**1. Introduction and Basic Chapter :**

1. What is java and explain feature of java?

**----------------------------------------------------------------------------------------------------------------------------------------------------------------**🡪

**Java is a class-based, object-oriented programming language. It follows WORA principle (Write once run anywhere).**

**Features of java :**

Simple

Object-Oriented

Portable

Platform independent

Secured

Robust

Architecture neutral

Interpreted

High Performance

Multithreaded

Distributed

Dynamic

**2. What is compiler?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

Compiler is application or software which is used for convert your source code in to byte code.

**3. Why java is platform independent language?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**Consider the following code :

public class Test{

public static void main(String args[]){

//logic

}

}

**Compilation**: javac Test.java

**Run**: java Test

When we compile above code compiler creates bytecode by name “Test.class”. This bytecode is same for all platforms

Bytecode is processed by JVM and gives output. Hence java is platform independent.

**4. Explain System.out.println(“Good Morning”);**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

It is output statement of java same like as printf() in c language.

System is class out is static reference of PrintStream class and System and PrintStream maintain HAS-A relationship between them and println() is overloaded method for displaying output on output screen.

**5. What is compilation?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

Compilation is process where we convert your source code to byte code in java

**6. What is byte code and importance of byte code?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

Byte code is machine understandable code means byte code is intermediate format which easily convert in machine code with the help of JVM.

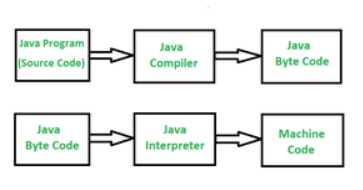
**7. What is difference between byte code and machine code?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

Byte code is an intermediate code between the source code and machine code. It is a low-level code that is the result of the compilation of a source code which is written in a high-level language. It is processed by a virtual machine like Java Virtual Machine (JVM).

Byte code is a non-runnable code after it is translated by an interpreter into machine code then it is understandable by the machine. It is compiled to run on JVM, any system having JVM can run it irrespective of their operating system. That’s why Java is platform-independent. Byte code is referred to as a Portable code.

Machine code is a set of instructions that is directly machine-understandable and it is processed by the Central Processing Unit (CPU). Machine code is in binary (0’s and 1’s) format which is completely different from the byte code and source code. It is regarded as the most lowest-level representation of the source code. Machine code is obtained after compilation or interpretation. It is also called machine language.



**8. What is OOP and explain depth?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

As the name suggests, Object-Oriented Programming or Java OOPs concept refers to languages that use objects in programming, they use objects as a primary source to implement what is to happen in the code. Objects are seen by the viewer or user, performing tasks you assign.

Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc. in programming. The main aim of OOPs is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOP languages has four pillars and 7 features i.e.

Pillars: 1. Inheritance 2. Abstraction 3.Encapsulation 4. Polymorphism

Features: 1. Class 2. object 3. inheritance 4. polymorphism 5. data abstraction and encapsulation

6. dynamic binding 7. message passing

**9. What are the pillars of OOP?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

OOP languages has four pillars i.e

**1. Abstraction :**

Abstraction is a process of hiding implementation details and exposing only the functionality to the user. In abstraction, we deal with ideas and not events. This means the user will only know “what it does” rather than “how it does”.

There are two ways to achieve abstraction in Java:

- Abstract class (0 to 100%)

- Interface (100%)

Real-Life Example: A driver will focus on the car functionality (Start/Stop -> Accelerate/ Break), he/she does not bother about how the Accelerate/ brake mechanism works internally. And this is how the abstraction works.

**2. Encapsulation**

Encapsulation is the process of wrapping code and data together into a single unit.

Real-Life Example: A capsule which is mixed of several medicines. The medicines are hidden data to the end user.

In order to achieve encapsulation in java follow certain steps as proposed below:

-Declare the variables as private

-Declare the setters and getters to set and get the variable values

**3.Inheritance :**

Inheritance is the process of one class inheriting properties and methods from another class in Java. Inheritance is used when we have is-a relationship between objects. Inheritance in Java is implemented using extends keyword.

