42. {
43. **public** **void** actionPerformed(ActionEvent e)
44. {
45. hello\_actionPerformed(e);
46. }
47. });
48. bye.addActionListener(**new** ActionListener()
49. {
50. **public** **void** actionPerformed(ActionEvent e)
51. {
52. bye\_actionPerformed(e);
53. }
54. });
55. aboutus.addActionListener(**new** ActionListener()
56. {
57. **public** **void** actionPerformed(ActionEvent e)
58. {
59. aboutus\_actionPerformed(e);
60. }
61. });
62. menu1.add(hello);
63. menu1.add(bye);
64. menu2.add(aboutus);
65. menuBar1.add(menu1);menuBar1.add(menu2);
66. }
67. **public** **static** **void** main(String args[])
68. {
69. MenuDemo obj=**new** MenuDemo();
70. }
71. **private** **void** hello\_actionPerformed(ActionEvent e)
72. {
73. JOptionPane.showMessageDialog
74. (**this**,"Hello Friends" );
75. }
76. **private** **void** bye\_actionPerformed(ActionEvent e)
77. {
78. JOptionPane.showMessageDialog
79. (**this**,"Bye Friends" );
80. System.exit(0);
81. }
82. **private** **void** aboutus\_actionPerformed(ActionEvent e)
83. {
84. JOptionPane.showMessageDialog
85. (**this**,"Hey Friends **this** **is** the
86. Techknow Heights Examples" );
87. }
88. }

##### **Output :**



#### Sub Menu

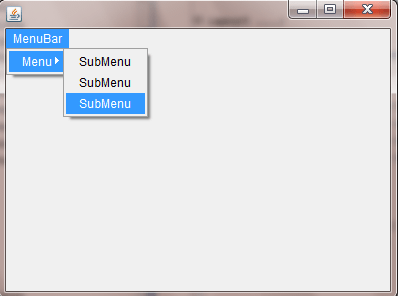
* SubMenu provides easy way of displaying menu items based on SubMenuModel.
* Each item in a menu must belong to the MenuItem class. It can be an instance of MenuItem, a submenu (an instance of Menu), or a check box (an instance of CheckboxMenuItem).
* In AWT creating a SubMenu is easy, you just add menu item to another menu item.

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-components-canvas-menu-submenu-nestedmenu.jsp)

1. package com.techknow.menu;
3. import java.awt.Dimension;
4. import java.awt.Frame;
5. import java.awt.Menu;
6. import java.awt.MenuBar;
7. import java.awt.MenuItem;
8. import java.awt.SystemColor;
10. **public** **class** SubMenuDemo extends Frame
11. {
12. **private** MenuBar menuBar1 = **new** MenuBar();
13. **private** Menu menu1 = **new** Menu("MenuBar");
14. **private** Menu menu2 = **new** Menu("Menu");
15. **private** MenuItem menu3 = **new** MenuItem("SubMenu");
16. **private** MenuItem menu4 = **new** MenuItem("SubMenu");
17. **private** MenuItem menu5 = **new** MenuItem("SubMenu");
18. **public** SubMenuDemo()
19. {
20. **try**
21. {
22. jbInit();
23. }
24. **catch** (Exception e)
25. {
26. e.printStackTrace();
27. }
28. }
29. **private** **void** jbInit() throws Exception
30. {
31. **this**.setLayout( **null** );
32. **this**.setSize( **new** Dimension( 400, 300 ) );
33. **this**.setBackground( SystemColor.control );
34. **this**.setLocation(150,150);
35. **this**.setVisible(**true**);
36. **this**.setMenuBar(menuBar1);
37. menuBar1.add(menu1);
38. menu1.add(menu2);
39. menu2.add(menu3);
40. menu2.add(menu4);
41. menu2.add(menu5);
42. }
43. **public** **static** **void** main(String args[])
44. {
45. SubMenuDemo obj=**new** SubMenuDemo();
46. }
47. }

##### **Output :**



#### Nested Menu

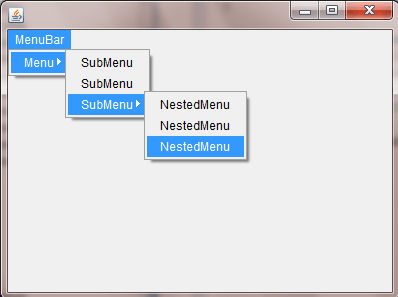
* NestedMenu provides easy way of displaying menu items based on NestedMenuModel.
* Each menu in a menu must belong to the Menu class.and futher the nested menu must be belong to the menuItem class .It can be an instance of MenuItem, a Nestedmenu (an instance of Menu), or a check box (an instance of CheckboxMenuItem).
* In AWT creating a NestedMenu is easy, you just add menu item to another menu item.

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-components-canvas-menu-submenu-nestedmenu.jsp)

1. package com.techknow.menu;
3. import java.awt.Dimension;
4. import java.awt.Frame;
5. import java.awt.Menu;
6. import java.awt.MenuBar;
7. import java.awt.MenuItem;
8. import java.awt.SystemColor;
10. **public** **class** NestedMenuDemo extends Frame
11. {
12. **private** MenuBar menuBar1 = **new** MenuBar();
13. **private** Menu menu1 = **new** Menu("MenuBar");
14. **private** Menu menu2 = **new** Menu("Menu");
15. **private** MenuItem menu3 = **new** MenuItem("SubMenu");
16. **private** MenuItem menu4 = **new** MenuItem("SubMenu");
17. **private** Menu menu5 = **new** Menu("SubMenu");
18. **private** MenuItem menu6 = **new** MenuItem("NestedMenu");
19. **private** MenuItem menu7 = **new** MenuItem("NestedMenu");
20. **private** MenuItem menu8 = **new** MenuItem("NestedMenu");
21. **public** NestedMenuDemo()
22. {
23. **try**
24. {
25. jbInit();
26. }
27. **catch** (Exception e)
28. {
29. e.printStackTrace();
30. }
31. }
32. **private** **void** jbInit() throws Exception
33. {
34. **this**.setLayout( **null** );
35. **this**.setSize( **new** Dimension( 400, 300 ) );
36. **this**.setBackground( SystemColor.control );
37. **this**.setLocation(150,150);
38. **this**.setVisible(**true**);
39. **this**.setMenuBar(menuBar1);
40. menuBar1.add(menu1);
41. menu1.add(menu2);
42. menu2.add(menu3);
43. menu2.add(menu4);
44. menu2.add(menu5);
45. menu5.add(menu6);
46. menu5.add(menu7);
47. menu5.add(menu8);
48. }
49. **public** **static** **void** main(String args[])
50. {
51. NestedMenuDemo obj=**new** NestedMenuDemo();
52. }
53. }

##### **Output :**



### Layouts in AWT

* The layout of components within a container remains a mystery.
* Why are the buttons placed side by side?
* Why are the buttons centered within the frame?

##### **Layout Managers**

* Layout manager classes are founded in the java.awt package same as the classes used for displaying graphical components.
* Every container has a default layout manager that determines the sizes and positions of components within the container.
* One reason that Java uses layout managers is so that containers can be resized gracefully.

##### **Layout Manager Classes**

* The java.awt package provides five layout manager classes

| **Name** | **Description** |
| --- | --- |
| BorderLayout | Arranges components along the sides of the container and in the middle. |
| CardLayout | Arrange components in "cards" Only one card is visible at a time. |
| FlowLayout | Arranges components in variable-length rows. |
| GridBagLayout | Aligns components horizontally and vertically; components can be of different sizes. |
| GridLayout | Arranges components in fixed-length rows and columns. |

#### Border Layout

**Package :** Java.awt   
  
**Class:** java.awt.BorderLayout   
  
**Note :**The BorderLayout layout manager can handle up to five components.

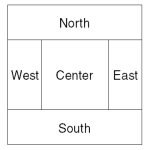
##### **Constructors of BorderLayout Class**

| **Name** | **Description** |
| --- | --- |
| BorderLayout() | This constructor creates new BorderLayout without any space between Components. |
| BorderLayout(int hgap, int vgap) | This constructor creates new BorderLayout with horizontal and Vertical space between Components. |

##### **Methods of BorderLayout Class**

| **Method** | **Description** |
| --- | --- |
| addLayoutComponent(Component comp,Object constraints) | This method adds the specific component. |
| getHgap() | This method gets the Horizontal gap between components. |
| getVgap() | This method gets the Vertical gap between components. |
| setHgap(int hgap) | This method sets the Horizontal gap between components. |
| setVgap() | This method sets the Vertical gap between components. |

* Four of the components can be positioned against the sides of the container, with the fifth occupying the center of the container.
* The positions in a BorderLayout are named North, South, East, West, and Center:



* The North and South components are stretched to the width of the container.
* The West and East components are stretched vertically to fill the gap between North and South.
* The Center component expands in both directions to fill any remaining space.
* The no-arg version of the BorderLayout constructor leaves no space between components:

setLayout(new BorderLayout());

* A different constructor is used if space is needed between components:

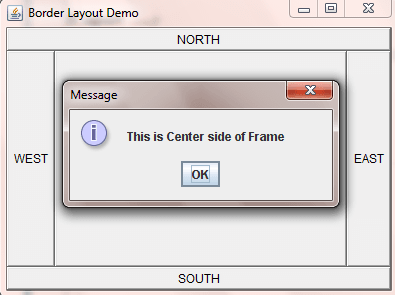
setLayout(new BorderLayout(20, 10));

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-layouts.jsp)

1. package com.techknow.layout;
2. import java.awt.BorderLayout;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Frame;
6. import java.awt.Rectangle;
7. import java.awt.SystemColor;
8. import java.awt.**event**.ActionEvent;
9. import java.awt.**event**.ActionListener;
10. import java.awt.**event**.WindowAdapter;
11. import java.awt.**event**.WindowEvent;
12. import javax.swing.JOptionPane;
14. **public** **class** BorderLayoutDemo extends Frame
15. {
16. **private** Button button1 = **new** Button();
17. **private** Button button2 = **new** Button();
18. **private** Button button3 = **new** Button();
19. **private** Button button4 = **new** Button();
20. **private** Button button5 = **new** Button();
21. **public** BorderLayoutDemo()
22. {
23. **try**
24. {
25. jbInit();
26. }
27. **catch** (Exception e)
28. {
29. e.printStackTrace();
30. }
31. }
32. **private** **void** jbInit() throws Exception
33. {
34. **this**.setLayout( **null** );
35. **this**.setSize( **new** Dimension( 400, 300 ) );
36. **this**.setBackground( SystemColor.control );
37. **this**.setLayout(**new** BorderLayout());
38. **this**.setVisible(**true**);
39. **this**.setTitle("Border Layout Demo");
40. **this**.addWindowListener(**new** WindowAdapter()
41. {
42. **public** **void** windowClosing(WindowEvent e)
43. {
44. this\_windowClosing(e);
45. }
46. });
47. **this**.setLocation(250,150);
48. button1.setLabel("SOUTH");
49. button1.setBounds(**new** Rectangle
50. (30, 55, 70, 22));
51. button1.addActionListener(**new** ActionListener()
52. {
53. **public** **void** actionPerformed(ActionEvent e)
54. {
55. button1\_actionPerformed(e);
56. }
57. });
58. button2.setLabel("NORTH");
59. button2.setBounds(**new** Rectangle
60. (240, 45, 70, 22));
61. button2.addActionListener(**new** ActionListener()
62. {
63. **public** **void** actionPerformed(ActionEvent e)
64. {
65. button2\_actionPerformed(e);
66. }
67. });
68. button3.setLabel("WEST");
69. button3.setBounds(**new** Rectangle
70. (185, 100, 70, 22));
71. button3.addActionListener(**new** ActionListener()
72. {
73. **public** **void** actionPerformed(ActionEvent e)
74. {
75. button3\_actionPerformed(e);
76. }
77. });
78. button4.setLabel("EAST");
79. button4.setBounds(**new** Rectangle
80. (215, 170, 70, 22));
81. button4.addActionListener(**new** ActionListener()
82. {
83. **public** **void** actionPerformed(ActionEvent e)
84. {
85. button4\_actionPerformed(e);
86. }
87. });
88. button5.setLabel("CENTER");
89. button5.setBounds(**new** Rectangle
90. (130, 200, 70, 22));
91. button5.addActionListener(**new** ActionListener()
92. {
93. **public** **void** actionPerformed(ActionEvent e)
94. {
95. button5\_actionPerformed(e);
96. }
97. });
98. **this**.add(button5, BorderLayout.CENTER);
99. **this**.add(button4, BorderLayout.EAST);
100. **this**.add(button3, BorderLayout.WEST);
101. **this**.add(button2, BorderLayout.NORTH);
102. **this**.add(button1, BorderLayout.SOUTH);
103. }
104. **public** **static** **void** main(String args[])
105. {
106. **new** BorderLayoutDemo();
107. }
108. **private** **void** button5\_actionPerformed(ActionEvent e)
109. {
110. JOptionPane.showMessageDialog
111. (**this**,"This is Center side of Frame");
112. }
113. **private** **void** button4\_actionPerformed(ActionEvent e)
114. {
115. JOptionPane.showMessageDialog
116. (**this**,"This is East side of Frame");
117. }
118. **private** **void** button3\_actionPerformed(ActionEvent e)
119. {
120. JOptionPane.showMessageDialog
121. (**this**,"This is West side of Frame");
122. }
123. **private** **void** button2\_actionPerformed(ActionEvent e)
124. {
125. JOptionPane.showMessageDialog
126. (**this**,"This is North side of Frame");
127. }
128. **private** **void** button1\_actionPerformed(ActionEvent e)
129. {
130. JOptionPane.showMessageDialog
131. (**this**,"This is South side of Frame");
132. }
133. **private** **void** this\_windowClosing(WindowEvent e)
134. {
135. System.exit(0);
136. }
137. }

##### **Output :**



#### Card Layout

**Package :** Java.awt   
  
**Class:** java.awt.CardLayout   
  
**Note :**The CardLayout class lets you implement an area that contains different components at different times.

##### **Constructors of CardLayout Class**

| **Name** | **Description** |
| --- | --- |
| CardLayout() | This constructor creates new CardLayout without any space between Components. |
| CardLayout(int hgap, int vgap) | This constructor creates new CardLayout with horizontal and Vertical space between Components. |

##### **Methods of CardLayout Class**

| **Method** | **Description** |
| --- | --- |
| addLayoutComponent(Component comp,Object constraints) | This method adds the specific component. |
| getHgap() | This method gets the Horizontal gap between components. |
| getVgap() | This method gets the Vertical gap between components. |
| setHgap(int hgap) | This method sets the Horizontal gap between components. |
| setVgap() | This method sets the Vertical gap between components. |
| first(Container parent) | This method switch to first card of the Container. |
| last(Container parent) | This method switch to last card of the Container. |
| next(Container parent) | This method switch to next card of the Container. |

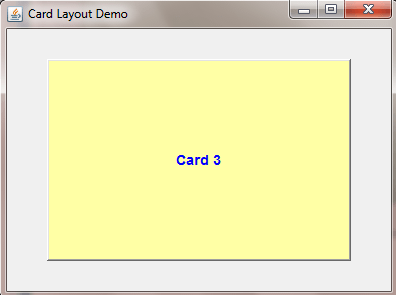
* A CardLayout is often controlled by a combo box, with the state of the combo box determining which panel (group of components) the CardLayout displays.
* The CardLayout class manages two or more components (usually JPanel instances) that share the same display space.

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-layouts.jsp)

1. package com.techknow.layout;
3. import java.awt.Button;
4. import java.awt.CardLayout;
5. import java.awt.Dimension;
6. import java.awt.Font;
7. import java.awt.Frame;
8. import java.awt.Rectangle;
9. import java.awt.SystemColor;
10. import java.awt.**event**.ActionEvent;
11. import java.awt.**event**.ActionListener;
12. import java.awt.**event**.WindowAdapter;
13. import java.awt.**event**.WindowEvent;
14. import javax.swing.JButton;
16. **public** **class** CardLayoutDemo extends Frame
17. {
18. **private** Button button1 = **new** Button();
19. **private** Button button2 = **new** Button();
20. **private** Button button3 = **new** Button();
21. CardLayout card=**new** CardLayout(40,30);
22. **public** CardLayoutDemo()
23. {
24. **try**
25. {
26. jbInit();
27. }
28. **catch** (Exception e)
29. {
30. e.printStackTrace();
31. }
32. }
33. **private** **void** jbInit() throws Exception
34. {
35. **this**.setLayout( **null** );
36. **this**.setSize( **new** Dimension( 400, 300 ) );
37. **this**.setBackground( SystemColor.control );
38. **this**.setLayout(card);
39. **this**.setVisible(**true**);
40. **this**.setTitle("Card Layout Demo");
41. **this**.addWindowListener(**new** WindowAdapter()
42. {
43. **public** **void** windowClosing(WindowEvent e)
44. {
45. this\_windowClosing(e);
46. }
47. });
48. **this**.setLocation(250,200);
49. button1.setLabel("Card 1");
50. button1.setFont(**new** Font("Tahoma", 1, 14));
51. button1.setForeground(java.awt.Color.red);
52. button1.setBackground
53. (**new** java.awt.Color(214, 255, 239));
54. button2.setBackground
55. (**new** java.awt.Color(214, 214, 255));
56. button3.setBackground
57. (**new** java.awt.Color(255, 255, 165));
58. button2.setFont(**new** Font
59. ("Tahoma", 1, 14));
60. button2.setForeground(java.awt.Color.green);
61. button3.setFont(**new** Font
62. ("Tahoma", 1, 14));
63. button3.setForeground(java.awt.Color.blue);
64. button1.addActionListener
65. (**new** ActionListener()
66. {
67. **public** **void** actionPerformed(ActionEvent e)
68. {
69. button1\_actionPerformed(e);
70. }
71. });
72. button2.addActionListener(**new** ActionListener()
73. {
74. **public** **void** actionPerformed(ActionEvent e)
75. {
76. button2\_actionPerformed(e);
77. }
78. });
79. button3.addActionListener(**new** ActionListener()
80. {
81. **public** **void** actionPerformed(ActionEvent e)
82. {
83. button3\_actionPerformed(e);
84. }
85. });
86. button2.setLabel("Card 2");
87. button3.setLabel("Card 3");
88. **this**.add(button1, "a");
89. **this**.add(button2, "b");
90. **this**.add(button3, "c");
91. }
92. **public** **static** **void** main(String args[])
93. {
94. **new** CardLayoutDemo();
95. }
96. **private** **void** this\_windowClosing(WindowEvent e)
97. {
98. System.exit(0);
99. }
100. **private** **void** button1\_actionPerformed
101. (ActionEvent e)
102. {
103. card.next(**this**);
104. }
105. **private** **void** button2\_actionPerformed
106. (ActionEvent e)
107. {
108. card.next(**this**);
109. }
110. **private** **void** button3\_actionPerformed
111. (ActionEvent e)
112. {
113. card.next(**this**);
114. }
115. }

##### **Output :**



#### Flow Layout

**Package :** Java.awt   
  
**Class:** java.awt.FlowLayout   
  
**Note :**The FlowLayout class lets you implement an area that contains different components at different times.

##### **Constructors of FlowLayout Class**

| **Name** | **Description** |
| --- | --- |
| FlowLayout() | This constructor creates new FlowLayout with centered alignment and 5-unit Horizontal and Vertical gap. |
| FlowLayout(int align) | This constructor creates new FlowLayout with specific alignment and 5-unit Horizontal and Vertical gap. |
| FlowLayout(int align, int hgap, int vgap) | This constructor creates new FlowLayout with specific alignment and specific Horizontal and Vertical gap. |

