S J P N Trust's

HIRASUGAR INSTITUTE OF TECHNOLOGY, NIDASOSHI.

Inculcating Values, Promoting Prosperity

Approved by AICTE, Recognized by Govt. of Karnataka and Affiliated to VTU Belagavi.

Accredited at 'A' Grade by NAAC

Programmes Accredited by NBA: CSE, ECE, EEE & ME

Subject: Object Oriented Concepts (18CS45)

Module 5: Event Handling, Applets and Java Swings

Dr. Mahesh G Huddar

Dept. of Computer Science and Engineering

Introduction

- Event means any activity that interrupts the current ongoing activity.
- For example: When a user clicks the mouse or presses some key from a keyboard during some processing, then it generates an event.
- In Java, events represent all activity that is carried out between the user and the application.
- Some specific actions are associated with these events.
- Java's Abstract Windowing Toolkit (AWT) conveys these actions to the programs.
- When the user interacts with a program let us say by clicking a button, the system creates an event representing the click action and passes it to the event-handling code within the program.
- This code is then responsible for giving an appropriate responses.

Introduction

Two Event Handling Mechanism

1. The event handling mechanism used in Java(1.0) is changed in modem versions of Java.
2. The event handling method used in the old version is still supported in latest version but it is not recommended to use these method of event handling.

Delegation Event Model (DEM)

- It defines standard and consistent mechanisms to generate and process events.
- Here the source generates an event and sends it to one or more listeners.
- The listener simply waits until it receives an event.
- Once it is obtained, It processes this event and returns.
- Listeners should register themselves with a source in order to receive an even notification.
- Notifications are sent only to listeners that want to receive them.

Three components of DEM

- Events
- Event Sources
- Event Listeners

Three components of DEM - Events

- In the delegation model, an event is an object that describes a state change in a source.
- It can be generated as a consequence of a person interacting with the elements in a graphical user interface.
- Eg: pressing a button, entering a character via the keyboard, selecting an item in a list, and clicking the mouse.
- Events may also occur that are not directly caused by interactions with a user interface.
- Eg: a timer expires, a counter exceeds a value, a software or hardware failure occurs, or an operation is completed

Three components of DEM – Event Sources

- A source is an object that generates an event. This occurs when the internal state of that object changes in some way. Sources may generate more than one type of event.
- A source must register listeners in order for the listeners to receive notifications about a specific type of event.
- Each type of event has its own registration method. Here is the general form:

  public void addTypeListener(TypeListener el)

  Type → the name of the event

  el → reference to the event listener

Three components of DEM – Event Sources

- When an event occurs, all registered listeners are notified and receive a copy of the event object. This is known as **multicasting** the event.
- Some sources may allow only one listener to register. The general form of such a method is this:

public void addTypeListener(TypeListener el) throws java.util.TooManyListenersException

- When such an event occurs, the registered listener is notified. This is known as **unicasting** the event.
- A source must also provide a method that allows a listener to unregister an interest in a specific type of event. The general form of such a method is this:

public void remove*TypeListener(TypeListener el)*

Three components of DEM – Event Classes

- These are the classes responsible for handling events in the event handling mechanism.
- The EventObject class is at the top of the event class hierarchy.
- It belongs to the java.util package.
- And other event classes are present in java.awtevent package.
- The getSource() and toString0 are the methods of the EventObject class.
- There is getId0 method belonging to java.awt.event package returns the nature of the event.

Three components of DEM – Event Classes

For example, if a keyboard event occurs, you can find out whether the event was key press or key release from the event object. Various event classes that are defined in java.awt.event class are –

1. An ActionEvent object is generated when a component is activated. For example if a button is pressed or a menu item is selected then this event occurs.
2. An AdjustmentEvent object is generated when scrollbars are used.
3. A TextEvent object is generated when text of a component or a text field is changed.

Three components of DEM – Event Classes

4. A ContainerEvent object is generated when component are added or removed from container.
5. A ComponentEvent object is generated when a component is resized, moved, hidden or made visible.
6. An ItemEvent is generated when an item from a list is selected. For example a choice is made or if checkbox is selected.
7. A FocusEvent object is generated when component receives keyboard focus for input.
8. A KeyEvent object is generated when key on keyboard is pressed or released.
9. A WindowEvent object is generated when a window activated, maximized or minimized.
10. A MouseEvent object is generated when a mouse is clicked, moved, dragged, released.

Event Listener Interfaces

- The task of handling an event is carried out by the event listener. When an event occurs, first of all, an event object of the appropriate type is created.
- This object is then passed to a Listener. A listener must implement the interface that has the method for event handling.
- The java.awt.event package contains definitions of all event classes and listener interface.
- An Interface contains constant values and method declaration.
- The methods in an interface are only declared and not implemented, i.e. the methods do not have a body.
- The interfaces are used to define behavior on the occurrence of event that can be implemented by any class anywhere in the class hierarchy.

Event Listener Interfaces

Various event listener interfaces are -

1. ActionListener: This interface defines the method actionPerformed() which is invoked when an ActionEvent occurs. The syntax of this method is void actionPerformed(ActionEvent act) where act is an object of ActionEvent class.
2. AdjustmentListener: This interface defines the method adjustmentValueChanged() which is invoked when an AdjustmentEvent occurs. The syntax of this method is

   void adjustmentValueChanged(ActionEvent act)

Event Listener Interfaces

3. TextListener: It has a method textChanged0 when a change in text area or text field occurs then this method is invoked.

   void textChanged(TextEvent tx)

4. ContainerListener: When a component is added to container then this interface is required. There are two methods for this interface and those are -

   void componentAdded(ContainerEvent ct)

   void componentRemoved(ContainerEvent ct)

   where ct represents the object of class ContainerEvent

Event Listener Interfaces

5. ComponentListener When component is shown, hidden, moved or resized then the corresponding methods are defined by ContainerListener interface. The syntax for the methods defined by this interface is

   void componentShown(ComponentEvent co)

   void componentHidden(ComponentEvent co)

   void componentMoved(ComponentEvent co)

   void componentResized(ComponentEvent co)

6. ItemListener: The itemStateChanged0 is the only method defined by the ItemListener interface. The syntax is

   void itemStateChanged(ItemEvent It)

Event Listener Interfaces

7. FocusListener: By this interface the methods related to keyboard focus are used. These methods are –

   void focusGained(FocusEvent fo) void focusLost(FocusEvent fo)

8. WindowFocusListener: By this interface the methods related to windows focus are used. These methods are –

   void windowGainedFocus(WindowEvent fo)

   void windowLostFocus(WindowEvent fo)

   These methods are called when window gains or loses focus.

Event Listener Interfaces

9. KeyListener This interface is defining the events such as keyPressed(),keyReleased() and keyTyped() are used. These methods are useful for key press, key release and when you type some characters.

void keyPressed(keyEvent k)

void keyReleased(keyEvent k)

void keyTyped(keyEvent k)

10. MouseListener: This interface defines five important methods for various activities such as mouse click, press, released, entered or exited. These are

void mouseClicked(MouseEvent m)

void mousePressed(MouseEvent m)

void mouseReleased(MouseEvent m)

void mouseEntered(MouseEvent m)

void mouseExited(MouseEvent m)

Event Listener Interfaces

11. 11. MouseMotionListener: For handling mouse drag and mouse move events the required methods are defined by MouseMotionListener interface. These methods are

    void mouseDragged(MouseEvent m)

    void mouseMoved(MouseEvent m)

12. WindowsListener: There are seven methods in which are related to windows activation and deactivation.

    void windowOpened(WindowEvent w)

    void windowClosed(WindowEvent w)

    void windowClosing(WindowEvent w)

    void windowActivated(WindowEvent w)

    void windowDeactivated(WindowEvent w)

    void windowIconified(WindowEvent w)

    void windowDeiconified(WindowEvent w)

Handling Mouse

- While handling the mouse events we use two interfaces MouseListener and MouseMotionListener.
- These interfaces has certain methods such as mouseClicked(), mousePressed(), mouseReleased(), mouseEntered(), mouseExited() and mouseDrag,ged(), mouseMoved() respectively.
- Let us see a simple Java program in which various mouse events are handled.

Handling Mouse

Handling Mouse

Program Explanation :

- In above program, we have used

```
import java.awt.event.*;
```

- because various commonly used events are defined in the package java.awt.event.

