## Applet Programming

- Applets are small Java applications that can be accessed on an Internet server, transported over Internet, and can be automatically installed and run as a part of a **web document**.
- An Applet is a Java 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 that is "**appletviewer**".
- Applets are designed to be embedded within an **HTML** page.
- When a user views an HTML page that contains an **applet**, the code for the applet is downloaded to the **user's machine**.

## Local Applet and Remote Applet

| Local Applet | Remote Applet |
|---|---|
| • It is developed and stored in local system. | • It is developed and stored on remote system. |
| • The web page will search the local system directories, find the local applet and execute it. | • The web page will require an internet connection to locate and load the remote applet from the remote computer. |
| • Execution of local applet does not require internet connection. | • Execution of remote applet must require internet connection. |
| • **Example:**<br><applet codebase="**path**" code="xyz.class" width=120 height=120 ><br></apple><br>**path = Path of an applet on local system.** | • **Example:**<br><applet codebase="**URL** " code="xyz.class" width=120 height=120 ><br></applet><br>**URL = Url where applet is located.** |

## Applet and Application

| Applet | Application |
|---|---|
| • It requires some third party tool like a **browser** to execute. | • It called as **stand-alone** application as application can be executed from command prompt. |
| • In applet **main()** method is **not present**. | • In application **main()** method is **present**. |
| • It cannot access anything on the system except **browser's services.** | • It can access any data or software available on the system. |
| • It requires **highest security** for the system as they are untrusted. | • It does not require any **security**. |
| • Applet starts execution from init() method. | • Applications start execution from main() method. |

## Life Cycle of Applet

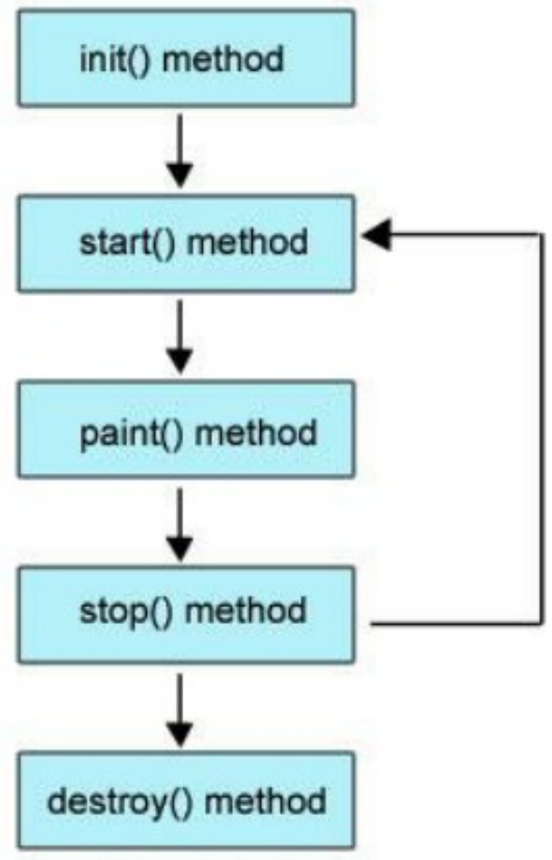



Figure: Life cycle of Applet

- In applet life cycle, there are 5 stages which are given in above figure and these stages are represented by 5 methods.
- These methods called **automatically** by the **browser** whenever required for the execution of the applet.
- No need to call these methods by the user.
- Following are the methods of the life cycle :
  1) init() method
  2) start() method
  3) paint() method
  4) stop() method
  5) destroy() method
- All above mention methods are defined in **java.applet.Applet** class except **paint()** method.
- **paint()** method is defined in **java.awt.Component** class.

**1) init() :**

- It is used to initialize the Applet. It is invoked only **once**.
- This method is called before all the other methods.

**2) Start() :**

- This method is automatically called after the browser call the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.

**3) Stop() :**

- An applet comes in idle state when its execution has been stopped either **implicitly** or **explicitly**.
- An applet is **implicitly** stopped when we **leave the page** containing the currently running applet.
- An applet is **explicitly** stopped when we call **stop()** method to stop its execution.

- So, this method called repeatedly in the same applet.

4) **Destroy():**
   - This method is only called when the browser shuts down normally. It is called only once.
   - It is called just before an applet object is removed from the memory.
5) **Paint():**
   - It is invoked immediately after the **start()** method, and also any time the applet needs to **repaint** itself in the browser.
   - The **paint()** method is actually inherited from the **java.awt**.
   - It provides **Graphics** class object that can be used for drawing **oval, rectangle** etc.

