Module V

String class - basics. Applet basics and methods. Event Handling: delegation event model, event classes, sources, listeners.

String class – basics

In java, string is basically an object that represents sequence of character values. An array of characters works same as java string.

The java.lang.String class is used to create string object.

How to create String object?

There are two ways to create String object:

1. By string literal
2. By new keyword

1) Java String literal is created by using double quotes. For Example:

```
1. String s="welcome";
```

2) By new keyword

```
1. String s=new String("Welcome");
```

Java String class methods

1. Java String compare

We can compare two strings in java on the basis of content

There are three ways to compare string in java:

1. **By equals() method**
2. **By = = operator**
3. **By compareTo() method**

a) String compare by equals() method

The String equals() method compares the original content of the string. It compares values of string for equality.

Example

```
class Test
 {
  public static void main(String args[])
  {
 String s1="Sachin";
 String s2="Sachin";
 String s3=new String("Sachin");
 String s4="Saurav";
 System.out.println(s1.equals(s2));//true
```

```
  System.out.println(s1.equals(s3));//true
  System.out.println(s1.equals(s4));//false
 }
}
```

```
Output:true
       true
       false
```

b) String compare by == operator

The = = operator compares references not values.

```
class Test
{
 public static void main(String args[])
{
  String s1="Sachin";
  String s2="Sachin";
  String s3=new String("Sachin");
  System.out.println(s1==s2);//true
  System.out.println(s1==s3);//false
 }
}
```

```
Output:true
       false
```

c) String compare by compareTo() method

The String compareTo() method compares values and returns an integer value that describes if first string is less than, equal to or greater than second string.

Suppose s1 and s2 are two string variables. If:

- **s1 == s2** :0
- **s1 > s2** :positive value
- **s1 < s2** :negative value

```
class Test
{
 public static void main(String args[])
{
   String s1="Sachin";
   String s2="Sachin";
   String s3="Ratan";
   System.out.println(s1.compareTo(s2));//0
   System.out.println(s1.compareTo(s3));//1(because s1>s3)
   System.out.println(s3.compareTo(s1));//-1(because s3 < s1 )
 }
}
```

```
Output:0
       1
       -1
```

2. String Concatenation in Java

In java, string concatenation forms a new string *that is* the combination of multiple strings. There are two ways to concat string in java:

1. By + (string concatenation) operator
2. By concat() method

a) String Concatenation by + (string concatenation) operator

Java string concatenation operator (+) is used to add strings. For Example:

```
class Test
{
 public static void main(String args[])
{
   String s="Sachin"+" Tendulkar";
   System.out.println(s);//Sachin Tendulkar
 }
}
```

```
Output:Sachin Tendulkar
```

b) String Concatenation by concat() method

The String concat() method concatenates the specified string to the end of current string.

```
class Test
```

```
{
public static void main(String args[])
{
  String s1="Sachin ";
  String s2="Tendulkar";
  String s3=s1.concat(s2);
  System.out.println(s3);//Sachin Tendulkar
 }
}
```

3)Substring in Java

A part of string is called substring. In other words, substring is a subset of another string.

Note: Index starts from 0.

You can get substring from the given string object by one of the two methods:

public String substring(int startIndex): This method returns new String object containing the substring of the given string from specified startIndex

public String substring(int startIndex, int endIndex): This method returns new String object containing the substring of the given string from specified startIndex to endIndex.

Example

```
String s="hello";
System.out.println(s.substring(0,2));//he
```

In the above substring, 0 points to h but 2 points to e (because end index is exclusive).

