What is the difference between Java and C++ ?

	Java	C++
1.	It does not support pointer	It supports pointer
2	Does not support union and structure	Does supports union and structure
3	Does not support operator overloading	Does supports operator overloading
4	Does not supports destructor	Does supports destructor
5	Does not support multiple Inheritance	Does support multiple Inheritance.
6	Does not support default argument	Does support default argument
7	Does not support scope resolution	Does support scope resolution
8	Java supports garbage collection	Does not support garbage collection
9	It supports threads.	It does not support threads
10	Java supports Interfaces	C++ does not supports Interfaces
11	Java is platform independent	C++ is platform dependent
12	Java supports compiler and interpreter	C++ only supports compiler
13	Runtime error detection is done by system.	Runtime error detection is done by programmer

★ Properties of Java :-

A list of most important features of Java programming language are as given below :-

1. Simple
2. Secure
3. Robust
4. Object oriented
5. Compiled and Interpreted
6. Platform independent and portable
7. Distributed
8. Multi threading
9. High performance
10. Dynamic

1) Simple :-

The Java programming language is simple easy to learn as well as Java code is easy to read and write.

→ Java syntax is based on C++, so that easier to learn it after C++.

2) Secure :-

Java programming language provides security by default.

→ Some security can also be provided by software developer explicitly through SSL, JAAS, Cryptography and so on

3) Robust :-

Java is the robust programming language because of,

① It uses strong memory management

② there is no use of pointer, that avoids security problem

③ There is automatic garbage collection which run on JVM. [Java Virtual Machine]

④ There are exception handling and type checking mechanism in Java. All this points makes Java robust.

4) Object oriented :-

Java is a pure object oriented programming language. Everything in Java is an object. Object oriented means we organize our software as a combination of different type of object that incorporates both data and behaviour.

Java supports following concepts OOPs are :- (1) Object (2) Class (3) Inheritance (4) Polymorphism (5) Abstraction (6) Encapsulation

5) Compiled and Interpreted :-

The any computer programming language is either compiled or interpreted. Java is the combination of both approaches.

The first stage Java compiler translates source code [Programmer written code] into byte code.

In second stage Java interpreter generate the machine code that can directly be execut by machine to run Java program.

6) Platform Independent and portable :-

Java code compiled into intermediate format [byte code] which can be executed on any system for which Java Virtual Machine is portable.

This means you can write Java progra once at any platform and run it on XP, Mac Linux, Solaries, etc without recompiling.

7) Distributed :-

Java is distributed because it facilitates users to execute distributed applicati In Java, EJB is used for creating distributed application. This feature of Java makes us able to access files by calling meth from any machine on internet.

8) Multi threading :-

Thread is like a seperate program, executing concurently. We can write Java program that deals with many tasks at once by defining multiple threads. The main advantage of multi threading is that it doesn occupy memory for each thread, but it share a common memory area. Threads are

important for multimedia, social media, etc.

9) High Performance :
Java code is compiled into byte code which is highly optimized by Java compiler, so that Java Virtual Machine can execute Java application at full speed.

10) Dynamic :
Java is a dynamic language. It support runtime loading of classes. It means classes are loaded on demand.
It also support function from native languages that is C and C++.
It supports dynamic compilation and automatic memory management (Garbedge collection).

What is variable ?
Variable is an entity which is used to store value and we can change that stored value in variable. It is called variable.

ex:- int num = 10;

Variable is the name given to a memory location.

Datatypes in Java :
Java has two types of datatype : (1) Primitive and (2) Non-Primitive

(1) Primitive :

1) Boolean :
Boolean datatype has two values : true and false. Boolean datatype has false value by default. Boolean datatype occupies one bit size in memory.

2) Char :
Char datatype is used to store characters in Java. Char datatype has default value '\u0000' in Java. Char datatype occupies two byte size in memory.

3) Byte :
Byte datatype is used to store binary data 0 and 1 in Java. Byte has a by default value of zero. Byte occupies one byte memory in Java.

4) Short :
Short datatype is used to store small amount of integer value in Java. Short has a by default value zero. It occupies two byte of memory as size.

5) Int :
Int is used to store the integer value upto four byte memorie in Java. Integer has by default value zero.

6) Long :
Long datatype is used to store long int

number upto 4 byte. It has by default value 0L.

7) Float :
Float datatype is used to store floating point value upto 4 byte. Its default value is 0.0F.

8) Double :
The by default value of double datatype is 0.0d. The double datatype is used to store large amount of float value upto 8 byte in Java.