##### **Methods of FlowLayout Class**

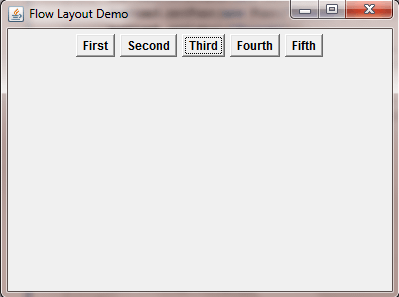
| **Method** | **Description** |
| --- | --- |
| addLayoutComponent(Component comp,Object constraints) | This method adds the specific component. |
| getHgap() | This method gets the Horizontal gap between components. |
| getVgap() | This method gets the Vertical gap between components. |
| setHgap(int hgap) | This method sets the Horizontal gap between components. |
| setVgap() | This method sets the Vertical gap between components. |
| getAlignment() | This method get alignment of the Container. |
| setAlignment(int align) | This method sets the alignment of the Container. |

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-layouts.jsp)

1. package com.techknow.layout;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.FlowLayout;
6. import java.awt.Font;
7. import java.awt.Frame;
8. import java.awt.Rectangle;
9. import java.awt.SystemColor;
10. import java.awt.**event**.ActionEvent;
11. import java.awt.**event**.ActionListener;
12. import java.awt.**event**.WindowAdapter;
13. import java.awt.**event**.WindowEvent;
15. **public** **class** FlowLayoutDemo extends Frame
16. {
17. **private** Button button1 = **new** Button();
18. **private** Button button2 = **new** Button();
19. **private** Button button3 = **new** Button();
20. **private** Button button4 = **new** Button();
21. **private** Button button5 = **new** Button();
22. FlowLayout flow=**new** FlowLayout(FlowLayout.CENTER);
23. **private** Button button6 = **new** Button();
24. **public** FlowLayoutDemo()
25. {
26. **try**
27. {
28. jbInit();
29. }
30. **catch** (Exception e)
31. {
32. e.printStackTrace();
33. }
34. }
35. **private** **void** jbInit() throws Exception
36. {
37. **this**.setLayout( flow );
38. **this**.setSize( **new** Dimension( 400, 300 ) );
39. **this**.setBackground( SystemColor.control );
40. **this**.setVisible(**true**);
41. **this**.setTitle("Flow Layout Demo");
42. **this**.addWindowListener(**new** WindowAdapter()
43. {
44. **public** **void** windowClosing(WindowEvent e)
45. {
46. this\_windowClosing(e);
47. }
48. });
49. **this**.setLocation(250,200);
50. button1.setLabel("First");
51. button1.setBounds(**new** Rectangle
52. (290, 45, 70, 22));
53. button1.setFont(**new** Font("Tahoma", 1, 12));
54. button2.setLabel("Second");
55. button2.setBounds(**new** Rectangle
56. (205, 75, 70, 22));
57. button2.setFont(**new** Font("Tahoma", 1, 12));
58. button3.setLabel("Third");
59. button3.setBounds(**new** Rectangle
60. (195, 115, 70, 22));
61. button3.setFont(**new** Font("Tahoma", 1, 12));
62. button4.setLabel("Fourth");
63. button4.setBounds(**new** Rectangle
64. (180, 145, 70, 22));
65. button4.setFont(**new** Font("Tahoma", 1, 12));
66. button5.setLabel("Fifth");
67. button5.setBounds(**new** Rectangle
68. (130, 160, 70, 22));
69. button5.setFont(**new** Font("Tahoma", 1, 12));
70. button6.setLabel("button6");
71. **this**.add(button1);
72. **this**.add(button2);
73. **this**.add(button3);
74. **this**.add(button4);
75. **this**.add(button5);
76. }
77. **public** **static** **void** main(String args[])
78. {
79. **new** FlowLayoutDemo();
80. }
81. **private** **void** this\_windowClosing(WindowEvent e)
82. {
83. System.exit(0);
84. }
85. }

##### **Output :**



#### GridBag Layout

**Package :** Java.awt   
  
**Class:** java.awt.GridBagLayout   
  
**Note :**GridBagLayout is one of the most flexible - and complex - layout managers the Java platform provides.

##### **Constructors of GridBagLayout Class**

| **Name** | **Description** |
| --- | --- |
| GridBagLayout() | This constructor creates new GridBag Layout. |

##### **Methods of GridBagLayout Class**

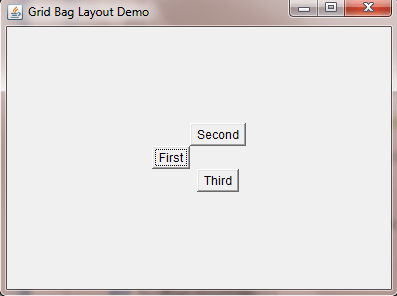
| **Method** | **Description** |
| --- | --- |
| addLayoutComponent(Component comp,Object constraints) | This method adds the specific component. |
| arrangeGrid(Container parent) | lays out the Grid. |
| getLayoutDimensions() | Find out the width and height of the cell. |
| location(int x, int y) | This method returns the location of cell in grid specified by specific x, y co-ordinates . |

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-layouts.jsp)

1. package com.techknow.layout;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Frame;
6. import java.awt.GridBagConstraints;
7. import java.awt.GridBagLayout;
8. import java.awt.Insets;
9. import java.awt.SystemColor;
10. import java.awt.**event**.WindowAdapter;
11. import java.awt.**event**.WindowEvent;
13. **public** **class** GridBagLayoutDemo extends Frame
14. {
15. GridBagLayout gridBag=**new** GridBagLayout();
16. **private** Button button1 = **new** Button();
17. **private** Button button2 = **new** Button();
18. **private** Button button3 = **new** Button();
19. **public** GridBagLayoutDemo()
20. {
21. **try**
22. {
23. jbInit();
24. }
25. **catch** (Exception e)
26. {
27. e.printStackTrace();
28. }
29. }
30. **private** **void** jbInit() throws Exception
31. {
32. **this**.setLayout( gridBag );
33. **this**.setVisible(**true**);
34. **this**.setLocation(250,200);
35. **this**.setSize( **new** Dimension( 400, 300 ) );
36. **this**.setBackground( SystemColor.control );
37. **this**.setTitle("Grid Bag Layout Demo");
38. **this**.addWindowListener(**new** WindowAdapter()
39. {
40. **public** **void** windowClosing(WindowEvent e)
41. {
42. this\_windowClosing(e);
43. }
44. });
45. button1.setLabel("First");
46. button2.setLabel("Second");
47. button3.setLabel("Third");
48. **this**.add(button1, **new**
49. GridBagConstraints(0, 1, 1, 1, 0.0, 0.0,
50. GridBagConstraints.CENTER,
51. GridBagConstraints.NONE,
52. **new** Insets(0, 0, 0, 0), 0, 0));
53. **this**.add(button2, **new**
54. GridBagConstraints(1, 0, 1, 1, 0.0, 0.0,
55. GridBagConstraints.CENTER,
56. GridBagConstraints.NONE,
57. **new** Insets(0, 0, 0, 0), 0, 0));
58. **this**.add(button3, **new**
59. GridBagConstraints(1, 2, 1, 1, 0.0, 0.0,
60. GridBagConstraints.CENTER,
61. GridBagConstraints.NONE,
62. **new** Insets(0, 0, 0, 0), 0, 0));
63. }
64. **private** **void** this\_windowClosing(WindowEvent e)
65. {
66. System.exit(0);
67. }
68. **public** **static** **void** main(String args[])
69. {
70. **new** GridBagLayoutDemo();
71. }
72. }

##### **Output :**



#### Grid Layout

**Package :** Java.awt   
  
**Class:** java.awt.GridLayout   
  
**Note :**The GridLayout class is a layout manager that lays out a container's components in a rectangular grid. 

##### **Constructors of GridBagLayout Class**

| **Name** | **Description** |
| --- | --- |
| GridLayout() | This constructor creates new Grid Layout. |
| GridLayout(int rows, int cols) | This constructor creates new Grid Layout with specific number of row and columns. |
| GridLayout(int rows, int cols,int hgap, int vgap) | This constructor creates new Grid Layout with specific number of row and columns with horizontal and vertical gap between components. |

##### **Methods of GridLayout Class**

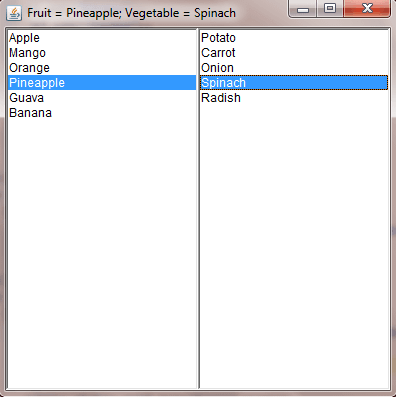
| **Method** | **Description** |
| --- | --- |
| addLayoutComponent(Component comp,Object constraints) | This method adds the specific component. |
| getColumns() | This method returns the columns. |
| getHgap() | This method returns the Horizontal gap between Components. |
| getRows() | This method returns the rows. |
| getVgap() | This method returns the Vertical gap between Components. |
| setColumns(int cols) | This method sets the specific columns. |
| setHgap(int hgap) | This method sets the specific Horizontal gap between Components. |
| setRows(int rows) | This method sets the specific rows. |
| setVgap(int vgap) | This method sets the specific Vertical gap between Components. |

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-layouts.jsp)

1. package com.techknow.layout;
3. import java.awt.\*;
4. import java.awt.**event**.\*;
6. **class** GridLayoutDemo extends Frame
7. {
8. List l1,l2;
9. **public** GridLayoutDemo()
10. {
11. // Set the frame properties
12. setTitle("GridLayout Demo");
13. setSize(400,400);
14. setLayout(**new** GridLayout());
15. setLocationRelativeTo(**null**);
16. setVisible(**true**);
17. // Create lists
18. l1=**new** List();
19. l2=**new** List();
20. // Add items to lists
21. l1.add("Apple");
22. l1.add("Mango");
23. l1.add("Orange");
24. l1.add("Pineapple");
25. l1.add("Guava");
26. l1.add("Banana");
27. l2.add("Potato");
28. l2.add("Carrot");
29. l2.add("Onion");
30. l2.add("Spinach");
31. l2.add("Radish");
32. // Add lists
33. add(l1);
34. add(l2);
35. // Add item listeners
36. l1.addItemListener(**new** ItemListener()
37. {
38. **public** **void** itemStateChanged(ItemEvent ie)
39. {
40. setTitle("Fruit =
41. "+l1.getSelectedItem()+";
42. Vegetable = "+l2.getSelectedItem());
43. }
44. });
45. l2.addItemListener(**new** ItemListener()
46. {
47. **public** **void** itemStateChanged(ItemEvent ie)
48. {
49. setTitle("Fruit =
50. "+l1.getSelectedItem()+";
51. Vegetable = "+l2.getSelectedItem());
52. }
53. });
54. }
55. **public** **static** **void** main(String args[])
56. {
57. **new** GridLayoutDemo();
58. }
59. }

##### **Output :**



### Key Listener

* The listener interface for receiving keyboard events which include (keypressed, keyReleased, KeyTyped).
* The class that is interested in processing a keyboard event either implements this interface (and all the methods it contains) or extends the abstract KeyAdapter class (overriding only the methods of interest).

##### **Methods of Key Listener Interface**

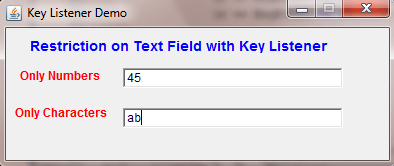
| **Method** | **Description** |
| --- | --- |
| keyPressed() | Invoke when Key pressed |
| keyReleased(KeyEvent e) | Invokes when Key Released. |
| keyTyped(KeyEvent e) | Invokes when Key Typed. |

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-listener.jsp)

1. package com.techknow.keylistener;
3. import java.awt.Dimension;
4. import java.awt.Font;
5. import java.awt.Frame;
6. import java.awt.Label;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.TextField;
10. import java.awt.**event**.KeyAdapter;
11. import java.awt.**event**.KeyEvent;
12. import javax.swing.JOptionPane;
14. **public** **class** KeyListnerDemo extends Frame
15. {
16. **private** TextField textField1 = **new** TextField();
17. **private** Label label1 = **new** Label();
18. **private** Label label2 = **new** Label();
19. **private** TextField textField2 = **new** TextField();
20. **private** Label label3 = **new** Label();
21. **public** KeyListnerDemo()
22. {
23. **try**
24. {
25. jbInit();
26. }
27. **catch** (Exception e)
28. {
29. e.printStackTrace();
30. }
31. }
32. **private** **void** jbInit() throws Exception
33. {
34. **this**.setLayout( **null** );
35. **this**.setSize(**new** Dimension(389, 160));
36. **this**.setBackground( SystemColor.control );
37. **this**.setResizable(**false**);
38. **this**.setTitle("Key Listener Demo");
39. **this**.setVisible(**true**);
40. **this**.setLocation(150,150);
41. textField1.setBounds(**new** Rectangle
42. (120, 65, 220, 20));
43. textField1.addKeyListener(**new** KeyAdapter()
44. {
45. **public** **void** keyPressed(KeyEvent e)
46. {
47. textField1\_keyPressed(e);
48. }
49. });
50. label1.setText("Only Numbers");
51. label1.setBounds(**new** Rectangle
52. (15, 60, 100, 25));
53. label1.setFont(**new** Font("Tahoma", 1, 12));
54. label1.setForeground(java.awt.Color.red);
55. label2.setText("Only Characters");
56. label2.setBounds(**new** Rectangle
57. (10, 100, 105, 20));
58. label2.setFont(**new** Font("Tahoma", 1, 12));
59. label2.setForeground(java.awt.Color.red);
60. textField2.setBounds(**new** Rectangle
61. (120, 105, 220, 20));
62. textField2.addKeyListener(**new** KeyAdapter()
63. {
64. **public** **void** keyTyped(KeyEvent e)
65. {
66. textField2\_keyTyped(e);
67. }
68. });
69. label3.setText
70. ("Restriction on Text Field with Key Listener");
71. label3.setBounds(**new** Rectangle(25, 35, 315, 15));
72. label3.setFont(**new** Font("Tahoma", 1, 14));
73. label3.setForeground(java.awt.Color.blue);
74. **this**.add(label3, **null**);
75. **this**.add(textField2, **null**);
76. **this**.add(label2, **null**);
77. **this**.add(label1, **null**);
78. **this**.add(textField1, **null**);
79. }
80. **private** **void** textField1\_keyPressed(KeyEvent e)
81. {
82. **char** c = e.getKeyChar();
83. **if** (!(Character.isDigit(c) ||
84. (c == KeyEvent.VK\_BACK\_SPACE) ||
85. (c == KeyEvent.VK\_DELETE)))
86. {
87. e.consume();
88. }
89. }
90. **private** **void** textField2\_keyTyped(KeyEvent e)
91. {
92. **char** c = e.getKeyChar();
93. **if** ((Character.isDigit(c) ||
94. (c == KeyEvent.VK\_BACK\_SPACE) ||
95. (c == KeyEvent.VK\_DELETE)))
96. {
97. e.consume();
98. }
99. }
100. **public** **static** **void** main(String args[])
101. {
102. **new** KeyListnerDemo();
103. }
104. }

##### **Output :**



#### Mouse Listener

* The listener interface for receiving "interesting" mouse events (press, release, click, enter, and exit) on a component. (To track mouse moves and mouse drags, use the MouseMotionListener.)
* A mouse event is also generated when the mouse cursor enters or leaves a component. When a mouse event occurs, the relevant method in the listener object is invoked, and the MouseEvent is passed to it.

##### **Methods of Mouse Listener Interface**

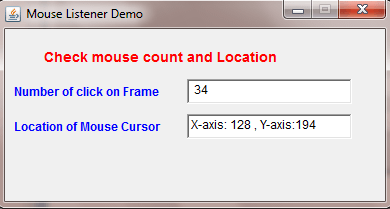
| **Method** | **Description** |
| --- | --- |
| mouseClicked(MouseEvent e) | Invokes when Mouse Click. |
| mouseEntered(MouseEvent e) | Invokes when Mouse Entered. |
| mouseExited(MouseEvent e) | Invokes when Mouse Exited. |
| mousePressed(MouseEvent e) | Invokes when Mouse Pressed. |
| mouseReleased(MouseEvent e) | Invokes when Mouse Released. |

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-listener.jsp)

1. package com.techknow.mouselistener;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Label;
8. import java.awt.Rectangle;
9. import java.awt.SystemColor;
10. import java.awt.TextField;
11. import java.awt.**event**.MouseAdapter;
12. import java.awt.**event**.MouseEvent;
14. **public** **class** MouseListenerDemo extends Frame
15. {
16. **private** **int** count=0;
17. **private** TextField textField1 = **new** TextField();
18. **private** Label label1 = **new** Label();
19. **private** Label label2 = **new** Label();
20. **private** Label label3 = **new** Label();
21. **private** TextField textField2 = **new** TextField();
22. **public** MouseListenerDemo()
23. {
24. **try**
25. {
26. jbInit();
27. }
28. **catch** (Exception e)
29. {
30. e.printStackTrace();
31. }
32. }
33. **private** **void** jbInit() throws Exception
34. {
35. **this**.setLayout( **null** );
36. **this**.setVisible(**true**);
37. **this**.setLocation(150,150);
38. **this**.setSize(**new** Dimension(386, 200));
39. **this**.setBackground( SystemColor.control );
40. **this**.setResizable(**false**);
41. **this**.setTitle("Mouse Listener Demo");
42. **this**.addMouseListener(**new** MouseAdapter()
43. {
44. **public** **void** mouseClicked(MouseEvent e)
45. {
46. this\_mouseClicked(e);
47. }
48. });
49. textField1.setBounds(**new** Rectangle
50. (185, 75, 165, 25));
51. label1.setText("Number of click on Frame ");
52. label1.setBounds(**new** Rectangle
53. (10, 80, 170, 15));
54. label1.setFont(**new** Font("Tahoma", 1, 12));
55. label1.setForeground(java.awt.Color.blue);
56. label2.setText("Check mouse count and Location");
57. label2.setBounds(**new** Rectangle
58. (40, 45, 290, 15));
59. label2.setFont(**new** Font("Tahoma", 1, 14));
60. label2.setForeground(java.awt.Color.red);
61. label3.setText("Location of Mouse Cursor ");
62. label3.setBounds(**new** Rectangle
63. (10, 115, 170, 15));
64. label3.setForeground(java.awt.Color.blue);
65. label3.setFont(**new** Font("Tahoma", 1, 12));
66. textField2.setBounds(**new** Rectangle
67. (185, 110, 165, 25));
68. **this**.add(textField2, **null**);
69. **this**.add(label3, **null**);
70. **this**.add(label2, **null**);
71. **this**.add(label1, **null**);
72. **this**.add(textField1, **null**);
73. }
74. **private** **void** this\_mouseClicked(MouseEvent e)
75. {
76. **int** x=e.getX();
77. **int** y=e.getY();
78. count=count+1;
79. textField1.setText(" "+count);
80. textField2.setText("X-axis: "+ x + " , Y-axis:"+y);
81. }
82. **public** **static** **void** main(String args[])
83. {
84. **new** MouseListenerDemo();
85. }
86. }

##### **Output :**



#### Window Listener

* The listener interface for receiving window events.
* The listener object created from that class is then registered with a Window using the window's addWindowListener method.