Example of java substring

```
public class Test
{
 public static void main(String args[])
{
    String s="SachinTendulkar";
   System.out.println(s.substring(6));//Tendulkar
   System.out.println(s.substring(0,6));//Sachin
 }
}
```

```
Tendulkar
Sachin
```

4)Java String toUpperCase() and toLowerCase() method

The java string toUpperCase() method converts this string into uppercase letter and string toLowerCase() method into lowercase letter.

```
String s="Sachin";
System.out.println(s.toUpperCase());//SACHIN
System.out.println(s.toLowerCase());//sachin
System.out.println(s);//Sachin(no change in original)
```

Output

```
SACHIN
sachin
Sachin
```

5)Java String trim() method

The string trim() method eliminates white spaces before and after string.

```
String s="  Sachin  ";
System.out.println(s);//  Sachin
System.out.println(s.trim());//Sachin
```

Output

```
  Sachin
Sachin
```

6)Java String charAt() method

The string charAt() method returns a character at specified index.

```
String s="Sachin";
System.out.println(s.charAt(0));//S
System.out.println(s.charAt(3));//h
```

Output

```
S
h
```

7)Java String length() method

The string length() method returns length of the string.

```
String s="Sachin";
System.out.println(s.length());//6
```

8) Java String contains

The java string contains() method searches the sequence of characters in this string. It returns true if sequence of char values are found in this string otherwise returns false.

Java String contains() method example

```
class ContainsExample
{
public static void main(String args[]){

String name="what do you know about me";
System.out.println(name.contains("do you know"));
System.out.println(name.contains("about"));
System.out.println(name.contains("hello"));
}}
```

Output

```
true
true
false
```

Write a java program to check whether given string is palindrome or not

```
import java.io.*;
class Palindrome
{
  public static void main(String  args[])
  {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("\nEnter a String : ");
    String str = br.readLine();
    String rev = "";
    int n = str.length();
    for(int i=n-1 ; i>=0 ; i--)
    {
      rev = rev + str.charAt(i);
```

```
        }
        if(str.equals(rev))
           System.out.println("\nGiven string is a palindrome");
        else
           System.out.println("\nGiven string is not a palindrome");
    }
}
```

Java Program to count number of Vowels present in a String

```
    import java.util.Scanner;
     class Vowels
    {
       public static void main(String args[])
    {
          String str;
          int count=0;
          char[] v = new char[] {'a','e','i','o','u'};
          BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
          System.out.print("\nEnter a String : ");
          String str = br.readLine();
          for(int i=0;i<str.length();i++)
           {
             for(int j=0;j<5;j++) {
                if((str.charAt(i)) == vowels[j])
                  {
                  count++;
                }
             }
          }
             System.out.println("Number of Vowels present in the given String: "+count);
       }}
```

Java Applet

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

Applets are small Java applications that can be accessed on an Internet server, transported over Internet, and can be automatically installed and run as apart of a web document.
An Applet class does not have any main() method.

Advantage of Applet

- There are many advantages of applet. They are as follows:
- It works at client side so less response time.
  -Secured
- It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

Applet Skelton(Applet Life cycle)

Applet is initialized.
Applet is started.
Applet is painted.
Applet is stopped.
Applet is destroyed.

The java.applet.Applet class 4 life cycle methods and java.awt.Component class provides 1 life cycle methods for an applet.
public void init(): is used to initialized the Applet. It is invoked only once.
public void start(): is invoked after the init() method or browser is maximized. It is used to start the Applet.
public void stop(): is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.
public void destroy(): is used to destroy the Applet. It is invoked only once.
The AWT provides 1 life cycle method of applet.
public void paint(Graphics g): is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.
When an applet is called AWT calls the methods in this order

init()
start()
paint()

When an applet is terminated these methods are called

stop()
destroy()

syntax of applet program

```
import java.awt.*;
import java.applet.*;
/*
<applet code=Skeleton.class width=200 height=60>
</applet>
*/
public class Skeleton extends Applet
{
 public void init()
 {
  // Initialization
 }
 public void start()
 {
  // Start or resume execution
 }
 public void stop()
 {
  //Suspends execution
 }
 public void destroy()
 {
  //shutdown activities
 }
 public void paint(Graphics g)
  {
    //redraw applet
  }
}
```

init()

first method to be called

initialization of variables

called only once

start()

called after init()

called each time when applet has stopped

ie when a browser is closed and opened

paint()

When redrawn

Graphics context – graphical environment

stop()

when browser is closed or leaves to another page

suspends the applet thread

destroy()

When the applet is completely removed from the memory

There are two ways to run an applet

By html file.