- There are two ways to run an applet :
  1) **Using HTML file**
  2) **Using AppletViewer tool**

- Example using **HTML** file :

**// Demo.java**
```
import java.applet.Applet;
import java.awt.Graphics;
public class Demo extends Applet
{
        public void paint(Graphics g)
        {
                g.drawString("welcome",200,200);
        }
}
```

**//Applet.html**
```
<html>
<body>
        <applet code="Demo.class"
        width="300" height="300">
        </applet>
</body>
</html>
```

- Example using **AppletViewer** tool :

**//Demo.java**
```
/*
<applet code="Demo.class" width="300" height="300">
</applet>
*/

import java.applet.Applet;
import java.awt.*;
public class Demo extends Applet
{
        public void paint(Graphics g)
        {
                g.drawString("welcome",200,200);
        }
}
```

- To run these above two code write following code in prompt :

```
c:\>javac Demo.java
c:\>appletviewer Demo.java
```

## Applet Tag

- The HTML <applet> tag specifies an applet.
- It is used for embedding a Java applet within an HTML document.

**Syntax :**

```
< APPLET
        [CODEBASE = codebaseURL]
        CODE = appletFile
        [ALT = alternateText]
        [NAME = appletInstanceName]
        WIDTH = pixels   HEIGHT = pixels
        [ALIGN = alignment]
        [VSPACE = pixels]     [HSPACE = pixels]
>
        [< PARAM NAME = AttributeName      VALUE = AttributeValue>]
        [< PARAM NAME = AttributeName2    VALUE = AttributeValue>]
        . . .
        [HTML Displayed in the absence of Java]
</APPLET>
```

- Attribute written in square brackets are optional.

| Attribute | Value | Description |
|---|---|---|
| **CODEBASE** | url | URL of the directory or folder that contains the applet code. |
| **CODE** | .Class file | Name of the file that contains the applet's compiled Applet subclass. |
| **ALT** | alternateText | It specifies any text that should be displayed if the browser understands the APPLET tag but can't run Java applets. |
| **NAME** | appletInstanceName | It specifies a name for the applet instance, which makes it possible for applets on the same page to find (and communicate with) each other. |
| **WIDTH** | pixels | It gives initial width (in pixels) of the applet display area. |
| **HEIGHT** | pixels | It gives initial height (in pixels) of the applet display area. |
| **ALIGN** | alignment | It specifies the alignment of the applet. |
| **VSPACE** | pixels | It specifies the number of pixels above and below the applet |
| **HSPACE** | pixels | It specifies the number of pixels on each side of the applet |
| **PARAM NAME and VALUE** | Attribute Name and Value | The PARAM tag allows you to specify applet-specific arguments in an HTML page. Applets access their attributes with the **getParameter( )** method. |

**Example :**

**//AppletParameter.html**

```
<HTML>
<HEAD>
        <TITLE>Java Applet Example</TITLE>
</HEAD>
<BODY>
        <APPLET CODE="
        AppletParameter.class"
                WIDTH="400" HEIGHT="50">
          <PARAM    NAME="Hello"
                VALUE="Hello, Welcome to Java
                World  :)" >
        </APPLET>
</BODY>
</HTML>
```

