Event Handling

----------------------

Mouse move

Mouse Click

Mouse Drag

key type

key presses

key released

Event

-------

An event is an object that describes a state change in a source

Event Sources

-------------------

A source is an object that generates an event(textbox,list,choice,button)

For event Handling

-------------------------

1. Register the source with some event class or event listeners

syntax:

--------

public void addTypeListener(TypeEvent el) (interface)

Type - Name of the event

el - reference to the event listener

For Eg:

---------

Button b;

b.addActionListener(this);

void actionPerformed(ActionEvent e) {

statements;

}

java.awt.event package

--------------------------------

Event Class Description

--------------- ---------------

ActionEvent Generated when a button is pressed, a list item is double clicked, or a menu item is selected

AdjustmentEvent Generated when a scroll bar is manipulated.

Component Event Generated when a component is hidden, moved,resized or becomes visible

Container Event Generated when a component is added or removed from a container

focus Event Generated when a component gains or loses keyboard focus

ItemEvent Generated when a check box or list item is clicked; also occurs when a choice selection is made or a checkable menu item is selected or deselected

KeyEvent Generated when input is received from keyboard

MouseEvent Generated when the mouse is dragged, clicked,pressed or released. also generated when the mouse enters or exits a component

TextEvent Generated when the value of the textarea or textfield is changed

WindowEvent Generated when a window is activated, closed, deactivated, deiconified,iconified,opened or quit

Listener Interfaces and a brief description of the methods

-------------------------------------------------------------------------------------

The ActionListener Interface

-----------------------------------

This interface defines the actionPerformed() method that is invoked when an action event occurs.

void actionPerformed(ActionEvent ae)

The AdjustmentListener interface

-------------------------------------------

This interface defines the adjustmentValueChanged() method that is invoked when an adjustment event occurs.

void adjustmentValueChanged(AdjustmentEvent ae)

The ComponentListener Interface

-----------------------------------------

This interface defines four methods that are invoked when a component is moved,shown,resized or hidded.

void componentResized(ComponentEvent e)

void componentMoved(ComponentEvent e)

void componentShown(ComponentEvent e)

void componentHidden(ComponentEvent e)

The ContainerListener Interface

-----------------------------------------

This interface contains two methods. When a component is added to a container, componentAdded() is invoked. When a component is removed from a container, componentRemoved() is invoked.

void componentAdded(ContainerEvent ce)

void componentRemoved(ContainerEvent ce)

The FocusListener Interface

-----------------------------------

Components got or lost focus.

void focusGained(FocusEvent fe)

void focusLost(focusEvent fe)

The ItemListener Interface

---------------------------------

This invoked when a state of the item is changed

void itemStateChanged(ItemEvent ie)

The KeyListener Interface

------------------------------------

key pressed,released,typed.

void keyPressed(KeyEvent ke)

void keyReleased(KeyEvent ke)

void keyTyped(KeyEvent ke)

The MouseListener Interface

-------------------------------------

mouse click,press,release,enter,exit

void mouseClicked(MouseEvent me)

void mouseEntered(MouseEvent me)

void mouseExited(MouseEvent me)

void mousePressed(MouseEvent me)

void mouseReleased(MouseEvent me)

The MouseMotionListener Interface

-----------------------------------------------

mouse drag,move

void mouseDragged(MouseEvent me)

void mouseMoved(MouseEvent me)

The TextListener interface

---------------------------------

Text field,TextArea

void textChanged(TextEvent te)

The WindowListener Interface

---------------------------------------

open,close,activate,deactivate,iconified,deiconified

void windowActivated(WindowEvent we)

void windowClosed(WindowEvent we)

void windowClosing(WindowEvent we)

void windowDeactivated(WindowEvent we)

void windowDeiconified(WindowEvent we)

void windowIconified(WindowEvent we)

void windowOpened(WindowEvent we)

Eg:

------

import java.awt.\*;

import java.applet.\*;

import java.awt.event.\*;

public class MouseEvents extends Applet implements Mouse

Listener,MouseMotionListener {

String msg = " ";

int mouseX=0,mouseY=0;

public void init() {

addMouseListener(this);

addMouseMotionListener(this);

}

public void mouseClicked(MouseEvent me) {

mouseX = 0;

mouseY = 10;

msg = "Mouse Clicked";

repaint();

}

public void mouseEntered(MouseEvent me) {

mouseX = 0;

mouseY = 10;

msg = "Mouse Entered";

repaint();

}

public void mouseExited(MouseEvent me) {

mouseX = 0;

mouseY = 10;

msg = "Mouse Exited";

repaint();

}

public void mousePressed(MouseEvent me) {

mouseX = me.getX();

mouseY = me.getY();

msg = "Down";

repaint();

}

public void mouseReleased(MouseEvent me) {

mouseX = me.getX();

mouseY = me.getY();

msg = "Up";

repaint();

}

public void mouseDragged(MouseEvent me) {

mouseX = me.getX();

mouseY = me.getY();

msg = "\*";

showStatus("Mouse Dragged at " + mouseX + ", " + mouseY);

repaint();

}

public void mouseMoved(MouseEvent me) {

showStatus("Mouse Moved at " + me.getX() + ", "+ me.getY());

}

public void paint(Graphics g) {

g.drawString(msg,mouseX,mouseY);

}

}

//<applet code="MouseEvents" width=500 height=500></applet>

Keyboard Events

----------------------

Key pressed, key released, key typed.

we can use the KeyListener

Eg:

-----

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class SimpleKey extends Applet implements KeyListener{

String msg=" ";

int X = 10,Y = 20;

public void init() {

addKeyListener(this);

requestFocus(); //set the input focus

}

public void keyPressed(KeyEvent ke) {

showStatus("Key Down");

msg="Key Pressed";

}

public void keyReleased(KeyEvent ke) {

showStatus("Key Up");

msg="Key released";

}

public void keyTyped(KeyEvent ke) {

msg += ke.getKeyChar();

repaint();

}

public void paint(Graphics g) {

g.drawString(msg,X,Y);

}

}

//<applet code="SimpleKey.class" width=300 height=100></applet>

Virtual Keys

-----------------

VK\_F1 - VK\_F12

VK\_A - VK\_Z

VK\_0 - VK\_9

VK\_ENTER

VK\_ESCAPE

VK\_DOWN

VK\_PAGE\_UP

VK\_LEFT

VK\_SHIFT

VK\_CANCEL

VK\_RIGHT

VK\_UP

VK\_PAGE\_DOWN

VK\_CONTROL

VK\_ALT

VK\_BACK

Demonstration of Virtual keycodes

---------------------------------------------

import java.awt.\*;

import java.applet.\*;

import java.awt.event.\*;

public class KeyEvents extends Applet implements KeyListener {

String msg = "";

int X = 10,Y = 20;

public void init() {

addKeyListener(this);

requestFocus();

}

public void keyPressed(KeyEvent ke) {

showStatus("Key Down");

int key = ke.getKeyCode();

switch(key) {

case KeyEvent.VK\_F1:

msg += "<F1>";

break;

case KeyEvent.VK\_F2:

msg += "<F2>";

break;

case KeyEvent.VK\_F3:

msg += "<F3>";

break;

case KeyEvent.VK\_PAGE\_DOWN:

msg += "<PgDn>";

break;

case KeyEvent.VK\_CANCEL:

msg += "<PgUp>";

break;

case KeyEvent.VK\_LEFT:

msg += "<Left Arrow>";

break;

case KeyEvent.VK\_RIGHT:

msg += "<Right Arrow>";

break;

}

repaint();

}

public void keyReleased(KeyEvent ke) {

showStatus("Key Up");

}

public void keyTyped(KeyEvent ke) {

msg += ke.getKeyChar();

repaint();

}

public void paint(Graphics g) {

g.drawString(msg,X,Y);