By appletViewer tool

Simple example of Applet by html file:

Java File

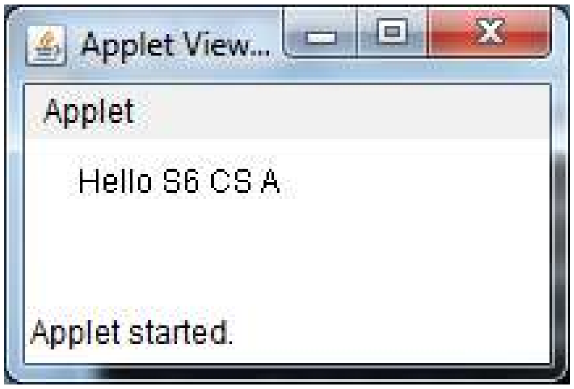
```
import java.awt.*;
import java.applet.*;
public class Simple extends Applet
{
 public void paint(Graphics g)
  {
    g.drawString("Hello S6 CS A ", 20,20);
  }
}
```

Html file

```
<html>
<body>
<applet code="Simpleapp.class" width="300" height="300">
</applet>
</body>
```

```
</html>
```

To execute the applet by html file, create an applet and compile it. After that create an html file and place the applet code in html file. Now click the html file.



```
Hello S6 CS A
```

Applet started.

Every Applet application must import two packages - java.awt and java.applet.
java.awt.* imports the Abstract Window Toolkit (AWT) classes. Applets interact with the user (either directly or indirectly) through the AWT. The AWT contains support for a window-based, graphical user interface. java.applet.* imports the applet package, which contains the class Applet. Every applet that you create must be a subclass of Applet class.
The class in the program must be declared as public, because it will be accessed by code that is outside the program.Every Applet application must declare a paint() method. This method is defined by AWT class and must be overridden by the applet. The paint() method is called each time when an applet needs to redisplay its output. Another important thing to notice about applet application is that, execution of an applet does not begin at main() method. In fact an applet application does not have any main() method.

<u>Simple example of Applet Using Applet viewer</u>

Applet viewer is a standard tool, that executes your applet in a window

```
import java.awt.*;
import java.applet.*;
public class Simple extends Applet
{
 public void paint(Graphics g)
  {
    g.drawString("Hello S6 CS A", 20,20);
  }
/*
<applet code=Simpleapp.class width=200 height=60>
</applet>
*/
}
```

Save file as Simpleapp.java

Compile javac Simpleapp.java

Execute appletviewer Simpleapp.java

Output same as above

<u>Displaying Graphics in Applet</u>

java.awt.Graphics class provides many methods for graphics programming.

Commonly used methods of Graphics class:

public abstract void drawString(String str, int x, int y): is used to draw the specified string.

public void drawRect(int x, int y, int width, int height): draws a rectangle with the specified width and height.

public abstract void fillRect(int x, int y, int width, int height): is used to fill rectangle with the default color and specified width and height.

public abstract void drawOval(int x, int y, int width, int height): is used to draw oval with the specified width and height.

public abstract void fillOval(int x, int y, int width, int height): is used to fill oval with the default color and specified width and height.

public abstract void drawLine(int x1, int y1, int x2, int y2): is used to draw line between the points(x1, y1) and (x2, y2).

public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer): is used draw the specified image.

public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle): is used draw a circular or elliptical arc.

public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle): is used to fill a circular or elliptical arc.

public abstract void setColor(Color c): is used to set the graphics current color to the specified color.

public abstract void setFont(Font font): is used to set the graphics current font to the specified font.