- The applet has to register itself as a listener to various events (here the same applet acts as a event source as well as event listener). Hence inside init() method applet registers itself as a listener by following statements

```
addMouseListener(this);
addMouseMotionListener(this);
```

Handling Mouse

- And then simple methods of mouse events are defined. The getXO and getYO methods return the current x and y positional values. To each of these methods an object of MouseEvent is passed which is shown by a variable 'm'.

Handling Keyboard

- When key from a keyboard is pressed then it causes an event.
- There are three commonly used methods from KeyListener interface and those are keyPressed(), keyReleased() and keyTyped().

Handling Keyboard

CSE, HIT, Nidasoshi

Handling Keyboard

Program Explanation :

- The keyPressed() event is for handling the event of pressing a key, similarly keyReleasedO event is for handling the event of release of key.
- Using keyTyped() method we can display the character typed on the screen. The only needed method for this is getKeyChar() which returns the currently typed character.
- Last but not the least, all these methods require one important method that has to be invoked in init() function and that is requestFocus(). This method has to be invoked to gain the focus in the init method and whenever you want to implement keyListener interface.

Introduction - Applets

- Applets are small Java programs that can be used in an internetworking environment.
- These programs can be transferred over the internet from one computer to another and can be displayed on various web browsers.
- Various applications of applets are in performing arithmetic operations, displaying graphics, playing sounds, creating animation, and so on.

Introduction - Applets

Following are the situations in which we need to use applet –

1. For displaying the dynamic web pages we need an applet. The dynamic web page is a kind of web page on which the contents are constantly changing. For example, an applet that can represent the sorting process of some numbers. During the process of sorting the positions of all the elements are changing continuously.
2. If we want some special effects such as sound, animation and much more then the applets are used.
3. If we want that the particular application should be used by any user who might be located remotely. Then in such situation, the applets are embedded into the web pages and can be transferred over the internet.

Advantages and Disadvantages of Applet

Advantages of Java applet

1. These are simple programs that provide good user interface.
2. The applets can run many platform such as Windows, Linux, Mac.
3. It is supported by most of the web browsers such as Internet Explorer, Netscape Navigator, Mozilla browser.
4. It is for supporting real time applications.
5. Using Applet client-server communication is possible.

Advantages and Disadvantages of Applet

Disadvantages of applet

1. It require Java plug-ins and can not be available by default on the web browsers.

2. It's GUI based programming is complex as compared to scripting languages/technologies such as HTML, DHTML, Flash and so on.
3. It takes lot of downloading time. Hence it is slower to execute.

Two Types of Applets

The applet can be created using two ways –

1) Using the Applet class. This class uses the Abstract Window Toolkit (AWT) to provide a Graphical User Interface (GUI)
2) The another way of creating an applet is using swing class JApplet. Swing components provide rich set of components while executing the Applet programs.

Applet Basics

<t> </t>

- All the applets are subclasses of class Applet.
- Applets are not stand-alone programs. Applets run either in web browser or using appletviewer provided by JDK.
- The execution of applet does not start using the main().
- For running the applet code in appletviewer, we need to add following comment statement in applet program,
- /* <applet code="Name_ofappletclass" width=300 height=100> </applet> */

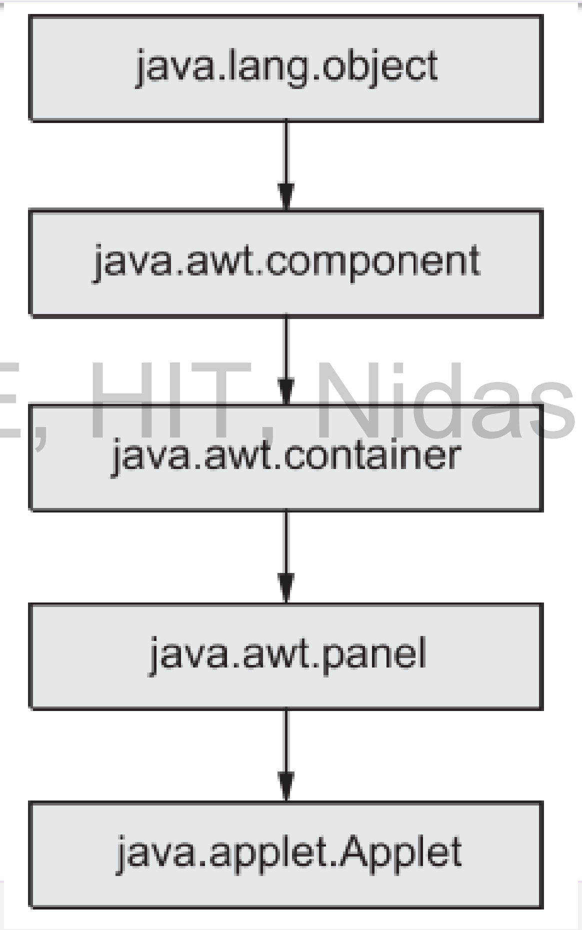
Applet Class Hierarchy

- The AWT allows us to use various graphical components.
- When we start writing any applet program we essentially import two packages namely - java.awt and java.applet.
- The java.applet package contains a class Applet that uses various interfaces such as AppletContext, AppletStub and AudioClip.
- The applet class is an extension of the Panel class belonging to java.awt package.

Applet Class Hierarchy

- To create a user-friendly graphical interface we need to place various components on the GUI window.
- There is a Component class from the java.awt package which derives several classes for components.
- These classes include Checkbox, Choice, List, buttons and so on.
- The Component class in java.awt is an abstract class.
- The class hierarchy for Applets is as shown in below Fig.

Applet Class Hierarchy

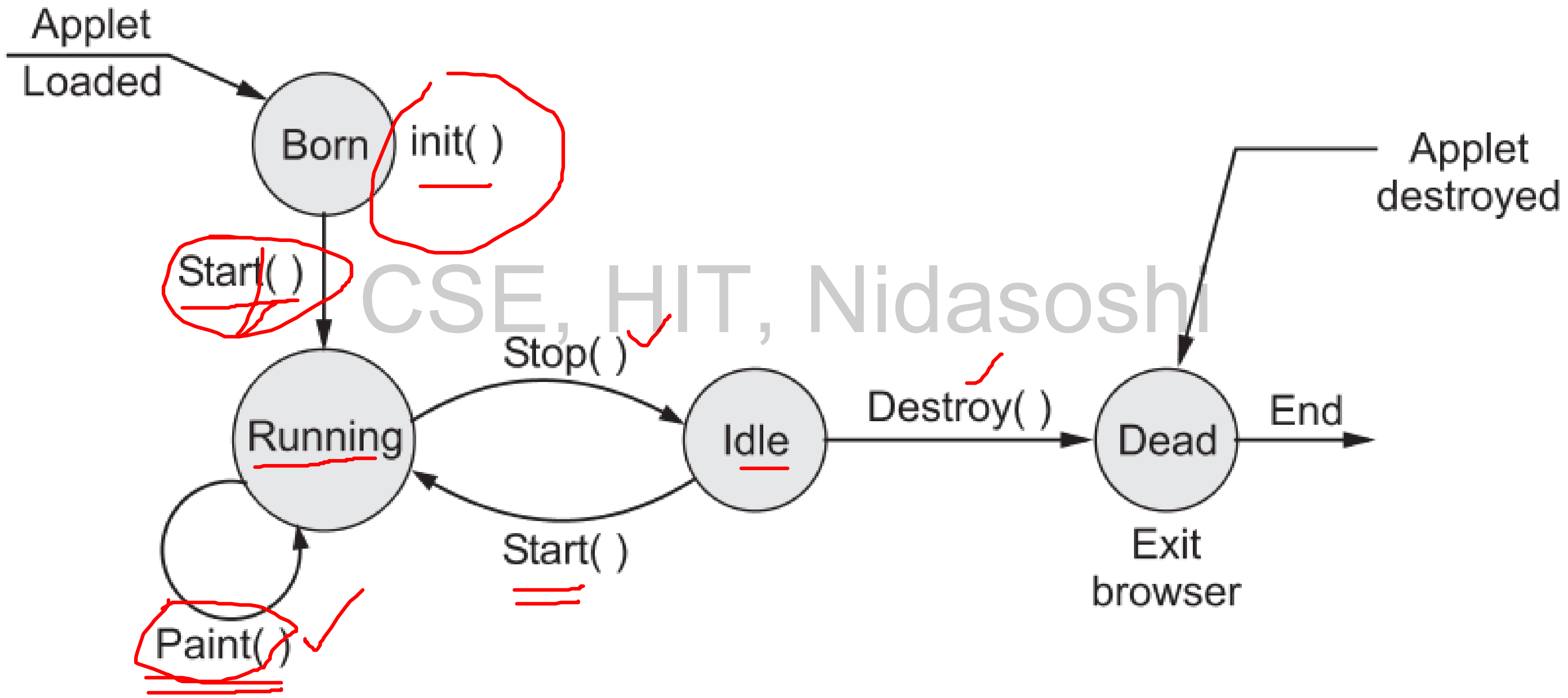


Applet Architecture

- There are various methods which are typically used in applet for initialization and termination purpose.
- These methods are