}

}

//<applet code="KeyEvents.class" width=300 height=100></applet>

Adapter Classes

---------------------

ComponentAdapter ComponentListener

ContainerAdapter ContainerListener

FocusAdapter FocusListener

KeyAdapter KeyListener

MouseAdapter MouseListener

MouseMotionAdapter MouseMotionListener

WindowAdapter WindowListener

Eg:

------

AdapterDemo.java

-------------------------

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class AdapterDemo extends Applet {

public void init() {

addMouseListener(new MyMouseAdapter(this));

addMouseMotionListener(new MyMouseMotionAdapter(this));

}

}

class MyMouseAdapter extends MouseAdapter {

AdapterDemo adapterDemo;

public MyMouseAdapter(AdapterDemo adapterDemo) {

this.adapterDemo = adapterDemo;

}

public void mouseClicked(MouseEvent me) {

adapterDemo.showStatus("Mouse Clicked");

}

}

class MyMouseMotionAdapter extends MouseMotionAdapter {

AdapterDemo adapterDemo;

public MyMouseMotionAdapter(AdapterDemo adapterDemo) {

this.adapterDemo = adapterDemo;

}

public void mouseDragged(MouseEvent me) {

adapterDemo.showStatus("Mouse Dragged");

}

}

//<applet code="AdapterDemo.class" width=400 height=300> </applet>

AdapterDemo1.java

--------------------------

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class AdapterDemo1 extends Applet {

String msg=" ";

public void init() {

addKeyListener(new MyKeyAdapter(this));

requestFocus();

}

public void paint(Graphics g) {

g.drawString(msg,100,50);

}

}

class MyKeyAdapter extends KeyAdapter {

AdapterDemo1 adapterdemo1;

public MyKeyAdapter(AdapterDemo1 adapterdemo1) {

this.adapterdemo1 = adapterdemo1;

}

public void keyTyped(KeyEvent ke) {

adapterdemo1.msg += ke.getKeyChar();

adapterdemo1.repaint();

}

public void keyPressed(KeyEvent ke) {

adapterdemo1.showStatus("Key Down");

}

}

//<applet code="AdapterDemo1" width=500 height=500></applet>

User Interface Components

------------------------------------

Object

Component

Label

Button

CheckBox

TextComponent

TextArea

TextField

Choice

List

ScrollBar

Common Methods

---------------------

setSize(int width,int height) - Resizes the correxponding component so tha it has width and height

setFont(font f) - Sets the font of the corresponding component

setEnabled(boolean b)- Enables or Disables the corresponding component, depending on the value of the parameter b.

setVisible(boolean b) - shows or hides the corresponding component depending on the value of the parameter b

setForeground(Color c) - sets the foreground color of the corresponding component.

setBounds(int x,int y,int width,int height) -Moves and resizes the corresponding component.

setBackground(Color c) - sets the background color of the component

Color getBackground() gets the background color of the corresponding component

getBounds() gets the bounds of the corresponding component in the form of a rectangle object

getFont() - gets the font of the corresponding component

getForeground() - gets the foreground color of the corresponding component

getSize() - Returns the size of the corresponding component in the form of a Dimension object.

Button

------

Constructor Description

------------ ------------

Button() Constructs a button with no label

Button(String label) Constructs a button with label specified.

Methods

-------

Method Description

------- -----------

addActionListener(ActionListener l) Adds the specified actionlistener to

receive the action events from the

corresponding button

getActionCommand() Returns the command name of the action fired by the corresponding button

getLabel() Returns the label of the corresponding button

setLabel(String label) sets the label of the button to the value specified

getSource() Returns the label of the command button

eg:

---

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class ButtonDemo extends Applet implements ActionListener {

String msg = " ";

Button yes,no,maybe;

public void init() {

yes = new Button("Yes");

no = new Button("No");

maybe = new Button("Undecided");

add(yes);

add(no);

add(maybe);

yes.addActionListener(this);

no.addActionListener(this);

maybe.addActionListener(this);