<u>Example of Graphics in applet:</u>

```
import java.applet.*;
import java.awt.*;
 public class Simpleapp extends Applet
{
        /*
        <applet code=Simpleapp.class width=200 height=60>
        </applet>
        */
 public void paint(Graphics g)
{
g.setColor(Color.red);
g.drawString("Welcome",50, 50);
g.drawLine(20,30,20,300);
g.drawRect(70,100,30,30);
g.fillRect(70,100,30,30);
g.drawOval(70,200,30,30);
 }
}
```

Displaying Image in Applet

Applet is mostly used in games and animation. For this purpose image is required to be displayed. The java.awt.Graphics class provide a method drawImage() to display the image.

Syntax of drawImage() method:

public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer): is used draw the specified image.

How to get the object of Image:

The java.applet.Applet class provides getImage() method that returns the object of Image. Syntax:

public Image getImage(URL u, String image){}

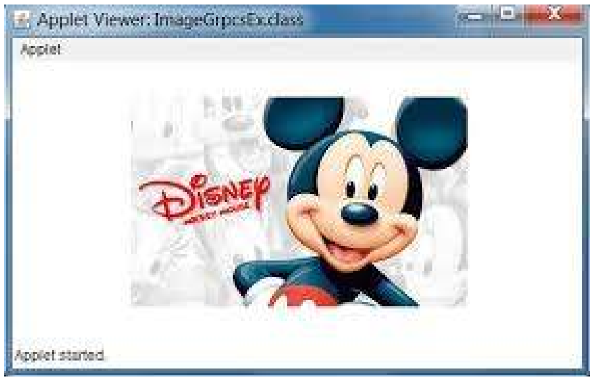
Other required methods of Applet class to display image:

public URL getDocumentBase(): is used to return the URL of the document in which applet is embedded.

Example of displaying image in applet:

```
import java.awt.*;
import java.applet.*;
```

```
  public class DisplayImage extends Applet
{
/*  <applet code="DisplayImage.class" width="300" height="300">
</applet>  */
  Image p;
    public void init() {
    p = getImage(getDocumentBase(),"sonoo.jpg");
  }
      public void paint(Graphics g) {
    g.drawImage(p, 30,30, this);
  }
       }
```



In the above example, drawImage() method of Graphics class is used to display the image. The 4th argument of drawImage() method of is ImageObserver object. The Component class implements ImageObserver interface. So current class object would also be treated as ImageObserver because Applet class indirectly extends the Component class.

Advantage of Applet

- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

Passing Parameters to Applets

We can get any information from the HTML file as a parameter. For this purpose, Applet class provides a method named getParameter().

Java File

```
public class UseParam extends Applet
{
public void paint(Graphics g){
String str=getParameter("msg");
g.drawString(str,50, 50);
}
  }
```

HTML File

```
<html>
<body>
<applet code="UseParam.class" width="300" height="300">
<param name="msg" value="Welcome to applet">
</applet>
</body>
</html>
```

Event Handling in Java

Applets are event driven program

Change in the state of an object is known as event

Package is- java.awt.event

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs.

Eg:-

Pressing a button

Enter a character using keyboard

Mouse clicks

Delegation event model

Java Uses the Delegation Event Model to handle the events.

Source generates an event and sends it to one or more listeners

Once received the listener processes the event and then returns

Source - The source is an object on which event occurs.

Listener - It is also known as event handler. Listener is responsible for generating response to an event.

Event Source

- Object that generates an event
- Due to change in internal state of the object
- Source must register listeners

Eg for registration:- addKeyListener(), addMouseMotionListener()

Event Listener

The object that is notified when an event occurs

Listeners must be

1. Registered with the source
2. Implement methods to receive and process the event

Eg: - MouseMotionListener interface

Important Event Classe and Interface

Event Class	Description	Listener Interface
ActionEvent	generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released also when the enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener

ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventListener
ContainerEvent	generated when component is added or removed from container	ContainerListener
AdjustmentEvent	generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