**//AppletParameter.java**

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

public class AppletParameter extends Applet
{
        public String myString;
        public void init()
        {
                myString = getParameter("Hello");
        }
        public void paint(Graphics g)
        {
                g.setColor(Color.red);
                g.drawString(myString, 20, 20);
        }
}
```

## Methods of Applet

| Methods | Description |
|---|---|
| void init( ) | It Called when an applet begins execution. |
| void start( ) | It Called by the browser when an applet starts (or resume) execution. |
| void stop( ) | It Called by the browser to suspend execution of the applet. Once stopped, an applet is restarted when the browser calls **start( )**. |
| void destroy( ) | It Called by the browser just before an applet is terminated. It **released** all resources allocated to the applet. |
| String getAppletInfo( ) | Returns a string that describes the applet. |
| URL getCodeBase( ) | Returns the URL associated with the invoking applet. |
| URL getDocumentBase( ) | Returns the URL of the HTML document that invokes the applet. |
| String getParameter(String *paramName*) | Returns the parameter associated with **paramName**. If not found then return NULL. |
| String[ ] [ ] getParameterInfo( ) | Returns information about the parameters that are understood by this applet. |
| boolean isActive( ) | Returns true if the applet has been started else returns False. |
| void play(URL *url*) | Plays the audio clip at the specified **absolute** URL. |

| | |
|---|---|
| void play(URL *url*, String *clipName*) | Plays the audio clip is found at the location specified by **url** with the name specified by **clipName**. |

| Methods | Description |
|---|---|
| void resize(Dimension *dim*) | Resizes the applet according to the dimensions specified by **dim**. Dimension class contains two field **width and height**. |
| void resize(int *width*, int *height*) | Resizes the applet according to the dimensions specified by width and height. |
| void showStatus(String *str*) | Displays **str** in the status window of the browser or applet viewer. |
| AudioClip getAudioClip(URL *url*) | Returns the AudioClip object specified by the url. |
| AudioClip getAudioClip(URL url, String clipName) | Returns the AudioClip object specified by the url and name specified by **clipName**. |
| Image getImage(URL *url*) | Returns the Image object specified by **the url**. |
| Image getImage(URL *url*, String *imageName*) | Returns the Image object specified by the url and name specified by **imageName**. |
| AppletContext getAppletContext( ) | Returns the context associated with the applet. |

**Example :** A simple applet that sets the foreground, background colors, draw rectangle, fill rectangle, get base code ,get document code and display as a string.

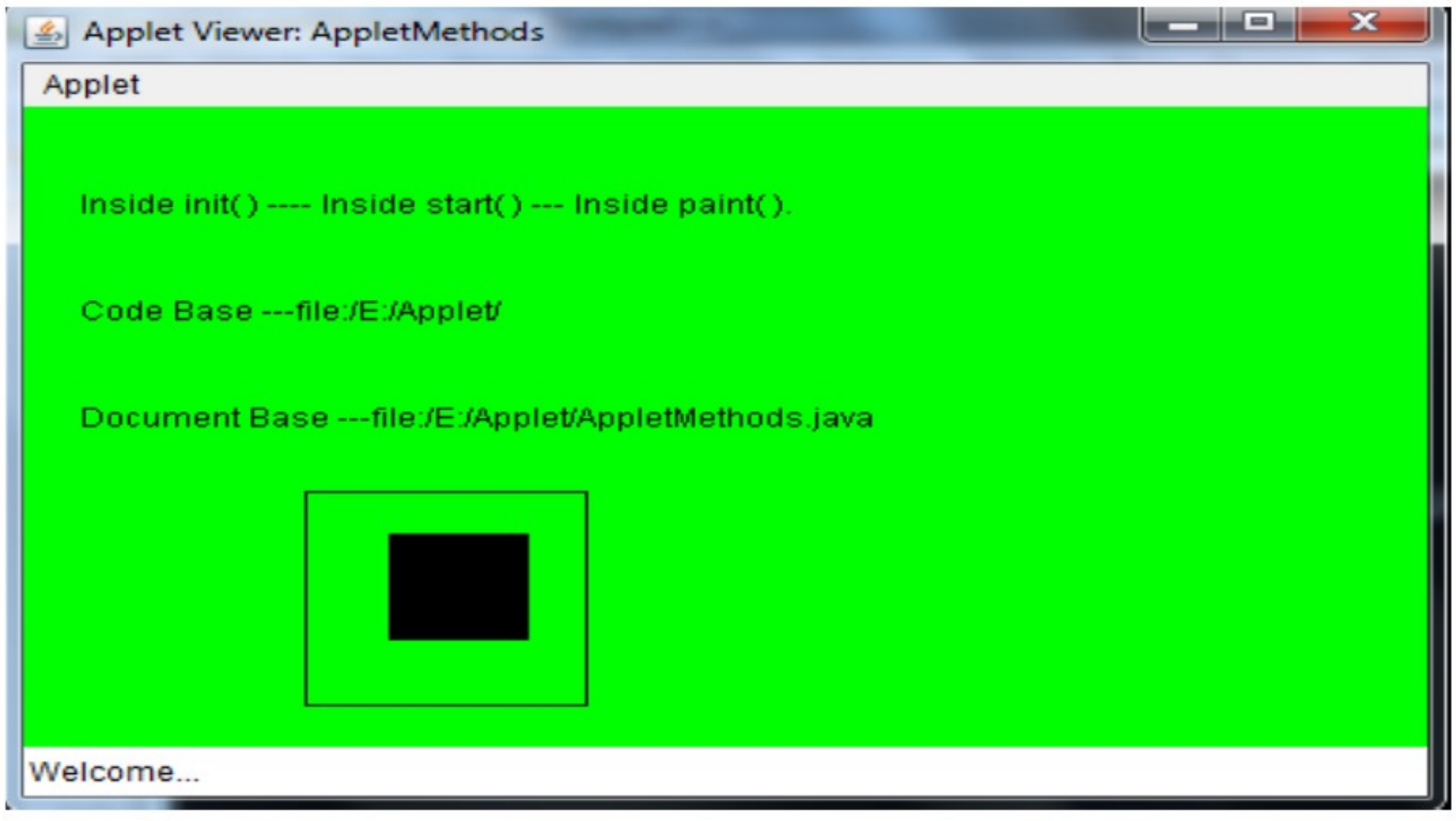
```
import java.awt.*;
import java.applet.*;
import java.net.*;
/*
<applet code="AppletMethods" width=10000 height=500 name = "HelloApplet">
</applet>
*/
public class AppletMethods extends Applet
{
String msg,displayUrl;
//set the foreground and background colors.
public void init()
{
        setBackground(Color.green);
        setForeground(Color.black);
        msg = "Inside init method ----";
}
```