}

public void actionPerformed(ActionEvent ae) {

String str = ae.getActionCommand();

if(str.equals("Yes")) {

msg = "You pressed Yes.";

}

else if(str.equals("No")) {

msg = "You pressed No.";

}else {

msg = "You pressed undecided.";

}

repaint();

}

public void paint(Graphics g) {

g.drawString(msg,6,100);

}

}

//<applet code="ButtonDemo.class" width = 300 height = 100></applet>

CheckBox

---------

Constructor Description

------------ ------------

Checkbox() creates a checkbox with no label.

Checkbox(String label) Creates a checkbox with specified label

Checkbox(String label,boolean state) Creates a checbox with the specified label and sets the state

Checkbox(String label,CheckboxGroup group,boolean state)

Creates a checkbox with the specified label and sets the specified state and places it in the specified group. The position of the checkbox group and state can be interchanged

Method Description

------ -----------

getCheckboxGroup() Determines the group of the corresponding Checkbox

getLabel() Gets the name of the corresponding checkbox

getSelectedObjects() Returns an array(length l) containing the checkbox label or null if the checkbox is not selected.

getState() Detrmines if the checkbox is 'true' or 'false' state

getItem() returns the item label

setCheckboxGroup(CheckboxGroup g) sets the corresponding checkbox group to the specified one

setLabel(String label) Set the label of the corresponding checkbox to the value specified

setState(boolean state) sets the state of the corresponding checkbox to the value specified

CheckboxDemo.java

-------------------------

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class CheckboxDemo extends Applet implements ItemListener {

String msg = new String();

Checkbox Win98,winNT,solaris,mac;

public void init() {

Win98 = new Checkbox("Windows 98",null,true);

winNT = new Checkbox("Windows NT/2000");

solaris = new Checkbox("Solaris");

mac = new Checkbox("MacOS");

add(Win98);

add(winNT);

add(solaris);

add(mac);

Win98.addItemListener(this);

winNT.addItemListener(this);

solaris.addItemListener(this);

mac.addItemListener(this);

}

public void itemStateChanged(ItemEvent ie) {

repaint();

}

public void paint(Graphics g) {

msg = "Current state : ";

g.drawString(msg,6,80);

msg = " Windows 98: " + Win98.getState();

g.drawString(msg,6,100);

msg = " Windows NT/2000: " + winNT.getState();

g.drawString(msg,6,120);

msg = " Solaris: " + solaris.getState();

g.drawString(msg,6,140);

msg = " MacOS: " + mac.getState();

g.drawString(msg,6,160);

}

}

//<applet code="CheckboxDemo.class" width=300 height=200></applet>

CBGroup.java

--------------------

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class CBGroup extends Applet implements ItemListener {

String msg = new String();

Checkbox Win98,winNT,solaris,mac;

CheckboxGroup cbg;

public void init() {

cbg = new CheckboxGroup();

Win98 = new Checkbox("Windows 98",cbg,true);

winNT = new Checkbox("Windows NT/2000",cbg,false);

solaris = new Checkbox("Solaris",cbg,false);

mac = new Checkbox("MacOS",cbg,false);

add(Win98);

add(winNT);

add(solaris);

add(mac);

Win98.addItemListener(this);

winNT.addItemListener(this);

solaris.addItemListener(this);

mac.addItemListener(this);

}

public void itemStateChanged(ItemEvent ie) {

repaint();

}

public void paint(Graphics g) {

msg = "Current Selection: ";

msg += cbg.getSelectedCheckbox().getLabel();

g.drawString(msg,6,100);

}

}

//<applet code="CBGroup.class" width=500 height=500></applet>

Choice

-----------

Constructor Description

----------- ------------

Choice() Creates a new Choice item

Choice c = new Choice();