##### **Methods of Window Listener Interface**

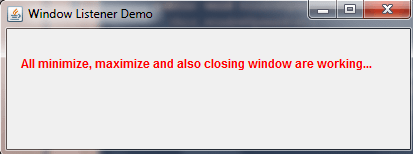
| **Method** | **Description** |
| --- | --- |
| windowActivated(WindowEvent e) | Invokes when Window Activated. |
| windowClosed(WindowEvent e) | Invokes when Window Closed. |
| windowClosing(WindowEvent e) | Invokes when Window going to close. |
| windowDeactivated(WindowEvent e) | Invokes when Window deactivated. |
| windowDeiconified(WindowEvent e) | Invokes when Window deiconified. |
| windowIconified(WindowEvent e) | Invokes when Window iconified. |
| windowOpened(WindowEvent e) | Invokes when Window opened. |

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-listener.jsp)

1. package com.techknow.windowlistener;
3. import java.awt.Dimension;
4. import java.awt.Font;
5. import java.awt.Frame;
6. import java.awt.Label;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.**event**.WindowAdapter;
10. import java.awt.**event**.WindowEvent;
11. import javax.swing.JOptionPane;
13. **public** **class** WindowListenerDemo extends Frame
14. {
15. **private** Label label1 = **new** Label();
16. **public** WindowListenerDemo()
17. {
18. **try**
19. {
20. jbInit();
21. }
22. **catch** (Exception e)
23. {
24. e.printStackTrace();
25. }
26. }
27. **private** **void** jbInit() throws Exception
28. {
29. **this**.setLayout( **null** );
30. **this**.setVisible(**true**);
31. **this**.setLocation(250,200);
32. **this**.setSize(**new** Dimension(419, 158));
33. **this**.setBackground( SystemColor.control );
34. **this**.setTitle("Window Listener Demo");
35. **this**.addWindowListener(**new** WindowAdapter()
36. {
37. **public** **void** windowClosing(WindowEvent e)
38. {
39. this\_windowClosing(e);
40. }
41. **public** **void** windowOpened(WindowEvent e)
42. {
43. this\_windowOpened(e);
44. }
45. });
46. label1.setText("All minimize, maximize
47. and also closing window are working...");
48. label1.setBounds(**new** Rectangle
49. (20, 0, 390, 130));
50. label1.setFont(**new** Font("Tahoma", 1, 12));
51. label1.setForeground(java.awt.Color.red);
52. **this**.add(label1, **null**);
53. }
54. **private** **void** this\_windowClosing(WindowEvent e)
55. {
56. System.exit(0);
57. }
58. **public** **static** **void** main(String args[])
59. {
60. **new** WindowListenerDemo();
61. }
62. **private** **void** this\_windowOpened(WindowEvent e)
63. {
64. JOptionPane.showMessageDialog
65. (**this**,"Welcome To Techknow Heights");
66. }
67. }

##### **Output :**



### Graphics Class

**Package :** Java.awt   
  
**Class:** java.awt.Graphics   
  
**Note :**The Graphics class is the abstract base class for all graphics contexts that allow an application to draw onto components that are realized on various devices, as well as onto off-screen images. 

##### **Constructors of Graphics Class**

| **Name** | **Description** |
| --- | --- |
| Graphics() | This constructor creates instance of Graphics class . |

##### **Methods of Panel Class**

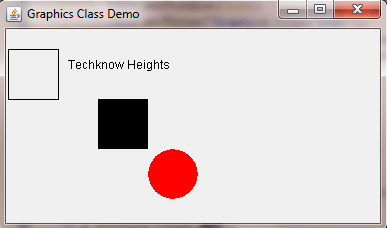
| **Method** | **Description** |
| --- | --- |
| create() | This method create the object of Graphics class which is the copy of another Graphics object. |
| draw3DRect(int x, int y, int width,  int height, boolean raised) | This method draw a 3-D outline of the rectangle. |
| drawArc(int x, int y, int width,  int height, int startAngle, int arcAngle) | This method draw outline of circular and elliptical rectangle. |
| drawBytes(byte[] data, int offset, int length, int x, int y) | This method draw the specified String at the specified position. |
| drawChars(char[] data, int offset, int length, int x, int y) | This method draw the character at specific position. |
| drawImage(Image img, int x, int y,  Color bgcolor, ImageObserver observer) | This method draw image at specific position. |
| drawLine(int x1, int y1, int x2,  int y2) | This method draw line with specified width and height. |
| drawOval(int x, int y, int width,  int height) | This method draw Oval with specified width and height. |
| drawPolygon(int[] xPoints,  int[] yPoints, int nPoints) | This method draw Polygon . |
| drawRect(int x, int y, int width,  int height) | This method draw Rectangle with specified width and height. |
| drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight) | This method draw Round Rectangle with specified width and height. |
| drawString (AttributedCharacterIteratoriterator, int x, int y) | This method draw String with specified width and height. |
| fill3DRect(int x, int y, int width,  int height, boolean raised) | This method draw a filled rectangle with specified width and height. |
| fillOval(int x, int y, int width, int height) | This method draw a filled Oval with specified width and height. |
| setColor(Color c) | This method sets the color. |
| setFont(Font font) | This method sets the font. |

##### **Example :**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/awt-graphics.jsp)

1. package com.techknow.graphics;
3. import java.awt.Color;
4. import java.awt.Dimension;
5. import java.awt.Frame;
6. import java.awt.Graphics;
7. import java.awt.SystemColor;
8. import java.awt.**event**.WindowAdapter;
9. import java.awt.**event**.WindowEvent;
11. **public** **class** GraphicsDemo extends Frame
12. {
13. **public** GraphicsDemo()
14. {
15. **try**
16. {
17. jbInit();
18. }
19. **catch** (Exception e)
20. {
21. e.printStackTrace();
22. }
23. }
24. **private** **void** jbInit() throws Exception
25. {
26. **this**.setLayout( **null** );
27. **this**.setSize(**new** Dimension(390, 232));
28. **this**.setBackground( SystemColor.control );
29. **this**.setVisible(**true**);
30. **this**.setTitle("Graphics Class Demo");
31. **this**.addWindowListener(**new** WindowAdapter()
32. {
33. **public** **void** windowClosing(WindowEvent e)
34. {
35. this\_windowClosing(e);
36. }
37. });
38. **this**.setLocation(250,200);
39. }
40. **public** **void** paint(Graphics g)
41. {
42. g.drawRect(10,50,50,50);
43. g.drawString("Techknow Heights ",70,70);
44. g.fillRect(100,100,50,50);
45. g.setColor(Color.RED);
46. g.fillOval(150,150,50,50);
47. }
48. **public** **static** **void** main(String args[])
49. {
50. **new** GraphicsDemo();
51. }
52. **private** **void** this\_windowClosing(WindowEvent e)
53. {
54. System.exit(0);
55. }
56. }

##### **Output :**



Examples:-

Label in AWT

**LabelDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. import java.awt.Button;
2. import java.awt.Dimension;
3. import java.awt.Font;
4. import java.awt.Frame;
5. import java.awt.Label;
6. import java.awt.Rectangle;
7. import java.awt.SystemColor;
8. import java.awt.**event**.ActionEvent;
9. import java.awt.**event**.ActionListener;
10. **public** **class** LabelDemo extends Frame
11. {
12. **private** Label label1 = **new** Label();
13. **private** Label label2 = **new** Label();
14. **private** Button button1 = **new** Button();
15. **private** Button button2 = **new** Button();
16. **private** Button button3 = **new** Button();
17. **public** LabelDemo()
18. {
19. **try**
20. {
21. jbInit();
22. }
23. **catch** (Exception e)
24. {
25. e.printStackTrace();
26. }
27. }
28. **private** **void** jbInit() throws Exception
29. {
30. **this**.setLayout( **null** );
31. **this**.setSize(**new** Dimension(451, 300));
32. **this**.setBackground( SystemColor.control );
33. label1.setText("Label for Demonstration");
34. label1.setBounds(**new** Rectangle(25, 45, 380, 35));
35. label1.setFont(**new** Font("Tahoma", 1, 14));
36. label1.setBackground(**new** java.awt.Color(247, 255, 214));
37. label1.setForeground(java.awt.Color.red);
38. label2.setText("Set the alignment of Label");
39. label2.setBounds(**new** Rectangle(65, 135, 280, 15));
40. label2.setFont(**new** Font("Tahoma", 1, 12));
41. button1.setLabel("Left");
42. button1.setBounds(**new** Rectangle(60, 170, 70, 25));
43. button1.setFont(**new** Font("Tahoma", 1, 12));
44. button1.addActionListener(
45. **new** ActionListener()
46. {
47. **public** **void** actionPerformed(ActionEvent e)
48. {
49. button1\_actionPerformed(e);
50. }
51. }
52. );
53. button2.setLabel("Center");
54. button2.setBounds(**new** Rectangle(150, 170, 80, 25));
55. button2.setFont(**new** Font("Tahoma", 1, 12));
56. button2.addActionListener(
57. **new** ActionListener()
58. {
59. **public** **void** actionPerformed(ActionEvent e)
60. {
61. button2\_actionPerformed(e);
62. }
63. }
64. );
65. button3.setLabel("Right");
66. button3.setBounds(**new** Rectangle(255, 170, 70, 25));
67. button3.setFont(**new** Font("Tahoma", 1, 12));
68. button3.addActionListener(
69. **new** ActionListener()
70. {
71. **public** **void** actionPerformed(ActionEvent e)
72. {
73. button3\_actionPerformed(e);
74. }
75. }
76. );
77. **this**.add(button3, **null**);
78. **this**.add(button2, **null**);
79. **this**.add(button1, **null**);
80. **this**.add(label2, **null**);
81. **this**.add(label1, **null**);
82. }
84. **private** **void** button1\_actionPerformed(ActionEvent e)
85. {
86. label1.setAlignment(0);
87. }
89. **private** **void** button2\_actionPerformed(ActionEvent e)
90. {
91. label1.setAlignment(1);
92. }
94. **private** **void** button3\_actionPerformed(ActionEvent e)
95. {
96. label1.setAlignment(2);
97. }
98. **public** **static** **void** main(String args[])
99. {
100. LabelDemo obj=**new** LabelDemo();
101. obj.setVisible(**true**);
102. obj.setLocation(150,150);
103. }
104. }

**Output**

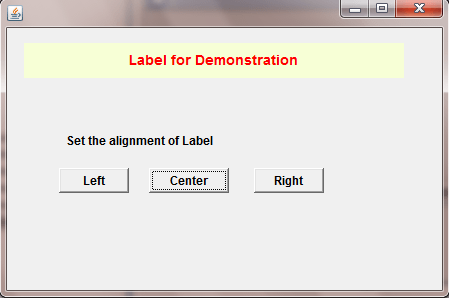
Label in AWT

**LabelDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. import java.awt.Button;
2. import java.awt.Dimension;
3. import java.awt.Font;
4. import java.awt.Frame;
5. import java.awt.Label;
6. import java.awt.Rectangle;
7. import java.awt.SystemColor;
8. import java.awt.**event**.ActionEvent;
9. import java.awt.**event**.ActionListener;
10. **public** **class** LabelDemo extends Frame
11. {
12. **private** Label label1 = **new** Label();
13. **private** Label label2 = **new** Label();
14. **private** Button button1 = **new** Button();
15. **private** Button button2 = **new** Button();
16. **private** Button button3 = **new** Button();
17. **public** LabelDemo()
18. {
19. **try**
20. {
21. jbInit();
22. }
23. **catch** (Exception e)
24. {
25. e.printStackTrace();
26. }
27. }
28. **private** **void** jbInit() throws Exception
29. {
30. **this**.setLayout( **null** );
31. **this**.setSize(**new** Dimension(451, 300));
32. **this**.setBackground( SystemColor.control );
33. label1.setText("Label for Demonstration");
34. label1.setBounds(**new** Rectangle(25, 45, 380, 35));
35. label1.setFont(**new** Font("Tahoma", 1, 14));
36. label1.setBackground(**new** java.awt.Color(247, 255, 214));
37. label1.setForeground(java.awt.Color.red);
38. label2.setText("Set the alignment of Label");
39. label2.setBounds(**new** Rectangle(65, 135, 280, 15));
40. label2.setFont(**new** Font("Tahoma", 1, 12));
41. button1.setLabel("Left");
42. button1.setBounds(**new** Rectangle(60, 170, 70, 25));
43. button1.setFont(**new** Font("Tahoma", 1, 12));
44. button1.addActionListener(
45. **new** ActionListener()
46. {
47. **public** **void** actionPerformed(ActionEvent e)
48. {
49. button1\_actionPerformed(e);
50. }
51. }
52. );
53. button2.setLabel("Center");
54. button2.setBounds(**new** Rectangle(150, 170, 80, 25));
55. button2.setFont(**new** Font("Tahoma", 1, 12));
56. button2.addActionListener(
57. **new** ActionListener()
58. {
59. **public** **void** actionPerformed(ActionEvent e)
60. {
61. button2\_actionPerformed(e);
62. }
63. }
64. );
65. button3.setLabel("Right");
66. button3.setBounds(**new** Rectangle(255, 170, 70, 25));
67. button3.setFont(**new** Font("Tahoma", 1, 12));
68. button3.addActionListener(
69. **new** ActionListener()
70. {
71. **public** **void** actionPerformed(ActionEvent e)
72. {
73. button3\_actionPerformed(e);
74. }
75. }
76. );
77. **this**.add(button3, **null**);
78. **this**.add(button2, **null**);
79. **this**.add(button1, **null**);
80. **this**.add(label2, **null**);
81. **this**.add(label1, **null**);
82. }
84. **private** **void** button1\_actionPerformed(ActionEvent e)
85. {
86. label1.setAlignment(0);
87. }
89. **private** **void** button2\_actionPerformed(ActionEvent e)
90. {
91. label1.setAlignment(1);
92. }
94. **private** **void** button3\_actionPerformed(ActionEvent e)
95. {
96. label1.setAlignment(2);
97. }
98. **public** **static** **void** main(String args[])
99. {
100. LabelDemo obj=**new** LabelDemo();
101. obj.setVisible(**true**);
102. obj.setLocation(150,150);
103. }
104. }

**Output**



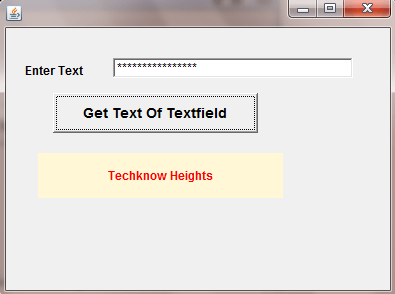
TextField in AWT

**TextfieldDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.textfields;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Label;
8. import java.awt.Rectangle;
9. import java.awt.SystemColor;
10. import java.awt.TextField;
11. import java.awt.**event**.ActionEvent;
12. import java.awt.**event**.ActionListener;
14. **public** **class** TextfieldDemo extends Frame
15. {
16. **private** TextField textField1 = **new** TextField();
17. **private** Button button1 = **new** Button();
18. **private** Label label1 = **new** Label();
19. **private** Label label2 = **new** Label();
21. **public** TextfieldDemo()
22. {
23. **try**
24. {
25. jbInit();
26. }
27. **catch** (Exception e)
28. {
29. e.printStackTrace();
30. }
31. }
33. **private** **void** jbInit() throws Exception
34. {
35. **this**.setLayout( **null** );
36. **this**.setSize( **new** Dimension( 400, 300 ) );
37. **this**.setBackground( SystemColor.control );
38. textField1.setBounds(**new** Rectangle(115, 60, 240, 20));
39. textField1.setEchoChar('\*');
40. button1.setLabel("Get Text Of Textfield");
41. button1.setBounds(**new** Rectangle(55, 95, 205, 40));
42. button1.setFont(**new** Font("Tahoma", 1, 14));
43. button1.addActionListener(
44. **new** ActionListener()
45. {
46. **public** **void** actionPerformed(ActionEvent e)
47. {
48. button1\_actionPerformed(e);
49. }
50. }
51. );
52. label1.setBounds(**new** Rectangle(40, 155, 245, 45));
53. label1.setFont(**new** Font("Tahoma", 1, 12));
54. label1.setAlignment(1);
55. label1.setBackground(**new** java.awt.Color(255, 247, 214));
56. label1.setForeground(java.awt.Color.red);
57. label2.setText("Enter Text");
58. label2.setBounds(**new** Rectangle(25, 65, 75, 15));
59. label2.setFont(**new** Font("Tahoma", 1, 12));
60. **this**.add(label2, **null**);
61. **this**.add(label1, **null**);
62. **this**.add(button1, **null**);
63. **this**.add(textField1, **null**);
64. }
65. **public** **static** **void** main(String args[])
66. {
67. TextfieldDemo obj=**new** TextfieldDemo();
68. obj.setVisible(**true**);
69. obj.setLocation(150,150);
70. }
72. **private** **void** button1\_actionPerformed(ActionEvent e)
73. {
74. String str=textField1.getText();
75. label1.setText(str);
76. }
77. }

**Output**



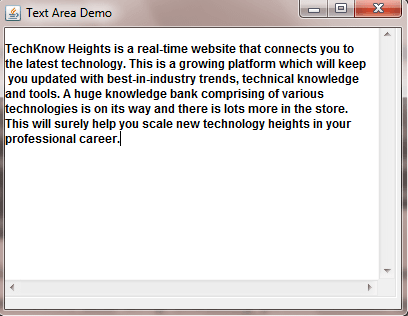
TextArea in AWT

**TextAreaDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.textarea;
3. import java.awt.Dimension;
4. import java.awt.Font;
5. import java.awt.Frame;
6. import java.awt.Rectangle;
7. import java.awt.SystemColor;
8. import java.awt.TextArea;
10. **public** **class** TextAreaDemo extends Frame
11. {
12. **private** TextArea textArea1 = **new** TextArea();
14. **public** TextAreaDemo()
15. {
16. **try**
17. {
18. jbInit();
19. }
20. **catch** (Exception e)
21. {
22. e.printStackTrace();
23. }
24. }
26. **private** **void** jbInit() throws Exception
27. {
28. **this**.setLayout( **null** );
29. **this**.setSize(**new** Dimension(412, 319));
30. **this**.setBackground( SystemColor.control );
31. **this**.setTitle("Text Area Demo");
32. **this**.setFont(**new** Font("Dialog", 1, 12));
33. textArea1.setBounds(**new** Rectangle(5, 25, 395, 275));
34. **this**.add(textArea1, **null**);
35. }
37. **public** **static** **void** main(String args[])
38. {
39. TextAreaDemo obj=**new** TextAreaDemo();
40. obj.setVisible(**true**);
41. obj.setLocation(150,150);
42. }
43. }