★ Looping :

Loops are used to execute a set of instruction / function repetably when some condition become true.

There are three types of loop in Java :
① For ② While ③ Do.... while ④ For each

(1) For :
→ For loop is the same as C or C++.
It has main three section.

1) Initialization section is the starting point of loop.

2) Condition Section is the second section which is used to put condition, which is executed each time to check condition of loop. It continues execution until condition is false. This section is ending point of loop.

3) Increament / Decrement section is used to specify intervals in between the loop cycle.

→ Syntax :

```
For (initialization ; condition ; inc/dec) {
    // code to executed
}
```

→ Exa :

```
Public class ForExa {
    public static void main (string args[]){
    For (int i=1; i<=10 ; i++)
    {
        System.out.println ("i" + i);
    }
}
```

(2) For each :

→ The for each loop is used to cross with the array or collection in Java. It is easier to use them simple for loop, because we do not need to increment value.

→ It works on element bases not index. It return elements one by one into to defined variable.

→ Syntax :

```
For (var : array) {
    // code to be executed
}
```

→ Exa :

```
Public class ForEachExa {
    public static void main (string[] args) {

    int arr[] = {10,20,30,40,50};

    For (int i : arr)
    {
        System.out.println(" i :- " + i);
    }
}
```

(3) While :

→ In while loop, first must check the condition then if condition is true then and then any body of loop is executed. If condition is false, the body of loop does not gets executed.

→ While keyword is used to execute while loop.

→ Syntax :

```
while (condition)
{
    // statement
}
```

→ Exa :

```
public class WhileExa {
    public static void main (string[] args) {

    int i=1;
    while (i<=10)
    {
        System.out.println (i);
        i = i+1;
    }
}
```

(4) Do....While :

→ To execute do.... while loop, do keyword is used at the first and while keyword is used at the end of loop.

→ Do.... while loop is executed at least once, even if condition is true or false.

→ Syntax :

```
do
{
    // statement
} while (condition);
```

→ Exa :

```
public class DoWhileExa {
    public static void main (string args[]) {

    int i=1;
    do
    {
        System.out.println (i);
        i++;
    } while (i<=10);

    }
}
```

★ Break and Countinue keyword :

Java break keyword is used to break loop or switch statements. It breaks the current flow of program at specified condition.

1) Widening conversion :

→ When we convert data of small size into a bigger size datatype, this process is known as widening conversion.

→ Data is never lost into this type of conversion.

→ Exa: int m=50;
long n=(long) m;

2) Narrowing conversion :

→ When we convert data of large size datatype into small size datatype, it is known as narrowing conversion.

→ In this type of conversion, data may be lost. But in java it give guarantee to reserve result in no loss of information. For following datatype conversion :-

From	to
Byte	short, char, int, long, float, double
short	int, long, float, double
char	int, long, float, double
long	float, double
float	double

→ Exa: int m=50;
byte n=(byte) m;

```
System.out.println("D = "+d);
System.out.println("Types Converted");

short s1 = (short) b;
short s2 = (short) i;
float m1 = (float) l;
int m1 = (int) f1;

System.out.println("S1 = "+s1);
System.out.println("S2 = "+s2);
System.out.println("M1 = "+m1);
System.out.println("M1 = "+m1);
}
}
```

String in Java :

Strings are a sequence of character. In Java programming language, Strings are used as objects. Java platform provides the string class to create and manipulate strings.

The Java.lang.string class is used to create a string object. Strings objects are stored in special memory are, which is known as "String constant pool."

*What are string object ?

→ There are two ways to create strings object : (1) By string literal (2) By new keyword

1> String literal :

Java string literal is created by using double quote. For exa. String str = "Sabarigam" ;

When you create the string literal, the JVM when you check the "string constant pool" first. If the string already exist in pool or not. If string already exist in pool then it will access from pool. If string does not exist in pool, then new string is created and placed in pool.

i.e String str1 = "Sabarigam" ;
String str2 = "Sabarigam" ; // It does not create new object.

In above example, only one object will be created. Firstly, JVM will not find any string object with true value "Sabarigam" in string constant pool, because it will create new object. After that it will find the string with value "Sabarigam" in pool, it will not create new object but it will return the reference to same object.

2> By new keyword :

We can create new string object by following syntax :

String s= new String("Sabarigam");

exa. public class StringExa {
public static void main(String args[]) {

```
String s1 = "Java" // created string using
                   // string literal
char ch[] = {'a','b','c','d'};
String s2 = new String(ch);
String s3 = new String("Sabarigam");

System.out.println("S1 :- "+s1);
System.out.println(s2);
System.out.println(s3);
}
}
```

Java String Buffer Class :

Java String Buffer class is used to create mutable (modifiable) string. The string buffer class in Java is same as string class, except it is mutable. That means it can be changed.