Method Description

------- -----------

add(String item) Adds an item to the corresponding choice menu

addItem(String item) Adds an item to the corresponding choice

getItem(int index) Gets the string at the specified index of the corresponding choice menu

getItemCount() Returns the number of items in the corresponding choice menu

getSelectedIndex() Returns the index of the currently selected item

getSelectedItem() Gets a representation of the current choice as a string

insert(String item, int index) Inserts the item into the corresponding choice at the specified position

remove(int position) Removes an item into the corresponding choice menu at the specified position

remove(String item) Removes the first occurence of item from the corresponding choice menu

removeAll() Removes all item from the corresponding choice menu

select(int pos) Sets the selected item in the corresponding choice menu to be the item at the specified position

select(String str) Sets the selected item in the coresponding Choice menu to be the item whose name is equal to the specified string

eg:

----

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class ChoiceDemo extends Applet implements ItemListener {

Choice os,browser;

String msg = new String();

public void init() {

os = new Choice();

browser = new Choice();

os.add("Windows 98");

os.add("Windows NT/2000");

os.add("Solaris");

os.add("MacOS");

browser.add("Netscape 1.1");

browser.add("Netscape 2.x");

browser.add("Netscape 3.x");

browser.add("Netscape 4.x");

browser.add("Internet Explorer 3.0");

browser.add("Internet Explorer 4.0");

browser.add("Internet Explorer 5.0");

browser.add("Lynx 3.4");

browser.select("Netscape 4.x");

add(os);

add(browser);

os.addItemListener(this);

browser.addItemListener(this);

}

public void itemStateChanged(ItemEvent ie) {

repaint();

}

public void paint(Graphics g) {

msg = "Current OS : ";

msg += os.getSelectedItem();

g.drawString(msg,6,120);

msg = "Current Browser : ";

msg += browser.getSelectedItem();

g.drawString(msg,6,140);

}

}

//<applet code="ChoiceDemo.class" width=500 height=500></applet>

ChoiceDemo1.java

---------------------------

import java.applet.\*;

import java.awt.\*;

import java.awt.event.\*;

public class ChoiceDemo1 extends Applet implements ActionListener

{

Choice names;

TextField tf;

Button ad,rm,ra;

Label na,cnt;

public void init()

{

setLayout(null);

na = new Label("Enter Name : ");

tf = new TextField(20);

names = new Choice();

ad = new Button("Add");

rm = new Button("Remove");

ra = new Button("RemoveAll");

cnt = new Label("Count : ");

na.setBounds(100,100,130,30);

tf.setBounds(240,100,100,30);

names.setBounds(100,150,100,30);

ad.setBounds(240,150,100,30);

rm.setBounds(240,190,100,30);

ra.setBounds(240,230,100,30);

cnt.setBounds(240,270,100,30);

add(na);

add(tf);

add(names);

add(ad);

add(rm);

add(ra);

add(cnt);

ad.addActionListener(this);

rm.addActionListener(this);

ra.addActionListener(this);

}

public void actionPerformed(ActionEvent ae)

{

String na;

if(ae.getSource() == ad)

{

na = tf.getText();

names.add(na);

tf.setText("");

cnt.setText("Count : " + names.getItemCount());

}

else if(ae.getSource() == rm)

{

int index = names.getSelectedIndex();

names.remove(index);

cnt.setText("Count : " + names.getItemCount());

}

else if(ae.getSource() == ra)

{

names.removeAll();

cnt.setText("Count : " + names.getItemCount());

}

}

}

//<applet code="ChoiceDemo1" width=400 height=400></applet>

Label

-----

For displaying text

Constructor Description

----------- ------------

Label() Constructs an empty label.

Label(String text) Constructs a string with the corresponding text

Label(String text,

int alignment) Constructs a string with the text, with the specified alignment. the alignment could be CENTER,LEFT,RIGHT

Method Description

------- ------------

getText() Gets the text of the coresponding label.

setText(String text) set the text for the corresponding label to the specified text

Eg:

----

import java.awt.\*;

import java.applet.\*;

public class LabelDemo extends Applet {

public void init() {

Label one = new Label("One");

Label two = new Label("Two");

Label three = new Label("Three");

add(one);

add(two);

add(three);

}

}

/\*

<applet code = "LabelDemo.class" width = 300 height = 100>

</applet>

\*/

List

----

List the items, the user can select a single or multiple items.