```
//Initialize the string to be displayed.
public void start()
{
        msg += " Inside start method ---";
}

//Display msg in applet window.
public void paint(Graphics g)
{
        msg += " Inside paint method .";
        g.drawString(msg, 20, 10);
        URL url = getCodeBase();      // get the path of the folder in which applet placed
        displayUrl = "Code Base ---"+url;
        g.drawString(displayUrl,20,100);

        url = getDocumentBase(); // get the path upto the source file
        displayUrl = "Document Base ---"+url;
        g.drawString(displayUrl,20,150);
        // Draw Rectangle  needs (x,y,width,height) four arguments
                g.drawRect(200, 100, 100, 100);
                showStatus("Welcome...");
        }
}
```

**Output :**



## Component Class

- A **component** is an object having a graphical representation that can be displayed on the screen and that can interact with the user.
- Examples of components are the buttons, checkboxes, radio buttons and textbox of a typical graphical user interface.

| Methods | Description |
|---|---|
| void add(Component c) | Inserts a component on this component. |
| void setSize(int width,int height) | Sets the width and height of the component. |
| void setForeground(Color)<br>void setBackground(Color) | Set the foreground or background color for the component |
| Color getForeground()<br>Color getBackground() | Get the foreground or background color for the component. |
| void setName(String)<br>String getName() | Set or get the name of the component |
| void setEnabled(boolean)<br>boolean isEnabled() | Set or get whether the component is enabled. |
| void setVisible(boolean)<br>boolean isVisible() | Set or get whether the component is visible. |
| int getWidth()<br>int getHeight() | Get the current width or height of the component measured in pixels. |
| int getX()<br>int getY() | Get the current *x* or y coordinate of the component |
| void repaint()<br>void repaint(int, int, int, int) | Request that all or part of the component be repainted. |
| void remove(Component)<br>void removeAll() | Remove one of or all of the components from this container. |
| void setLayout(LayoutManager)<br>LayoutManager getLayout() | Set or get the component's layout manager. |
| Rectangle getBounds()<br>Rectangle getBounds(Rectangle) | Gets the bounds of this component in the form of a Rectangle object. |
| void setBounds(int, int, int, int)<br>void setBounds(Rectangle) | Moves and resizes this component. |

**Example :** Write a applet program to create simple GUI consist of Button, Radio Button, TextField, Checkbox and Label.

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

public class AppletComponent extends Applet
{
```

```
// Button to click
    Button submit;
 // A textField to get text input
    TextField text;
 // A group of radio buttons necessary to only allow one radio button to be selected at the
//same time.
    CheckboxGroup radioGroup;
 // The radio buttons to be selected
    Checkbox male;
    Checkbox female;
 // An independant selection box
    Checkbox option;
 //Label dispaly as text only
    Label name;

public  void  init()
{
        // Tell the applet not to use a layout manager.
            setLayout(null);
        // Initialize the button and give it a text.
            submit = new Button("Submit");
// Text and length of the field
        text = new TextField("Insert name..",100);
// initialize the radio buttons group
        radioGroup = new CheckboxGroup();
// first radio button. Gives the label text, tells to which group it belongs and sets the
//default state selected(true)
        male = new Checkbox("Male", radioGroup,true);
// same but not selected
        female = new Checkbox("Female", radioGroup,false);
// Label and state of the checkbox
        option = new Checkbox("Option",false);
// Label named Name
        name = new Label("Name : ");

// now we will specify the positions of the GUI components. This is done by specifying
the x and y coordinate and the width and height.
            submit.setBounds(100,200,100,30);
            text.setBounds(20,50,150,25);
            male.setBounds(20,120,100,30);
            female.setBounds(120,120,100,30);
            option.setBounds(20,160,100,30);
            name.setBounds(20, 15, 50, 50);
```

```
// now that all is set we can add these components to the applet.
        add(submit);
        add(text);
        add(male);
        add(female);
        add(option);
        add(name1);
            }
    }
```

**Output :**