**Output**



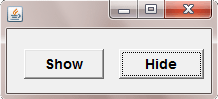
Button in AWT

**ButtonDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.buttonDemo;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.**event**.ActionEvent;
10. import java.awt.**event**.ActionListener;
12. **public** **class** ButtonDemo extends Frame
13. {
15. **private** Button button1 = **new** Button();
16. **private** Button button2 = **new** Button();
18. **public** ButtonDemo()
19. {
20. **try**
21. {
22. jbInit();
23. }
24. **catch** (Exception e)
25. {
26. e.printStackTrace();
27. }
28. }
30. **private** **void** jbInit() throws Exception
31. {
32. **this**.setLayout( **null** );
33. **this**.setSize(**new** Dimension(247, 114));
34. **this**.setBackground( SystemColor.control );
35. **this**.setResizable(**false**);
36. button1.setLabel("Show");
37. button1.setBounds(**new** Rectangle(25, 50, 80, 30));
38. button1.setFont(**new** Font("Tahoma", 1, 14));
39. button1.addActionListener(
40. **new** ActionListener()
41. {
42. **public** **void** actionPerformed(ActionEvent e)
43. {
44. button1\_actionPerformed(e);
45. }
46. }
47. );
48. button2.setLabel("Hide");
49. button2.setBounds(**new** Rectangle(120, 50, 85, 30));
50. button2.setFont(**new** Font("Tahoma", 1, 14));
51. button2.addActionListener(
52. **new** ActionListener()
53. {
54. **public** **void** actionPerformed(ActionEvent e)
55. {
56. button2\_actionPerformed(e);
57. }
58. }
59. );
60. **this**.add(button2, **null**);
61. **this**.add(button1, **null**);
62. }
64. **public** **static** **void** main(String args[])
65. {
66. ButtonDemo obj=**new** ButtonDemo();
67. obj.setSize(220,102);
68. obj.setLocation(150,150);
69. obj.setVisible(**true**);
70. }
72. **private** **void** button1\_actionPerformed(ActionEvent e)
73. {
74. button1.setLabel("Hide");
75. button2.setLabel("Show");
77. }
79. **private** **void** button2\_actionPerformed(ActionEvent e)
80. {
81. button2.setLabel("Hide");
82. button1.setLabel("Show");
84. }
85. }

**Output**



CheckBox in AWT

**CheckBoxDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.checkbox;
3. import java.awt.Button;
4. import java.awt.Checkbox;
5. import java.awt.Dimension;
6. import java.awt.Font;
7. import java.awt.Frame;
8. import java.awt.Label;
9. import java.awt.Rectangle;
10. import java.awt.SystemColor;
11. import java.awt.**event**.ActionEvent;
12. import java.awt.**event**.ActionListener;
14. import java.awt.**event**.MouseAdapter;
16. import java.awt.**event**.MouseEvent;
18. import java.beans.PropertyChangeEvent;
19. import java.beans.PropertyChangeListener;
21. **public** **class** CheckBox extends Frame
22. {
23. **private** Checkbox checkbox1 = **new** Checkbox();
24. **private** Label label1 = **new** Label();
25. **private** Label label2 = **new** Label();
26. **private** boolean counter=**true**;
28. **public** CheckBox()
29. {
30. **try**
31. {
32. jbInit();
33. }
34. **catch** (Exception e)
35. {
36. e.printStackTrace();
37. }
38. }
40. **private** **void** jbInit() throws Exception
41. {
42. **this**.setLayout( **null** );
43. **this**.setSize(**new** Dimension(270, 202));
44. **this**.setBackground( SystemColor.control );
45. checkbox1.setLabel("Hello");
46. checkbox1.setBounds(**new** Rectangle(85, 40, 105, 40));
47. checkbox1.setFont(**new** Font("Tahoma", 1, 14));
48. checkbox1.addMouseListener(
49. **new** MouseAdapter()
50. {
51. **public** **void** mouseClicked(MouseEvent e)
52. {
53. checkbox1\_mouseClicked(e);
54. }
55. });
56. label1.setBounds(**new** Rectangle(0, 100, 230, 70));
57. label1.setAlignment(1);
58. label1.setFont(**new** Font("Tahoma", 1, 14));
59. label2.setText("Check any Checkbox");
60. label2.setBounds(**new** Rectangle(35, 5, 170, 25));
61. label2.setFont(**new** Font("Tahoma", 1, 14));
62. label2.setAlignment(1);
63. **this**.add(label2, **null**);
64. **this**.add(label1, **null**);
65. **this**.add(checkbox1, **null**);
66. }
68. **public** **static** **void** main(String args[])
69. {
70. CheckBox obj=**new** CheckBox();
71. obj.setSize(200,200);
72. obj.setVisible(**true**);
73. obj.setLocation(150,150);
74. }
76. **private** **void** checkbox1\_mouseClicked(MouseEvent e)
77. {
78. **if**(counter)
79. {
80. String str=checkbox1.getLabel();
81. label1.setText(" "+str);
82. counter=**false**;
83. }
84. **else**
85. {
86. String str=checkbox1.getLabel();
87. label1.setText(" ");
88. counter=**true**;
89. }
90. }
91. }

**Output**



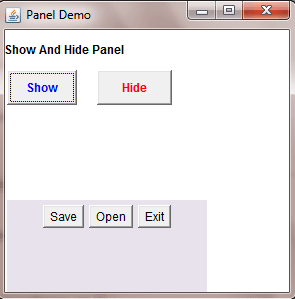
Panel in AWT

**PanelDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.panel;
3. import java.awt.BorderLayout;
4. import java.awt.Button;
5. import java.awt.Color;
6. import java.awt.Font;
7. import java.awt.Frame;
8. import java.awt.Label;
9. import java.awt.Panel;
10. import java.awt.Rectangle;
11. import java.awt.**event**.ActionEvent;
12. import java.awt.**event**.ActionListener;
14. **public** **class** PanelDemo extends Frame implements ActionListener
15. {
16. **private** Panel panel1 = **new** Panel();
17. **private** Panel panel2 = **new** Panel();
18. Button ok = **new** Button("Show");
19. Button cancel = **new** Button("Hide");
20. Button save = **new** Button("Save");
21. Button open = **new** Button("Open");
22. Button exit = **new** Button("Exit");
23. **private** Label label1 = **new** Label();
25. **public** PanelDemo()
26. {
27. **try**
28. {
29. jbInit();
30. }
31. **catch** (Exception e)
32. {
33. e.printStackTrace();
34. }
35. }
37. **public** **static** **void** main(String[] args)
38. {
39. PanelDemo panelDemo = **new** PanelDemo();
40. panelDemo.setSize(300,300);
41. panelDemo.setVisible(**true**);
42. }
44. **private** **void** jbInit() throws Exception
45. {
46. **this**.setLayout(**null**);
47. **this**.setTitle("Panel Demo");
48. panel1.setBounds(**new** Rectangle(0, 0, 395, 120));
49. panel1.setLayout(**null**);
50. panel1.add(label1, **null**);
51. panel1.add(ok);
52. panel1.add(cancel, **null**);
53. **this**.add(panel1);
54. panel1.setSize(200, 200);
55. ok.setBounds(10, 20, 50, 50);
56. cancel.setBounds(90, 20, 50, 50);
57. ok.setBounds(**new** Rectangle(10, 70, 70, 35));
58. ok.setFont(**new** Font("Tahoma", 1, 12));
59. ok.setForeground(Color.blue);
60. ok.addActionListener(**this**);
61. cancel.setBounds(**new** Rectangle(100, 70, 75, 35));
62. cancel.setFont(**new** Font("Tahoma", 1, 12));
63. cancel.setForeground(Color.red);
64. cancel.addActionListener(
65. **new** PanelDemo\_cancel\_actionAdapter(**this**)
66. );
67. exit.addActionListener(**this**);
68. label1.setText("Show And Hide Panel");
69. label1.setBounds(**new** Rectangle(5, 35, 240, 30));
70. label1.setFont(**new** Font("Tahoma", 1, 12));
71. panel1.setLayout(**null**);
73. panel2.setBounds(**new** Rectangle(10, 200, 395, 265));
74. panel2.setSize(200, 200);
75. panel2.setBackground(**new** Color(232, 226, 237));
76. save.setBounds(10, 220, 50, 50);
77. open.setBounds(90, 220, 50, 50);
78. exit.setBounds(140, 220, 50, 50);
79. panel2.add(save);
80. panel2.add(open);
81. panel2.add(exit);
82. **this**.add(panel2);
83. panel2.setVisible(**false**);
84. }
86. **public** **void** actionPerformed(ActionEvent e)
87. {
88. panel2.setVisible(**true**);
89. }
91. **void** cancel\_actionPerformed(ActionEvent e)
92. {
93. panel2.setVisible(**false**);
94. }
96. **void** exit\_actionPerformed(ActionEvent e)
97. {
98. System.exit(0);
99. }
100. }
102. final **class** PanelDemo\_cancel\_actionAdapter implements ActionListener
103. {
104. **private** PanelDemo adaptee;
106. PanelDemo\_cancel\_actionAdapter(PanelDemo adaptee)
107. {
108. **this**.adaptee = adaptee;
109. }
111. **public** **void** actionPerformed(ActionEvent e)
112. {
113. adaptee.cancel\_actionPerformed(e);
114. }
115. }

**Output**



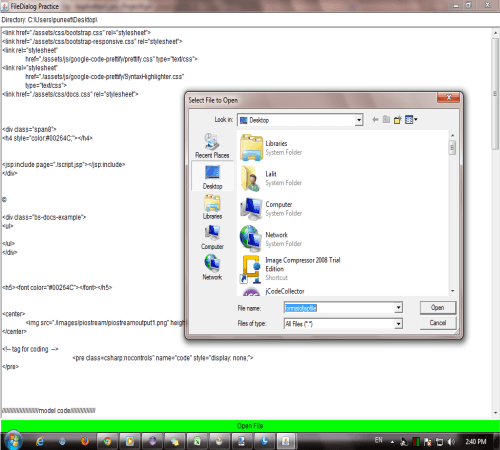
FileDialog in AWT

**FileDialogDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. import java.io.\*;
2. import java.awt.\*;
3. import java.awt.**event**.\*;
5. **public** **class** FileDialogDemo extends Frame implements ActionListener
6. {
7. FileDialog fd1;
8. Button openPlease;
9. Label lab1;
10. TextArea ta1;
11. **public** FileDialogDemo()
12. {
13. fd1 = **new** FileDialog(**this**, "Select File to Open");
14. openPlease = **new** Button("Open File");
15. openPlease.setBackground(Color.green);
16. lab1 = **new** Label("Complete path of the selected file");
17. ta1 = **new** TextArea(40, 20);
18. add(openPlease, "South");
19. add(ta1, "Center");
20. add(lab1, "North");
21. openPlease.addActionListener(**this**);
22. setTitle("FileDialog Practice");
23. setSize(525, 325);
24. setVisible(**true**);
26. // a shortcut to close the frame
27. addWindowListener(
28. **new** WindowAdapter()
29. {
30. **public** **void** windowClosing(WindowEvent e)
31. {
32. System.exit(0);
33. }
34. });
35. }
36. **public** **void** actionPerformed(ActionEvent e)
37. {
38. fd1.setVisible(**true**);
39. lab1.setText("Directory: " + fd1.getDirectory());
40. display(fd1.getDirectory() + fd1.getFile());
41. }
42. **public** **void** display(String fname)
43. {
44. // this method is for reading a file
45. **try**
46. {
47. FileInputStream fis1 = **new** FileInputStream(fname);
48. **int** fileSize = fis1.available();
49. **byte** buf1[] = **new** **byte**[fileSize];
50. fis1.read(buf1);
51. String str1 = **new** String(buf1);
52. ta1.setText(str1);
53. }
54. **catch**(IOException e)
55. {
56. System.exit(0);
57. }
58. }
60. **public** **static** **void** main(String args[])
61. {
62. **new** FileDialogDemo();
63. }
64. }

**Output**



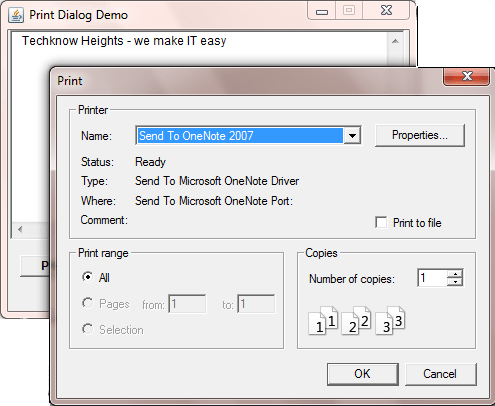
PrintDialog in AWT

**PrintDialogDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.printdialog;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.TextArea;
10. import java.awt.**event**.ActionEvent;
11. import java.awt.**event**.ActionListener;
12. import java.awt.print.PrinterJob;
13. import javax.print.PrintService;
15. **public** **class** PrintDialogDemo extends Frame
16. {
17. **private** Button button1 = **new** Button();
18. **private** TextArea textArea1 = **new** TextArea();
20. **public** PrintDialogDemo()
21. {
22. **try**
23. {
24. jbInit();
25. }
26. **catch** (Exception e)
27. {
28. e.printStackTrace();
29. }
30. }
32. **private** **void** jbInit() throws Exception
33. {
34. **this**.setLayout( **null** );
35. **this**.setSize(**new** Dimension(407, 309));
36. **this**.setBackground( SystemColor.control );
37. **this**.setTitle("Print Dialog Demo");
38. **this**.setVisible(**true**);
39. **this**.setResizable(**false**);
40. **this**.setLocation(250,200);
41. button1.setLabel("Print");
42. button1.setBounds(**new** Rectangle(15, 250, 70, 22));
43. button1.setFont(**new** Font("Tahoma", 1, 12));
44. button1.addActionListener(
45. **new** ActionListener()
46. {
47. **public** **void** actionPerformed(ActionEvent e)
48. {
49. button1\_actionPerformed(e);
50. }
51. });
52. textArea1.setBounds(**new** Rectangle(5, 25, 395, 210));
53. **this**.add(textArea1, **null**);
54. **this**.add(button1, **null**);
55. }
57. **public** **static** **void** main(String args[])
58. {
59. **new** PrintDialogDemo();
60. }
62. **private** **void** button1\_actionPerformed(ActionEvent e)
63. {
64. // PrinterJob Class controls printing to a particular
65. // print service (such as a printer or fax capability)
66. PrinterJob printJob = PrinterJob.getPrinterJob();
68. PrintService printer = printJob.getPrintService();
70. printJob.printDialog();
71. }
72. }

**Output**



Canvas in AWT

**CanvasDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

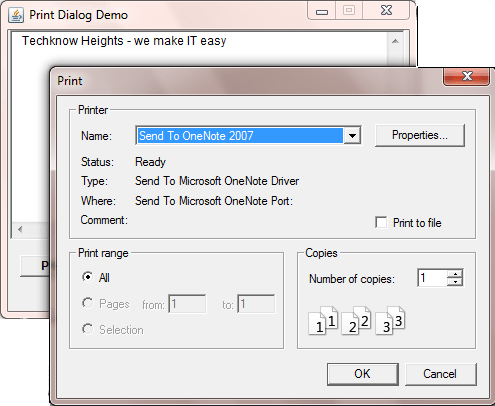
1. package com.techknow.canvas;
3. import java.awt.Canvas;
4. import java.awt.Graphics;
5. import java.awt.PrintJob;
6. import java.awt.Toolkit;
7. import java.util.Properties;
8. import java.awt.Dimension;
9. import java.awt.Frame;
10. import java.awt.SystemColor;
12. **public** **class** CanvasDemo extends Frame
13. {
14. MyCanvas canvas = **new** MyCanvas();
16. **public** CanvasDemo()
17. {
18. add("Center", canvas);
19. setSize(500, 500);
20. setVisible(**true**);
21. String name = "Test print job";
22. Properties properties = **new** Properties();
23. PrintJob pj = Toolkit.getDefaultToolkit().getPrintJob(
24. CanvasDemo.**this**, name, properties);
25. **if** (pj != **null**)
26. {
27. canvas.printAll(pj.getGraphics());
28. pj.end();
29. }
30. }
31. **private** **void** jbInit() throws Exception
32. {
33. **this**.setLayout( **null** );
34. **this**.setSize(**new** Dimension(390, 217));
35. **this**.setBackground( SystemColor.control );
36. }
37. **public** **static** **void** main(String args[])
38. {
39. CanvasDemo app = **new** CanvasDemo();
40. }
41. }

**MyCanvas.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.canvas;
3. **class** MyCanvas extends Canvas
4. {
5. **public** **void** paint(Graphics g)
6. {
7. Dimension size = getSize();
8. **int** width = size.width;
9. **int** height = size.height;
10. **int** x1 = (**int**) (width \* 0.1);
11. **int** x2 = (**int**) (width \* 0.9);
12. **int** y1 = (**int**) (height \* 0.1);
13. **int** y2 = (**int**) (height \* 0.9);
14. g.drawRect(x1, y1, x2 - x1, y2 - y1);
15. g.drawOval(x1, y1, x2 - x1, y2 - y1);
16. g.drawLine(x1, y1, x2, y2);
17. g.drawLine(x2, y1, x1, y2);
18. String text = "Print Me! ";
19. text += text;
20. text += text;
21. g.drawString(text, x1, (**int**) ((y1 + y2) / 2));
22. g.dispose();
23. }
24. }

**Output**



Menu in AWT

**MenuDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.menu;
3. import java.awt.Dimension;
4. import java.awt.Frame;
5. import java.awt.Menu;
6. import java.awt.MenuBar;
7. import java.awt.MenuItem;
8. import java.awt.SystemColor;
9. import java.awt.**event**.ActionEvent;
10. import java.awt.**event**.ActionListener;
12. import javax.swing.JOptionPane;
14. **public** **class** MenuDemo extends Frame
15. {
16. **private** MenuBar menuBar1 = **new** MenuBar();
17. **private** Menu menu1 = **new** Menu("File");
18. **private** Menu menu2 = **new** Menu("Help");
19. **private** MenuItem hello=**new** MenuItem("Hello");
20. **private** MenuItem bye=**new** MenuItem("Bye");
21. **private** MenuItem aboutus=**new** MenuItem("About Us");
23. **public** MenuDemo()
24. {
25. **try**
26. {
27. jbInit();
28. }
29. **catch** (Exception e)
30. {
31. e.printStackTrace();
32. }
33. }
35. **private** **void** jbInit() throws Exception
36. {
37. **this**.setLayout( **null** );
38. **this**.setSize( **new** Dimension( 400, 300 ) );
39. **this**.setBackground( SystemColor.control );
40. **this**.setVisible(**true**);
41. **this**.setLocation(150,150);
43. **this**.setMenuBar(menuBar1);
44. menu1.setLabel("File");
45. hello.addActionListener(
46. **new** ActionListener()
47. {
48. **public** **void** actionPerformed(ActionEvent e)
49. {
50. hello\_actionPerformed(e);
51. }
52. });
53. bye.addActionListener(**new** ActionListener()
54. {
55. **public** **void** actionPerformed(ActionEvent e)
56. {
57. bye\_actionPerformed(e);
58. }
59. });
60. aboutus.addActionListener(
61. **new** ActionListener()
62. {
63. **public** **void** actionPerformed(ActionEvent e)
64. {
65. aboutus\_actionPerformed(e);
66. }
67. });
68. menu1.add(hello);
69. menu1.add(bye);
70. menu2.add(aboutus);
71. menuBar1.add(menu1);menuBar1.add(menu2);
72. }
74. **public** **static** **void** main(String args[])
75. {
76. MenuDemo obj=**new** MenuDemo();
77. }
79. **private** **void** hello\_actionPerformed(ActionEvent e)
80. {
81. JOptionPane.showMessageDialog(**this**,"Hello Friends" );
82. }
84. **private** **void** bye\_actionPerformed(ActionEvent e)
85. {
86. JOptionPane.showMessageDialog(**this**,"Bye Friends" );
87. System.exit(0);
88. }
90. **private** **void** aboutus\_actionPerformed(ActionEvent e)
91. {
92. JOptionPane.showMessageDialog(
93. **this**,"Hey Friends this is the Techknow Heights Examples");
94. }
95. }