Event classes

1.Action Event Class

This class is defined in java.awt.event package

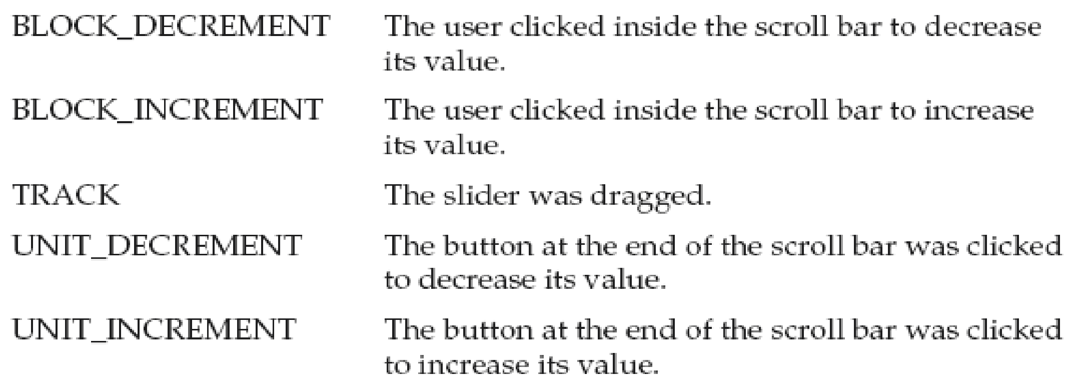
Defines 4 integers which defines modifiers which is passed along with the event **ALT_MASK,CTRL_MASK,META_MASK,SHIFT_MASK**

Modifiers indicates which modifier keys were pressed (ALT,CTRL,META,SHIFT)

getActionCommand() – For example, button press returns command name equal to label of the button

2.AdjustmentEvent Class

- Generated by **scrollbar**
- Define the following integer constants



BLOCK_DECREMENT	The user clicked inside the scroll bar to decrease its value.
BLOCK_INCREMENT	The user clicked inside the scroll bar to increase its value.
TRACK	The slider was dragged.
UNIT_DECREMENT	The button at the end of the scroll bar was clicked to decrease its value.
UNIT_INCREMENT	The button at the end of the scroll bar was clicked to increase its value.

Three **methods** are used in this class

getAdjustable() – returns object

getAdjustmentType() – returns one of the above constants.

getValue() – retuns amount of adjustment

3. **The ComponentEvent class**

   Generated when size, position or visibility of a **component** is changed (component such as button , text field etc)

COMPONENT_HIDDEN	The component was hidden.
COMPONENT_MOVED	The component was moved.
COMPONENT_RESIZED	The component was resized.
COMPONENT_SHOWN	The component became visible.

ComponentEvent has this constructor:

ComponentEvent(Component *src*, int *type*)

Method is **getComponent( )** – returns component that generated the event.

4) ContainerEvent Class

Generated when a component is added/removed from a container

Two Constants

COMPONENT_ADDED, COMPONENT_REMOVED

Two methods

getContainer( ) –obtain a reference to the container that generated this event.

getChild( )- returns a reference to the component that was added to or removed from the container

5)FocusEvent class

When a component gains or looses input focus

Two constants

FOCUS_GAINED, FOCUS_LOST

Two methods

getOppositeComponent( )

boolean isTemporary( ) – true if the change is temporary

InputEvent class

Sub classes are KeyEvent and MouseEvent

InputEvent defines several integer constants that represent any modifiers, such as the control key being pressed, that might be associated with the event.

<u>**6)KeyEvent Class**</u>

- **When keyboard input occurs**

  KEY_PRESSED, KEY_RELEASED, KEY_TYPED

getKeyChar( )- which returns the character that was entered.

getKeyCode( )- which returns the key code

7)MouseEvent Class

MOUSE_CLICKED	The user clicked the mouse.
MOUSE_DRAGGED	The user dragged the mouse.
MOUSE_ENTERED	The mouse entered a component.
MOUSE_EXITED	The mouse exited from a component.
MOUSE_MOVED	The mouse moved.
MOUSE_PRESSED	The mouse was pressed.
MOUSE_RELEASED	The mouse was released.
MOUSE_WHEEL	The mouse wheel was moved (Java 2, v1.4).