The Java.lang.StringBuffer class is the thread-safe. Following are important points for String Buffer class :-

> String Buffer is like a string but can be modified.
> It contain some particular sequence of character but length and content of string can be changed through different method.
3> They are safe for use by multiple threads.
4> Every string buffer has a capacity.

* Constructor of String Buffer class :

→ StringBuffer ()
This constructor is used to create an empty StringBuffer with the initial capacity of 16.

2> StringBuffer (String str)
→ This constructor is used to create a string buffer with specified string.

3> StringBuffer (int capacity)
→ This constructor is used to create an empty String Buffer with given capacity as a length.

* Method of String Buffer Class :

1> Append (String str) :
→ Append method is used to concate the string with given argument.
→ Exa :-

```
public class AppendExa {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Saba");
        sb.append("gam");
```

9) Delete (start_index, end_index) :-
→ Delete method is used to delete the string from specified starting index to ending index.

→ Exa :-

```
public class DeleteExa {
    public static void main (String [] args) {

        StringBuffer sb = new StringBuffer ("Sabarigam");
        sb.delete (0,4);

        System.out.println ("SB = " + sb);
        }
}
```

→ O/P :- garm

Difference between String and StringBuffer :-

There are many difference between String and StringBuffer

String	StringBuffer
String class is immutable	1) StringBuffer class is mutable (modifiable).
String is slow and consumes more memory when you concate too many string, because	2) It is consumes less memory when you concate string.
everytime it creates new instance	
3) String class overrides the equals method of object class so that we can compare two strings.	3) StringBuffer class does not overrides equals method of object class.

★ Java String Comparision :-

→ We can compare string in java on the bases of content and reference.

It is used in authentication [by equal method], sorting [By compare to method] method reference matching [by == operator]

There are 3 ways to compare various strings in Java :
(1) By equal method
(2) By == operator
(3) By compare to method

(1) Compare by equal() method :-
The String equal() method compares the original content of string. It compare values of string for equality.

String class provides the two equals method.

★ Common Java Exception :-

1) ArithmaticException
2) ArrayIndexOutOfBoundException
3) ArrayStoreException
4) FileNotFoundException
5) IOException
6) NullPointerException
7) NumberFormatException
8) OutOfMemoryException
9) SecurityException
10) StackOverflowException
11) StringIndexOutOfBoundException

★ Difference :-

Throw	Throws
1. Throw keyword is used to throw single exception only.	1. Throws keyword is used to throw multiple exception seperated by comma.
2. Throw keyword is followed by variable.	2. Throws keyword is followed by exception class.
3. checked exception can't be thrown by 'throw' keyword. only Unchecked exception can be thrown.	3. Unchecked exception can't be thrown by 'throws' keyword only checked exception can be thrown.

★ Visibility control or modifiers
Or
Access modifiers :-

Java provides main 3 type of access specifiers. (1) Public (2) Private (3) Protected.

1) Public Access :-
Public keyword is used to specify the public access specifiers. Public member visible to all the classes in the same package or outside package. that means public member accessible everywhere.

For exa public int a;
public void get.data();

2) Private Access :-
Private keyword is used to specify the private access specifiers. Public members are only visible for same class or same block only. It can not be accessed outside class or block. Private access specifier is more secure than all other access specifier.

For exa. private int a;
private void getdata();

3) Protected Access :-
Protected keyword is used to specify protected member. Protected members will be visible upto level of inheritance. The

protected modifiers are visible not only to all classes and sub classes in same package, but also to sub classes in other packages. None sub classes in other package cannot access the protected member.

For exa.

4) Friendly Access :-
When no access modifier is specified then the member by defaul be come the version of public access specifier. it is known as friendly access. The difference between public & friendly access is that the public modifier is visible in all classes & all packages, while friendly access is visible only within the package, but not in other package.

5) Private protected access :-
A member can be declared with two keyword private protected like,
private protected int a;

⇒ This modifier is visible in all sub classes means upto level of inheritance, within same package.

