History of Java

1. Brief history of Java
2. Java Version History

The history of Java is very interesting. Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time. The history of java starts from Green Team. Java team members (also known as **Green Team**), initiated this project to develop a language for digital devices such as set-top boxes, televisions etc. But, it was suited for internet programming. Later, Java technology was incorporated by Netscape.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describe the history of java.

VM (Java Virtual Machine)

1. Java Virtual Machine
2. Internal Architecture of JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

What is JVM

It is:

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

What it does

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code

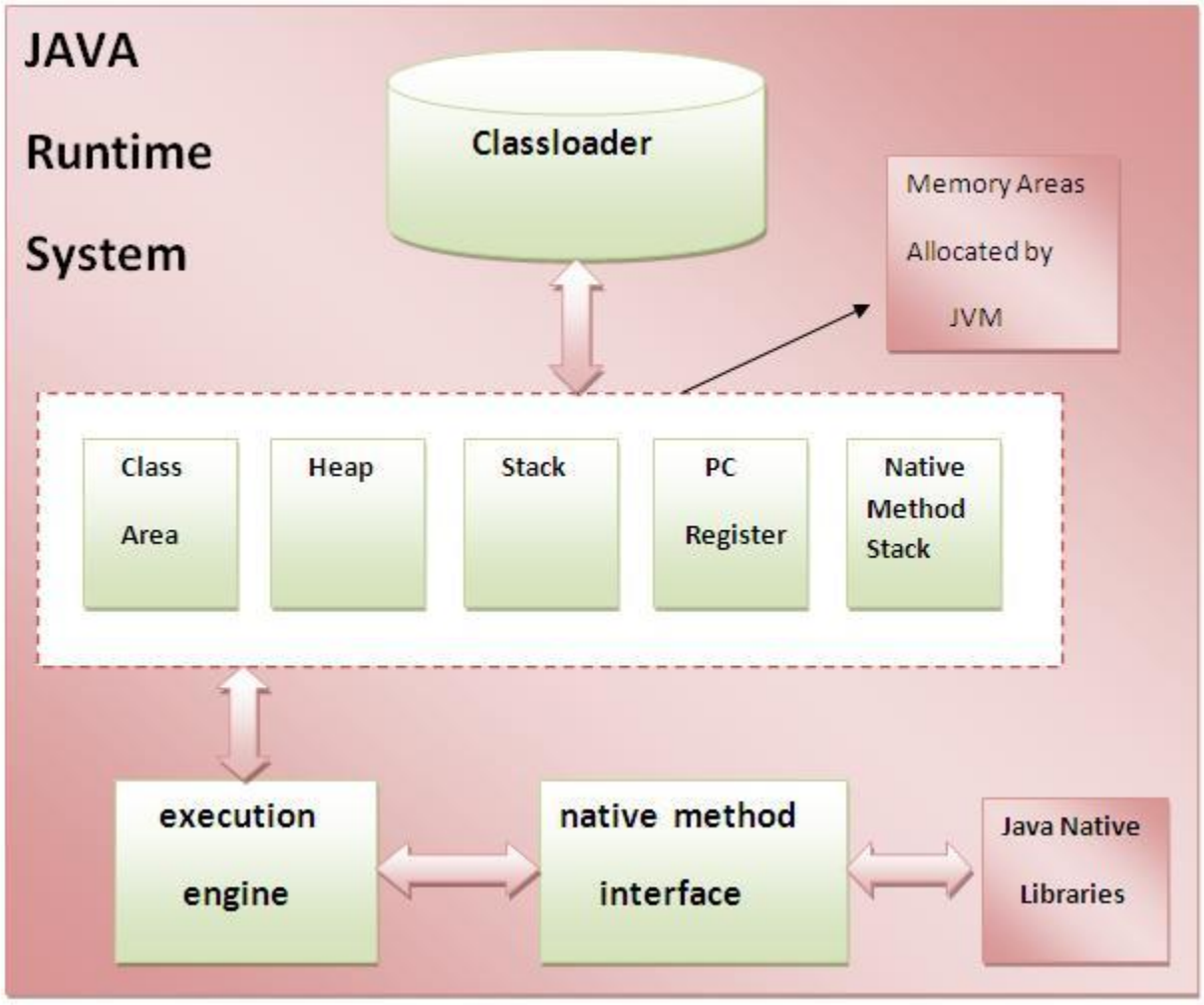
- Provides runtime environment

JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.