1. Initialization ✓
2. Running state ✓
3. Idle state ✓
4. Dead or destroyed state ✓

Applet Architecture



Applet Architecture

- When applet begins, the AWT calls following methods in sequence - a) init() b) start() c) paint()
- When applet is terminated following method are invoked in sequence. a) stop() b) destroy()

Applet Architecture

1. Initialization state

- When applet gets loaded it enters in the initialization state. For this purpose the init() method is used. In this method you can initialize the required variables. This method is called only once initially at the execution of the program. The syntax can be

```
public void init()
{
    //initialization of variables
}
```

Applet Architecture

In this method various tasks of initialization can be performed such as –

1. Creation of objects needed by applet
2. Setting up of initial values
3. Loading of image ✓
4. Setting up of colors. ✓

Applet Architecture

2. Running state

- When the applet enters in the running state, it invokes the start() method of Applet class.
- This method is called only after the init method. After stopping the applet when we restart the applet at that time also this method is invoked.
- The syntax can be

```
public void start()  { }
```

Applet Architecture

3. Display state

- Applet enters in the display state when it wants to display some output.
- This may happen when applet enters in the running state.
- The paint() method is for displaying or drawing the contents on the screen. The syntax is: **public void paint(Graphics g) { }**
- An instance of Graphics class has to be passed to this function as an argument. In this method various operations such as display of text, circle, line are invoked.

Applet Architecture

4. Idle state

- This is an idle state in which applet becomes idle.
- The stop() method is invoked when we want to stop the applet.
- When an applet is running if we go to another page then this method is invoked.
- The syntax is **public void stop() { }**

Applet Architecture

5. Dead state

- When applet is said to be dead then it is removed from memory.
- The method destroy() is invoked when we want to terminate applet completely and want to remove it from the memory.
- The syntax is

public void destroy() { } ✓

- It is good practice to call stop prior to destroy method.

An Applet Skeleton

- Normally the applet code makes use of two classes –

  Applet and Graphics.

- For the applet class the package java.applet is required.
- This class provides the applet's life cycle method such as init(), start() and paint().
- The Graphics class is supported by the package java.awt.
- Here is a simple applet

An Applet Skeleton

```
/*This is my First Applet program */
import java.awt.*;
import java.applet.*;
public class FirstApplet extends Applet
{
        public void paint(Graphics g)
        {
                g.drawString("This is my First Applet",50,30);
        }
}
```

An Applet Skeleton

- In the above given small applet program, the main intention is to display the message **"This is my First Applet".**
- Let us start from the beginning of the program –
- The first three lines represent a comment statement.
- Then next comes **import java.awt.*; import java.applet.*;**
- We always need these two packages to be imported into the program.
- The java.awt package consists of java awt classes. Here awt stands for Abstract Window Toolkit.
- The AWT provides support for window-based graphical interfaces such as for drawing screens, windows, buttons, text boxes, menus, and so on.
- The other imported package is java.applet. This is essential because we need to use the Applet class in our applet program

An Applet Skeleton

- which is included in java.applet package. The functionalities that are required to run applet inside the web browser are supported by java.applet.
- Then comes **public class FirstApplet extends Applet**
- The class FirstApplet is a subclass of class Applet. Hence the keyword extends is used.
- Java requires that your applet subclass (here it is FirstApplet) should be declared as public. Hence is the declaration! We have then defined a method
- **public void paint(Graphics g)**

An Applet Skeleton

- This method is used to paint something on the screen and it can be text, line, circle, rectangle or anything.
- Thus paint is a method which provides actual appearance on the screen. And this method requires a parameter to be passed as an object of class graphics.
- Therefore an object g of class Graphics is passed. Using this object the method drawstring of Graphics class is invoked. [Note that Graphics class is a part of java.awt package].
- **g.drawString("This is my First Applet",50,30);**
- To method drawstring, firstly we have passed a string which we want to be displayed, then 50 and 30 represents the position of the string on the screen i.e. x and y positions respectively.

How run Applet?

- There are two methods to run the applet
    1. Using web browser
    2. Using Appletviewer

- Let us learn both the methods with necessary illustrations

1. Using web browser

Step 1: Compile your Applet source program using javac compiler, i.e.

```
D:\test>javac FirstApplet.java
```

How run Applet?

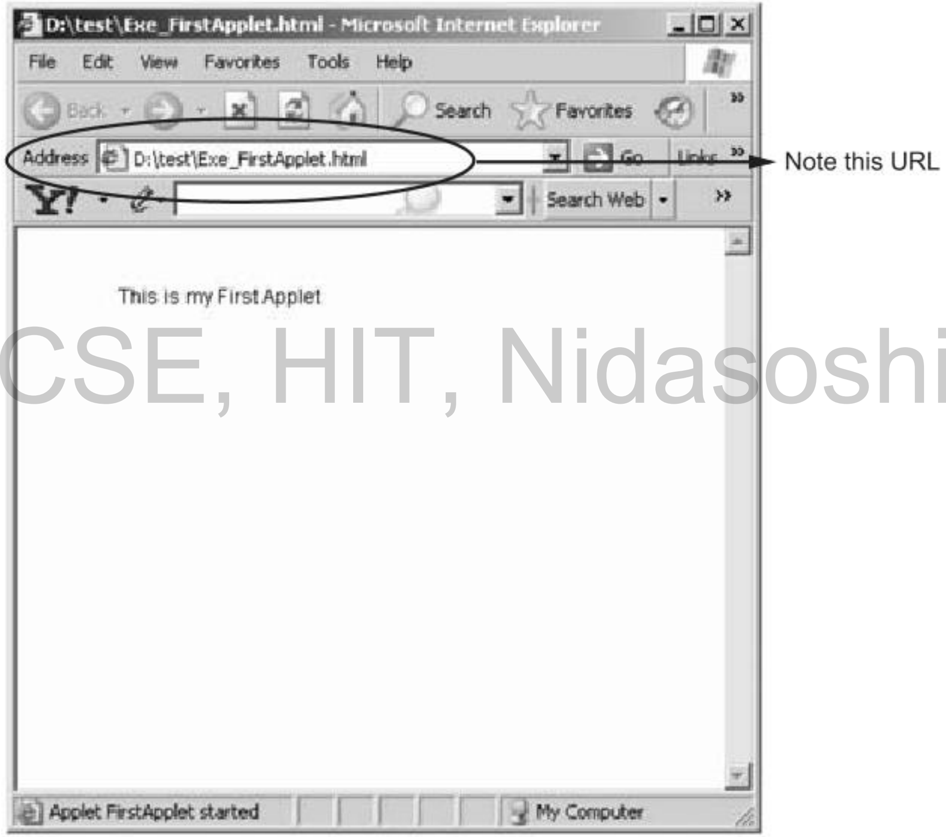
Step 2: Write following code in Notepad/Wordpad and save it with filename and extension .html. For example following code is saved as Exe_FirstApplet.html, The code is

```
<applet code="FirstApplet" width=300 height=100> </applet>
```

Step 3: Load html file with some web browser, This will cause to execute your html file.

How run Applet?

It will look this -


D:\test\Exe_FirstApplet.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address D:\test\Exe_FirstApplet.html Go Links
Search Web
Note this URL

This is my First Applet

Applet FirstApplet started

My Computer

How run Applet?