Page No.
Date
6

Access Location	Public	Private	Protected	Friendly (default)	Private Protected
Same class	✓	✓	✓	✓	✓
Sub class in same package	✓	X	✓	✓	✓
Other classes in same package	✓	X	✓	✓	X
Sub class in other package	✓	X	✓	X	✓
Non-sub classes in other package	✓	X	X	X	X

* Rules :-
Simple rules for applying appropriate access specifier :-
(1) Use public if the field is to be visible everywhere.
(2) Use protected if the field is to be visible everywhere in current package and also sub classes an another package.
(3) Use Friend if the field is to be visible everywhere in current package only.
(4) Use private protected if field is to be visible only in sub class, regardless of packag
(5) Use private if the field is not to be

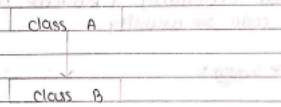
* Types of Inheritance in Java :

There are 5 types of Inheritance, which are as given below :-

1) Single
2) Multilevel
3) Hierarchical
4) Multiple
5) Hybrid

1) Single Inheritance :

In single inheritance, one class extends properties and behaviours into another class (one class only).



In above diagram, class B extends only class A. class A is 'super class' and class B is 'sub class'.

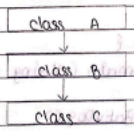
Exa.

```
public class Animal {

    void eat()
    {
        System.out.println("eating....!!");
    }
}
```

```
public class Dog extends Animal {

    void bark()
    {
        System.out.println("Barking....");
    }
}

public class SingleInheritance {
    public static void main(String[] args) {

        Dog d = new Dog();
        d.eat();
        d.bark();
    }
}
```

2) Multilevel Inheritance :

In multilevel inheritance, one class can inherite the properties & behaviour into the derived class, then after that derived class act as base class for other new child class.



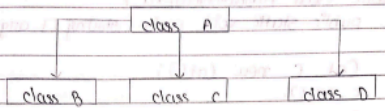
In above diagram, Class C is the sub class of class B and class B is sub class of class A.

Exa.

```
public class Animal {
    void eat()
    {
        System.out.println("eating");
    }
}

public class Dog extends Animal {
    void bark()
    {
        System.out.println("Barking...");
    }
}

public class BabyDog extends Dog {
    void weep()
    {
        System.out.println("Weeping");
    }
}

public class MultilevelExa {
    public static void main(String args[]) {

    BabyDog d = new BabyDog();
    d.eat();
    d.bark();
    d.weep();
    }
}
```

3) Hierarchical Inheritance :

In hierarchical Inheritance one class gets inherited into more than one sub class.



In above diagram, class B, C, & D inherit the same class A.

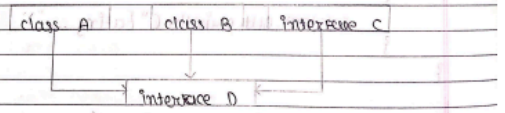
Exa.

```
public class Animal {
    void eat()
    {
        System.out.println("Eating....");
    }
}

public class Dog extends Animal {
    void bark()
    {
        System.out.println("Barking....");
    }
}

public class cat extends Animal {
    void mew()
    {
```

```
        System.out.println("Meowing");
    }
}

public class HierarchicalExa {
    public static void main(String[] args) {

        Cat c = new Cat();
        c.eat();
        c.meow();
    }
}
```

A) Multiple Inheritance :-

In multiple inheritance, one interface extending more than one interfaces.



In above diagram, interface D extends interface A, B and C.

Java does not support multiple inheritance, but we can achieve multiple inheritance with the help of interfaces.

ex:-

```
public interface One {
    void print_subarayam();
    }
    }
}

interface Two {
    public void subar()
    {
    }
}

interface Three extends One, Two {
    public void subarayam_surat()   subar()
    {                               // name should be same
    }
}

public class Child implements Three
    public void subar()
    {
        System.out.println("Subarayam");
    }
    public void print_Far()
    {
        System.out.println("Far");
    }
}
```

Abstract Class	Interface
1. It can have abstract & non-abstract method.	1. It can only have abstract method.
2. It does not support multiple inheritance.	2. It supports multiple inheritance.
3. It can have final, non-final, static & non-static variables and methods.	3. It has only static and final variables and methods.
4. It can provide the implementation of interface.	4. It can not provide the implementation of abstract class.
5. "Abstract" keyword is used to declare abstract class.	5. "Interface" keyword is used to declare interface.
6. Abstract class can extend another Java class and implement multiple interface.	6. An interface can extend only another interfaces.
7. It can be extended using "extends" keyword.	7. It can be implemented using "implements" keyword.
8. It can have member like private, protected, public.	8. It can have only public by default.

Package in Java :-

Package is the collection of similar type of classes, interfaces and sub package.

