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 plateforms, including Linux, Windows, Mac Os etc.

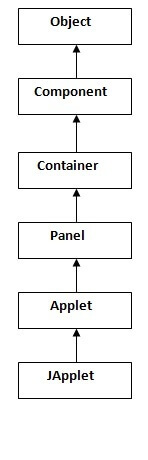
Drawback of Applet

* Plugin is required at client browser to execute applet.

Do You Know

* Who is responsible to manage the life cycle of an applet ?
* How to perform animation in applet ?
* How to paint like paint brush in applet ?
* How to display digital clock in applet ?
* How to display analog clock in applet ?
* How to communicate two applets ?

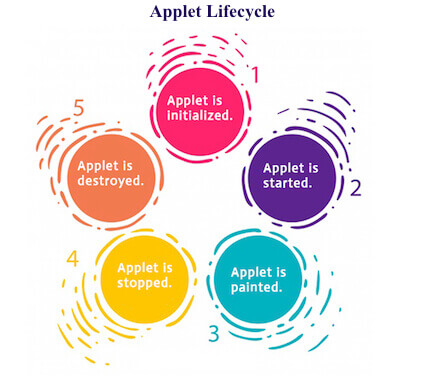
Hierarchy of Applet



|  |
| --- |
| As displayed in the above diagram, Applet class extends Panel. Panel class extends Container which is the subclass of Component. |

Lifecycle of Java Applet

1. Applet is initialized.
2. Applet is started.
3. Applet is painted.
4. Applet is stopped.
5. Applet is destroyed.



Lifecycle methods for Applet:

The java.applet.Applet class 4 life cycle methods and java.awt.Component class provides 1 life cycle methods 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.

1. **public void init():** is used to initialized the Applet. It is invoked only once.
2. **public void start():** is invoked after the init() method or browser is maximized. It is used to start the Applet.
3. **public void stop():** is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.
4. **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.

1. **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.

### Who is responsible to manage the life cycle of an applet?

Java Plug-in software.

### How to run an Applet?

There are two ways to run an applet

1. By html file.
2. By appletViewer tool (for testing purpose).

### Simple example of Applet by html file:

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

# Displaying Graphics in Applet

java.awt.Graphics class provides many methods for graphics programming.

## **Commonly used methods of Graphics class:**

1. **public abstract void drawString(String str, int x, int y):** is used to draw the specified string.
2. **public void drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height.
3. **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.
4. **public abstract void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.
5. **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.
6. **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).
7. **public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image.
8. **public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used draw a circular or elliptical arc.
9. **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.
10. **public abstract void setColor(Color c):** is used to set the graphics current color to the specified color.
11. **public abstract void setFont(Font font):** is used to set the graphics current font to the specified font.

## **Example of Graphics in applet:**

**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);

}

}

**myapplet.html**

<html>

<body>

<applet code="GraphicsDemo.class" width="300" height="300">

</applet>

</body>

</html>

### The methods paint(), repaint() and update()

The class Component defines methods called paint(), repaint() and update().

The way these methods interact can sometimes create strange results.

First, consider what the methods are intended for:

* paint(): paint() is where you place code for drawing, writing etc. Initial placement of buttons and other widgets should not be done in paint(). Whenever a rendering is thought to be "damaged" (e.g, partially covered by another window), paint() is called to re-draw the scene. Note: AWT maintains information about the size of the "drawn" area, so that if a portion of the window outside this area is covered and un-covered, paint() may not be called. Thus, if buttons are created in paint, new buttons will be created each time paint() is called. This is why you shouldn't create buttons in paint(). Note: paint() is given a Graphics context as a parameter.
* update(): update is called when the window is re-sized. The default implementation of update(): first clears the background; then calls paint().
* repaint(): The repaint() is intended to allow various methods to call for a re-rendering of the component. No graphics context is needed for repaint(). A call to repaint() calls update().