**Differences between Java and others[C and C++]?**

1.C and C++ are static programming languages but JAVA is dynamic programming language: If any programming language allows memory allocation for primitive data types at compilation time [Static Time] then that programming language is called as Static Programming language. EX: C and C++.

Int a;

Int a=8;

Printf(“a”);

Example the memory is allocated for declared variables by the compiler is called static.

the memory is allocated is done at time of execution in runtime is called dynamic.

In C and C++ applications, memory will be allocated for primitive data types at compilation time only, not at runtime.

If any programming language allows memory allocation for primitive data types at runtime, not at compilation time then that programming language is called as Dynamic Programming Language.

EX: JAVA

In java applications, memory will be allocated for primitive data types at runtime only, not at compilation time.

Note: In Java applications, memory will be allocated for primitive data types at the time of creating objects only, in java applications, objects are created at runtime only.

2.Pre-Processor is required in C and C++ , but, Pre-Processor is not required in Java:

In case of C and C++, the complete predefined library is provided in the form of header files

EX: stdio.h

conio.h

math.h --- ----

If we want to use predefined library in C and C++ applications, we have to include header files in C and C++ applications, for this, we have to use #include<> statement.

EX: #include<stdio.h>

#include< conio.h>

#include<Math.h>

If we compile C and C++ applications then Pre-Processor will perform the following actions.

1.Pre-Processor will recognize all #include<> statement.

2.Pre-Processor will take all the specified header files from #include<> statements.

3.Pre-Processor will check whether the specified header files are existed or not in C and C++ softwares

. 4.If the specified header files are not existed the Pre-Processor will generate some error messages. 5.If the specified header files are existed then Pre-Processor will load the specified header files to the memory, this type of loading predefined library at compilation time is called as "Static Loading".

In C and C++ applications, Pre-Processor is required to recognize #include<> statements inorder to load header files to the memory.

In java , the complete predefined library is provided in the form of classes and interfaces in packages EX: java.io

java.util

java.sql

If we want to use predefined library in java applications then we have to include packages in java application, for this we have to use "import" statements

EX: import java.io.\*;

import java.util.\*;

import java.sql.\*;

If we compile java program then compiler will perform the following actions.

1.Compiler will recognize all the import statements.

2.Compiler will take the specified package names from import statements.

3.Compiler will check whether the specified packages are existed or not in java software.

4.If the specified packages are not existed in java predefined library then compiler will rise an error "package xxx does not exist".

5.If the specified packages are existed in java predefined library then Compielr will not rise any error and compiler will not load any package content to the memory.

While executing java program, when JVM[Java Virtual Machine] encounter any class or interface from the specified package then only JVM will load the required classes and interfaces to the memory at runtime, loading predefined library at runtime is called as "Dynamic Loading".

Pre-Processor is not required in JAVA , because, java does not include header files and #include<> statements, alternativily, JAVA has classes and interfaces in the form of packages and import statements.

Q)What are the differences between #include<> statement and import statement?

Ans: 1. #include<> statement is available upto C and C++. import statement is available upto JAVA.

2. #include<> statements are used to include the predefined library which is available in the form of header files. import statements are used to include the predefined library which are available in the form of packages.

3. #include<> statement is providing static loading. import statement is providing dynamic loading. 4. #include<> statements are recognized by Pre-Processor. import statements are recognized by both Compiler and JVM.

5. By using Single #include<> statement we are able to include only one header file.

EX: #include<conio.h>

#include <stdio.h>

#include <Math.h>

By using single import statement we are able to include more than one class or more than one interface of the same package.

EX: import java.io.\*;

3.C and C++ are platform dependent programming languages, but, JAVA is platform Independent programming language:

Q)What are the differences between .exe file and .class file?

Ans: 1. .exe file is available upto C and C++ only. .class file is available upto Java.

2. .exe file contains directly executable code. .class file contains bytecode, it is not executable code directly, it is an intermediate code.

3. .exe file is platform dependent file. .class file is platform independent file.

4. .exe file is less secured file. .class file is more secured file.

4.Pointers are existed in C and C++, but, Pointers are not existed in Java:

Q)What are the differences between pointer variables and reference variables?

Ans: 1. Pointer variables are available upto C and C++. Reference variables are available upto JAVA mainly.

2. Pointer variables are able to refer a block of memory by storing its address locations. Reference variables are able to refer a block of memory[Object] by storing object reference values, where Object reference value is hexa decimal form of hashcode, where hashcode is an unique identity provided by Heap manager.