2. Using Appletviewer

Step 1 : To run the applet without making use of web browser or using command prompt we need to modify the code little bit. This modification is as shown below

How run Applet?

```
/* This is my First Applet program */
import java.awt.*;
import java.applet.*;
/* <applet code="FirstApplet" width=300 height=100> </applet> */
public class FirstApplet extends Applet
{
        public void paint(Graphics g)
        {
                g.drawString("This is my First Applet",50,30);
        }
}
```

How run Applet?

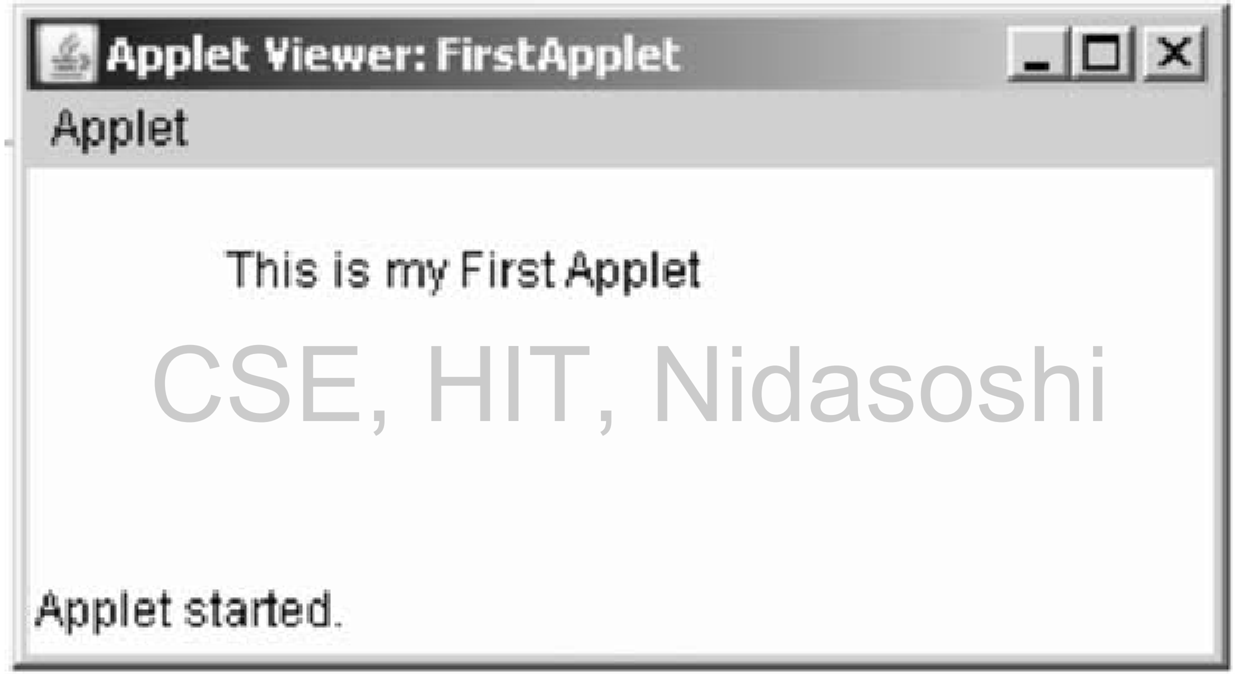
- In above code we have added one more comment at the beginning of the program
- <applet code="FirstApplet" width=300 height =100> </applet>
- This will help to understand Java that the source program is an applet with the name FirstApplet.
- By this edition you can run your applet program merely by Appletviewer command.

How run Applet?

Step 2 :

- D:\Atest>javac FirstApplet.java
- D:\test>Appletviewer FirstApplet.java
- And we will get

How run Applet?



```
This is my First Applet
```

```
Applet started.
```

- Note one fact about applet is that: it does not require main function.

Simple Applet Display Methods

- The output message can be displayed using the method drawstring().
- This method is defined in class Graphics.
- This method is typically called from update() or paint().
- Syntax of drawString

void drawString(String msg, int x, int y)

- where msg is a message to be displayed on the console.
- The x and y represent the coordinate x and y respectively at which the message is to be displayed.
- The upper-left corner of the screen is at co-ordinate 0,0.
- The drawString method does not recognize the newline character.
- You have to manually specify the message at the desired location using x and y coordinates.

Colors in Applet

- When you want to specify the color in order to make your applet colourful, there are some methods supported by AWT system.
- The syntax of specifying color is

```
Color(int R,int G,int B);
Color(int RGBVal);   //here RGB val denotes the value of color
```

Colors in Applet

- Similarly to set the background color of applet window we use

  void setBackground(Color colorname)

- where colorname denotes the name of the background color. In the same manner we can also specify the foreground color i.e. color of the text by using method

  void setForeground(Color colomame)

Colors in Applet

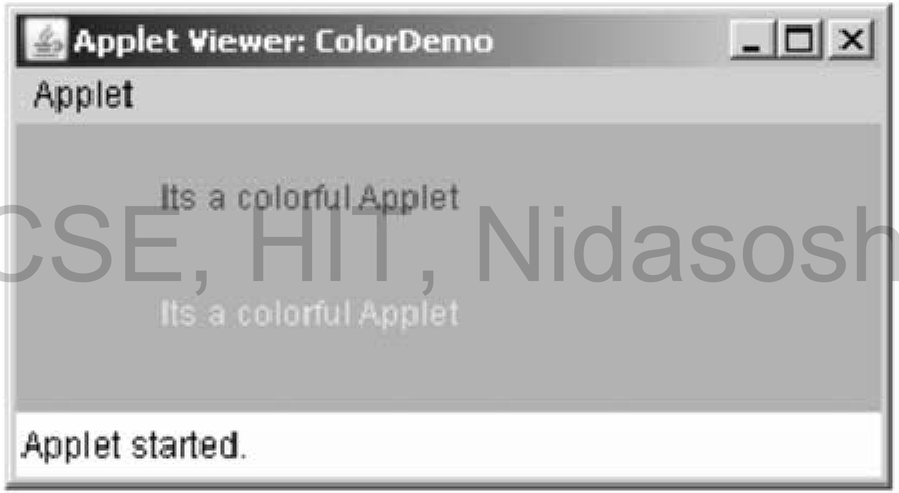
- We can specify the colorname using the Color object as follows –

Color.black	Color.lightGray
Color.blue	Color.darkGray
Color.magenta	Color.pink
Color.cyan	Color.gray
Color.orange	Color.red
Color.green	Color.white
Color.yellow	

Colors in Applet

```
import java.awt.*;
import java.applet.*;
/* <applet code="ColorDemo" width=300 height=100> </applet> */
public class ColorDemo extends Applet
{
        public void paint(Graphics g)
        {
                setBackground(Color.cyan);
                setForeground(Color.red);
                g.drawString("Its a colorful Applet",50,30);
                Color newColor=new Color(255,255,0);    //creating red+green=yellow color
                g.setColor(newColor);
                g.drawString("Its a colorful Applet",50,70);
        }
}
```

Colors in Applet



```
Its a colorful Applet

Its a colorful Applet
```

Applet started.

Colors in Applet

- In above program we see one more method related to color and that is setColor.
- To set some specific color this method is used.
- The color name has to be passed as a parameter to this method.
- We can create our own color by a method Color(int R,int G,int B).
- In above program we have created yellow color (Red + Green = Yellow) and then using newColor in setColor method the string 'It's a colourful applet' is written on the new line.

Requesting Repainting

- The repaint method requests an erase and redraw or update after some delay. There are four forms of repaint method.
- These are as follows –
- 1. public void repaint() This method indicates that AWT calls the component's update method as soon as possible. This is most common form of repaint method.
- 2. public void repaint(long time) For this methods, the time represents the maximum time in milliseconds before update.

Requesting Repainting

- 3. public void repaint(int x, int y,int width,int height) Repaints the specified rectangle of this component. The values x and y represents the co-ordinates and width and height represents the width and height of the rectangle.
- 4. public void repaint(long time,int x,int y,int width,int height) This method repaints the specified rectangle of this component within time milliseconds. The values x and y represents the co-ordinates and width and height represents the width and height of the rectangle.
- Following is a simple Applet which makes use of repaint method.