SubMenu in AWT

**SubMenuDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.menu;
3. import java.awt.Dimension;
4. import java.awt.Frame;
5. import java.awt.Menu;
6. import java.awt.MenuBar;
7. import java.awt.MenuItem;
8. import java.awt.SystemColor;
10. **public** **class** SubMenuDemo extends Frame
11. {
12. **private** MenuBar menuBar1 = **new** MenuBar();
13. **private** Menu menu1 = **new** Menu("MenuBar");
14. **private** Menu menu2 = **new** Menu("Menu");
15. **private** MenuItem menu3 = **new** MenuItem("SubMenu");
16. **private** MenuItem menu4 = **new** MenuItem("SubMenu");
17. **private** MenuItem menu5 = **new** MenuItem("SubMenu");
18. **public** SubMenuDemo()
19. {
20. **try**
21. {
22. jbInit();
23. }
24. **catch** (Exception e)
25. {
26. e.printStackTrace();
27. }
28. }
30. **private** **void** jbInit() throws Exception
31. {
32. **this**.setLayout( **null** );
33. **this**.setSize( **new** Dimension( 400, 300 ) );
34. **this**.setBackground( SystemColor.control );
35. **this**.setLocation(150,150);
36. **this**.setVisible(**true**);
37. **this**.setMenuBar(menuBar1);
38. menuBar1.add(menu1);
39. menu1.add(menu2);
40. menu2.add(menu3);
41. menu2.add(menu4);
42. menu2.add(menu5);
43. }
44. **public** **static** **void** main(String args[])
45. {
46. SubMenuDemo obj=**new** SubMenuDemo();
47. }
48. }

**Output**

Nested Menu in AWT

**NestedMenuDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.menu;
3. import java.awt.Dimension;
4. import java.awt.Frame;
5. import java.awt.Menu;
6. import java.awt.MenuBar;
7. import java.awt.MenuItem;
8. import java.awt.SystemColor;
10. **public** **class** NestedMenuDemo extends Frame
11. {
12. **private** MenuBar menuBar1 = **new** MenuBar();
13. **private** Menu menu1 = **new** Menu("MenuBar");
14. **private** Menu menu2 = **new** Menu("Menu");
15. **private** MenuItem menu3 = **new** MenuItem("SubMenu");
16. **private** MenuItem menu4 = **new** MenuItem("SubMenu");
17. **private** Menu menu5 = **new** Menu("SubMenu");
18. **private** MenuItem menu6 = **new** MenuItem("NestedMenu");
19. **private** MenuItem menu7 = **new** MenuItem("NestedMenu");
20. **private** MenuItem menu8 = **new** MenuItem("NestedMenu");
21. **public** NestedMenuDemo()
22. {
23. **try**
24. {
25. jbInit();
26. }
27. **catch** (Exception e)
28. {
29. e.printStackTrace();
30. }
31. }
33. **private** **void** jbInit() throws Exception
34. {
35. **this**.setLayout( **null** );
36. **this**.setSize( **new** Dimension( 400, 300 ) );
37. **this**.setBackground( SystemColor.control );
38. **this**.setLocation(150,150);
39. **this**.setVisible(**true**);
40. **this**.setMenuBar(menuBar1);
41. menuBar1.add(menu1);
42. menu1.add(menu2);
43. menu2.add(menu3);
44. menu2.add(menu4);
45. menu2.add(menu5);
46. menu5.add(menu6);
47. menu5.add(menu7);
48. menu5.add(menu8);
49. }
51. **public** **static** **void** main(String args[])
52. {
53. NestedMenuDemo obj=**new** NestedMenuDemo();
54. }
55. }

**Output**

Border Layout in AWT

**BorderLayoutDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.layout;
2. import java.awt.BorderLayout;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Frame;
6. import java.awt.Rectangle;
7. import java.awt.SystemColor;
8. import java.awt.**event**.ActionEvent;
9. import java.awt.**event**.ActionListener;
10. import java.awt.**event**.WindowAdapter;
11. import java.awt.**event**.WindowEvent;
12. import javax.swing.JOptionPane;
13. **public** **class** BorderLayoutDemo extends Frame {
14. **private** Button button1 = **new** Button();
15. **private** Button button2 = **new** Button();
16. **private** Button button3 = **new** Button();
17. **private** Button button4 = **new** Button();
18. **private** Button button5 = **new** Button();
19. **public** BorderLayoutDemo() {
20. **try** {
21. jbInit();
22. } **catch** (Exception e) {
23. e.printStackTrace();
24. }
25. }
26. **private** **void** jbInit() throws Exception {
27. **this**.setLayout( **null** );
28. **this**.setSize( **new** Dimension( 400, 300 ) );
29. **this**.setBackground( SystemColor.control );
30. **this**.setLayout(**new** BorderLayout());
31. **this**.setVisible(**true**);
32. **this**.setTitle("Border Layout Demo");
33. **this**.addWindowListener(**new** WindowAdapter() {
34. **public** **void** windowClosing(WindowEvent e) {
35. this\_windowClosing(e);
36. }
37. });
38. **this**.setLocation(250,150);
39. button1.setLabel("SOUTH");
40. button1.setBounds(**new** Rectangle(30, 55, 70, 22));
41. button1.addActionListener(**new** ActionListener() {
42. **public** **void** actionPerformed(ActionEvent e) {
43. button1\_actionPerformed(e);
44. }
45. });
46. button2.setLabel("NORTH");
47. button2.setBounds(**new** Rectangle(240, 45, 70, 22));
48. button2.addActionListener(**new** ActionListener() {
49. **public** **void** actionPerformed(ActionEvent e) {
50. button2\_actionPerformed(e);
51. }
52. });
53. button3.setLabel("WEST");
54. button3.setBounds(**new** Rectangle(185, 100, 70, 22));
55. button3.addActionListener(**new** ActionListener() {
56. **public** **void** actionPerformed(ActionEvent e) {
57. button3\_actionPerformed(e);
58. }
59. });
60. button4.setLabel("EAST");
61. button4.setBounds(**new** Rectangle(215, 170, 70, 22));
62. button4.addActionListener(**new** ActionListener() {
63. **public** **void** actionPerformed(ActionEvent e) {
64. button4\_actionPerformed(e);
65. }
66. });
67. button5.setLabel("CENTER");
68. button5.setBounds(**new** Rectangle(130, 200, 70, 22));
69. button5.addActionListener(**new** ActionListener() {
70. **public** **void** actionPerformed(ActionEvent e) {
71. button5\_actionPerformed(e);
72. }
73. });
74. **this**.add(button5, BorderLayout.CENTER);
75. **this**.add(button4, BorderLayout.EAST);
76. **this**.add(button3, BorderLayout.WEST);
77. **this**.add(button2, BorderLayout.NORTH);
78. **this**.add(button1, BorderLayout.SOUTH);
79. }
80. **public** **static** **void** main(String args[]) {
81. **new** BorderLayoutDemo();
82. }
83. **private** **void** button5\_actionPerformed(ActionEvent e) {
84. JOptionPane.showMessageDialog(
85. **this**,"This is Center side of Frame");
86. }
87. **private** **void** button4\_actionPerformed(ActionEvent e) {
88. JOptionPane.showMessageDialog(
89. **this**,"This is East side of Frame");
90. }
91. **private** **void** button3\_actionPerformed(ActionEvent e) {
92. JOptionPane.showMessageDialog(
93. **this**,"This is West side of Frame");
94. }
95. **private** **void** button2\_actionPerformed(ActionEvent e) {
96. JOptionPane.showMessageDialog(
97. **this**,"This is North side of Frame");
98. }
99. **private** **void** button1\_actionPerformed(ActionEvent e) {
100. JOptionPane.showMessageDialog(
101. **this**,"This is South side of Frame");
102. }
103. **private** **void** this\_windowClosing(WindowEvent e) {
104. System.exit(0);
105. }
106. }

**Output**

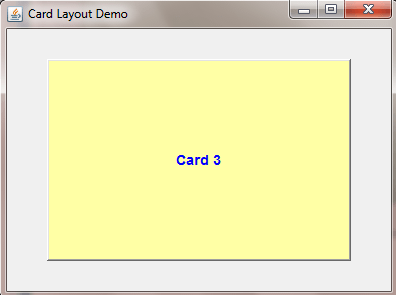
Card Layout in AWT

**CardLayoutDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.layout;
2. import java.awt.Button;
3. import java.awt.CardLayout;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.**event**.ActionEvent;
10. import java.awt.**event**.ActionListener;
11. import java.awt.**event**.WindowAdapter;
12. import java.awt.**event**.WindowEvent;
13. import javax.swing.JButton;
14. **public** **class** CardLayoutDemo extends Frame {
15. **private** Button button1 = **new** Button();
16. **private** Button button2 = **new** Button();
17. **private** Button button3 = **new** Button();
18. CardLayout card=**new** CardLayout(40,30);
19. **public** CardLayoutDemo() {
20. **try** {
21. jbInit();
22. } **catch** (Exception e) {
23. e.printStackTrace();
24. }
25. }
26. **private** **void** jbInit() throws Exception
27. {
28. **this**.setLayout( **null** );
29. **this**.setSize( **new** Dimension( 400, 300 ) );
30. **this**.setBackground( SystemColor.control );
31. **this**.setLayout(card);
32. **this**.setVisible(**true**);
33. **this**.setTitle("Card Layout Demo");
34. **this**.addWindowListener(**new** WindowAdapter() {
35. **public** **void** windowClosing(WindowEvent e) {
36. this\_windowClosing(e);
37. }
38. });
39. **this**.setLocation(250,200);
40. button1.setLabel("Card 1");
41. button1.setFont(**new** Font("Tahoma", 1, 14));
42. button1.setForeground(java.awt.Color.red);
43. button1.setBackground(**new** java.awt.Color(214, 255, 239));
44. button2.setBackground(**new** java.awt.Color(214, 214, 255));
45. button3.setBackground(**new** java.awt.Color(255, 255, 165));
46. button2.setFont(**new** Font("Tahoma", 1, 14));
47. button2.setForeground(java.awt.Color.green);
48. button3.setFont(**new** Font("Tahoma", 1, 14));
49. button3.setForeground(java.awt.Color.blue);
50. button1.addActionListener(**new** ActionListener() {
51. **public** **void** actionPerformed(ActionEvent e) {
52. button1\_actionPerformed(e);
53. }
54. });
55. button2.addActionListener(**new** ActionListener() {
56. **public** **void** actionPerformed(ActionEvent e) {
57. button2\_actionPerformed(e);
58. }
59. });
60. button3.addActionListener(**new** ActionListener() {
61. **public** **void** actionPerformed(ActionEvent e) {
62. button3\_actionPerformed(e);
63. }
64. });
65. button2.setLabel("Card 2");
66. button3.setLabel("Card 3");
67. **this**.add(button1, "a");
68. **this**.add(button2, "b");
69. **this**.add(button3, "c");
70. }
71. **public** **static** **void** main(String args[]) {
72. **new** CardLayoutDemo();
73. }
74. **private** **void** this\_windowClosing(WindowEvent e) {
75. System.exit(0);
76. }
77. **private** **void** button1\_actionPerformed(ActionEvent e) {
78. card.next(**this**);
79. }
80. **private** **void** button2\_actionPerformed(ActionEvent e) {
81. card.next(**this**);
82. }
83. **private** **void** button3\_actionPerformed(ActionEvent e) {
84. card.next(**this**);
85. }
86. }

**Output**



Flow Layout in AWT

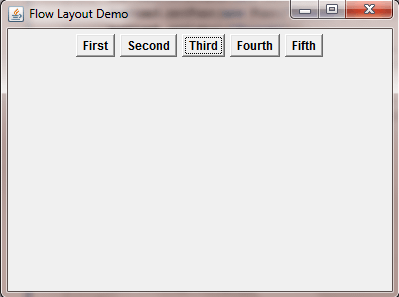
**FlowLayoutDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.layout;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.FlowLayout;
6. import java.awt.Font;
7. import java.awt.Frame;
8. import java.awt.Rectangle;
9. import java.awt.SystemColor;
10. import java.awt.**event**.ActionEvent;
11. import java.awt.**event**.ActionListener;
12. import java.awt.**event**.WindowAdapter;
13. import java.awt.**event**.WindowEvent;

16. **public** **class** FlowLayoutDemo extends Frame {
17. **private** Button button1 = **new** Button();
18. **private** Button button2 = **new** Button();
19. **private** Button button3 = **new** Button();
20. **private** Button button4 = **new** Button();
21. **private** Button button5 = **new** Button();
22. FlowLayout flow=**new** FlowLayout(FlowLayout.CENTER);
23. **private** Button button6 = **new** Button();
25. **public** FlowLayoutDemo() {
26. **try** {
27. jbInit();
28. } **catch** (Exception e) {
29. e.printStackTrace();
30. }
31. }
33. **private** **void** jbInit() throws Exception {
34. **this**.setLayout( flow );
35. **this**.setSize( **new** Dimension( 400, 300 ) );
36. **this**.setBackground( SystemColor.control );
37. **this**.setVisible(**true**);
38. **this**.setTitle("Flow Layout Demo");
39. **this**.addWindowListener(**new** WindowAdapter() {
40. **public** **void** windowClosing(WindowEvent e) {
41. this\_windowClosing(e);
42. }
43. });
44. **this**.setLocation(250,200);
45. button1.setLabel("First");
46. button1.setBounds(**new** Rectangle(290, 45, 70, 22));
47. button1.setFont(**new** Font("Tahoma", 1, 12));
48. button2.setLabel("Second");
49. button2.setBounds(**new** Rectangle(205, 75, 70, 22));
50. button2.setFont(**new** Font("Tahoma", 1, 12));
51. button3.setLabel("Third");
52. button3.setBounds(**new** Rectangle(195, 115, 70, 22));
53. button3.setFont(**new** Font("Tahoma", 1, 12));
54. button4.setLabel("Fourth");
55. button4.setBounds(**new** Rectangle(180, 145, 70, 22));
56. button4.setFont(**new** Font("Tahoma", 1, 12));
57. button5.setLabel("Fifth");
58. button5.setBounds(**new** Rectangle(130, 160, 70, 22));
59. button5.setFont(**new** Font("Tahoma", 1, 12));
60. button6.setLabel("button6");
61. **this**.add(button1);
62. **this**.add(button2);
63. **this**.add(button3);
64. **this**.add(button4);
65. **this**.add(button5);
66. }
68. **public** **static** **void** main(String args[]) {
69. **new** FlowLayoutDemo();
70. }
72. **private** **void** this\_windowClosing(WindowEvent e) {
73. System.exit(0);
74. }
75. }

**Output**



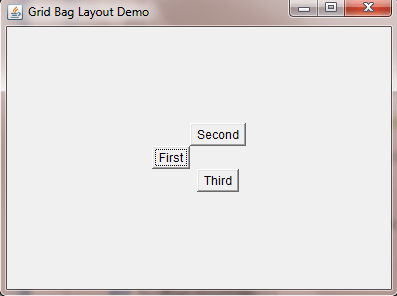
GridBag Layout in AWT

**GridBagLayoutDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.layout;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Frame;
6. import java.awt.GridBagConstraints;
7. import java.awt.GridBagLayout;
8. import java.awt.Insets;
9. import java.awt.SystemColor;
10. import java.awt.**event**.WindowAdapter;
11. import java.awt.**event**.WindowEvent;
13. **public** **class** GridBagLayoutDemo extends Frame {
14. GridBagLayout gridBag=**new** GridBagLayout();
15. **private** Button button1 = **new** Button();
16. **private** Button button2 = **new** Button();
17. **private** Button button3 = **new** Button();
18. **public** GridBagLayoutDemo() {
19. **try** {
20. jbInit();
21. } **catch** (Exception e) {
22. e.printStackTrace();
23. }
24. }
26. **private** **void** jbInit() throws Exception {
27. **this**.setLayout( gridBag );
28. **this**.setVisible(**true**);
29. **this**.setLocation(250,200);
30. **this**.setSize( **new** Dimension( 400, 300 ) );
31. **this**.setBackground( SystemColor.control );
32. **this**.setTitle("Grid Bag Layout Demo");
33. **this**.addWindowListener(**new** WindowAdapter() {
34. **public** **void** windowClosing(WindowEvent e) {
35. this\_windowClosing(e);
36. }
37. });
38. button1.setLabel("First");
39. button2.setLabel("Second");
40. button3.setLabel("Third");
41. **this**.add(button1, **new** GridBagConstraints(
42. 0, 1, 1, 1, 0.0, 0.0,
43. GridBagConstraints.CENTER,
44. GridBagConstraints.NONE,
45. **new** Insets(0, 0, 0, 0), 0, 0)
46. );
47. **this**.add(button2, **new** GridBagConstraints(
48. 1, 0, 1, 1, 0.0, 0.0,
49. GridBagConstraints.CENTER,
50. GridBagConstraints.NONE,
51. **new** Insets(0, 0, 0, 0), 0, 0)
52. );
53. **this**.add(button3, **new** GridBagConstraints(
54. 1, 2, 1, 1, 0.0, 0.0,
55. GridBagConstraints.CENTER,
56. GridBagConstraints.NONE,
57. **new** Insets(0, 0, 0, 0), 0, 0)
58. );
59. }
60. **private** **void** this\_windowClosing(WindowEvent e) {
61. System.exit(0);
62. }
63. **public** **static** **void** main(String args[]){
64. **new** GridBagLayoutDemo();
65. }
66. }

**Output**



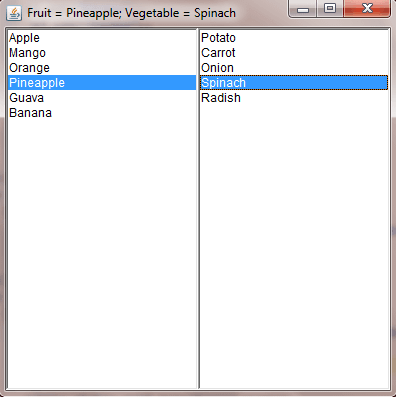
Grid Layout in AWT

**GridLayoutDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.layout;
3. import java.awt.\*;
4. import java.awt.**event**.\*;
6. **class** GridLayoutDemo extends Frame{
7. List l1,l2;
8. **public** GridLayoutDemo(){
9. // Set the frame properties
10. setTitle("GridLayout Demo");
11. setSize(400,400);
12. setLayout(**new** GridLayout());
13. setLocationRelativeTo(**null**);
14. setVisible(**true**);
15. // Create lists
16. l1=**new** List();
17. l2=**new** List();
18. // Add items to lists
19. l1.add("Apple");
20. l1.add("Mango");
21. l1.add("Orange");
22. l1.add("Pineapple");
23. l1.add("Guava");
24. l1.add("Banana");
25. l2.add("Potato");
26. l2.add("Carrot");
27. l2.add("Onion");
28. l2.add("Spinach");
29. l2.add("Radish");
30. // Add lists
31. add(l1);
32. add(l2);
34. // Add item listeners
35. l1.addItemListener(**new** ItemListener(){
36. **public** **void** itemStateChanged(ItemEvent ie){
37. setTitle("Fruit = "+l1.getSelectedItem()
38. +"; Vegetable = "+l2.getSelectedItem());
39. }
40. });
42. l2.addItemListener(**new** ItemListener(){
43. **public** **void** itemStateChanged(ItemEvent ie){
44. setTitle("Fruit = "+l1.getSelectedItem()
45. +"; Vegetable = "+l2.getSelectedItem());
46. }
47. });
48. }
50. **public** **static** **void** main(String args[]){
51. **new** GridLayoutDemo();
52. }
53. }

