Dr. Ambedkar Institute of Technology, Bengaluru

Department of Computer Science & Engg.

Faculty in-charge

Dr. Smitha Shekar B
Associate Professor
Dept.of CSE,
Dr.A.I.T

UG Programme

Subject Name: JAVA Programming
Subject Code: 18CS52

Dr. Smitha Shekar B

UNIT-1
APPLET FUNDAMENTALS

Life Cycle of an Applet,

Example of a simple Applet

Dr. Smitha Shekar B

Introduction

Dr. Smitha Shekar B

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.

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 platforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

Plugin is required at client browser to execute applet.

Concepts of Applets

- Applets are small applications that are accessed on an Internet server, transported over the Internet, automatically installed, and run as part of a Web document.

- After an applet arrives on the client, it has limited access to resources, so that it can produce an arbitrary multimedia user interface and run complex computations without introducing the risk of viruses or breaching data integrity.

Applications and Applets

- A Java program can be an application, applet or both.
- A Java application is a stand-alone, unrestricted program.

- A Java applet is a restricted program that relies on another program to execute.
- A Java applet executes under a Web browser or applet viewer.
- An applet is defined by extending the Applet or the JApplet class.

Differences between applets and applications

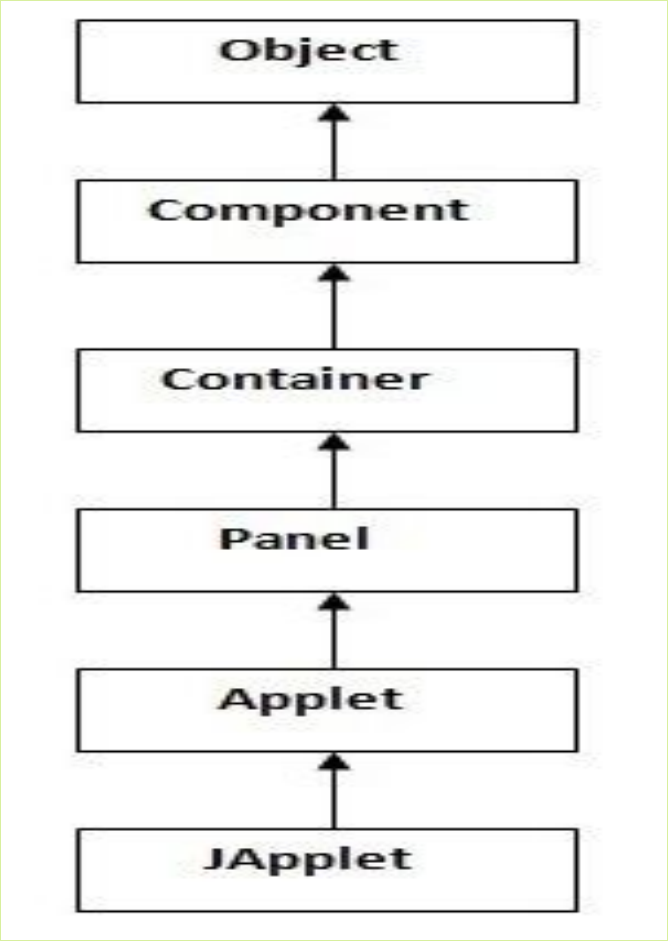
- **Java can be used to create two types of programs: applications and applets.**

- An ***application*** is a program that runs on your computer, under the operating system of that Computer(i.e an application created by Java is more or less like one created using C or C++).
- When used to create applications, Java is not much different from any other computer language.

- An ***applet*** is an application designed to be transmitted over the Internet and executed by a Java-compatible Web browser.
- An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image, sound file, or video clip.

- The important difference is that an applet is an *intelligent program*, not just an animation or media file(i.e an applet is a program that can react to user input and dynamically change—not just run the same animation or sound over and over).
- Applications require main method to execute.
- Applets do not require main method.
- Java's console input is quite limited
- Applets are graphical and window-based.

