Crazy notes untuk

Unit-IV :- APPLET

⇒ Applet :-

→ An applet represents a java byte code, that can be embeded with html code. Here html stands for hyper text markup language.

→ An applet can be represented as-

Applet = Java Byte code + Html code

→ By using the applet, we can design webpages, animation and games.

→ An applet does not contents the main method.

→ Applets are executed by using the command appletviewer.

* creating a Simple applet :-

```
import java.awt.*;
import java.applet.*;
Public class myapp extends applet
{
 Public Void init()
 {
 Set Background(Color.yellow);
 }
 Public void paint(Graphics g)
 {
 g.drow string("Hello applet");
 }
}
<html>
<applet code="myapp.class" height=300 width=200>
</applet>
</html>
```

dp :-

*Animation using applet:-

```
import java.awt.*;
import java.applet.*;
Public class Animation extends Applet
{
 Image picture;
 Public void init()
 {
  picture=getImage(getDocumentBase(),"plane.gif");
 }
 Public void paint(Graphics g)
 {
  for(int i=0; i<500; i++)
  {
   g.drawImage(picture, i, 30, this);
   try
   {
    Thread.sleep(100);
   }
   catch(Exception e)
   {
   }
  }
 }
}
<html>
<applet code="animation.class" height=600 width=700>
</applet>
</html>
```

O/P:-

⇒ Applet life cycle:-

→ An applet is born with init() method, and remove the applet by using destroy method. The process or the functioning from an applet from starting to ending of an applet is known as Applet life cycle.

→ The method in applet life cycle are-

① Public void init():-

→ This is the first method in applet life cycle where the programmer can be used this method to initialize variable and creating the code of an applet.

→ This method is executed only one's in a program-

② Public void start:-

This method is called after init() method and the execution of an applet can be started in this method.

③ public void stop():-

This method can be used to stop an applet for example, an applet with some animation can be stoped by using this method.

④ public void destroy():-

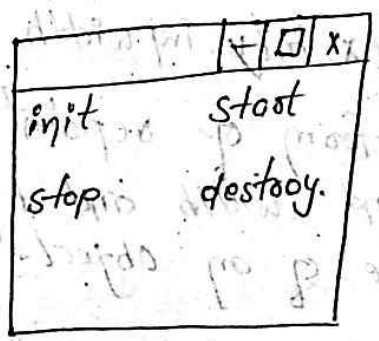
→ whenever a programmer want to remove an applet from the memory. we can use the destroy() method.

eg: import java.awt.*;
import java.applet.*;
Public class App1 extends Applet
{

```
string msg=" ";
Public void init()
{
setBackground(color.red);
setforeground(color.green);
font f=new font("Arial",font.BOLD,20);
set font(f);
msg+="init";
}
Public void start()
{
 msg+="start";
}
Public void stop()
{
 msg+="stop";
 }
Public void destroy()
{
 msg+="destroy";
}
```

```
<html>
<applet code="APPL.class" Height=200 weight=300>
<lapplet>
</html>
```

O/P:



⇒ Applet class:-

→ Every applet class is an extension of java.applet class.

→ It contents the following classes:-

① getting the applet parameter
② getting the network location of html file that contents an applet.
③ getting the applet class directory
④ print the message in the browser.
⑤ Resize an applet.

* The methods in applet class are-

paint():-

→ If we want to change the look of a component like changing the color animation of an object. we can use the paint() method.

→ The paint() method support with the help of graphics object.

update():-

→ By default, the update() feel the drawing area of a component with it's background color and send the paint method to it's object.

Repaint():-

→ It controls update() and paint() method.

→ The repaint() has two form.

Repaint()
Repaint(int x, int y, int width, int height);

→ The second form of repaint x and y are the co-ordinates, and width and height specifies the background size of an object.