Requesting Repainting

```
import java.awt.*;
import java.awlevent.*;
import java.applet.*;
/* <applet code="ScatterDemo" width=300 height=200> </applet> */
public class ScatterDemo extends Applet implements ActionListener
{
        private static Color[] colors =
        {
                Color.black, Color.gray, Color.blue, Color.red,
                Color.yellow, Color.orange, Color.cyan,
                Color.pink, Color.magenta, Color.green
        };
```

Requesting Repainting

```
public void init()
{
        Button button = new Button("Click");
        add(button);
        button.addActionListener(this);
}
public void paint(Graphics g)
{
        for(int i=0; i < 100; ++i)
        {
                int x = (int)(Math.random()*100);
                int y = (int)(Math.random()*100);
                g.setColor(colors[(int)(Math.random()*10)]);
                g.drawString("Java",x,y);
        }
}
```

Requesting Repainting

```
        public void actionPerformed(ActionEvent event)
        {
                repaint(1000,10,10,50,50);
        }
}
```

Following commands can be given at the command prompt to execute an applet

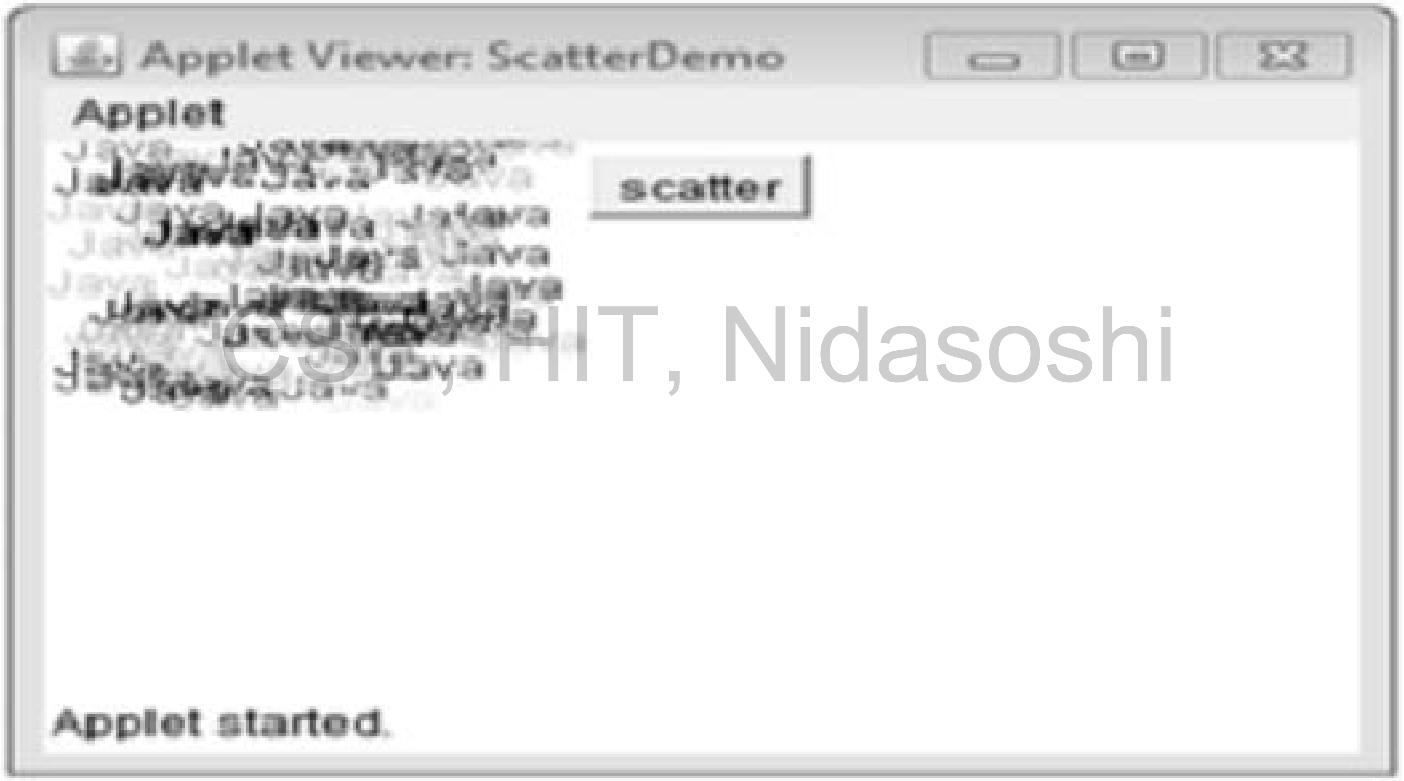
```
D:\JavaPrograms>javac ScatterDemo.java
D:\JavaPrograms>AppletViewer ScatterDemo.java
```

Requesting Repainting

Note

- When an applet window will be displayed just click the button pressed.
- You can see a small portion gets changed due to repaint method as the repaint method (repaint(1000,10,10,50,50)) repaint specific rectangular portion.

Requesting Repainting



Requesting Repainting

- Design an applet to display three buttons "Red", "green", and "blue". The color of the background changes according to the button presses by the user. Also, write the HTML code to display the applet.

Requesting Repainting

Step 1 :

```
<html>
        <body>
                <applet code=13gColorChange" width=350 height=200>
</applet>
        </body>
</html>
```

Step 2 :

```
<BgColorChange.java>
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
```

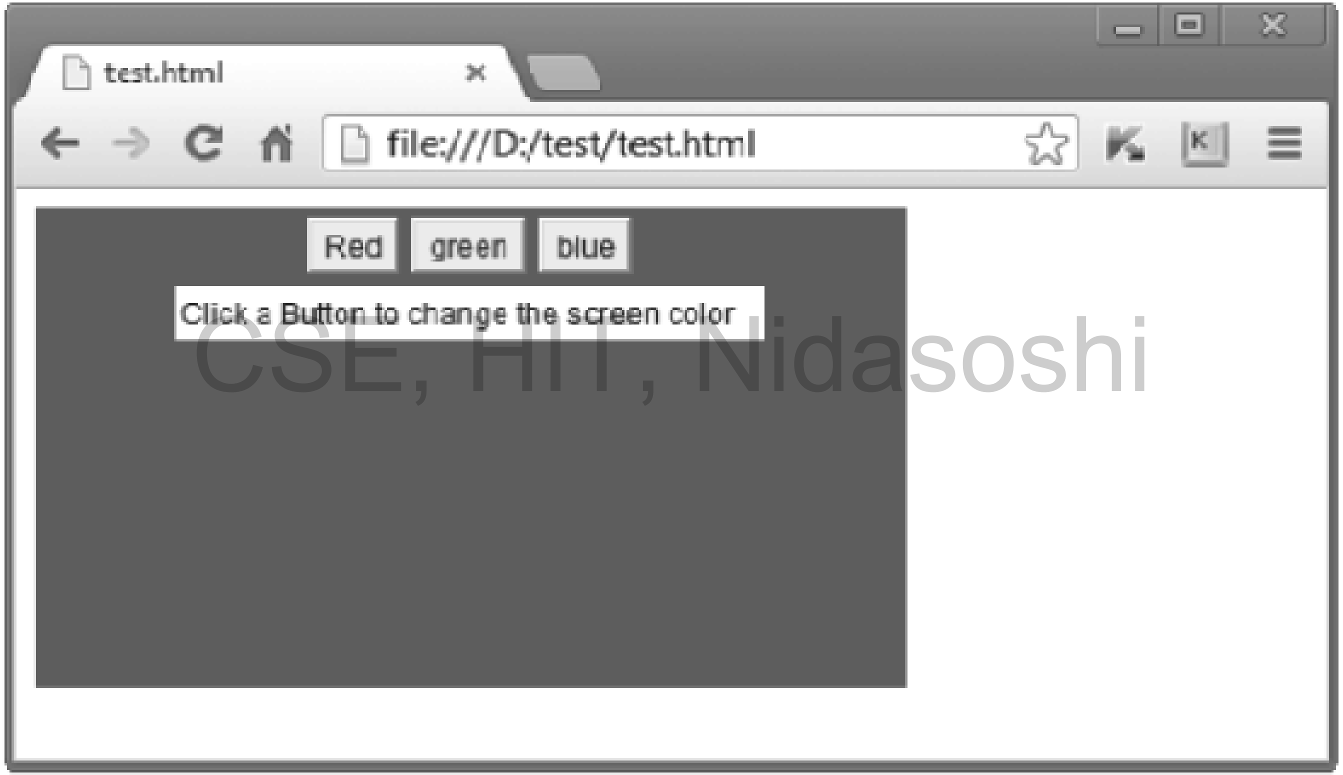
Requesting Repainting

```
public class BgColorChange extends Applet implements ActionListener
{
        int flag;
        Button buttonl=new Button("lled");
        Button button2=new Button("green");
        Button button3=new Button("blue");
        public void init()
        {
                add(buttonl);
                buttonl.addActionListener(this);
                add(button2);
                button2.addActionListener(this);
                add(button3);
                button3.adclActionListener(this);
                add(new Label("Click a Button to change the screen color"));
        }
```