Internal Architecture of JVM

Let's understand the internal architecture of JVM. It contains classloader, memory area, execution engine etc.



1) Classloader

Class loader is a subsystem of JVM that is used to load class files.

2) Class(Method) Area

Class (Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

It contains:

1) A virtual processor

2) Interpreter: Read bytecode stream then execute the instructions.

3) Just-In-Time (JIT) compiler: It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term compiler refers to a translator from the

instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

Differences among C, C++, Java :

The main difference between c++ and java is that C++ does not support database connection while Java supports database connection.The other differences can be summarized as :

Aspects	C	C++	Java
Developed Year	1972	1979	1991
Developed By	Dennis Ritchie	Bjarne Stroustrup	James Gosling
Successor of	BCPL	C	C(Syntax) & C++ (Structure)
Paradigms	Procedural	Object Oriented	Object Oriented
Platform Dependency	Dependent	Dependent	Independent
Keywords	32	63	50 defined (goto, const unusable)
Datatypes : union, structure	Supported	Supported	Not Supported
Pre-processor directives	Supported (#include, #define)	Supported (#include, #define)	Not Supported

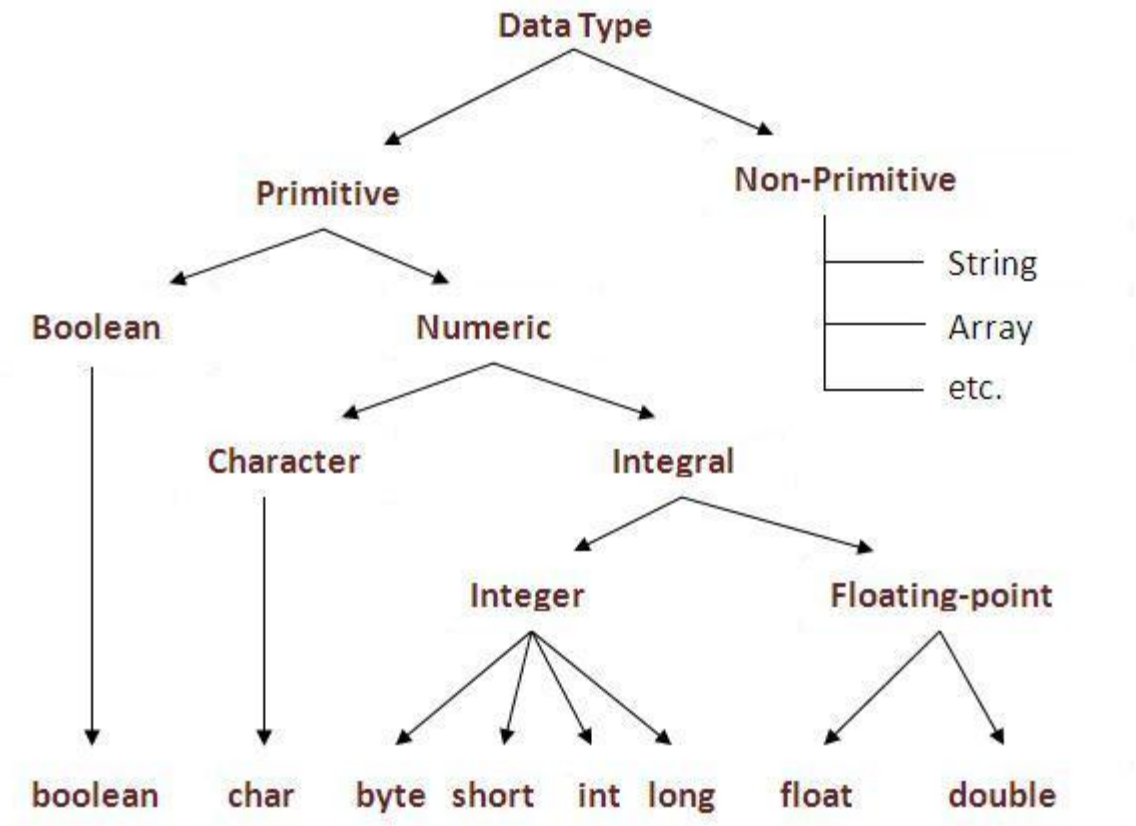
Header files	Supported	Supported	Use Packages (import)
Inheritance	No Inheritance	Supported	Multiple Inheritance not Supported
Overloading	No Overloading	Supported	Operator Overloading not Supported
Pointers	Supported	Supported	No Pointers
Code Translation	Compiled	Compiled	Interpreted
Storage Allocation	Uses malloc, calloc	Uses new , delete	uses garbage collector
Multi-threading and Interfaces	Not Supported	Not Supported	Supported
Exception Handling	No Exception handling	Supported	Supported
Templates	Not Supported	Supported	Not Supported
Storage class: auto, extern	Supported	Supported	Not Supported
Destructors	No Constructor or Destructor	Supported	Not Supported