Hierarchy of Applet



Applet class extends Panel. Panel class extends Container which is the subclass of Component.

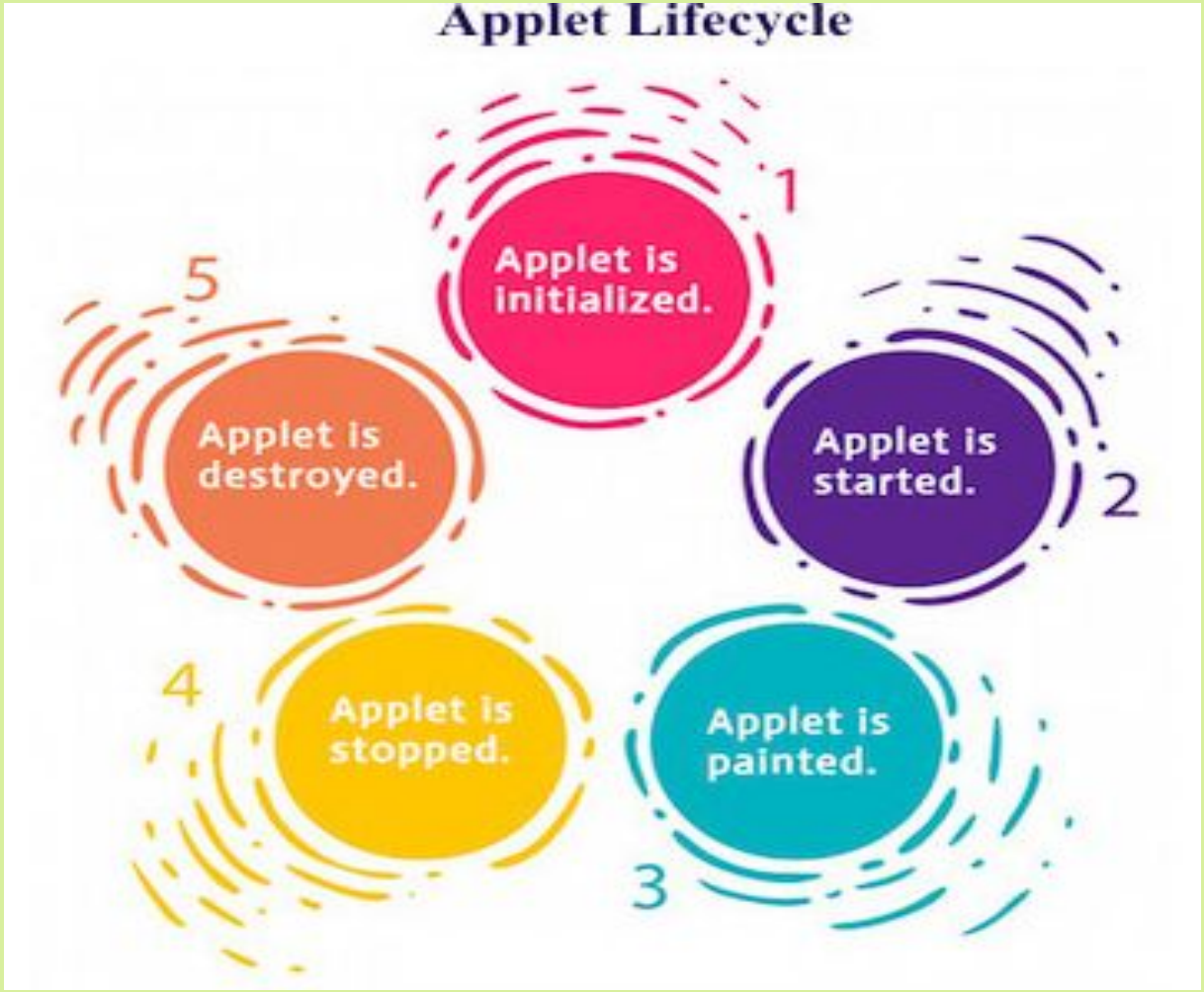
Life cycle of an applet

- Applets life cycle includes the following methods
  1. **init( )**
  2. **start( )**
  3. **paint( )**
  4. **stop( )**
  5. **destroy( )**