**Output**



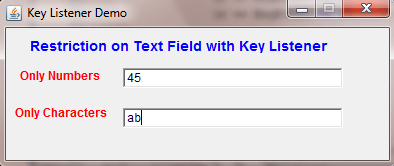
KeyListener in AWT

**KeyListenerDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.keylistener;
3. import java.awt.Dimension;
4. import java.awt.Font;
5. import java.awt.Frame;
6. import java.awt.Label;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.TextField;
10. import java.awt.**event**.KeyAdapter;
11. import java.awt.**event**.KeyEvent;
13. import javax.swing.JOptionPane;
15. **public** **class** KeyListnerDemo extends Frame
16. {
17. **private** TextField textField1 = **new** TextField();
18. **private** Label label1 = **new** Label();
19. **private** Label label2 = **new** Label();
20. **private** TextField textField2 = **new** TextField();
21. **private** Label label3 = **new** Label();
23. **public** KeyListnerDemo()
24. {
25. **try**
26. {
27. jbInit();
28. }
29. **catch** (Exception e)
30. {
31. e.printStackTrace();
32. }
33. }
35. **private** **void** jbInit() throws Exception
36. {
37. **this**.setLayout( **null** );
38. **this**.setSize(**new** Dimension(389, 160));
39. **this**.setBackground( SystemColor.control );
40. **this**.setResizable(**false**);
41. **this**.setTitle("Key Listener Demo");
42. **this**.setVisible(**true**);
43. **this**.setLocation(150,150);
44. textField1.setBounds(**new** Rectangle(120, 65, 220, 20));
45. textField1.addKeyListener(
46. **new** KeyAdapter()
47. {
48. **public** **void** keyPressed(KeyEvent e)
49. {
50. textField1\_keyPressed(e);
51. }
52. });
53. label1.setText("Only Numbers");
54. label1.setBounds(**new** Rectangle(15, 60, 100, 25));
55. label1.setFont(**new** Font("Tahoma", 1, 12));
56. label1.setForeground(java.awt.Color.red);
57. label2.setText("Only Characters");
58. label2.setBounds(**new** Rectangle(10, 100, 105, 20));
59. label2.setFont(**new** Font("Tahoma", 1, 12));
60. label2.setForeground(java.awt.Color.red);
61. textField2.setBounds(**new** Rectangle(120, 105, 220, 20));
62. textField2.addKeyListener(
63. **new** KeyAdapter()
64. {
65. **public** **void** keyTyped(KeyEvent e)
66. {
67. textField2\_keyTyped(e);
68. }
69. });
70. label3.setText("Restriction on Text Field with Key Listener");
71. label3.setBounds(**new** Rectangle(25, 35, 315, 15));
72. label3.setFont(**new** Font("Tahoma", 1, 14));
73. label3.setForeground(java.awt.Color.blue);
74. **this**.add(label3, **null**);
75. **this**.add(textField2, **null**);
76. **this**.add(label2, **null**);
77. **this**.add(label1, **null**);
78. **this**.add(textField1, **null**);
79. }
81. **private** **void** textField1\_keyPressed(KeyEvent e)
82. {
83. **char** c = e.getKeyChar();
84. **if** (!(Character.isDigit(c)
85. ||  (c == KeyEvent.VK\_BACK\_SPACE)
86. || (c == KeyEvent.VK\_DELETE)
87. ))
88. {
89. e.consume();
90. }
91. }
93. **private** **void** textField2\_keyTyped(KeyEvent e)
94. {
95. **char** c = e.getKeyChar();
96. **if** ((Character.isDigit(c)
97. || (c == KeyEvent.VK\_BACK\_SPACE)
98. || (c == KeyEvent.VK\_DELETE)
99. ))
100. {
101. e.consume();
102. }
104. }
106. **public** **static** **void** main(String args[])
107. {
108. **new** KeyListnerDemo();
109. }
110. }

**Output**



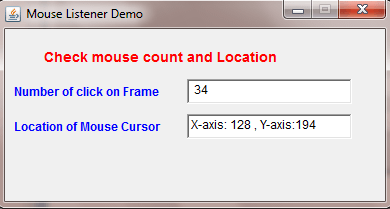
MouseListener in AWT

**MouseListenerDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.mouselistener;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Label;
8. import java.awt.Rectangle;
9. import java.awt.SystemColor;
10. import java.awt.TextField;
11. import java.awt.**event**.MouseAdapter;
12. import java.awt.**event**.MouseEvent;
14. **public** **class** MouseListenerDemo extends Frame
15. {
16. **private** **int** count=0;
17. **private** TextField textField1 = **new** TextField();
18. **private** Label label1 = **new** Label();
19. **private** Label label2 = **new** Label();
20. **private** Label label3 = **new** Label();
21. **private** TextField textField2 = **new** TextField();
23. **public** MouseListenerDemo()
24. {
25. **try**
26. {
27. jbInit();
28. }
29. **catch** (Exception e)
30. {
31. e.printStackTrace();
32. }
33. }
35. **private** **void** jbInit() throws Exception
36. {
37. **this**.setLayout( **null** );
38. **this**.setVisible(**true**);
39. **this**.setLocation(150,150);
40. **this**.setSize(**new** Dimension(386, 200));
41. **this**.setBackground( SystemColor.control );
42. **this**.setResizable(**false**);
43. **this**.setTitle("Mouse Listener Demo");
44. **this**.addMouseListener(
45. **new** MouseAdapter()
46. {
47. **public** **void** mouseClicked(MouseEvent e)
48. {
49. this\_mouseClicked(e);
50. }
51. });
52. textField1.setBounds(**new** Rectangle(185, 75, 165, 25));
53. label1.setText("Number of click on Frame ");
54. label1.setBounds(**new** Rectangle(10, 80, 170, 15));
55. label1.setFont(**new** Font("Tahoma", 1, 12));
56. label1.setForeground(java.awt.Color.blue);
57. label2.setText("Check mouse count and Location");
58. label2.setBounds(**new** Rectangle(40, 45, 290, 15));
59. label2.setFont(**new** Font("Tahoma", 1, 14));
60. label2.setForeground(java.awt.Color.red);
61. label3.setText("Location of Mouse Cursor ");
62. label3.setBounds(**new** Rectangle(10, 115, 170, 15));
63. label3.setForeground(java.awt.Color.blue);
64. label3.setFont(**new** Font("Tahoma", 1, 12));
65. textField2.setBounds(**new** Rectangle(185, 110, 165, 25));
66. **this**.add(textField2, **null**);
67. **this**.add(label3, **null**);
68. **this**.add(label2, **null**);
69. **this**.add(label1, **null**);
70. **this**.add(textField1, **null**);
71. }
73. **private** **void** this\_mouseClicked(MouseEvent e)
74. {
75. **int** x=e.getX();
76. **int** y=e.getY();
77. count=count+1;
78. textField1.setText(" "+count);
79. textField2.setText("X-axis: "+ x + " , Y-axis:"+y);
80. }
82. **public** **static** **void** main(String args[])
83. {
84. **new** MouseListenerDemo();
85. }
86. }

**Output**



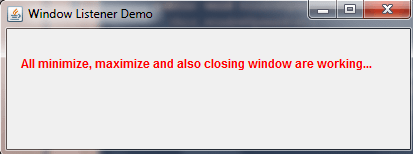
WindowListener in AWT

**WindowListenerDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.windowlistener;
3. import java.awt.Dimension;
4. import java.awt.Font;
5. import java.awt.Frame;
6. import java.awt.Label;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.**event**.WindowAdapter;
10. import java.awt.**event**.WindowEvent;
12. import javax.swing.JOptionPane;
14. **public** **class** WindowListenerDemo extends Frame
15. {
16. **private** Label label1 = **new** Label();
18. **public** WindowListenerDemo()
19. {
20. **try**
21. {
22. jbInit();
23. }
24. **catch** (Exception e)
25. {
26. e.printStackTrace();
27. }
28. }
30. **private** **void** jbInit() throws Exception
31. {
32. **this**.setLayout( **null** );
33. **this**.setVisible(**true**);
34. **this**.setLocation(250,200);
35. **this**.setSize(**new** Dimension(419, 158));
36. **this**.setBackground( SystemColor.control );
37. **this**.setTitle("Window Listener Demo");
38. **this**.addWindowListener(
39. **new** WindowAdapter()
40. {
41. **public** **void** windowClosing(WindowEvent e)
42. {
43. this\_windowClosing(e);
44. }
45. **public** **void** windowOpened(WindowEvent e)
46. {
47. this\_windowOpened(e);
48. }
49. });
50. label1.setText("All minimize, maximize and
51. also closing window are working...");
52. label1.setBounds(**new** Rectangle(20, 0, 390, 130));
53. label1.setFont(**new** Font("Tahoma", 1, 12));
54. label1.setForeground(java.awt.Color.red);
55. **this**.add(label1, **null**);
56. }
58. **private** **void** this\_windowClosing(WindowEvent e)
59. {
60. System.exit(0);
61. }
63. **public** **static** **void** main(String args[])
64. {
65. **new** WindowListenerDemo();
66. }
68. **private** **void** this\_windowOpened(WindowEvent e)
69. {
70. JOptionPane.showMessageDialog(**this**,
71. "Welcome To Techknow Heights");
72. }
73. }

**Output**



Cut, Copy and Paste in AWT

**CutCopyPasteDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.cutcopypaste;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Rectangle;
8. import java.awt.SystemColor;
9. import java.awt.TextArea;
10. import java.awt.datatransfer.Clipboard;
11. import java.awt.datatransfer.DataFlavor;
12. import java.awt.datatransfer.StringSelection;
13. import java.awt.datatransfer.Transferable;
14. import java.awt.**event**.ActionEvent;
15. import java.awt.**event**.ActionListener;
16. import java.awt.**event**.WindowAdapter;
17. import java.awt.**event**.WindowEvent;
19. **public** **class** CutCopyPasteDemo extends Frame
20. {
21. **private** TextArea textArea1 = **new** TextArea();
22. **private** Clipboard clipbd=**this**.getToolkit().getSystemClipboard();
23. **private** Button button1 = **new** Button();
24. **private** Button button2 = **new** Button();
25. **private** Button button3 = **new** Button();
27. **public** CutCopyPasteDemo()
28. {
29. **try**
30. {
31. jbInit();
32. }
33. **catch** (Exception e)
34. {
35. e.printStackTrace();
36. }
37. }
39. **private** **void** jbInit() throws Exception
40. {
41. **this**.setLayout( **null** );
42. **this**.setVisible(**true**);
43. **this**.setLocation(250,200);
44. **this**.setSize(**new** Dimension(421, 344));
45. **this**.setBackground( SystemColor.control );
46. **this**.setTitle("Cut Copy Paste Demo");
47. **this**.addWindowListener(
48. **new** WindowAdapter()
49. {
50. **public** **void** windowClosing(WindowEvent e)
51. {
52. this\_windowClosing(e);
53. }
54. });
55. textArea1.setBounds(**new** Rectangle(20, 30, 395, 245));
56. button1.setLabel("Cut");
57. button1.setBounds(**new** Rectangle(75, 290, 70, 22));
58. button1.setFont(**new** Font("Tahoma", 1, 12));
59. button1.setForeground(java.awt.Color.blue);
60. button1.addActionListener(
61. **new** ActionListener()
62. {
63. **public** **void** actionPerformed(ActionEvent e)
64. {
65. button1\_actionPerformed(e);
66. }
67. });
68. button2.setLabel("Copy");
69. button2.setBounds(**new** Rectangle(180, 290, 70, 22));
70. button2.setFont(**new** Font("Tahoma", 1, 12));
71. button2.setForeground(java.awt.Color.blue);
72. button2.addActionListener(**new** ActionListener()
73. {
74. **public** **void** actionPerformed(ActionEvent e)
75. {
76. button2\_actionPerformed(e);
77. }
78. });
79. button3.setLabel("Paste");
80. button3.setBounds(**new** Rectangle(285, 290, 70, 22));
81. button3.setFont(**new** Font("Tahoma", 1, 12));
82. button3.setForeground(java.awt.Color.blue);
83. button3.addActionListener(
84. **new** ActionListener()
85. {
86. **public** **void** actionPerformed(ActionEvent e)
87. {
88. button3\_actionPerformed(e);
89. }
90. });
91. **this**.add(button3, **null**);
92. **this**.add(button2, **null**);
93. **this**.add(button1, **null**);
94. **this**.add(textArea1, **null**);
95. }
97. **public** **static** **void** main(String args[])
98. {
99. **new** CutCopyPasteDemo();
100. }
102. **private** **void** this\_windowClosing(WindowEvent e)
103. {
104. System.exit(0);
105. }
107. **private** **void** button1\_actionPerformed(ActionEvent e)
108. {
109. String selection = textArea1.getSelectedText();
110. StringSelection clipString =  **new** StringSelection(selection);
111. clipbd.setContents(clipString, clipString);
112. textArea1.replaceRange("",textArea1.
113. getSelectionStart(),textArea1.getSelectionEnd());
114. }
116. **private** **void** button2\_actionPerformed(ActionEvent e)
117. {
118. String selection = textArea1.getSelectedText();
119. StringSelection clipString =  **new** StringSelection(selection);
120. clipbd.setContents(clipString, clipString);
121. }
123. **private** **void** button3\_actionPerformed(ActionEvent e)
124. {
125. Transferable clipData =  clipbd.getContents(**this**);
126. **try**
127. {
128. String clipString =(String)clipData.getTransferData(DataFlavor.stringFlavor);
129. textArea1.replaceRange(
130. clipString,textArea1.getSelectionStart(),
131. textArea1.getSelectionEnd()
132. );
133. }
134. **catch**(Exception ex)
135. {
136. System.**out**.println("not String flavor");
137. }
138. }
139. }