Real-life Example: The planet Earth and Mars inherits the super class Solar System and Solar system inherits the Milky Way Galaxy. So Milky Way Galaxy is the top super class for Class Solar System, Earth and Mars.

**4. Polymorphism**

Polymorphism is the ability to perform many things in many ways. The word Polymorphism is from two different Greek words- poly and morphs. “Poly” means many, and “Morphs” means forms. So polymorphism means many forms. The polymorphism can be present in the case of inheritance also. The functions behave differently based on the actual implementation.

Real-life Example:A delivery person delivers items to the user. If it’s a postman he will deliver the letters. If it’s a food delivery boy he will deliver the foods to the user. Like this polymorphism implemented different ways for the delivery function.

There are two types of polymorphism as listed below:

-Static or Compile-time Polymorphism

-Dynamic or Run-time Polymorphism

**10. What is diff between semi object oriented and pure object oriented?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

**Semi object oriented**: Semi object oriented means those language can use OOP concept but there is no compulsion to use object oriented every time called as semi object oriented.

Example of semi object oriented language is: CPP

**Pure Object oriented:** those languages must be use OOP concept compulsory called as pure object oriented language means in java we cannot run program without class so java is pure object oriented.

**11. What is JDK, JRE and JVM? OR What is difference between JDK, JRE and JVM?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

JDK stands for java development kit it is application which contain compiler as well as JVM which is responsible create code and compile it and run time means using JDK we can develop the application and run the application and check the output

JRE stands for run time environment It is physical existence of JVM means JRE contain JVM and which is responsible for run the java application means using JRE we cannot compile java program just you can run time so normally JRE refer in test environment or in production environment

JVM stands for java virtual machine it is logically present in JRE means when run program the internally JVM get executed and the role is JVM is load the byte in code in memory and manage the memory for java application as well as create object at run time ,delete object from memory at run time as well as manage thread etc

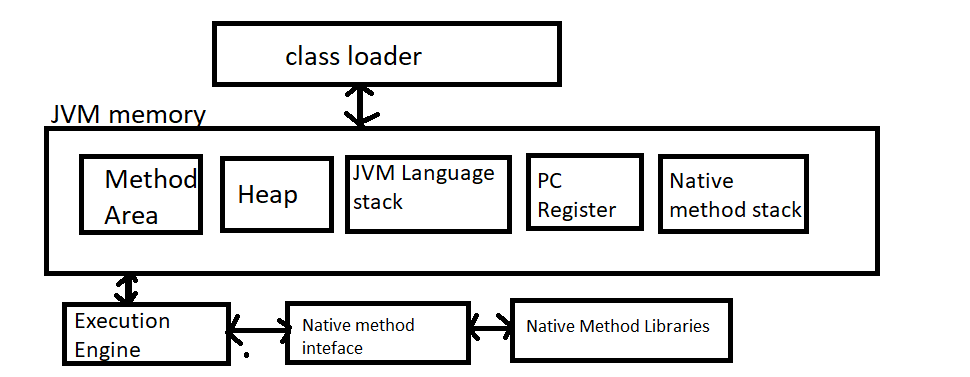
**12. What is JVM and explain its architecture?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

JVM is responsible for manage the overall activity of Java program execution

Means when we compile java program then java compiler create .class file and store byte code in it and load .class file by JVM in memory and perform different operation on it means

Allocate memory for class and its member, allocate memory for objects, destroy objects from memory as well as manage n number of things for java program shown in following diagram or show by JVM architecture.

****

**If we think about above diagram then we have first component in JVM architecture is class loader.**

**What is purpose of class loader?**

The major goal of class loader is read .class file generated by java compiler and perform three different operations on it i.e. loading, linking and initialization of .class file**.**

**Q. what is loading?**

when compiler generate .class file then JVM class loader reads .class file and generate corresponding binary data from .class file and save it in the method area and JVM store the following information in method area

I) Fully Qualifier name of the loaded class and its immediate parent class

ii) Store information about interface, class or enum

iii) Store information related with modifier, variables and method information.