- When an applet begins, the AWT calls the following methods, in this sequence:
  init( )
  start( )
  paint( )

- When an applet is terminated, the following sequence of method calls takes place:
  stop( )
  destroy( )

Applet Life Cycle



Dr. Smitha Shekar B

Life cycle methods for Applet

The java.applet.Applet class has **4** life cycle methods and java.awt.Component class provides **1** life cycle method for an applet.

java.applet.Applet class

For creating any applet java.applet.Applet class must be inherited. It provides **4 life cycle methods** of 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.

java.awt.Component class

The Component class 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.

- applets – Java program that runs within a Java-enabled browser, invoked through a "applet" reference on a web page, dynamically downloaded to the client computer
- **Example**

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

- There are two ways to run an applet:

1. Executing the applet within a Java-compatible Web browser, such as Netscape Navigator, Google Chrome,IE
2. Using an applet viewer, such as the standard JDK tool, **appletviewer**.

- An appletviewer executes your applet in a window. This is generally the fastest and easiest way to test an applet.
- To execute an applet in a Web browser, you need to write a short HTML text file that contains the appropriate APPLET tag.

```
<applet code="SimpleApplet.class" width=200 height=60>
</applet>
```

- **init( ):** The **init( )** method is the first method to be called. This is where you should initialize variables. This method is called only once during the run time of your applet.

- **start( ):** The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped.

  - Whereas **init( )** is called once—the first time an applet is loaded—**start( )** is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at **start( )**.

- **paint( ):** The **paint( )** method is called each time applet's output must be redrawn.
- **paint( )** is also called when the applet begins execution.
  - Whatever the cause, whenever the applet must redraw its output, **paint( )** is called.
  - The **paint( )** method has one parameter of type **Graphics**. This parameter will contain the graphics context, which describes the graphics environment in which the applet is running.
  - This context is used whenever output to the applet is required.

- **stop( ):** The **stop( )** method is called when a web browser leaves the HTML document containing the applet—when it goes to another page.
  - for example. When **stop( )** is called, the applet is probably running.
  - Applet uses **stop( )** to suspend threads that don't need to run when the applet is not visible.
  - To restart **start( )** is called if the user returns to the page.

- **destroy( ):** The **destroy( )** method is called when the environment determines that your applet needs to be removed completely from memory.
  - The **stop( )** method is always called before **destroy( )**.

Types of applets

- Applets are two types
  1. Simple applets
  2. JApplets

- Simple applets can be created by extending Applet class
- JApplets can be created by extending JApplet class of **javax.swing.JApplet** package.

Simple Programs

Dr. Smitha Shekar B

1.Simple applets

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

2.JApplets

JApplet class in Applet

As we prefer Swing to AWT. Now we can use JApplet that can have all the controls of swing. The JApplet class extends the Applet class.

A Simple Java Applet: Drawing a String

- Create an applet to display
  "Welcome to Java Programming!"
- Show applet and HTML file, then discuss them line by line

```
1    // WelcomeApplet.java
2    // A first applet in Java.
3
4    // Java core packages
5    import java.awt.Graphics;     // import class Graphics
6
7    // Java extension packages
8    import javax.swing.JApplet;  // import class JApplet
9
10   public class WelcomeApplet extends JApplet {
11
12      // draw text on applet's background
13      public void paint( Graphics g )
14      {
15         // call inherited version of method paint
16         super.paint( g );
17
18         // draw a String at x-coordinate 25 and y-coordinate 25
19         g.drawString( "Welcome to Java Programming!", 25, 25 );
20
21      }  // end method paint
22
23   }  // end class WelcomeApplet
```

import allows us to use predefined classes (allowing us to use applets and graphics, in this case).

extends allows us to inherit the capabilities of class **JApplet**.

Method **paint** is guaranteed to be called in all applets. Its first line must be defined as above.