**Output**

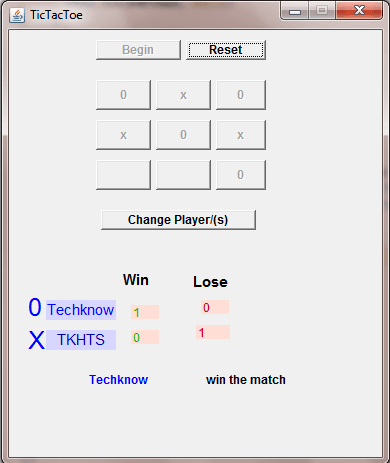
TicTacToe

**TicTacToe.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.tecknow.tictactoe;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
7. import java.awt.Label;
8. import java.awt.Rectangle;
9. import java.awt.SystemColor;
10. import java.awt.**event**.ActionEvent;
11. import java.awt.**event**.ActionListener;
12. import java.awt.**event**.WindowAdapter;
13. import java.awt.**event**.WindowEvent;
14. import javax.swing.JOptionPane;
16. **public** **class** TicTacToe extends Frame
17. {
18. **static** boolean flage=**true**;
19. **static** boolean winnerflage=**true**;
20. **static** **int** counter;
21. **int** player1Win;
22. **int** player1Lose;
23. **int** player2Win;
24. **int** player2Lose;
25. String playerName1;
26. String playerName2;
27. **private** Button button1 = **new** Button();
28. **private** Button button2 = **new** Button();
29. **private** Button button3 = **new** Button();
30. **private** Button button4 = **new** Button();
31. **private** Button button5 = **new** Button();
32. **private** Button button6 = **new** Button();
33. **private** Button button7 = **new** Button();
34. **private** Button button8 = **new** Button();
35. **private** Button button9 = **new** Button();
36. **private** Button button10 = **new** Button();
37. **private** Button button11 = **new** Button();
38. **private** Label label3 = **new** Label();
39. **private** Label label4 = **new** Label();
40. **private** Label label5 = **new** Label();
41. **private** Label label6 = **new** Label();
42. **private** Button button12 = **new** Button();
43. **private** Label label7 = **new** Label();
44. **private** Label label8 = **new** Label();
45. **private** Label label9 = **new** Label();
46. **private** Label label10 = **new** Label();
47. **private** Label label11 = **new** Label();
48. **private** Label label12 = **new** Label();
49. **private** Label label13 = **new** Label();
50. **private** Label label14 = **new** Label();
51. **private** Label label15 = **new** Label();
53. **public** TicTacToe()
54. {
55. **try**
56. {
57. jbInit();
58. }
59. **catch** (Exception e)
60. {
61. e.printStackTrace();
62. }
63. }
65. **private** **void** jbInit() throws Exception
66. {
67. **this**.setLayout( **null** );
68. **this**.setLocation(300,200);
69. **this**.setSize(**new** Dimension(379, 455));
70. **this**.setBackground( SystemColor.control );
71. **this**.setVisible(**true**);
72. **this**.setForeground(java.awt.Color.black);
73. **this**.setTitle("TicTacToe");
74. **this**.setResizable(**false**);
75. **this**.addWindowListener(**new** WindowAdapter()
76. {
77. **public** **void** windowClosing(WindowEvent e)
78. {
79. this\_windowClosing(e);
80. }
81. });
82. button1.setLabel("Begin");
83. button1.setBounds(**new** Rectangle(90, 35, 85, 20));
84. button1.setFont(**new** Font("Tahoma", 1, 12));
85. button1.addActionListener(
86. **new** ActionListener()
87. {
88. **public** **void** actionPerformed(ActionEvent e)
89. {
90. button1\_actionPerformed(e);
91. }
92. });
93. button2.setLabel("Reset");
94. button2.setBounds(**new** Rectangle(180, 35, 80, 20));
95. button2.setFont(**new** Font("Tahoma", 1, 12));
96. button2.addActionListener(
97. **new** ActionListener()
98. {
99. **public** **void** actionPerformed(ActionEvent e)
100. {
101. button2\_actionPerformed(e);
102. }
103. });
104. button3.setBounds(**new** Rectangle(90, 75, 55, 30));
105. button3.setFont(**new** Font("Tahoma", 1, 12));
106. button3.addActionListener(
107. **new** ActionListener()
108. {
109. **public** **void** actionPerformed(ActionEvent e)
110. {
111. button3\_actionPerformed(e);
112. }
113. });
114. button4.setBounds(**new** Rectangle(150, 75, 55, 30));
115. button4.setFont(**new** Font("Tahoma", 1, 12));
116. button4.addActionListener(
117. **new** ActionListener()
118. {
119. **public** **void** actionPerformed(ActionEvent e)
120. {
121. button4\_actionPerformed(e);
122. }
123. });
124. button5.setBounds(**new** Rectangle(210, 75, 50, 30));
125. button5.setFont(**new** Font("Tahoma", 1, 12));
126. button5.addActionListener(
127. **new** ActionListener()
128. {
129. **public** **void** actionPerformed(ActionEvent e)
130. {
131. button5\_actionPerformed(e);
132. }
133. });
134. button6.setBounds(**new** Rectangle(90, 115, 55, 30));
135. button6.setFont(**new** Font("Tahoma", 1, 12));
136. button6.addActionListener(
137. **new** ActionListener()
138. {
139. **public** **void** actionPerformed(ActionEvent e)
140. {
141. button6\_actionPerformed(e);
142. }
143. });
144. button7.setBounds(**new** Rectangle(150, 115, 55, 30));
145. button7.setFont(**new** Font("Tahoma", 1, 12));
146. button7.addActionListener(
147. **new** ActionListener()
148. {
149. **public** **void** actionPerformed(ActionEvent e)
150. {
151. button7\_actionPerformed(e);
152. }
153. });
154. button8.setBounds(**new** Rectangle(210, 115, 50, 30));
155. button8.setFont(**new** Font("Tahoma", 1, 12));
156. button8.addActionListener(
157. **new** ActionListener()
158. {
159. **public** **void** actionPerformed(ActionEvent e)
160. {
161. button8\_actionPerformed(e);
162. }
163. });
164. button9.setBounds(**new** Rectangle(90, 155, 55, 30));
165. button9.setFont(**new** Font("Tahoma", 1, 12));
166. button9.addActionListener(
167. **new** ActionListener()
168. {
169. **public** **void** actionPerformed(ActionEvent e)
170. {
171. button9\_actionPerformed(e);
172. }
173. });
174. button10.setBounds(**new** Rectangle(150, 155, 55, 30));
175. button10.setFont(**new** Font("Tahoma", 1, 12));
176. button10.addActionListener(
177. **new** ActionListener()
178. {
179. **public** **void** actionPerformed(ActionEvent e)
180. {
181. button10\_actionPerformed(e);
182. }
183. });
184. button11.setBounds(**new** Rectangle(210, 155, 50, 30));
185. button11.setFont(**new** Font("Tahoma", 1, 12));
186. button11.addActionListener(
187. **new** ActionListener()
188. {
189. **public** **void** actionPerformed(ActionEvent e)
190. {
191. button11\_actionPerformed(e);
192. }
193. });
194. label3.setBounds(**new** Rectangle(40, 295, 70, 20));
195. label3.setFont(**new** Font("Tahoma", 0, 15));
196. label3.setForeground(java.awt.Color.blue);
197. label3.setBackground(**new** java.awt.Color(214, 214, 255));
198. label3.setAlignment(1);
199. label4.setText("Win");
200. label4.setBounds(**new** Rectangle(115, 260, 40, 30));
201. label4.setFont(**new** Font("Tahoma", 1, 15));
202. label5.setText("Lose");
203. label5.setBounds(**new** Rectangle(185, 265, 55, 25));
204. label5.setFont(**new** Font("Tahoma", 1, 15));
205. label6.setBounds(**new** Rectangle(40, 325, 70, 20));
206. label6.setFont(**new** Font("Tahoma", 0, 15));
207. label6.setForeground(**new** java.awt.Color(0, 0, 165));
208. label6.setAlignment(1);
209. label6.setBackground(**new** java.awt.Color(214, 214, 255));
210. button12.setLabel("Change Player/(s)");
211. button12.setBounds(**new** Rectangle(95, 205, 155, 20));
212. button12.setFont(**new** Font("Tahoma", 1, 12));
213. button12.addActionListener(
214. **new** ActionListener()
215. {
216. **public** **void** actionPerformed(ActionEvent e)
217. {
218. button12\_actionPerformed(e);
219. }
220. });
221. label7.setBounds(**new** Rectangle(125, 300, 28, 14));
222. label7.setForeground(**new** java.awt.Color(0, 181, 0));
223. label7.setBackground(**new** java.awt.Color(255, 222, 214));
224. label8.setBounds(**new** Rectangle(195, 295, 28, 14));
225. label8.setForeground(**new** java.awt.Color(198, 0, 0));
226. label8.setBackground(**new** java.awt.Color(255, 222, 214));
227. label9.setBounds(**new** Rectangle(125, 325, 28, 14));
228. label9.setForeground(**new** java.awt.Color(0, 181, 0));
229. label9.setBackground(**new** java.awt.Color(255, 222, 214));
230. label10.setBounds(**new** Rectangle(190, 320, 34, 14));
231. label10.setForeground(**new** java.awt.Color(198, 0, 0));
232. label10.setBackground(**new** java.awt.Color(255, 222, 214));
233. label11.setBounds(**new** Rectangle(160, 360, 160, 30));
234. label11.setAlignment(1);
235. label11.setFont(**new** Font("Tahoma", 1, 12));
236. label12.setBounds(**new** Rectangle(60, 360, 105, 30));
237. label12.setAlignment(1);
238. label12.setForeground(java.awt.Color.blue);
239. label12.setFont(**new** Font("Tahoma", 1, 12));
240. label13.setBounds(**new** Rectangle(220, 380, 100, 30));
241. label13.setFont(**new** Font("Tahoma", 0, 14));
242. label14.setText("0");
243. label14.setBounds(**new** Rectangle(20, 285, 20, 35));
244. label14.setFont(**new** Font("Tahoma", 0, 25));
245. label14.setForeground(java.awt.Color.blue);
246. label15.setText("X");
247. label15.setBounds(**new** Rectangle(20, 320, 20, 30));
248. label15.setFont(**new** Font("Tahoma", 0, 25));
249. label15.setForeground(java.awt.Color.blue);
250. **this**.add(label15, **null**);
251. **this**.add(label14, **null**);
252. **this**.add(label13, **null**);
253. **this**.add(label12, **null**);
254. **this**.add(label11, **null**);
255. **this**.add(label10, **null**);
256. **this**.add(label9, **null**);
257. **this**.add(label8, **null**);
258. **this**.add(label7, **null**);
259. **this**.add(button12, **null**);
260. **this**.add(label6, **null**);
261. **this**.add(label5, **null**);
262. **this**.add(label4, **null**);
263. **this**.add(label3, **null**);
264. **this**.add(button11, **null**);
265. **this**.add(button10, **null**);
266. **this**.add(button9, **null**);
267. **this**.add(button8, **null**);
268. **this**.add(button7, **null**);
269. **this**.add(button6, **null**);
270. **this**.add(button5, **null**);
271. **this**.add(button4, **null**);
272. **this**.add(button3, **null**);
273. **this**.add(button2, **null**);
274. **this**.add(button1, **null**);
275. }
277. **public** **static** **void** main(String[] arg)
278. {
279. TicTacToe fm=**new** TicTacToe();
280. }
282. **private** **void** button1\_actionPerformed(ActionEvent e)
283. {
284. playerName1=JOptionPane.showInputDialog(**this**,
285. "Enter First Player Name");
286. playerName2=JOptionPane.showInputDialog(**this**,
287. "Enter Second Player Name");
288. JOptionPane.showMessageDialog(**this**,
289. playerName1+"=="+playerName2);
290. counter=0;
291. buttonEnable();
292. setStatusDefault();
293. setButtonLebelDefault();
294. flage=**true**;
295. //    playerName1=textField1.getText();
296. //    playerName2=textField2.getText();
297. **if**("".equals(playerName1))
298. {
299. playerName1="player1";
300. label3.setText(playerName1);
301. }
302. **else**
303. {
304. label3.setText(playerName1);
305. }
307. **if**("".equals(playerName2))
308. {
309. playerName2="player2";
310. label6.setText(playerName2);
311. }
312. **else**
313. {
314. label6.setText(playerName2);
315. }
317. button1.setEnabled(**false**);
318. }
320. **private** **void** button12\_actionPerformed(ActionEvent e)
321. {
322. button1.setEnabled(**true**);
323. }
325. //reset all the button and game
326. **private** **void** button2\_actionPerformed(ActionEvent e)
327. {
328. flage=**true**;
329. buttonEnable();
330. setButtonLebelDefault();
331. label11.setText("");
332. label12.setText("");
333. label13.setText("");
334. counter=0;
335. winnerflage=**true**;
336. }
338. **private** **void** button3\_actionPerformed(ActionEvent e)
339. {
340. **if**("".equals(button3.getLabel()))
341. {
342. counter++;
343. **if**(flage)
344. {
345. button3.setLabel("0");
346. flage=!flage;
347. }
348. **else**
349. {
350. button3.setLabel("x");
351. flage=!flage;
352. }
354. **if**(counter>=5)
355. {
356. String buttonLabe3=button3.getLabel();
357. String buttonLabe4=button4.getLabel();
358. String buttonLabe5=button5.getLabel();
359. String buttonLabe6=button6.getLabel();
360. String buttonLabe7=button7.getLabel();
361. String buttonLabe9=button9.getLabel();
362. String buttonLabe11=button11.getLabel();
363. **if**((buttonLabe3.equals(buttonLabe4)
364. && (buttonLabe3.equals(buttonLabe5)))
365. ||(buttonLabe3.equals(buttonLabe6)
366. && (buttonLabe3.equals(buttonLabe9)))
367. ||(buttonLabe3.equals(buttonLabe7)
368. && (buttonLabe3.equals(buttonLabe11)))
369. )
370. {
371. //winning code place here
372. winner(buttonLabe3);
373. buttonDisable();
374. showStatus();
375. winnerflage=**false**;
377. //matchDraw(winnerflage);
378. }
379. }
380. matchDraw(winnerflage);
381. }
382. }
384. **private** **void** button4\_actionPerformed(ActionEvent e)
385. {
386. **if**("".equals(button4.getLabel()))
387. {
388. counter++;
389. **if**(flage)
390. {
391. button4.setLabel("0");
392. flage=!flage;
393. }
394. **else**
395. {
396. button4.setLabel("x");
397. flage=!flage;
398. }
399. **if**(counter>=5)
400. {
401. String buttonLabe4=button4.getLabel();
402. String buttonLabe3=button3.getLabel();
403. String buttonLabe5=button5.getLabel();
404. String buttonLabe7=button7.getLabel();
405. String buttonLabe10=button10.getLabel();
406. **if**((buttonLabe4.equals(buttonLabe3)
407. && (buttonLabe4.equals(buttonLabe5)))
408. ||(buttonLabe4.equals(buttonLabe7)
409. && (buttonLabe4.equals(buttonLabe10)))
410. )
411. {
412. //winning code place here
413. winner(buttonLabe4);
414. buttonDisable();
415. showStatus();
416. winnerflage=**false**;
417. }
418. }
419. matchDraw(winnerflage);
420. }
421. }
423. **private** **void** button5\_actionPerformed(ActionEvent e)
424. {
425. **if**("".equals(button5.getLabel()))
426. {
427. counter++;
428. **if**(flage)
429. {
430. button5.setLabel("0");
431. flage=!flage;
432. }
433. **else**
434. {
435. button5.setLabel("x");
436. flage=!flage;
437. }
438. **if**(counter>=5)
439. {
440. String buttonLabe5=button5.getLabel();
441. String buttonLabe3=button3.getLabel();
442. String buttonLabe4=button4.getLabel();
443. String buttonLabe7=button7.getLabel();
444. String buttonLabe8=button8.getLabel();
445. String buttonLabe9=button9.getLabel();
446. String buttonLabe11=button11.getLabel();
448. **if**((buttonLabe5.equals(buttonLabe4)
449. && (buttonLabe5.equals(buttonLabe3)))
450. ||(buttonLabe5.equals(buttonLabe8)
451. && (buttonLabe5.equals(buttonLabe11)))
452. ||(buttonLabe5.equals(buttonLabe7)
453. && (buttonLabe5.equals(buttonLabe9)))
454. )
455. {
456. //winning code place here
457. winner(buttonLabe5);
458. //label11.setText("win the match");
459. buttonDisable();
460. showStatus();
461. winnerflage=**false**;
462. }
463. }
464. matchDraw(winnerflage);
465. }
466. }
467. **private** **void** button6\_actionPerformed(ActionEvent e)
468. {
469. **if**("".equals(button6.getLabel()))
470. {
471. counter++;
472. **if**(flage)
473. {
474. button6.setLabel("0");
475. flage=!flage;
476. }
477. **else**
478. {
479. button6.setLabel("x");
480. flage=!flage;
481. }
482. **if**(counter>=5)
483. {
484. String buttonLabe6=button6.getLabel();
485. String buttonLabe3=button3.getLabel();
486. String buttonLabe9=button9.getLabel();
487. String buttonLabe7=button7.getLabel();
488. String buttonLabe8=button8.getLabel();
489. **if**((buttonLabe6.equals(buttonLabe3)
490. && (buttonLabe6.equals(buttonLabe9)))
491. ||(buttonLabe6.equals(buttonLabe7)
492. && (buttonLabe6.equals(buttonLabe8)))
493. )
494. {
495. //winning code place here
496. winner(buttonLabe6);
497. buttonDisable();
498. showStatus();
499. winnerflage=**false**;
500. }
501. }
502. matchDraw(winnerflage);
503. }
504. }
505. **private** **void** button7\_actionPerformed(ActionEvent e)
506. {
507. **if**("".equals(button7.getLabel()))
508. {
509. counter++;
510. **if**(flage)
511. {
512. button7.setLabel("0");
513. flage=!flage;
514. }
515. **else**
516. {
517. button7.setLabel("x");
518. flage=!flage;
519. }
520. **if**(counter>=5)
521. {
522. String buttonLabe7=button7.getLabel();
523. String buttonLabe8=button8.getLabel();
524. String buttonLabe9=button9.getLabel();
525. String buttonLabe10=button10.getLabel();
526. String buttonLabe11=button11.getLabel();
527. String buttonLabe3=button3.getLabel();
528. String buttonLabe4=button4.getLabel();
529. String buttonLabe5=button5.getLabel();
530. String buttonLabe6=button6.getLabel();
532. **if**((buttonLabe7.equals(buttonLabe3)
533. && (buttonLabe7.equals(buttonLabe11)))
534. ||(buttonLabe7.equals(buttonLabe5)
535. && (buttonLabe7.equals(buttonLabe9)))
536. ||(buttonLabe7.equals(buttonLabe4)
537. && (buttonLabe7.equals(buttonLabe10)))
538. ||(buttonLabe7.equals(buttonLabe6)
539. && (buttonLabe7.equals(buttonLabe8)))
540. )
541. {
542. //winning code place here
543. winner(buttonLabe7);
545. buttonDisable();
546. showStatus();
547. winnerflage=**false**;
548. }
549. }
550. matchDraw(winnerflage);
551. }
552. }
554. **private** **void** button8\_actionPerformed(ActionEvent e)
555. {
556. **if**("".equals(button8.getLabel()))
557. {
558. counter++;
559. **if**(flage)
560. {
561. button8.setLabel("0");
562. flage=!flage;
563. }
564. **else**
565. {
566. button8.setLabel("x");
567. flage=!flage;
568. }
569. **if**(counter>=5)
570. {
571. String buttonLabe8=button8.getLabel();
572. String buttonLabe5=button5.getLabel();
573. String buttonLabe11=button11.getLabel();
574. String buttonLabe6=button6.getLabel();
575. String buttonLabe7=button7.getLabel();
576. **if**((buttonLabe8.equals(buttonLabe5)
577. && (buttonLabe8.equals(buttonLabe11)))
578. ||(buttonLabe8.equals(buttonLabe7)
579. && (buttonLabe8.equals(buttonLabe6)))
580. )
581. {
582. //winning code place here
583. winner(buttonLabe8);
585. buttonDisable();
586. showStatus();
587. winnerflage=**false**;
588. }
589. }
590. matchDraw(winnerflage);
591. }
592. }
594. **private** **void** button9\_actionPerformed(ActionEvent e)
595. {
596. **if**("".equals(button9.getLabel()))
597. {
598. counter++;
599. **if**(flage)
600. {
601. button9.setLabel("0");
602. flage=!flage;
603. }
604. **else**
605. {
606. button9.setLabel("x");
607. flage=!flage;
608. }
609. **if**(counter>=5)
610. {
611. String buttonLabe9=button9.getLabel();
612. String buttonLabe3=button3.getLabel();
613. String buttonLabe6=button6.getLabel();
614. String buttonLabe10=button10.getLabel();
615. String buttonLabe11=button11.getLabel();
616. String buttonLabe7=button7.getLabel();
617. String buttonLabe5=button5.getLabel();
619. **if**((buttonLabe9.equals(buttonLabe3)
620. && (buttonLabe9.equals(buttonLabe6)))
621. ||(buttonLabe9.equals(buttonLabe10)
622. && (buttonLabe9.equals(buttonLabe11)))
623. ||(buttonLabe9.equals(buttonLabe7)
624. && (buttonLabe9.equals(buttonLabe5)))
625. )
626. {
627. //winning code place here
628. winner(buttonLabe9);
630. buttonDisable();
631. showStatus();
632. winnerflage=**false**;
633. }
634. }
635. matchDraw(winnerflage);
636. }
637. }
639. **private** **void** button10\_actionPerformed(ActionEvent e)
640. {
641. **if**("".equals(button10.getLabel()))
642. {
643. counter++;
644. **if**(flage)
645. {
646. button10.setLabel("0");
647. flage=!flage;
648. }
649. **else**
650. {
651. button10.setLabel("x");
652. flage=!flage;
653. }
654. **if**(counter>=5)
655. {
656. String buttonLabe10=button10.getLabel();
657. String buttonLabe9=button9.getLabel();
658. String buttonLabe11=button11.getLabel();
659. String buttonLabe4=button4.getLabel();
660. String buttonLabe7=button7.getLabel();
661. **if**((buttonLabe10.equals(buttonLabe9)
662. && (buttonLabe10.equals(buttonLabe11)))
663. ||(buttonLabe10.equals(buttonLabe7)
664. && (buttonLabe10.equals(buttonLabe4)))
665. )
666. {
667. //winning code place here
668. winner(buttonLabe10);
670. buttonDisable();
671. showStatus();
672. winnerflage=**false**;
673. }
674. }
675. matchDraw(winnerflage);
676. }
677. }
679. **private** **void** button11\_actionPerformed(ActionEvent e)
680. {
681. **if**("".equals(button11.getLabel()))
682. {
683. counter++;
684. **if**(flage)
685. {
686. button11.setLabel("0");
687. flage=!flage;
688. }
689. **else**
690. {
691. button11.setLabel("x");
692. flage=!flage;
693. }
694. **if**(counter>=5)
695. {
696. String buttonLabe11=button11.getLabel();
697. String buttonLabe8=button8.getLabel();
698. String buttonLabe5=button5.getLabel();
699. String buttonLabe10=button10.getLabel();
700. String buttonLabe9=button9.getLabel();
701. String buttonLabe7=button7.getLabel();
702. String buttonLabe3=button3.getLabel();
704. **if**((buttonLabe11.equals(buttonLabe8)
705. && (buttonLabe11.equals(buttonLabe5)))
706. ||(buttonLabe11.equals(buttonLabe10)
707. && (buttonLabe11.equals(buttonLabe9)))
708. ||(buttonLabe11.equals(buttonLabe7)
709. && (buttonLabe11.equals(buttonLabe3)))
710. )
711. {
712. //winning code place here
713. winner(buttonLabe11);
715. buttonDisable();
716. showStatus();
717. winnerflage=**false**;
718. }
719. }
720. matchDraw(winnerflage);
721. }
722. }
724. **public**  **void** buttonDisable()
725. {
726. button3.enable(**false**);
727. button4.enable(**false**);
728. button5.enable(**false**);
729. button6.enable(**false**);
730. button7.enable(**false**);
731. button8.enable(**false**);
732. button9.enable(**false**);
733. button10.enable(**false**);
734. button11.enable(**false**);
735. counter=0;
736. }
738. **public** **void** showStatus()
739. {
740. label7.setText(""+player1Win);
741. label8.setText(""+player1Lose);
742. label9.setText(""+player2Win);
743. label10.setText(""+player2Lose);
744. }
746. **public** **void** setButtonLebelDefault()
747. {
748. button3.setLabel("");
749. button4.setLabel("");
750. button5.setLabel("");
751. button6.setLabel("");
752. button7.setLabel("");
753. button8.setLabel("");
754. button9.setLabel("");
755. button10.setLabel("");
756. button11.setLabel("");
757. counter=0;
758. }
760. **public**  **void** buttonEnable()
761. {
762. button3.enable(**true**);
763. button4.enable(**true**);
764. button5.enable(**true**);
765. button6.enable(**true**);
766. button7.enable(**true**);
767. button8.enable(**true**);
768. button9.enable(**true**);
769. button10.enable(**true**);
770. button11.enable(**true**);
771. }
773. **public** **void** setStatusDefault()
774. {
775. label7.setText("");
776. label8.setText("");
777. label9.setText("");
778. label10.setText("");
779. label11.setText("");
780. label12.setText("");
781. player1Win=0;
782. player1Lose=0;
783. player2Win=0;
784. player2Lose=0;
785. }
787. **public** **void** winner(String value)
788. {
789. **if**("0".equals(value))
790. {
791. label12.setText(playerName1);
792. label11.setText("win the match");
793. player1Win++;
794. player2Lose++;
795. }
796. **else**
797. {
798. label12.setText(playerName2);
799. label11.setText("win the match");
800. player1Lose++;
801. player2Win++;
802. }
803. }
805. **public** **void** matchDraw(boolean value)
806. {
807. **if**(counter==9 && value==**true**)
808. {
809. label11.setText("");
810. label12.setText("");
811. label13.setText("Draw the Match");
812. buttonDisable();
813. }
814. }
816. **private** **void** button13\_actionPerformed(ActionEvent e)
817. {
818. System.exit(0);
819. }
821. **private** **void** this\_windowClosing(WindowEvent e)
822. {
823. System.exit(0);
824. }
825. }