Database Connectivity	Not Supported	Not Supported	Supported

Data Types in Java

Data types represent the different values to be stored in the variable. In java, there are two types of data types:

- Primitive data types
- Non-primitive data types



Data Type	Default Value	Default size
boolean	false	1 bit

char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Java If-else Statement

The Java *if statement* is used to test the condition. It checks boolean condition: *true* or *false*. There are various types of if statement in java.

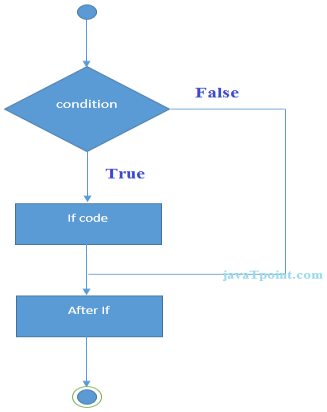
- if statement
- if-else statement
- if-else-if ladder
- nested if statement

Java if Statement

The Java if statement tests the condition. It executes the *if block* if condition is true.

Syntax:

```
1. if(condition){
2. //code to be executed
```

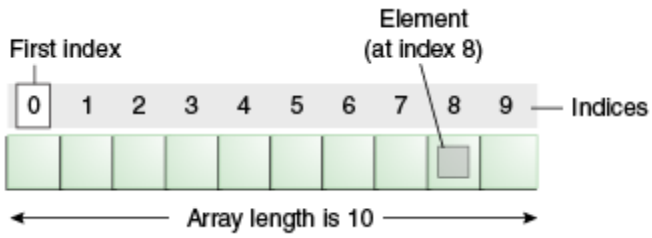


Java Array

Normally, array is a collection of similar type of elements that have contiguous memory location.

Java array is an object the contains elements of similar data type. It is a data structure where we store similar elements. We can store only fixed set of elements in a java array.

Array in java is index based first element of the array is stored at 0 index.



Advantage of Java Array

- **Code Optimization:** It makes the code optimized, we can retrieve or sort the data easily.
- **Random access:** We can get any data located at any index position.

Disadvantage of Java Array

- **Size Limit:** We can store only fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in java.

Types of Array in java

There are two types of array.

- Single Dimensional Array
- Multidimensional Array

Object and Class in Java

1. Object in Java
2. Class in Java
3. Instance Variable in Java
4. Method in Java
5. Example of Object and class that maintains the records of student
6. Anonymous Object

In this page, we will learn about java objects and classes. In object-oriented programming technique, we design a program using objects and classes.

Object is the physical as well as logical entity whereas class is the logical entity only.

Object in Java

An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical (tangible and intangible). The example of intangible object is banking system.

An object has three characteristics:

- **state:** represents data (value) of an object.
- **behavior:** represents the behavior (functionality) of an object such as deposit, withdraw etc.
- **identity:** Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.

For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior.

Object is an instance of a class. Class is a template or blueprint from which objects are created. So object is the instance (result) of a class.

Object Definitions:

- Object is *a real world entity*.
- Object is *a run time entity*.
- Object is *an entity which has state and behavior*.
- Object is *an instance of a class*.