3. Pointer variables are recognized and initialized at compilation time.

4. Reference variables are recognized and initialized at runtime.

5.Multiple inheritance is not possible in Java:

1.Single Inheritance 2.Multiple Inheritance

6.Destructors are required in C++, but, Destructors are not required in JAVA:

Create Objects----> Constructors

Destroy object----> Destructor

C++: To destroy objects developers must use destructors.

Java: Garbage Collector

Note: Developers are able to destro objects explicitly also

7.Operator Overloading is not supported in Java: Object Oriented Features

1.Class 2.Object 3.Encapsulation 4.Abstraction 5.Inheritance 6.Polymorphism 7.Message Passing

Polymorphism: If one thing is existed in more than one form then it is called as Polymorphism. Polymorphism is a Greak word, where Poly means many and morphism means structers or forms. 1.Static Polymorphism 2.Dynamic Polymorphism

1.Static Polymorphism: If polymorphism is existed at compilation time then it is called as Static Polymorhism. EX: Overloading

2.Dynamic Polymorphism: If the polymorphism is existed at runtime then that polymorphismn is called as Dynamic Polymorphism. EX: Overriding

Overloading: 1.Method Overloading 2.Operator Overloading 1.Method Overloading: If we declare more than one method with the same name and with the different parameter list then it is called as Method Overloading

EX: class A{

void add(int i, int j)

{ }

void add(float f1, float f2){

}

void add(String str1, String str2){

} }

2.Operator Overloading: If we define more than one functionality for any single operator then it is called as Operator Overloding.

EX: int a=10;

int b=20;

int c=a+b;// + is for Arithmetic Addition.

System.out.println(c);// 30

String str1="abc";

String str2="def";

String str3=str1+str2;// + is for String concatination.

System.out.println(str3);// abcdef

1.Operator overloading is very rarely used feature in application development. 2.Operator overloading is a bit confusion oriented feature.

Note: In java programming language, some of the few predefiend operators are declared as overloaded operators with fixed functionalities implicitly as per JAVA requirement, but, JAVA has not provided any environment explicitly to perform operator overloading at developers level.

EX: +, \*, %,....

8.C and C++ are following Call By Value and Call by reference parameter passing mechanisms, but, JAVA is following only call by value parameter passing mechanism:

primitive data: byte, short, int, long, float, double, boolean, char

Address locations

Pointer variable: Call by reference

JAVA: ref var call by value only,--->

9.In C and C++ , integers will take only 2 bytes of memory and characters will take 1 byte of memory, but, in JAVA integers will take 4 bytes of memory and characters will take 2 bytes of memory:

C and C++: Memory allocation for primitive data types is variable depending on the OS which we used.

JAVA: Memory allocation for primitive data types is fixed irrespective of the operating system which we used.

Primitives Sizes ---------- ------ byte ------> 1 byte

short------> 2 bytes

int--------> 4 bytes

long-------> 8 bytes

float------> 4 bytes

double-----> 8 bytes

char-------> 2 bytes

boolean----> 1 bit-T or F

Q)In case of C and C++, characters will take only 1 byte of memory then what is the required for JAVA to assign two bytes of memory for characters?

Ascii value for a-z=97-122java

A-Z=65-90

In case of C and C++, all the characters are represented in the form of ASCII values, here to store ASCII values one byte of memory is sufficient. In case of JAVA, all the characters are represented in the form UNICODE values, to store UNICODE value 2 bytes of memory is required.

Java Features: To show the nature of java programming language, JAVA has provided the following features.

1.Simple 2.Object Oriented 3.Platform independent 4.Arch Nuetral 5.Portable 6.Robust 7.Secure 8.Dynamic 9.Distributed 10.Multi Threadded 11.Interpretive 12.High Performance

1. Simple: Java is simple programming language, because, 1.Java applications will take less memory and less execution time. 2.Java has removed all most all the confusion oriented features like pointers, multiple inheritance,..... 3.Java is using all the simplified syntaxes from C and C++.

2. Object Oriented: Java is an object oriented programming language, because, JAVA is able to store data in the form of Objects only.

3. Platform Independent: Java is platform independent programming Language, because, Java allows its applications to compile on one operating system and to execute on another operating system

4. Arch Nuetral: Java is an Arch Nuetral Programming language, because, Java allows its applications to compile on one H/W Arch and to execute on another H/W Arch.

5. Portable: Java is a portable programming language, because, JAVA is able to run its applications under all the operating systems and under all the H/W Systems.

6. Robust: Java is Robust programming language, because,

1.Java is having very good memory management system in the form of heap memory Management SYstem, it is a dynamic memory management system, it allocates and deallocates memory for the objects at runtime.

