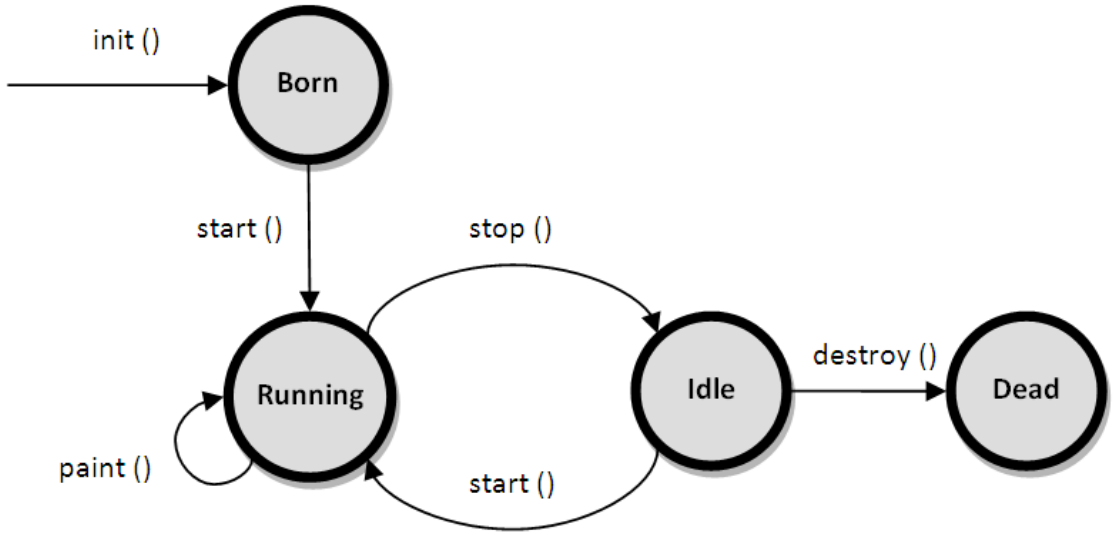
Applet in Java

- 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.
- After a user receives an applet, the applet can produce a graphical user interface. It has limited access to resources so that it can run complex computations without introducing the risk of viruses or breaching data integrity.
- Any applet in Java is a class that extends the java.applet.Applet class.
- An Applet class does not have any main() method. It is viewed using JVM. The JVM can use either a plug-in of the Web browser or a separate runtime environment to run an applet application.
- JVM creates an instance of the applet class and invokes **init()** method to initialize an Applet.

Applet Life Cycle

The life cycle of an applet is as shown in the figure below:



As shown in the above diagram, the life cycle of an applet starts with *init()* method and ends with *destroy()* method. Other life cycle methods are *start(), stop()* and *paint()*. The methods to execute only once in the applet life cycle are *init()* and *destroy()*. Other methods execute multiple times.

Below is the description of each applet life cycle method:

init(): The init() method is the first method to execute when the applet is executed. Variable declaration and initialization operations are performed in this method.

start(): The start() method contains the actual code of the applet that should run. The start() method executes immediately after the *init()* method. It also executes whenever the applet is restored, maximized or moving from one tab to another tab in the browser.

stop(): The stop() method stops the execution of the applet. The stop() method executes when the applet is minimized or when moving from one tab to another in the browser.

destroy(): The destroy() method executes when the applet window is closed or when the tab containing the webpage is closed. *stop()* method executes just before when destroy() method is invoked. The destroy() method removes the applet object from memory.

paint(): The paint() method is used to redraw the output on the applet display area. The paint() method executes after the execution of *start()* method and whenever the applet or browser is resized.

The method execution sequence when an applet is executed is:

- init()
- start()
- paint()

The method execution sequence when an applet is closed is:

- stop()
- destroy()

Applet HTML tag

The <applet> tag embeds a Java applet (mini Java applications) on the page. An applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included in a page.

Ex.

<applet code = demo.class width = 400 height = 200>

</applet>

Applet tag having three attribute

1. code – specifies name of the applet (.class file)
2. width – specifies width of the applet (in pixel)
3. height – specifies height of the applet (in pixel)

//program to demonstrate applet in html document

Code for demo.java

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

public class newClass extends Applet
{
  public void paint (Graphics g)
            {
    g.drawString("Hello World", 300, 150);
  }
}
```

Html code : saved as demo.html

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML applet Tag</title>
  </head>

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

</html>
```

To run an applet we require one of the following tools :

1. java enabled web browser (such as HotJava or Netscape)
2. Java appletviewer

Appletviewer is available with the JDK(java development kit) .we run the applet as follows:

C:\> appletviewer demo.html

Output :

Passing parameters to Applet

We can supply user defined parameter to an applet using <param …> tag .
Param tag is used inside the applet tag.
Param tag has two attributes :

1. Name
2. Value

we will show you how to pass some parameters to an applet and how to read those parameters in an applet to display their values in the output.

Steps to accomplish this task -:

1. To pass the parameters to the Applet we need to use the **param** attribute of **<applet>** tag.
2. To retrieve a parameter's value, we need to use the **getParameter()** method of **Applet** class.