Advantages of package are as following :-

1) Package is used to categorise classes & interfaces so that they can be easily maintained.
2) Application development time is less because of reusability of code.
3) Application's execution time is less.
4) Application performance is enhanced / improved.
5) Redundancy of code is avoided / minimised.
6) It provides access protection.
7) Package removes the naming collision / ambiguity.

* Types of Package :-

Packages are categorised in two categories in Java :-

(1) Predefine or built-in package
(2) User defined package

1) Predefine or built-in package :-
→ Packages which are designed by "sun.microsystem" are called predefine package.

→ Predefine package is collection of predefine classes, interfaces & sub packages. For eg. lang, util, awt, javax, swing, net, io, util, sql, etc.

Static variable	non-Static variable

1. class A {
static int a;
}
	1. class B {
	int b;
	}

2. Static keyword is used to declare static variable	2 Non-static variables are not declared by static keyword.
3 Memory is allocated to variable at the time of loading of class in memory	3 Memory is allocated whenever object of class is created.
4 Also known as class variable.	4 Also known as instance variable.
5 Static variable are common for every object.	5 Non-static variable are specific to object.
6 Static variable can access with class reference. (classname.varname)	6 Non-static variable can access with object reference (obj.var-name)

★ Life Cycle of applet :-

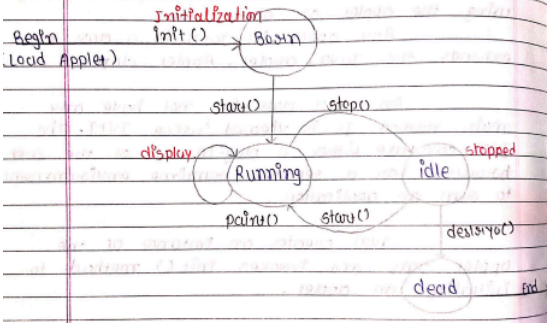


fig: An applet life cycle

The applet has mainly four stages.
(1) initialization states (2) Running state
(3) idle state (4) Destroyed state

★ life cycle method for applet :-

The java.applet.Applet class has four life cycle methods.

The java.awt.component class provide one life cycle method for applet.

* Java.applet.Applet :- To create any apple this class must be inherited. It provides four life cycle method for applet. those are (1) public void init() :- It is used to initialize applet & is convoked only once. (2) public void start() :- It

is invalked after init() method and is used to start applet. (3) public void stop() :- It is used to stop the applet. It is run when applet is stopped or browser is minimised (4) public void destroy() :- It is used to destroy the applet & is invalked only once.

* Java.awt.Compont :- This class provide one life cycle method for applet, those are (1) public void paint(Graphics g) :- It is used to paint the applet. It provides graphics class object that can be used for drawing oval, rectange, etc.

★ life Cycle of thread :-

The life cycle of thread in Java is controlled by JVM. Java thread states are as follows :-

1) new
2) Runnable
3) Running
4) Not Runnable [blocked]
5) Terminated

* Wrapper Class :-

Vector cannot handle primitive datatype like int, float, char and double. Primitive datatype maybe converted into object type, by using wrapper classes.

Primitive Data type	Wrapper class
boolean	Boolean
char	Character
float	Float
int	Integer
long	Long

* Conversion of Primitive to Object using constructor

Constructor Calling	Conversion Action
Integer intval = new Integer(i)	Primitive int i, to Integer object
Float f = new Float(f)	Primitive float into Float Object
Double d = new Double(d)	Primitive double to Double object
Long lv = new Long(l)	Primitive long to Long object

* Conversion of object type to Primitive type :

Method calling	Conversion Action
int i = intval.intValue()	Object to primitive integer
float F = FloatVal.floatValue()	
long l = LongVal.longValue()	
double d = DoubleVal.doubleValue()	

* Converting number to string using String()

Method calling	Conversion Action
Str = Integer.toString(int i)	Primitive datatype to string
Str = Float.toString(float f)	
Str = Long.toString(long l)	
Str = Double.toString(double d)	

* Converting string to numeric datatype :-

Method calling	Conversion Action
int i = Integer.parseInt(string)	Converts string object into primitive datatype
long l = Long.parseLong(Str)	

* Thread Priority :

→ Each thread is assigned a priority in Java, which affect the order in which it is scheduled for running. The threads of the same priority are given by java scheduler and, therefore they share the processor on the first come first serve bases.

Java allow us to set priority of thread using the setPriority() method.

*Syntax :-

Thread.name.setPriority(int i);

The thread class define several priority constant. 1) MIN_PRIORITY = 1,
2) NORM_PRIORITY = 5
3) MAX_PRIORITY = 10