A Simple Java Applet: Drawing a tring


```
1    // WelcomeApplet.java
2    // A first applet in Java.
```

- Comments
  - Name of source code and description of applet

```
5    import java.awt.Graphics;    // import class Graphics
8    import javax.swing.JApplet;  // import class JApplet
```

- Import predefined classes grouped into packages
  - **`import`** statements tell compiler where to locate classes used
  - When you create applets, **`import`** the **`JApplet`** class (package **`javax.swing`**)
  - **`import`** the **`Graphics`** class (package **`java.awt`**) to draw graphics
    - Can draw lines, rectangles ovals, strings of characters
  - **`import`** specifies directory structure

A Simple Java Applet: Drawing a String

- Applets have at least one class definition (like applications)
  - Rarely create classes from scratch
    - Use pieces of existing class definitions
    - Inheritance - create new classes from old ones

```
10   public class WelcomeApplet extends JApplet {
```

- Begins **`class`** definition for class **`WelcomeApplet`**
  - Keyword **`class`** then class name
- **`extends`** followed by class name
  - Indicates class to inherit from (**`JApplet`**)
    - **`JApplet`** : superclass (base class)
    - **`WelcomeApplet`** : subclass (derived class)
  - **`WelcomeApplet`** now has methods and data of **`JApplet`**

A Simple Java Applet: Drawing a String

```
10   public class WelcomeApplet extends JApplet {
```

- Class **`JApplet`** defined for us
  - Someone else defined "what it means to be an applet"
    - Applets require over 200 methods!
  - **`extends JApplet`**
    - Inherit methods, do not have to define them all
  - Do not need to know every detail of class **`JApplet`**

A Simple Java Applet: Drawing a String

```
10    public class WelcomeApplet extends JApplet {
```

- Class **`WelcomeApplet`** is a blueprint
  - **`appletviewer`** or browser creates an object of class **`WelcomeApplet`**
    - Keyword **`public`** required
    - File can only have one **`public`** class
    - **`public`** class name must be file name

A Simple Java Applet: Drawing a String

```
13      public void paint( Graphics g )
```

- Our class inherits method **`paint`** from **`JApplet`**
  - By default, **`paint`** has empty body
  - Override (redefine) **`paint`** in our class
- Methods **`paint`**, **`init`**, and **`start`**
  - Guaranteed to be called automatically
  - Our applet gets "free" version of these by inheriting from **`JApplet`**
    - Free versions have empty body (do nothing)
    - Every applet does not need all three methods
      - Override the ones you need
- Applet container “draws itself” by calling method **`paint`**

A Simple Java Applet: Drawing a String

```
13      public void paint( Graphics g )
```

- Method **paint**
  - Lines 13-21 are the definition of paint
  - Draws graphics on screen
  - **void** indicates **paint** returns nothing when finishes task
  - Parenthesis define parameter list - where methods receive data to perform tasks
    - Normally, data passed by programmer, as in **JOptionPane.showMessageDialog**
  - **paint** gets parameters automatically
    - **Graphics** object used by **paint**
  - Mimic **paint**'s first line

A Simple Java Applet: Drawing a String

```
16          super.paint( g );
```

- Calls version of method paint from superclass **`JApplet`**
- Should be first statement in every applet's paint method

```
19          g.drawString( "Welcome to Java Programming!", 25, 25 );
```

- Body of **`paint`**
  - Method **`drawString`** (of class **`Graphics`**)
  - Called using **`Graphics`** object **`g`** and dot operator (**`.`**)
  - Method name, then parenthesis with arguments
    - First argument: **`String`** to draw
    - Second: x coordinate (in pixels) location
    - Third: y coordinate (in pixels) location
- Java coordinate system
  - Measured in pixels (picture elements)
  - Upper left is (**`0`**,**`0`**)