Two methods: getX() and getY()

8)The MouseWheelEvent Class

Two constants:WHEEL_BLOCK_SCROLL, WHEEL_UNIT_SCROLL

Methods:

- getWheelRotation( )- returns the no of rotational units
- getScrollType( ) –returns one of the above constants.
- getScrollAmount( )

9)ItemEvent Class

When a **check box** or a **list item** is clicked or when a **checkable menu item** is selected or deselected. Two constants:

DESELECTED, SELECTED

Methods: getItem( ) - obtain reference to the item

getItemSelectable( )- obtain reference to the selected item

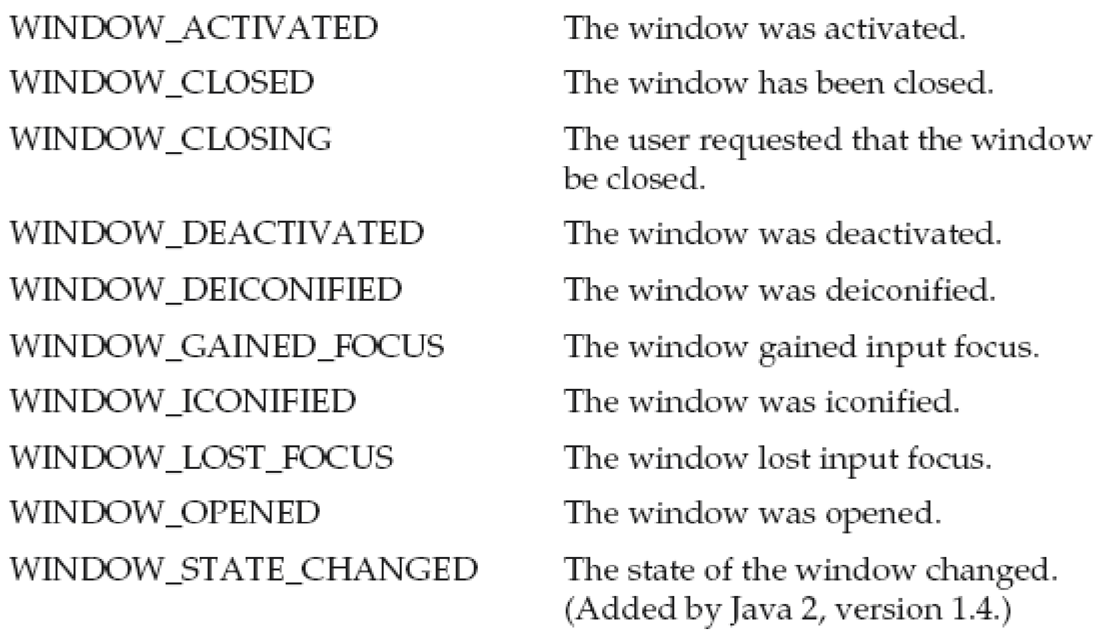
StateChange( ) -returns the state change(selected or deselected)

10)Text Event Class

When char are entered in the **text fields**

One constant: TEXT_VALUE_CHANGED

11)Window Event class



WINDOW_ACTIVATED	The window was activated.
WINDOW_CLOSED	The window has been closed.
WINDOW_CLOSING	The user requested that the window be closed.
WINDOW_DEACTIVATED	The window was deactivated.
WINDOW_DEICONIFIED	The window was deiconified.
WINDOW_GAINED_FOCUS	The window gained input focus.
WINDOW_ICONIFIED	The window was iconified.
WINDOW_LOST_FOCUS	The window lost input focus.
WINDOW_OPENED	The window was opened.
WINDOW_STATE_CHANGED	The state of the window changed. (Added by Java 2, version 1.4.)

getWindow( ) – returns the window object that generated the event.