**Output**



Login Form in AWT

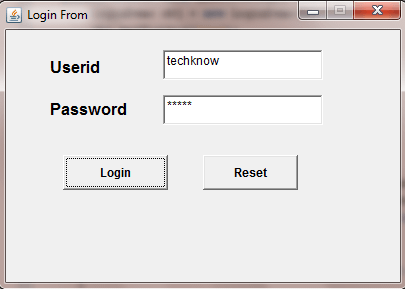
**LoginDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.ducat.ui;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.Font;
6. import java.awt.Frame;
8. import java.awt.Label;
9. import java.awt.Rectangle;
10. import java.awt.SystemColor;
11. import java.awt.TextField;
12. import java.awt.**event**.ActionEvent;
13. import java.awt.**event**.ActionListener;
15. import java.awt.**event**.WindowAdapter;
17. import java.awt.**event**.WindowEvent;
19. import javax.swing.JOptionPane;

22. **public** **class** LoginDemo extends Frame
23. {
24. **private** Label label2 = **new** Label();
25. **private** Label label3 = **new** Label();
26. **private** TextField textField1 = **new** TextField();
27. **private** TextField textField2 = **new** TextField();
28. **private** Button button1 = **new** Button();
29. **private** Button button2 = **new** Button();
31. **public** LoginDemo()
32. {
33. **try**
34. {
35. jbInit();
36. }
37. **catch** (Exception e)
38. {
39. e.printStackTrace();
40. }
41. }
43. **private** **void** jbInit() throws Exception
44. {
45. **this**.setLayout( **null** );
46. **this**.setSize(**new** Dimension(400, 280));
47. **this**.setBackground( SystemColor.control );
48. **this**.setResizable(**false**);
49. **this**.setTitle("Login From");
50. **this**.addWindowListener(
51. **new** WindowAdapter()
52. {
53. **public** **void** windowClosing(WindowEvent e)
54. {
55. this\_windowClosing(e);
56. }
57. });
58. label2.setText("Userid");
59. label2.setBounds(**new** Rectangle(45, 45, 115, 35));
60. label2.setFont(**new** Font("Tahoma", 1, 16));
61. label3.setText("Password");
62. label3.setBounds(**new** Rectangle(45, 90, 110, 30));
63. label3.setFont(**new** Font("Tahoma", 1, 16));
64. textField1.setBounds(**new** Rectangle(160, 45, 160, 30));
65. textField2.setBounds(**new** Rectangle(160, 90, 160, 30));
66. textField2.setEchoChar('\*');
67. button1.setLabel("Login");
68. button1.setBounds(**new** Rectangle(60, 150, 105, 35));
69. button1.setFont(**new** Font("Tahoma", 1, 12));
70. button1.addActionListener(
71. **new** ActionListener()
72. {
73. **public** **void** actionPerformed(ActionEvent e)
74. {
75. button1\_actionPerformed(e);
76. }
77. });
78. button2.setLabel("Reset");
79. button2.setBounds(**new** Rectangle(200, 150, 95, 35));
80. button2.setFont(**new** Font("Tahoma", 1, 12));
81. button2.addActionListener(
82. **new** ActionListener()
83. {
84. **public** **void** actionPerformed(ActionEvent e)
85. {
86. button2\_actionPerformed(e);
87. }
88. });
89. **this**.add(button2, **null**);
90. **this**.add(button1, **null**);
91. **this**.add(textField2, **null**);
92. **this**.add(textField1, **null**);
93. **this**.add(label3, **null**);
94. **this**.add(label2, **null**);
95. **this**.setLocation(300,300);
96. }
98. **public** **static** **void** main(String[] args)
99. {
100. LoginDemo obj = **new** LoginDemo();
101. obj.setVisible(**true**);
102. }
104. **private** **void** button1\_actionPerformed(ActionEvent e)
105. {
106. String userid = textField1.getText();
107. String password = textField2.getText();
108. checkLogin(userid,password);
109. }
110. **public** **void**  checkLogin(String userid,String password)
111. {
112. **if**(userid.equalsIgnoreCase("Techknow")
113. && password.equalsIgnoreCase("tkhts"))
114. {
115. JOptionPane.showMessageDialog(**this**,
116. "Welcome in Techknow Heights User");
117. }
118. **else**
119. {
120. JOptionPane.showMessageDialog(**this**,
121. "UserName, Password is Incorrect !!!");
122. }
123. }
125. **private** **void** button3\_actionPerformed(ActionEvent e)
126. {
127. textField1.setText("");
128. textField2.setText("");
129. }
131. **private** **void** this\_windowClosing(WindowEvent e)
132. {
133. **this**.setVisible(**false**);
134. **this**.dispose();
135. }
137. **private** **void** button2\_actionPerformed(ActionEvent e)
138. {
139. textField1.setText("");
140. textField2.setText("");
141. }
142. }

**Output**



Notepad in AWT

**NotepadDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.notepad;
3. import java.awt.Dimension;
4. import java.awt.FileDialog;
5. import java.awt.Frame;
6. import java.awt.Menu;
7. import java.awt.MenuBar;
8. import java.awt.MenuItem;
9. import java.awt.MenuShortcut;
10. import java.awt.Rectangle;
11. import java.awt.SystemColor;
12. import java.awt.TextArea;
13. import java.awt.datatransfer.Clipboard;
14. import java.awt.datatransfer.DataFlavor;
15. import java.awt.datatransfer.StringSelection;
16. import java.awt.datatransfer.Transferable;
17. import java.awt.**event**.ActionEvent;
18. import java.awt.**event**.ActionListener;
19. import java.awt.**event**.ComponentAdapter;
20. import java.awt.**event**.ComponentEvent;
21. import java.awt.**event**.WindowAdapter;
22. import java.awt.**event**.WindowEvent;
23. import java.awt.print.PrinterJob;
25. import java.io.FileNotFoundException;
26. import java.io.IOException;
28. import javax.print.PrintService;
30. **public** **class** NotepadDemo extends Frame
31. {
32. **public** **static** TextArea textArea1 = **new** TextArea();
33. **private** MenuBar menuBar1 = **new** MenuBar();
34. **private** Menu filemenu = **new** Menu();
35. **private** MenuItem newItem = **new** MenuItem("New");
36. **private** MenuItem openItem = **new** MenuItem("Open...");
37. **private** MenuItem saveItem = **new** MenuItem("Save");
38. **private** MenuItem printItem = **new** MenuItem("Print");
39. **private** MenuItem exitItem = **new** MenuItem("Exit");
40. **private** MenuItem cutItem =
41. **new** MenuItem("Cut", **new** MenuShortcut('X',**true**));
42. **private** MenuItem copyItem = **new** MenuItem("Copy");
43. **private** MenuItem pasteItem =
44. **new** MenuItem("Paste",**new** MenuShortcut('V',**true**));
45. **private** Menu editMenu = **new** Menu("Edit");
46. Clipboard clipbd = **this**.getToolkit().getSystemClipboard();
48. **public** NotepadDemo()
49. {
50. **try**
51. {
52. jbInit();
53. }
54. **catch** (Exception e)
55. {
56. e.printStackTrace();
57. }
58. }
60. **private** **void** jbInit() throws Exception
61. {
62. **this**.setLayout( **null** );
63. **this**.setSize(**new** Dimension(400, 328));
64. **this**.setBackground( SystemColor.control );
65. **this**.addComponentListener(**new** ComponentAdapter() {
66. **public** **void** componentResized(ComponentEvent e) {
67. this\_componentResized(e);
68. }
69. });
70. textArea1.setBounds(**new** Rectangle(10, 70, 335, 220));
71. filemenu.setLabel("File");
72. filemenu.add(newItem);
73. filemenu.add(openItem);
74. filemenu.add(saveItem);
75. filemenu.addSeparator();
76. filemenu.add(printItem);
77. filemenu.add(exitItem);
78. menuBar1.add(filemenu);
79. menuBar1.add(editMenu);
80. editMenu.add(cutItem);
81. editMenu.add(copyItem);
82. editMenu.addSeparator();
83. editMenu.add(pasteItem);
84. **this**.setMenuBar(menuBar1);
85. **this**.addWindowListener(**new** WindowAdapter() {
86. **public** **void** windowClosing(WindowEvent e) {
87. this\_windowClosing(e);
88. }
89. });
90. **this**.add(textArea1, **null**);
91. newItem.addActionListener(**new** ActionListener() {
92. **public** **void** actionPerformed(ActionEvent ee) {
93. newItem\_Click(ee);
94. }
95. }
96. );

99. openItem.addActionListener(**new** ActionListener() {
100. **public** **void** actionPerformed(ActionEvent ee) {
101. openItem\_Click(ee);
102. }
103. }
104. );
105. saveItem.addActionListener(**new** ActionListener() {
106. **public** **void** actionPerformed(ActionEvent ee) {
107. saveItem\_Click(ee);
108. }
109. }
110. );
111. cutItem.addActionListener(**new** ActionListener() {
112. **public** **void** actionPerformed(ActionEvent ee) {
113. cutItem\_Click(ee);
114. }
115. }
116. );
117. copyItem.addActionListener(**new** ActionListener() {
118. **public** **void** actionPerformed(ActionEvent ee) {
119. copyItem\_Click(ee);
120. }
121. }
122. );
123. pasteItem.addActionListener(**new** ActionListener() {
124. **public** **void** actionPerformed(ActionEvent ee) {
125. pasteItem\_Click(ee);
126. }
127. }
128. );
129. printItem.addActionListener(**new** ActionListener() {
130. **public** **void** actionPerformed(ActionEvent ee) {
131. printItem\_Click(ee);
132. }
133. }
134. );
135. }
137. **public** **static** **void** main(String[] args)
138. {
139. NotepadDemo obj = **new** NotepadDemo();
140. obj.setVisible(**true**);
141. }
143. **private** **void** this\_componentResized(ComponentEvent e)
144. {
145. **int** height = **this**.getHeight();
146. **int** width = **this**.getWidth();
147. height = height- 50;
148. width = width - 30;
150. textArea1.setSize(width,height);
151. }
153. **private** **void** newItem\_Click(ActionEvent e)
154. {
155. String data = textArea1.getText();
156. **if**(data.length()>0)
157. {
158. NewFrame obj = **new** NewFrame();
159. obj.setVisible(**true**);
160. obj.setLocation(200,200);
161. }
162. }
164. **private** **void** openItem\_Click(ActionEvent e)
165. {
166. FileDialog fd =
167. **new** FileDialog(**this**,"My Open Dialog",FileDialog.LOAD);
168. fd.setVisible(**true**);
169. String dirName = fd.getDirectory();
170. String fileName = fd.getFile();
171. String path = dirName + fileName;
172. ReadWriteDemo rw= **new** ReadWriteDemo();
173. String data = **null**;
174. **try** {
175. data = rw.readData(path);
176. }
177. **catch** (FileNotFoundException f)
178. {
179. }
180. **catch** (IOException f)
181. {
182. }
184. NotepadDemo.textArea1.setText(data);
185. }
187. **private** **void** saveItem\_Click(ActionEvent e)
188. {
189. FileDialog fd =
190. **new** FileDialog(**this**,"My Save Dialog",FileDialog.SAVE);
191. fd.setVisible(**true**);
192. String dirName = fd.getDirectory();
193. String fileName = fd.getFile();
194. String path = dirName + fileName;
195. ReadWriteDemo rw = **new** ReadWriteDemo();

198. **try**
199. {
200. rw.writeData(NotepadDemo  .textArea1.getText(),path);
201. }
202. **catch** (FileNotFoundException f)
203. {
204. }
205. **catch** (IOException f)
206. {
207. }
208. }
210. **private** **void** this\_windowClosing(WindowEvent e)
211. {
212. String data = textArea1.getText();
213. **if**(data.length()>0)
214. {
215. NewFrame obj = **new** NewFrame();
216. obj.setVisible(**true**);
217. obj.setLocation(200,200);
218. }
219. }
221. **private** **void** cutItem\_Click(ActionEvent e)
222. {
223. String selection = textArea1.getSelectedText();
224. StringSelection clipString =  **new** StringSelection(selection);
225. clipbd.setContents(clipString, clipString);
226. textArea1.replaceRange("",
227. textArea1.getSelectionStart(),textArea1.getSelectionEnd());
228. }
230. **private** **void** copyItem\_Click(ActionEvent e)
231. {
232. String selection = textArea1.getSelectedText();
233. StringSelection clipString =  **new** StringSelection(selection);
234. clipbd.setContents(clipString, clipString);
235. }
237. **private** **void** pasteItem\_Click(ActionEvent e)
238. {
239. Transferable clipData =  clipbd.getContents(**this**);
240. **try**
241. {
242. String clipString =(String)clipData.
243. getTransferData(DataFlavor.stringFlavor);
244. textArea1.replaceRange(clipString,
245. textArea1.getSelectionStart(),
246. textArea1.getSelectionEnd()
247. );
248. }
249. **catch**(Exception ex)
250. {
251. System.**out**.println("not String flavor");
252. }
253. }
255. **private** **void** printItem\_Click(ActionEvent e)
256. {
257. // PrinterJob Class controls printing to a particular print
258. //service (such as a printer or fax capability).
259. PrinterJob printJob = PrinterJob.getPrinterJob();
260. PrintService printer = printJob.getPrintService();
261. printJob.printDialog();
262. }
263. }

**NewFrame.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.notepad;
3. import java.awt.Button;
4. import java.awt.Dimension;
5. import java.awt.FileDialog;
6. import java.awt.Font;
7. import java.awt.Frame;
8. import java.awt.Label;
9. import java.awt.Rectangle;
10. import java.awt.SystemColor;
11. import java.awt.**event**.ActionEvent;
12. import java.awt.**event**.ActionListener;
14. import java.io.FileNotFoundException;
15. import java.io.IOException;
17. **public** **class** NewFrame extends Frame {
18. **private** Label label1 = **new** Label();
19. **private** Button button1 = **new** Button();
20. **private** Button button2 = **new** Button();
21. **private** Button button3 = **new** Button();
23. **public** NewFrame() {
24. **try** {
25. jbInit();
26. } **catch** (Exception e) {
27. e.printStackTrace();
28. }
29. }
31. **private** **void** jbInit() throws Exception {
32. **this**.setLayout( **null** );
33. **this**.setSize(**new** Dimension(315, 217));
34. **this**.setBackground( SystemColor.control );
35. **this**.setResizable(**false**);
36. label1.setText("Do u want to Save or not");
37. label1.setBounds(**new** Rectangle(15, 45, 285, 55));
38. label1.setFont(**new** Font("Tahoma", 1, 18));
39. button1.setLabel("Save");
40. button1.setBounds(**new** Rectangle(25, 150, 75, 24));
41. button1.addActionListener(**new** ActionListener() {
42. **public** **void** actionPerformed(ActionEvent e) {
43. button1\_actionPerformed(e);
44. }
45. });
46. button2.setLabel("Don't Save");
47. button2.setBounds(**new** Rectangle(115, 150, 75, 24));
48. button2.addActionListener(**new** ActionListener() {
49. **public** **void** actionPerformed(ActionEvent e) {
50. button2\_actionPerformed(e);
51. }
52. });
53. button3.setLabel("Cancel");
54. button3.setBounds(**new** Rectangle(210, 150, 75, 24));
55. button3.addActionListener(**new** ActionListener() {
56. **public** **void** actionPerformed(ActionEvent e) {
57. button3\_actionPerformed(e);
58. }
59. });
60. **this**.add(button3, **null**);
61. **this**.add(button2, **null**);
62. **this**.add(button1, **null**);
63. **this**.add(label1, **null**);
64. }
66. **private** **void** button1\_actionPerformed(ActionEvent e) {
67. FileDialog fd =
68. **new** FileDialog(**this**,"My Save Dialog",FileDialog.SAVE);
69. fd.setVisible(**true**);
70. String dirName = fd.getDirectory();
71. String fileName = fd.getFile();
72. String path = dirName + fileName;
73. ReadWriteDemo rw = **new** ReadWriteDemo();

76. **try** {
77. rw.writeData(NotepadDemo  .textArea1.getText(),path);
78. } **catch** (FileNotFoundException f) {
79. } **catch** (IOException f) {
80. }
81. }
83. **private** **void** button2\_actionPerformed(ActionEvent e) {
84. NotepadDemo.textArea1.setText("");
85. **this**.setVisible(**false**);
86. }
88. **private** **void** button3\_actionPerformed(ActionEvent e) {
89. **this**.setVisible(**false**);
90. }
91. }

**ReadWriteDemo.java**

[view plaincopy to clipboardprint?](http://www.tkhts.com/awt-swing/awt/example.jsp)

1. package com.techknow.notepad;
3. import java.io.FileInputStream;
4. import java.io.FileNotFoundException;
5. import java.io.FileOutputStream;
6. import java.io.IOException;
8. **public** **class** ReadWriteDemo {
9. **public** ReadWriteDemo() {
10. }
12. **public** String readData(String path ) throws
13. FileNotFoundException, IOException {
14. StringBuffer sb = **new** StringBuffer();
15. path  = path.replace("\\","\\\\");
17. FileInputStream fi = **new** FileInputStream(path);
18. **int** z = fi.read();
20. **while**(z!=-1) {
21. sb.append((**char**)z);
22. System.**out**.print((**char**)z);
23. z = fi.read();
24. }
25. fi.close();
26. **return** sb.toString();
27. }
28. **public** **void** writeData(String data, String path)
29. throws FileNotFoundException, IOException {
31. path = path.replace("\\","\\\\");
32. FileOutputStream fo = **new** FileOutputStream(path);
33. fo.write(data.getBytes());
34. fo.close();
35. }
36. }

**Output**