2.JAVA is having very good Exception Handling mechanisms, because, Java has provided very good predefined library to represent and handle almost all the frequently generated exceptions in java applications.

7. Secure: Java is very good Secure programming language, because, 1.JAVA has provided an implicit component inside JVM in the form of "Security Manager" to provide implicit security. 2.JAVA has provided a seperate middleware service in the form of JAAS [Java Authetication And Autherization Service] inorder to provide web security. 3.Java has provided very good predefined implementations for almost all well known network security alg.

8. Dynamic: If any programming language allows memory allocation for primitive data types at RUNTIME then that programming language is called as Dynamic Programming Language. JAVA is a dynamic programming language, because, JAVA allows memory allocation for primitive data types at RUNTIME

9. Distributed: By using JAVA we are able to prepare two types of applications

a)Standalone Applications

b)Distributed Applications

a)Standalone Applications: If we design any java application with out using client-Server arch then that java application is called as Standalone application.

b)Distributed Applications: If we design any java application on the basis of client-server arch then that java application is called as Distributed application. To prepare Distributed applications, JAVA has provided a seperate module that is "J2EE/JAVA EE".

10. Multi Threadded: Thread is a flow of execution to perform a particular task. There are two thread models a)Single Thread Model b)Multi Thread Model a)Single Thread Model: It able to allow only one thread to execute the complete application,it follows sequential execution, it will take more execution time, it will reduce application performance. b)Multi Thread Model: It able to allow more than one thread to execute application, It follows parallel execution, it will reduce execution time, it will improve application performance. JAVA is following Multi Thread Model, JAVA is able to provide very good environment to create and execute more than one thread at a time, due to this reason, JAVA is Multi threaded Programming Language.

11. Interpretive: JAVA is both compilative programming language and Interpretive programming language. 1.To check developers mistakes in java applications and to translate java program from High level representations to low level representation we need to compile java programs 2.To execute java programs , we need an interpretor inside JVM

12. High Performance: JAVA is high performance programming language due to its rich set of features like Platform independent, Arch Nuetral, Portable, Robust, Dynamic,.....

Language Fundamentals:

To prepare java applications , we need some fundamentals provided by Java programming language.

1. Tokens 2. Data Types 3. Type Casting 4. Java Statements 5. Arrays

1. Tokens: Smallest logical unit in java programming is called as "Lexeme". The Collection of Lexemes come under a particular group is called as "Token"

int a=b+c\*d; Lexemes: int, a, =, b,+, c, \*, d, ;-----> 9 Tokens:

1)Data Types: int 2)Identifiers: a, b, c, d 3)Operators: =, +, "\* 4)Special Symbol: ; Types of tokens: 4 To prepare java applications, java has provided the following list of tokens.

1.Identifiers

2.Literals

3.Keywords/ Reserved Word

4.Operators

1.Identifiers Identifier is a name assigned to the programming elements like variables, methods, classes, abstract classes, interfaces,.....

EX: int a=10; int ----> Data Types a ------> variable[Identifier] = ------> Operator 10 -----> constant ; ------> Terminato

To provide identifiers in java programming, we have to use the following rules and regulations. 1. Identifiers should not be started with any number, identifiers may be started with an alphabet, '\_' symbol, '$' symbol, but, the subsequent symbols may be a number, an alphabet, '\_' symbol, '$' symbol.

int eno=111;------> Valid

int 9eno=999;-----> Invalid

String \_eaddr="Hyd";---> Valid

float $esal=50000.0f;----> valid

String emp9No="E-9999";----> Valid

String emp\_Name="Durga";----> Valid

float emp$Sal=50000.0f;------> Valid

2. Identifiers are not allowing all operators and all special symbols except '\_' and '$' symbols. int empNo=111; -------> valid int emp+No=111;-------> Invalid String emp\*Name="Durga";-----> Invalid String #eaddr="Hyd";--->Invalid String emp@Hyd="Durga";-----> Invalid float emp.Sal=50000.0f;-----> Invalid String emp-Addr="Hyd";------> Invalid String emp\_Addr="Hyd";------> Valid

4. Identifiers should not be duplicated with in the same scope, identifiers may be duplicated in two different scopes.

class A{ int i=10;-----> Class level short

i=20;----> Error double

f=33.33---> No Error

void m1(){

float f=22.22f; -----> local variable double

f=33.33;---> Error long

i=30;---> No Error

} }

package firstprog;

public class Datatype {

public static void main(String[] args) {

int x1=10;

Integer x2=100;

System.***out***.println(x1);

System.***out***.println(x2);

short y1=7878;

Short y2=7898;

System.***out***.println(y1);

System.***out***.println(y2);

}

}