Event Listener Interfaces

The ActionListener Interface

This interface defines the **actionPerformed( )** method that is invoked when an action event occurs. Its general form is shown here:

void actionPerformed(ActionEvent *ae*)

The AdjustmentListener Interface

This interface defines the **adjustmentValueChanged( )** method that is invoked when an adjustment event occurs. Its general form is shown here:

void adjustmentValueChanged(AdjustmentEvent *ae*)

The ComponentListener Interface

This interface defines four methods that are invoked when a component is resized, moved, shown, or hidden. Their general forms are shown here:

void componentResized(ComponentEvent *ce*)

void componentMoved(ComponentEvent *ce*)

void componentShown(ComponentEvent *ce*)

void componentHidden(ComponentEvent *ce*)

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. Their general forms are shown here:

void componentAdded(ContainerEvent *ce*)

void componentRemoved(ContainerEvent *ce*)

The FocusListener Interface

This interface defines two methods. When a component obtains keyboard focus, **focusGained( )** is invoked. When a component loses keyboard focus, **focusLost( )** is called. Their general forms are shown here:

void focusGained(FocusEvent *fe*)

void focusLost(FocusEvent *fe*)

The ItemListener Interface

This interface defines the **itemStateChanged( )** method that is invoked when the state of an item changes. Its general form is shown here:

void itemStateChanged(ItemEvent *ie*)

The KeyListener Interface

This interface defines three methods. The **keyPressed( )** and **keyReleased( )** methods are invoked when a key is pressed and released, respectively. The **keyTyped( )** method is invoked when a character has been entered. The general forms of these methods are shown here:

void keyPressed(KeyEvent *ke*)

void keyReleased(KeyEvent *ke*)

void keyTyped(KeyEvent *ke*)

The MouseListener Interface

This interface defines five methods. If the mouse is pressed and released at the same point, m**ouseClicked( )** is invoked. When the mouse enters a component, the **mouseEntered( )** method is called. When it leaves, **mouseExited( )** is called. The **mousePressed( )** and **mouseReleased( )** methods are invoked when the mouse is pressed and released, respectively.

The general forms of these methods are shown here:

void mouseClicked(MouseEvent *me*)

void mouseEntered(MouseEvent *me*)

void mouseExited(MouseEvent *me*)

void mousePressed(MouseEvent *me*)

void mouseReleased(MouseEvent *me*)

The MouseMotionListener Interface

This interface defines two methods. The **mouseDragged( )** method is called multiple times as the mouse is dragged. The **mouseMoved( )** method is called multiple times as the mouse is moved. Their general forms are shown here:

void mouseDragged(MouseEvent *me*)

void mouseMoved(MouseEvent *me*)

The MouseWheelListener Interface

This interface defines the **mouseWheelMoved( )** method that is invoked when the mouse wheel is moved. Its general form is shown here.

void mouseWheelMoved(MouseWheelEvent *mwe*)

The TextListener Interface

This interface defines the **textChanged( )** method that is invoked when a change occurs in a text area or text field. Its general form is shown here:

void textChanged(TextEvent *te*)

The WindowFocusListener Interface

This interface defines two methods: **windowGainedFocus( )** and **windowLostFocus( )**.

These are called when a window gains or losses input focus. Their general forms are shown here.

void windowGainedFocus(WindowEvent *we*)

void windowLostFocus(WindowEvent *we*)

The WindowListener Interface

This interface defines seven methods. The **windowActivated( )** and **windowDeactivated( )** methods are invoked when a window is activated or deactivated, respectively. If a window is iconified, the **windowIconified( )** method is called. When a window is deiconified, the **windowDeiconified( )** method is called. When a window is opened or closed, the **windowOpened( )** or **windowClosed( )** methods are called, respectively. The **windowClosing( )** method is called when a window is being closed. The general forms of these methods are

void windowActivated(WindowEvent *we*)

void windowClosed(WindowEvent *we*)

void windowClosing(WindowEvent *we*)

void windowDeactivated(WindowEvent *we*)

void windowDeiconified(WindowEvent *we*)