Constructors Description

------------ ------------

List() Creates a new scrolling list of items

List(int rows) Creates a new scrolling list of items with the specified number of visible lines

List(int rows,boolean

mutilplemode) Creates a new scrolling list of items to display the specified number of rows

Method Description

------- -----------

add(String item) Adds the specified item at the end of the scrolling list

add(String item,int index) Adds the specified item at the position specified

deselect(int index) deselects the item at the specified index

getItem(int index) Gets the item at the specified index

getItemCount() gets the number of items in the list

getItems() gets the items in the list

getRows() Gets the number of visible lines in the corresponding list

getSelectedIndexes() Gets the index of the selected item

in the list

getSelectedItem() Gets the selected item in the

corresponding list

select(int index) Selects the item at the specfied index in the corresponding list

setMultipleMode(boolean b) Sets the flag that allows multiple selection in the corresponding list

Eg:

----

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class ListDemo extends Applet implements ItemListener {

List os,browser;

String msg=new String();

public void init() {

os = new List(4,true);

browser = new List(4,false);

os.add("Windows 98");

os.add("Windows NT/2000");

os.add("Solaris");

os.add("MacOS");

browser.add("Netscape 1.1");

browser.add("Netscape 2.x");

browser.add("Netscape 3.x");

browser.add("Netscape 4.x");

browser.add("Internet Explorer 3.0");

browser.add("Internet Explorer 4.0");

browser.add("Internet Explorer 5.0");

browser.add("Lynx 2.4");

browser.select(1);

add(os);

add(browser);

os.addItemListener(this);

browser.addItemListener(this);

}

public void itemStateChanged(ItemEvent ae) {

repaint();

}

public void paint(Graphics g) {

int idx[];

msg = "Current OS: ";

idx= os.getSelectedIndexes();

for(int i=0;i<idx.length;i++)

msg += os.getItem(idx[i]) + " " ;

g.drawString(msg,6,120);

msg = "Current browser:";

msg += browser.getSelectedItem();

g.drawString(msg,6,140);

}

}

/\*

<applet code="ListDemo.class" width=500 height=500>

</applet>

\*/

ScrollBars

--------------

Vertical

Horizontal

ScrollBar Contructors

----------------------------

Scrollbar()

Scrollbar(int style)

Scrollbar(int style,int initial value,int thumbsize,int min, int max)

style

------

Scrollbar.VERTICAL

Scrollbar.HORIZONTAL

initial value - Starting value of the scrollbar

thumbsize - Defint the thumbsize

min - Minimum value

max - Maximum value

Methods

-----------

setValues(int initialvalue,int thumbsize,int min,int max) - Set the values

int getMinimum() - Get minimum value

int getMaximum() - Get Maximum value

int getValue() - Get the current value

void setUnitIncrement(int newincr) - Defualt 1,

void setBlockIncrement(int newIncr) - Set the block increment (ie) click at the centre space of the scrollbar

To process ScrollBar events, the AdjustmentListener is used. Each time a user interacts with a scroll bar, an AdjustmentEvent object is generated.

getAdjustmentType () - Used to determine the type of the adjustment.