Requesting Repainting

```
        public void paint(Graphics g)
        {
                if (flag==1) setBackground(Color.red);
                if (flag= =2) setBackground(Color.green);
                if (flag= =3) setBackground(Color.blue);
        }
        public void actionPerformed(ActionEvent e)
        {
                String str=e.getActionCommand();
                if(str.equals("Red")) { flag=1; repaint(); }
                if(str.equals("green")) { flag=2; repaint(); }
                if(str.equals("blue")) { flag=3; repaint(); }
        }
}
```

Requesting Repainting



Click a Button to change the screen color

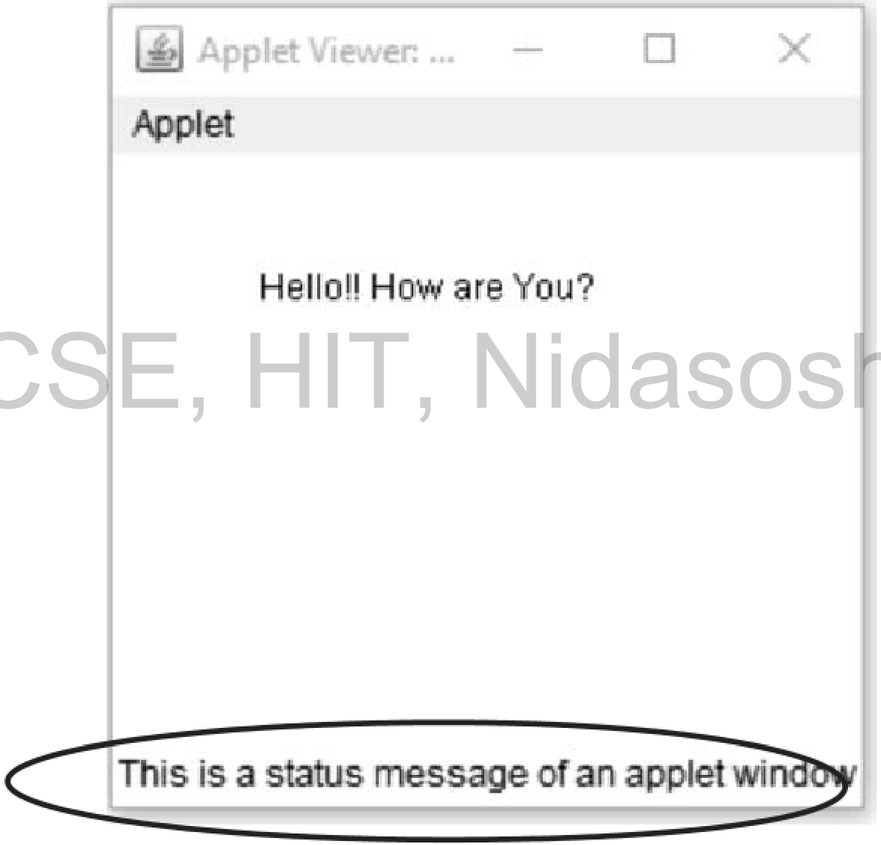
Using the Status Window

- The desired output can be displayed on the applet window.
- There is a status window, which is a good place to give the user feedback about what is occurring in the applet, suggest options, or possibly report some types of errors.
- The status window is also helpful for debugging because it gives you an easy way to output information about your applet.
- One can pass the message string as an argument to showStatus method. The syntax is void showStatus(String msg);
- Following example shows how to display message in status window

Using the Status Window

```
import java.applet.Applet;
import java.awt.Graphics;
/* <applet code="StatusWindowDemo" width=250 height =200> </applet> */
public class StatusWindowDemo extends Applet
{
        public void paint(Graphics g)
        {
                g.drawString("Hello!! How are You?", 50, 50);
                showStatus("This is a status message of an applet window");
        }
}
```

Using the Status Window

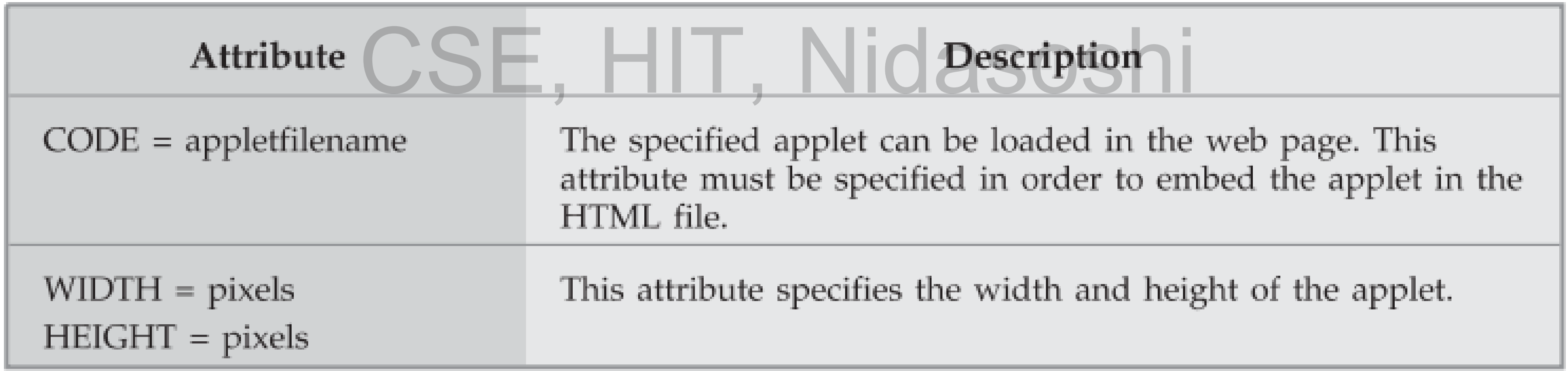


Hello!! How are You?