*Applet parameter:-

→ By using the tag <param>, we can pass the parameter to an applet.
→ The <param> tag contents two attributes such as name and value.
→ name represents the name of the parameter and value indicates, the value of a parameter.

eg:
```
import java.awt.*;
import java.applet.*;
Public class Tax extends Applet
{
 String name, str;
 float salTax;
 Public void init()
 {
  name=getparameter("t1");
  str=getparameter("t2");
  sal = float.parsefloat(str);
  calculate Tax(sal);
 }
 Public void calculate Tax(float sal)
 {
  if(sal <=1000000)
    tax=0.0f;
  else if (sal<=2000000)
     tax=sal x 0.01f;
  else
     tax=sal x 0.02f;
 }
 Public void paint(Graphics g)
 {
```

```
g.drawString("Hello"+name,20,100);
g.drawString("your salary"+sal,20,100);
g.drawString("pay tax:"+tax,20,100);
}
}
<html>
<applet code="Tax.class" width=300 Height=200>
<param name="t1" value="Ravi">
<param name="t2" value="1500000.50>
</applet>
</html>
```

o/p's

Hellow Ravi
your salary 1500000.50
paytax = 150000.050

⇒ **Event delegation model:-**

→ when we create a component generally the component will be display on the screen but it is not capable for performing any action.

→ for example if we create a button the button will display on the screen but it can not perform any action even the user click on the button. but a user want to perform an action when he click on the button. The user have to add some listener interfaces, depending on the event, that is performed by the button an event represents specific action that can be performed on a component.

* **Event Listener:-**

→ A listener is an object, that notifies when an event occurs.

→ It has two major requirements-

① it must have registed with one or more sources to receive a notification about

specific type of event.

② It must be implemented methods to receive and process these notification.

* Some of the Listener interfaces and methods for a component.

d.m.s

component	Listener Interface	Listener method
1. Button	action listener	public void action performed (ActionEvent e)
2. Checkbox	item listener	public void ~~actionperformed~~ itemstatechange (ItemEvent e)
3. Checkboxgroup	item listener	public void action performed (ItemEvent e)
4. Text field	action listener	public void actionperformed (ActionEvent e)
5. Choice	action listener and item listener	public void actionperformed (ActionEvent e) and public void itemstatechange (ItemEvent e)
6. List	action listener and item listener	public void actionperformed (ActionEvent e), and public void itemstatechange (ItemEvent e)
7. Scrollbar	adjustment listener and mouse motion listener	public void adjustment value changed (adjustment Listener e) public void dragged (mouse Event e)
8. keyboard	key listener	public void keypressed (key Event e) " " keyReleased (" " e) " " keyTyped (" " e)
9. frame	window listener	public void windowclosed (window Event e)

* Event Source:-

→ A Source is an object that generates an event Source for more than one type of event.
→ A Source must register a listener in order to receive the notification about specific type of event.
→ Each event has it's own register methods.
→ This can be represented as—

Public void add type (Type Listener el)

→ In the above statement type is the name of the event and el is a reference to the event Listener.

⇒ Inner class:-

→ if we declare a class within another class that type of class can be known as inner class
→ Inner classes are used for security purpose
→ A class can not be declare by using the private keyword, but if a class is a member of another class means an inner class can be declare by using the private keyword.
→ The Syntax of another class is—

```
class outer class
{
private class inner
{
=
}
}
```

eg:-
```
class outer
{
```

```
int num;
private class inner
{
Public void input()
{
System.out.println("Inner class");
}
void display inner()
{
 inner in = new inner();
 in.input();
 }
}
class my
{
Public static void main(string args[])
{
 outer out=new outer();
    out.display inner();
  }
}
```

O/P:-

inner class

⇒ Adapter class:-

→ An adapter class provides an empty implement of all the method of listener interface.

→ These classes are useful when we want to receive and process some of the events that are handled by a particular listener interfaces.

→ The classes of an adapter class and they conced Listener interfaces are-

Adapted class	Listener class
→ Component adapter	→ Component Listener
→ Contained adapter	→ Container listener

→ focus adapter
→ key adapter
→ mouse adapter
→ mouse motion adapter
→ window adapter

→ focus listener
→ key listener
→ mouse listener
→ mouse motion listener
→ window listener.