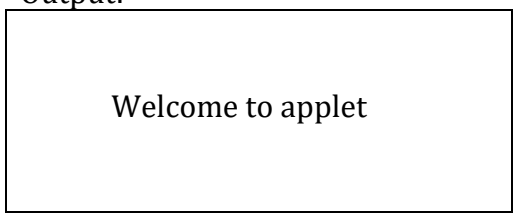
```
//program to demonstrate passing the parameters to applet
Java program saved as  : UseParam.java
import java.applet.Applet;
java.awt.Graphics;
public class UseParam extends Applet
{
public void paint(Graphics g)
```

```
{
String str=getParameter("msg");
g.drawString(str,50, 50);
}
}
```

Html document saved as : Myapplet.html

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

Output:



Repaint() Method

The **paint()** method is called by the JVM implicitly in two circumstances. One is when the first time frame is created and displayed. The other is when the frame is resized (by dragging the frame border with mouse) by the user. If the programmer would like to call the paint() method in the middle of the coding, he is permitted to call **repaint()** method. He is not permitted to call paint() method directly (Java repaint() Call paint()).
This method again call the paint method in the program. So the graphics get repaint again.
The second case, when paint() calls are generated is when the program calls repaint() or update().

The repaint() method is the one invoked by a program to do drawing. Their are 4 versions of this method but the one with no arguments is usually used. Drawing via repaint() most often takes place in response to user input.

repaint() ==> update() ==(usually calls)==> paint()

repaint() does not invoke paint() directly. It ***schedules*** a call to an intermediate method, update(). Finally, update() calls paint() (unless you override update).

The reason for this complexity is Java's support for concurrent programming.

Update() Method

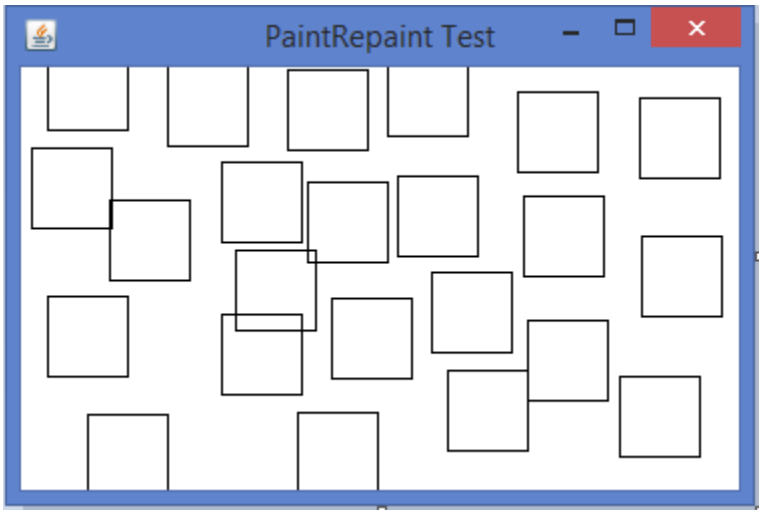
The **update()** method job is to **erase** the earlier drawings on the frame. On the cleared surface, the paint() method draws a fresh with the latest graphics. If the programmer is allowed to call paint() method directly, there is every possibility that he may forget to call the update() method explicitly. To avoid this and to make Java as a **robust language**, the designers do not allow to call paint() directly.

```
//program to demonstrate repaint( ) and update( )
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
public class PaintRepaintTest extends JPanel implements MouseListener
{
  private Vector v;
  public PaintRepaintTest()
            {
    v = new Vector();
    setBackground(Color.white);
    addMouseListener(this);
  }
  public void paint(Graphics g) // paint() method
            {
             super.paint(g);
    g.setColor(Color.black);
    Enumeration enumeration = v.elements();
    while(enumeration.hasMoreElements()) {
      Point p = (Point)(enumeration.nextElement());
      g.drawRect(p.x-20, p.y-20, 40, 40);
    }
  }
 public void mousePressed(MouseEvent me) {
    v.add(me.getPoint());
    repaint(); // call repaint() method
  }
  public void mouseClicked(MouseEvent me) {}
  public void mouseEntered(MouseEvent me) {}
  public void mouseExited(MouseEvent me) {}
```

```
  public void mouseReleased(MouseEvent me) {}
  public static void main(String args[])
        {
    JFrame frame = new JFrame();
    frame.getContentPane().add(new PaintRepaintTest());
    frame.setTitle("PaintRepaint Test");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLocationRelativeTo(null);
    frame.setSize(375, 250);
    frame.setVisible(true);
  }
}
```

In the above program, if we click on the screen able to draw squares. In the **mousePressed()** method, we can call the **repaint()** method.

Output



This program involves Frame and event handling mechanism.in which Frame creates a window as shown in output and when you press the mouse the mousepressed event will be occur then it will execute mousePressed method code and run the repaint( ) method and cursor goes to the paint method and draw a rectangle by using drawRect( ) method on the window.so we get the multiple rectangles on the window.

By [redacted]
Prof. Suchitra K. Kasbe
Department of Computer Science

****