This is a status message of an applet window

The HTML APPLET tag

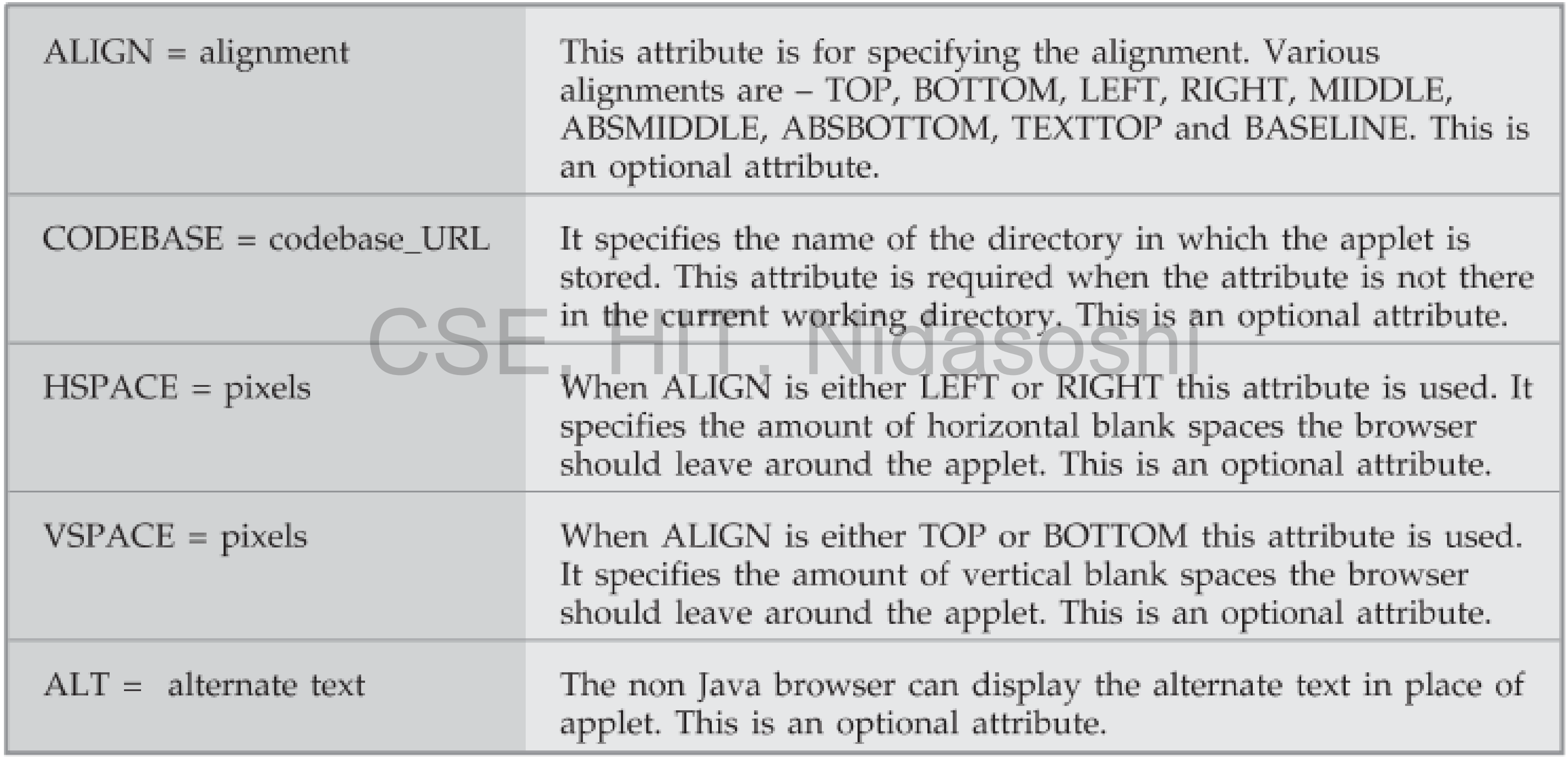
The <APPLET> tag can be specified with the help of various attributes. These attributes help to integrate the applet into overall design of the web page. Various attributes of APPLET tag are –



Attribute	Description
CODE = appletfilename	The specified applet can be loaded in the web page. This attribute must be specified in order to embed the applet in the HTML file.

WIDTH = pixels
HEIGHT = pixels
	This attribute specifies the width and height of the applet.

The HTML APPLET tag



ALIGN = alignment	This attribute is for specifying the alignment. Various alignments are – TOP, BOTTOM, LEFT, RIGHT, MIDDLE, ABSMIDDLE, ABSBOTTOM, TEXTTOP and BASELINE. This is an optional attribute.
CODEBASE = codebase_URL	It specifies the name of the directory in which the applet is stored. This attribute is required when the attribute is not there in the current working directory. This is an optional attribute.
HSPACE = pixels	When ALIGN is either LEFT or RIGHT this attribute is used. It specifies the amount of horizontal blank spaces the browser should leave around the applet. This is an optional attribute.
VSPACE = pixels	When ALIGN is either TOP or BOTTOM this attribute is used. It specifies the amount of vertical blank spaces the browser should leave around the applet. This is an optional attribute.
ALT = alternate text	The non Java browser can display the alternate text in place of applet. This is an optional attribute.

The HTML APPLET tag

- Develop an applet that receives two numerical values as input from the user and then displays the sum of these numbers on the screen.
- Write the HTML code that calls the applet.
- Solution :
- Step 1 :
- test.html
- <html> <body> <applet code="NumOperations" width=300 height=300> </applet> </body> </html>

The HTML APPLET tag

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class NumOperations extends Applet implements ActionListener
{
        Label L1,L2,L3;
        TextField T1,T2,T3;
        Button B1;
        public void init()
        {
                L1=new Label("Enter Num1 :");
                add(L1);
                T1 = new TextField(15);   //TextField for Numl
                add(T1);
```

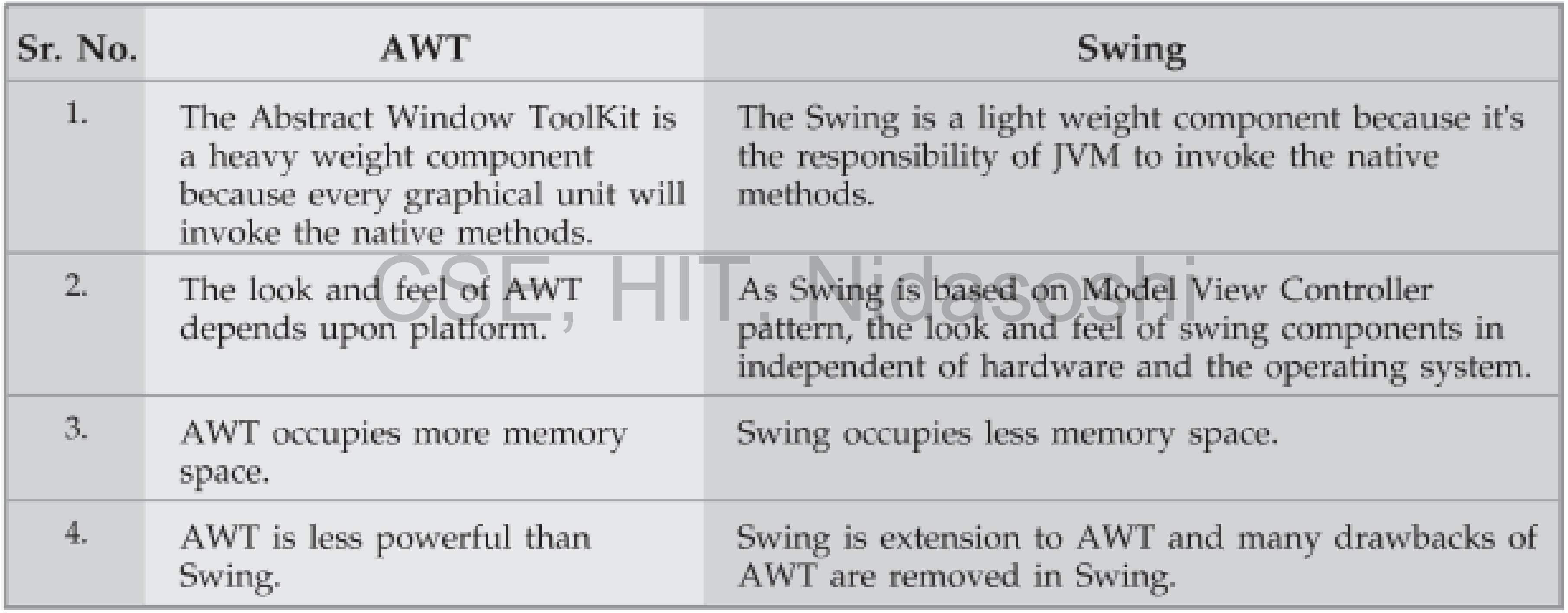
The HTML APPLET tag