**Example: suppose consider we have class name as**

**class ABC**

**{ private int a;**

**public void show()**

**{**

**}**

**}**

when we compile this program then compiler generate file name as ABC.class and when we run this program then JVM class loader read file name as ABC.class and store class name , private,public int a and void show() its information in method area means it load all class details in method area.

**Q. what is linking?**

Linking is process to verify and prepare byte code as well as binary code which loaded by class loader.

means in short we can in the linking processes check .class file is created properly or not means in linking we have BytecodeVerifier which is used for verify byte code generated property or not if byte code not generated properly it will throw error at run time java.lang.VerifyError

as well as linking section JVM allocates memory for class static variable and initialization the default in it means JVM allocate memory static variable before creating and give some default value to them means if static variable is integer then 0 default if static variable is string then give null value etc.

**Q. what is initialization?**

after verify byte code and allocate memory to static variable after that JVM try to initialize code for execution purpose from top to bottom.

There are three types of initialize in JVM

1) Bootstrap class loader 2) extension class loader 3) System class loader.

**1) Bootstrap class loader:** bootstrap class loader is used for load the core API classes present in JAVA\_HOME/lib folder.

**2) Extension class loader :** it is used for load classes present under JAVA\_HOME/jre/lib/ext folder

**3)System class loader or Application class loader :** it is child of extension class loader it is responsible load all classes from classpath.

**JVM Memory?**

JVM memory is distribution of memory management in ram means when your program running in memory then JVM memory decides which content program store in which section of memory.

We have different types of section in memory

**Method Area:** in this method area all class level information like as class name, its immediate parent class name , methods and variables and access specifier used with variable and method etc stored in method area as well as store static variable and its default value stored in method area.

**Heap Section**: in this section memory all objects stored in heap section of memory as well as all shared resources stored in heap section means if we perform any thing using new keyword in java it is stored in heap section.

**Stack Area:** so JVM create separate stack memory for every thread in java.

**PC Register:** store address of current execution instruction of thread

**13. What is array and how many ways to declare array in java?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

Array is a collection of similar data type.

Ways to declare array:

----------------------------------------------------------------------------------------------------------------------------------------------------------------

1. Declaration without Initialization:

int[] arr; // Preferred syntax

int arr[]; // Alternative syntax

2. Declaration with Size:

int[] arr = new int[5]; // Array of integers with 5 elements

3. Declaration with Initialization:

int[] arr = {1, 2, 3, 4, 5}; // Array of integers with 5 elements

4. Declaration with new Keyword and Initialization:

int[] arr = new int[]{1, 2, 3, 4, 5}; // Explicitly using 'new' to initialize the array

5. Multidimensional Arrays:

int[][] matrix = new int[3][3]; // 2D array with 3 rows and 4 columns

int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}; // Initialized 2D array

**14. What is diff between C array and java array?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

**1. Memory Allocation:**

*C Array:*

In C, arrays are statically allocated, meaning their size must be defined at compile-time.

The array memory is contiguous, and the array name acts as a pointer to the first element.

Dynamic arrays can be created using pointers and manual memory management with functions like malloc or calloc.

*Java Array:*

In Java, arrays are dynamically allocated objects on the heap. You can create arrays at runtime with a size determined during execution.

The array is a reference to a memory location that holds the array elements.

Java arrays are always initialized with default values (e.g., 0 for integers, null for objects).

**2. Bounds Checking:**

*C Array:*

C does not perform bounds checking. Accessing an out-of-bounds index results in undefined behavior, potentially leading to memory corruption or program crashes.

*Java Array:*

Java performs bounds checking. If you attempt to access an index outside the array bounds, a java.lang.ArrayIndexOutOfBoundsException is thrown, helping prevent memory errors.

**3. Type Safety:**

*C Array:*

C is not type-safe. Since arrays are closely related to pointers, it's possible to mix array and pointer types in unsafe ways, leading to potential bugs or security vulnerabilities.

*Java Array:*

Java enforces strict type safety with arrays. You cannot assign an array of one type to another incompatible type, except for arrays of objects where inheritance relationships allow it.

**4. Array Length:**

*C Array:*

The size of an array is not inherently stored with the array itself. Programmers need to manually track the size of the array, typically using a separate variable.