Compiling and Executing

Dr. Smitha Shekar B

Compiling and Executing WelcomeApplet

- Running the applet
  - Compile
    - **`javac WelcomeApplet.java`**
    - If no errors, bytecodes stored in **`WelcomeApplet.class`**
  - Create an HTML file
    - Loads the applet into **`appletviewer`** or a browser
    - Ends in **`.htm`** or **`.html`**
  - To execute an applet
    - Create an HTML file indicating which applet the browser (or **`appletviewer`**) should load and execute

Compiling and Executing WelcomeApplet

```
1  <html>
2  <applet code = "WelcomeLines.class" width = "300" height = "40">
3  </applet>
4  </html>
```

- Simple HTML file (**`WelcomeApplet.html`**)
  - Usually in same directory as **`.class`** file
  - Remember, **`.class`** file created after compilation
- HTML codes (tags)
  - Usually come in pairs
  - Begin with **<** and end with **>**
- Lines 1 and 4 - begin and end the HTML tags
- Line 2 - begins **`<applet>`** tag
  - Specifies code to use for applet
  - Specifies **`width`** and **`height`** of display area in pixels
- Line 3 - ends **`<applet>`** tag

Compiling and Executing WelcomeApplet

```
1  <html>
2  <applet code = "WelcomeLines.class" width = "300" height = "40">
3  </applet>
4  </html>
```

- **`appletviewer`** only understands **`<applet>`** tags
  - Ignores everything else
  - Minimal browser
- Executing the applet
  - **`appletviewer WelcomeApplet.html`**
  - Perform in directory containing **`.class`** file

Example of EventHandling in JApplet

```
import java.applet.*;
import javax.swing.*;
import java.awt.event.*;
public class EventJApplet extends JApplet implements ActionListener{
JButton b;
JTextField tf;
public void init(){

tf=new JTextField();
tf.setBounds(30,40,150,20);

b=new JButton("Click");
b.setBounds(80,150,70,40);

add(b);add(tf);
b.addActionListener(this);

setLayout(null);
}

public void actionPerformed(ActionEvent e){
tf.setText("Welcome");
}
}
```

In the above example, we have created all the controls in init() method because it is invoked only once.

myapplet.html

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

Displaying Graphics in Applet

Dr. Smitha Shekar B

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.


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

public class GraphicsDemo extends 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(170,100,30,30);
g.drawOval(70,200,30,30);

g.setColor(Color.pink);
g.fillOval(170,200,30,30);
g.drawArc(90,150,30,30,30,270);
g.fillArc(270,150,30,30,0,180);

}
}
```

Creating applets

- Applets are created by extending the Applet class.

```
import java.awt.*;
import java.applet.*;
/*<applet code="AppletSkel" width=300 height=100></applet> */

public class AppletSkel extends Applet {
public void init() {
// initialization
}
public void start() {
// start or resume execution
}
public void stop() {
// suspends execution
}
public void destroy() {
// perform shutdown activities
}
public void paint(Graphics g) {
// redisplay contents of window
}
 }
```

How to run an Applet?
There are two ways to run an applet
By html file.
By appletViewer tool (for testing purpose).

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.

```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{

public void paint(Graphics g){
g.drawString("welcome",150,150);
}
}
```

Note: class must be public because its object is created by Java Plugin software that resides on the browser.

myapplet.html

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

Simple example of Applet by appletviewer tool:

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it.

After that run it by: appletviewer First.java.

Now Html file is not required but it is for testing purpose only.