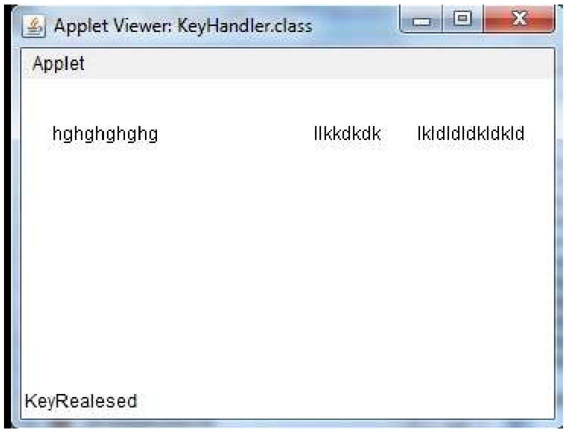
void windowIconified(WindowEvent *we*)

void windowOpened(WindowEvent *we*)

Program for handling keyboard events in Java

```
import java.awt.*;
import java.applet.*;

public class Test extends Applet implements KeyListener
{
                /*
                <applet code= Test.class width=200 height=60>
                </applet>
                */
 String msg="";
 public void init()
 {
  addKeyListener(this);
 }
 public void keyPressed(KeyEvent k)
 {
  showStatus("KeyPressed");
 }
 public void keyReleased(KeyEvent k)
 {
  showStatus("KeyRealesed");
 }
 public void keyTyped(KeyEvent k)
 {
  msg = msg+k.getKeyChar();
  repaint();
 }
 public void paint(Graphics g)
 {
  g.drawString(msg, 20, 40);
 }
}
```



```
hghghghghg          llkkdkdk    lkldldldkldkld
```

```
KeyRealesed
```

Example program for handling button events in java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
/*
<applet code= AppletWithButtons width=200 height=60>
</applet>
*/
public class  AppletWithButtons extends Applet implements ActionListener
{
  Button button1, button2;
        public void init()
        {
                button1 = new Button("Button 1");
                add(button1);
                button1.addActionListener(this);
                button2 = new Button("Button 2");
                add(button2);
                button2.addActionListener(this);
         }
```

```
                    public void actionPerformed(ActionEvent e)
                    {
                            if (e.getSource() == button1)
                                    System.out.println("Button 1 was pressed");
                            else
                                    System.out.println("Button 2 was pressed");
                    }
            }
```

Write an example program for mouse handling in java

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="Mouse.class" width=500 height=500>
</applet>
*/
public class Mouse extends Applet implements MouseListener,MouseMotionListener
{
          int X=0,Y=20;
          String msg="MouseEvents";
          public void init()
          {
                    addMouseListener(this);
                    addMouseMotionListener(this);
                    setBackground(Color.black);
                    setForeground(Color.red);
          }
          public void mouseEntered(MouseEvent m)
          {
                    setBackground(Color.magenta);
                    showStatus("Mouse Entered");
                    repaint();
          }
```

```
public void mouseExited(MouseEvent m)
{
        setBackground(Color.black);
        showStatus("Mouse Exited");
        repaint();
}
public void mousePressed(MouseEvent m)
{
        X=10;
        Y=20;
        msg="NEC";
        setBackground(Color.green);
        repaint();
}
public void mouseReleased(MouseEvent m)
{
        X=10;
        Y=20;
        msg="Engineering";
        setBackground(Color.blue);
        repaint();
}
public void mouseMoved(MouseEvent m)
{
        X=m.getX();
        Y=m.getY();
        msg="College";
        setBackground(Color.white);
        showStatus("Mouse Moved");
        repaint();
}
public void mouseDragged(MouseEvent m)
{
        msg="CSE";
```

```
                setBackground(Color.yellow);
                showStatus("Mouse Moved"+m.getX()+" "+m.getY());
                repaint();
        }
        public void mouseClicked(MouseEvent m)
        {
                msg="Students";
                setBackground(Color.pink);
                showStatus("Mouse Clicked");
                repaint();
        }
        public void paint(Graphics g)
        {
                g.drawString(msg,X,Y);
        }
}
```