BLOCK\_DECREMENT A Page Down event has been generated

BLOCK\_INCREMENT A Page Up event is generated

TRACK An absoulte tracking event is generated

UNIT\_DECREMENT The line down button in a scrollbar is pressed

UNIT\_INCREMENT The line up button in a scrollbar is pressed

Eg:

----

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class SBDemo extends Applet implements AdjustmentListener,MouseMotionListener {

String msg = new String();

Scrollbar vertSB,horzSB;

public void init() {

int width = Integer.parseInt(getParameter("width"));

int height= Integer.parseInt(getParameter("height"));

vertSB = new Scrollbar(Scrollbar.VERTICAL,0,1,0,height);

horzSB = new Scrollbar(Scrollbar.HORIZONTAL,0,1,0,width);

add(vertSB);

add(horzSB);

vertSB.addAdjustmentListener(this);

horzSB.addAdjustmentListener(this);

addMouseMotionListener(this);

}

public void adjustmentValueChanged(AdjustmentEvent ae) {

repaint();

}

public void mouseDragged(MouseEvent me) {

int x = me.getX();

int y = me.getY();

vertSB.setValue(x);

horzSB.setValue(y);

repaint();

}

public void mouseMoved(MouseEvent me){}

public void paint(Graphics g) {

msg = "Vertical: "+ vertSB.getValue();

msg += ", Horizontal: " + horzSB.getValue();

g.drawString(msg,6,160);

g.drawString("\*",horzSB.getValue(),vertSB.getValue());

}

}

/\*<applet code="SBDemo.class" width=500 height=500>

<param name="width" value="255">

<param name="height value="255">

</applet>

TextField

------------

For Single-line data Entry.

Constructors

----------------

TextField()

TextField(int length)

TextField(String str)

TextField(String str,int length)

Methods

------------

String getText() Used to get the text from TextField

void setText(String str) Used to set the text

String getSelectedText() returns the selected text

void select(int startindex,int endindex) Used to select the text

void setEditable(boolean canEdit) If it is false, the text cannot be altered

boolean isEditable() Used to check if the text is editable or not

void setEchoChar(char ch) Set the printed character in text (password field)

boolean echoCharIsSet() check if the echo char is set or not

char getEchoChar() get the echoed character

Eg:

---

import java.awt.\*;

import java.applet.\*;

import java.awt.event.\*;

public class TextFieldDemo extends Applet implements TextListener,ActionListener {

TextField name,pass;

public void init() {

Label namep = new Label("Name : ",Label.RIGHT);

Label passp = new Label("Password : ",Label.RIGHT);

name = new TextField(12);

pass = new TextField(8);

pass.setEchoChar('\*');

add(namep);

add(name);

add(passp);

add(pass);

name.addTextListener(this);

pass.addTextListener(this);

name.addActionListener(this);

pass.addActionListener(this);

}

public void textValueChanged(TextEvent te) {

repaint();

}

public void actionPerformed(ActionEvent ae) {

repaint();

}

public void paint(Graphics g) {

g.drawString("Name : "+ name.getText(),6,60);

g.drawString("Selected Text in name :" + name.getSelectedText(),6,80);

g.drawString("Password : " + pass.getText(),6,100);

}

}

//<applet code="TextFieldDemo" width=500 height=500></applet>

TextArea

------------

Multiline Data Entry

Constructors

-----------------

TextArea()

TextArea(int numLines,int numChars)

TextArea(String str)

TextArea(String str,int numLines,int numChars)

TextArea(String str,int numLines,int numChars,int sBars)

sBars

-----

SCROLLBARS\_BOTH -0

SCROLLBARS\_HORIZONTAL\_ONLY -2

SCROLLBARS\_NONE - 3

SCROLLBARS\_VERTICAL\_ONLY -1

Methods

-------------

String getText()

void setText(String str)

void select(int startIndex,int endIndex)

String getSelectedText()

boolean isEditable()

void setEditable(boolean canEdit)

void append(String str) - Appends the text at end of the line

void insert(String str,int index) - Insert the string in the correspoding index position

void repalceRange(String str,int startIndex,int endIndex) - Replace the string with the corresponding start and end index

Eg:

-----

import java.awt.\*;

import java.applet.\*;

public class TextAreaDemo extends Applet {

TextArea ta;

String str="This is a sample string for adding text in text area";

public void init() {

ta = new TextArea(str,10,25,3);

add(ta);

}

}

//<applet code="TextAreaDemo" width=500 height=500></applet>