```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{

public void paint(Graphics g){
g.drawString("welcome to applet",150,150);
}

}
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
```

To execute the applet by appletviewer tool, write in command prompt:
c:\>javac First.java **c:\>**appletviewer First.java

Simple example of Applet by appletviewer tool:

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by: appletviewer First.java. Now Html file is not required but it is for testing purpose only.

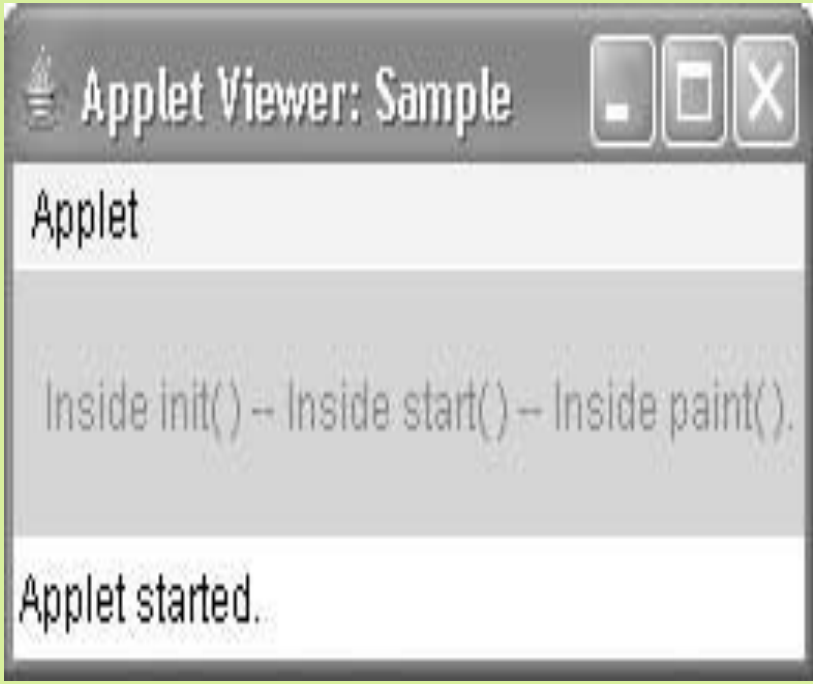
```
//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{

public void paint(Graphics g){
g.drawString("welcome to applet",150,150);
}

}
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
```

A simple applet that sets the foreground and background colors and outputs a string

{ Order of Execution }

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Sample" width=300 height=50>
</applet>
*/

public class Sample extends Applet{
String msg;

// set the foreground and background colors.

public void init() {
setBackground(Color.cyan);
setForeground(Color.red);
msg = "Inside init( ) --";
}
```

```
// Initialize the string to be displayed.

public void start() {
msg += " Inside start( ) --";
}

// Display msg in applet window.

public void paint(Graphics g) {
msg += " Inside paint( ).";
g.drawString(msg, 10, 30);
}
}
```



Status Window

```
import java.awt.*;
import java.applet.*;
/*
<applet code="StatusWindow" width=300 height=50>
</applet>
*/
public class StatusWindow extends Applet {

public void init() {
setBackground(Color.cyan);
}
// Display msg in applet window.
public void paint(Graphics g) {
g.drawString("This is in the applet window.", 10, 20);
showStatus("This is shown in the status window.");
}
}
```

Syntax of standard HTML - Applet tag [Bracketed items are optional]

```
<APPLET
   [CODEBASE = URL to an applet]
    CODE = applel class filename
   [ARCHIVES = Name of a JAR file]
   [OBJECT Serialized applet filename]
   [ALT = alternate text]
   [NAME = applet instance name]
    WIDTH = pixels HEIGHT = pixels
   [ALIGN = alignment type]
   [VSPACE = pixels] [HSPACE = pixels]
>
   [< PARAM NAME = AttributeName VALUE = AttributeValue>]
   [< PARAM NAME = AttributeName2 VALUE = AttributeValue>]
   . . .
   [Some text to be displayed in the absence of Applet]
</APPLET>
```

Passing parameters to applets

Dr. Smitha Shekar B

The area between the opening and closing APPLET tag is also used to pass parameters to applets.

This is done through the use of the PARAM HTML tag and the getParameter method of the java.applet.Applet class.

To demonstrate this we'll convert HelloWorldApplet into a generic string drawing applet.

To do this we'll need to pass the applet parameters that define the string to be drawn.


```
import java.applet.Applet;
import java.awt.Graphics;
/*
<applet code="AppletParam.class" width="300" height="300">
<param name="msg" value="Hello World">
</applet>
*/
public class AppletParam extends Applet
{
 String str;
 public void init()
 {
        str=getParameter("Name");
        str = " World";
                    str = "Hello " + str + "!";

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