Class in Java

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

- **fields**
- **methods**
- **constructors**
- **blocks**
- **nested class and interface**

Syntax to declare a class:

```
1. class <class name>{
2.     field;
3.     method;
4. }
```

Instance variable in Java

A variable which is created inside the class but outside the method, is known as instance variable. Instance variable doesn't get memory at compile time. It gets memory at run time when object(instance) is created. That is why, it is known as instance variable.

Method in Java

In java, a method is like function i.e. used to expose behavior of an object.

Advantage of Method

- Code Reusability
- Code Optimization

Inheritance in Java

1. Inheritance
2. Types of Inheritance
3. Why multiple inheritance is not possible in java in case of class?

Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object.

The idea behind inheritance in java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.

Inheritance represents the **IS-A relationship**, also known as *parent-child* relationship.

Why use inheritance in java

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

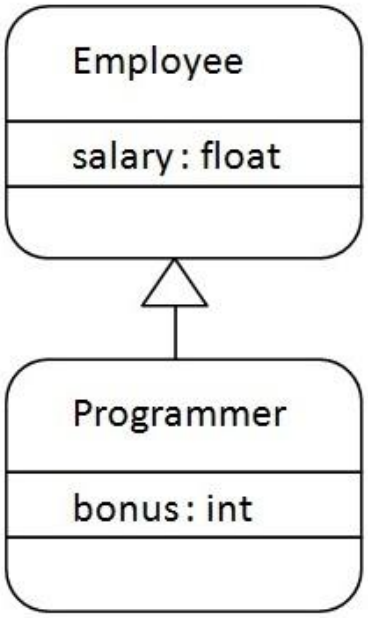
Syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called parent or super class and the new class is called child or subclass.

Java Inheritance Example



As displayed in the above figure, Programmer is the subclass and Employee is the superclass. Relationship between two classes is **Programmer IS-A Employee**.It means that Programmer is a type of Employee.

```
1. class Employee{
2.  float salary=40000;
3. }
4. class Programmer extends Employee{
5.  int bonus=10000;
6.  public static void main(String args[]){
7.    Programmer p=new Programmer();
8.    System.out.println("Programmer salary is:"+p.salary);
9.    System.out.println("Bonus of Programmer is:"+p.bonus);
10. }
11. }
```

Test it Now

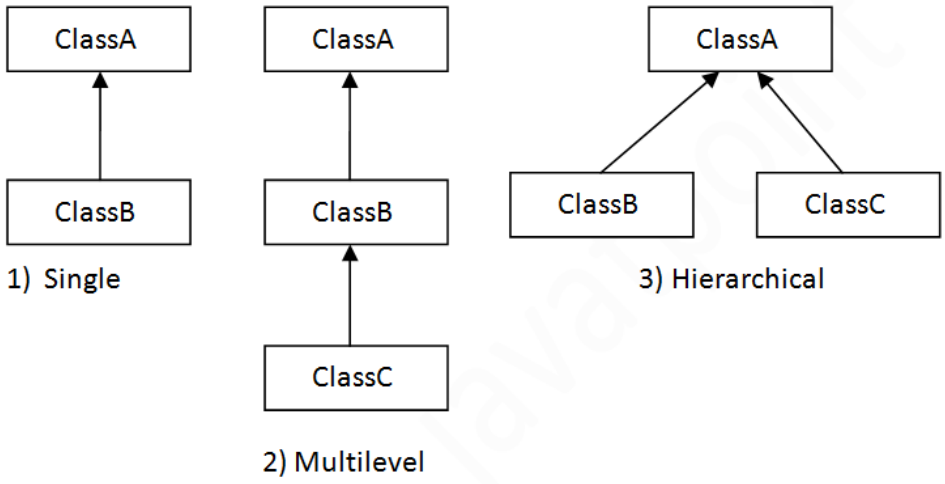
```
Programmer salary is:40000.0
Bonus of programmer is:10000
```

In the above example, Programmer object can access the field of own class as well as of Employee class i.e. code reusability.