```
        L2=new Label("Enter Num2 :");
        add(L2);
        T2=new TextField(15);    //TextField for Num2
        add(T2);
        L3=new Label("The Result :");
        add(L3);
        T3=new TextField(15);    //TextField for result
        add(T3);
        B1=new Button("Sum");
        add(B1); //Button to invoke addition operation
        B1.addActionListener(this);
}
```

The HTML APPLET tag

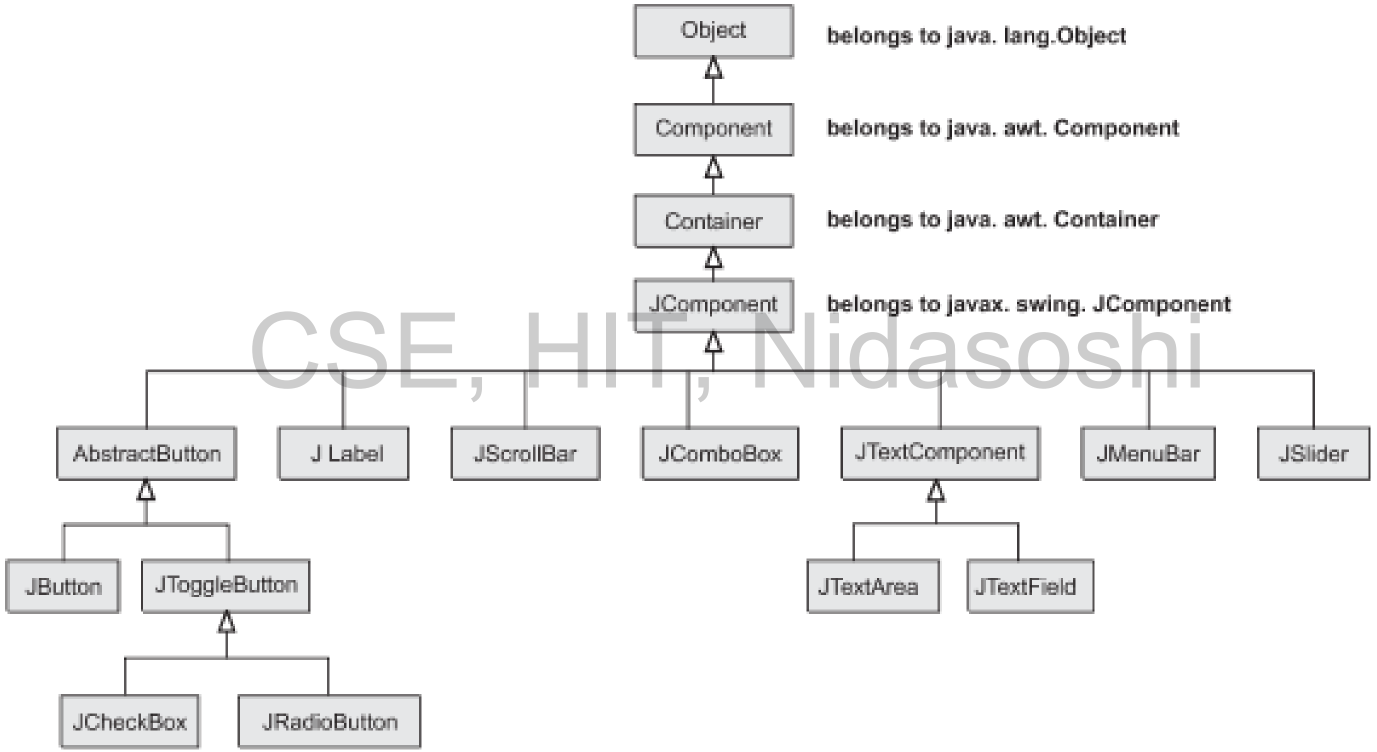
```
        public void actionPerformed(ActionEvent e)
        {
                if(e.getSource()==B1)
                {
                        int a=Integer.parseInt(T1.getText());
                        int b=Integer.parseInt(T2.getText());
                        int c=a+b;
                        T3.setText(String.valueOf(c));
                }
        }
}
```

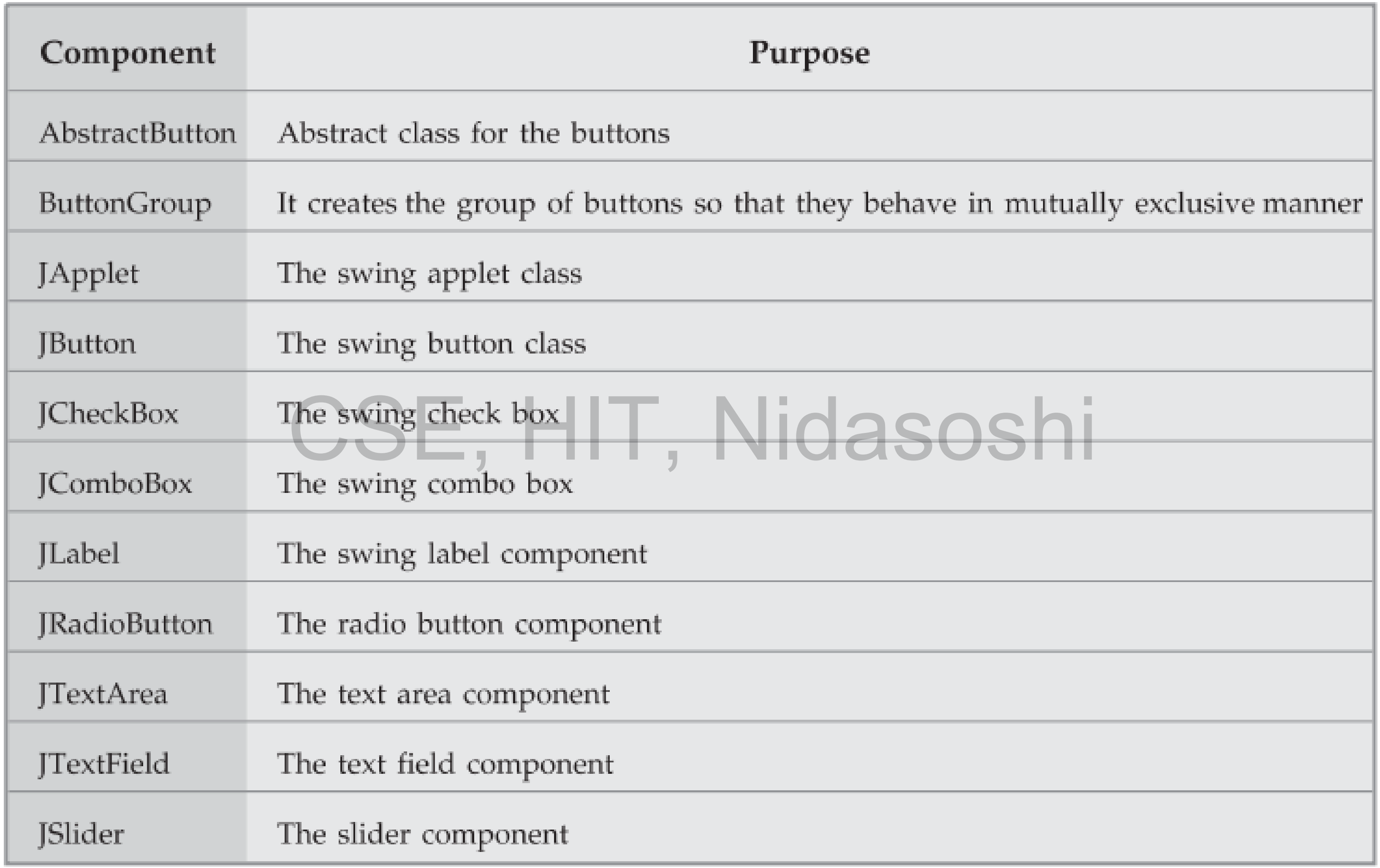
Swings



Sr. No.	AWT	Swing
1.	The Abstract Window ToolKit is a heavy weight component because every graphical unit will invoke the native methods.	The Swing is a light weight component because it's the responsibility of JVM to invoke the native methods.
2.	The look and feel of AWT depends upon platform.	As Swing is based on Model View Controller pattern, the look and feel of swing components in independent of hardware and the operating system.
3.	AWT occupies more memory space.	Swing occupies less memory space.
4.	AWT is less powerful than Swing.	Swing is extension to AWT and many drawbacks of AWT are removed in Swing.

Swings


Object
belongs to java. lang.Object
Component
belongs to java. awt. Component
Container
belongs to java. awt. Container
JComponent
belongs to javax. swing. JComponent
AbstractButton
J Label
JScrollBar
JComboBox
JTextComponent
JMenuBar
JSlider
JButton
JToggleButton
JTextArea
JTextField
JCheckBox
JRadioButton



Component	Purpose
AbstractButton	Abstract class for the buttons
ButtonGroup	It creates the group of buttons so that they behave in mutually exclusive manner
JApplet	The swing applet class
JButton	The swing button class
JCheckBox	The swing check box
JComboBox	The swing combo box
JLabel	The swing label component
JRadioButton	The radio button component
JTextArea	The text area component
JTextField	The text field component
JSlider	The slider component