Types of inheritance in java

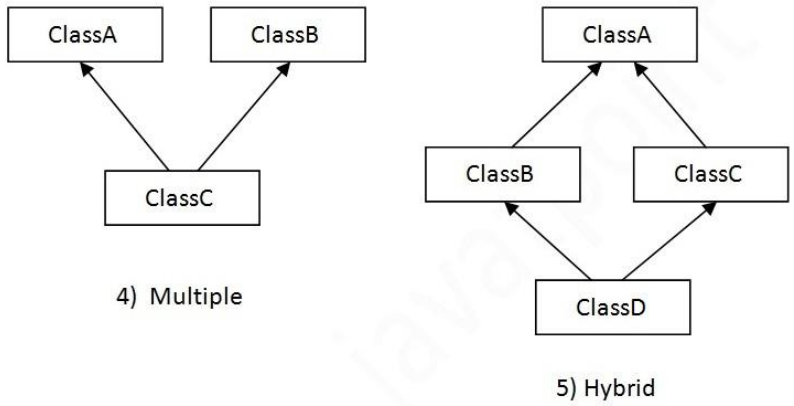
On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.



> ***Note: Multiple inheritance is not supported in java through class.***

When a class extends multiple classes i.e. known as multiple inheritance. For Example:



.

Interface in Java

1. Interface
2. Example of Interface
3. Multiple inheritance by Interface
4. Why multiple inheritance is supported in Interface while it is not supported in case of class.
5. Marker Interface
6. Nested Interface

An **interface in java** is a blueprint of a class. It has static constants and abstract methods.

The interface in java is **a mechanism to achieve abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve abstraction and multiple inheritance in Java.

Java Interface also **represents IS-A relationship**.

It cannot be instantiated just like abstract class.

Why use Java interface?

There are mainly three reasons to use interface. They are given below.

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritances.
- It can be used to achieve loose coupling.

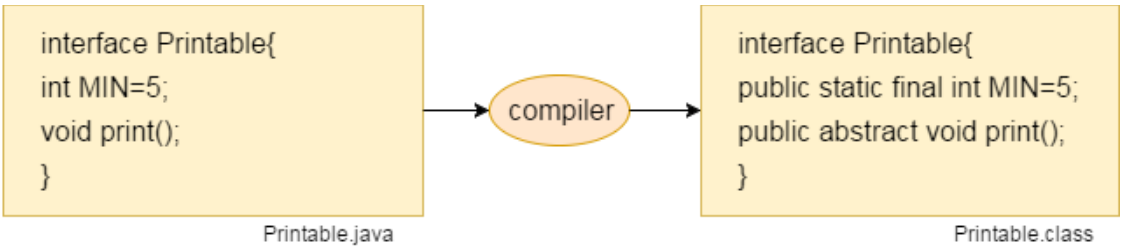
Java 8 Interface Improvement

Since Java 8, interface can have default and static methods which are discussed later.

Internal addition by compiler

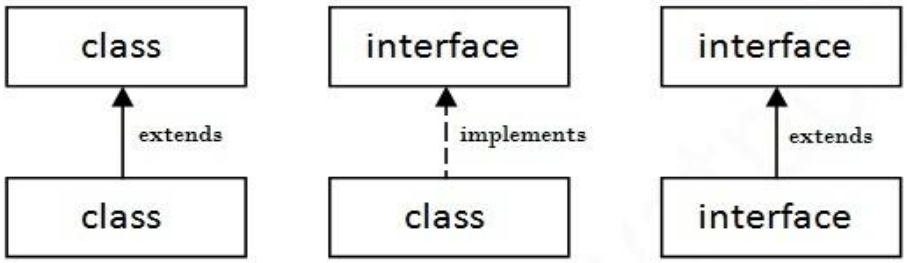
> ***The java compiler adds public and abstract keywords before the interface method. More, it adds public, static and final keywords before data members.***

In other words, Interface fields are public, static and final by default, and methods are public and abstract.



Understanding relationship between classes and interfaces

As shown in the figure given below, a class extends another class, an interface extends another interface but a **class implements an interface**.



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:

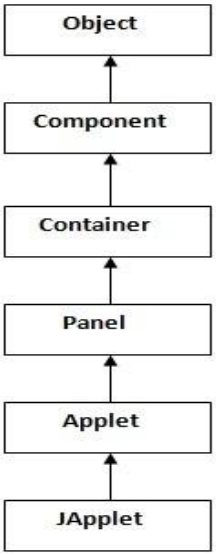
- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

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).