*Java Array:*

The length of an array can be accessed using the length attribute, making it easier to work with arrays without needing to track their size manually.

**5. Multidimensional Arrays:**

*C Array:*

C supports true multidimensional arrays, where all dimensions must be specified at compile time. They are stored in a contiguous block of memory.

*Java Array:*

Java supports multidimensional arrays as arrays of arrays. Each sub-array can have a different length, leading to jagged arrays. This structure offers flexibility but also means that the memory layout is not necessarily contiguous.

**6. Array Initialization:**

*C Array:*

Arrays can be initialized at the time of declaration, e.g., int arr[3] = {1, 2, 3};.

Uninitialized arrays contain garbage values.

Java Array:

Arrays are automatically initialized to default values (e.g., 0 for integers, false for booleans, null for objects).

Arrays can be initialized at the time of declaration or using a loop or static initializer block.

**7. Object Orientation:**

*C Array:*

C is a procedural language and does not support object-oriented programming. Arrays are simple data structures.

*Java Array:*

Java is an object-oriented language, and arrays themselves are objects, which means they inherit methods from the Object class, although they don't have many methods other than length.

**8. Garbage Collection:**

*C Array:*

C requires manual memory management. If you dynamically allocate an array, you must also manually free the memory to avoid memory leaks.

*Java Array:*

Java has automatic garbage collection. Once an array is no longer referenced, the memory is automatically reclaimed by the garbage collector.

These differences reflect the distinct design philosophies of C as a low-level systems programming language and Java as a higher-level, object-oriented language.

**15. What is Jagged Array in java?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

Jagged Array is used for create matrix where every row having a different column list

**Example :**

1 2 3

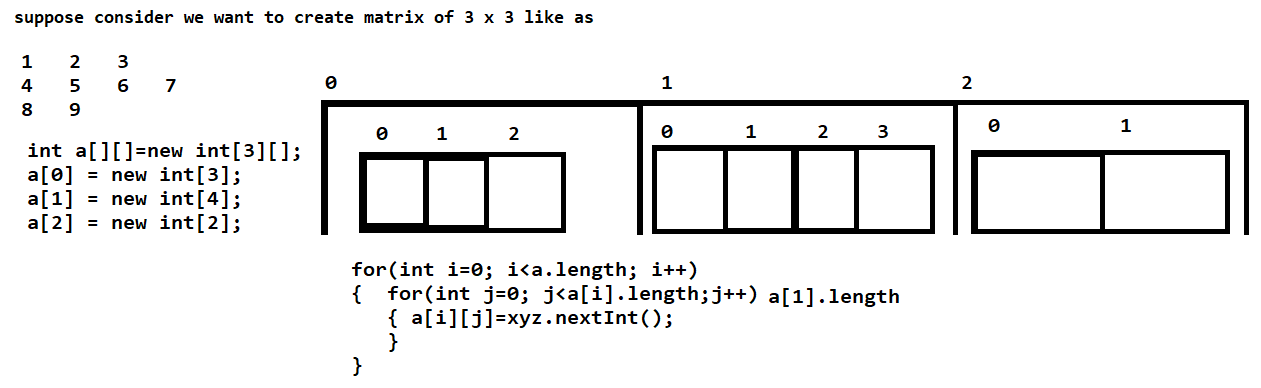
4 5 6 7

8 9

**Syntax:**

data type variablename[][]=new datatype[rowsize][];

variablename=new datatype[rowindex][colsize];



import java.util.\*;

public class JaggedArray

{

public static void main(String x[])

{

Scanner xyz = new Scanner(System.in);

int a[][]=new int[3][];

a[0]=new int[3];

a[1] = new int[4];

a[2] = new int[2];

System.out.println("Enter values in matrix");

for(int i=0; i<a.length;i++)

{

for(int j=0; j<a[i].length;j++)

{

a[i][j]=xyz.nextInt();

}

}

System.out.println("Display matrix");

for(int i=0; i<a.length;i++)

{

for(int j=0; j<a[i].length;j++)

{

System.out.printf("%d\t",a[i][j]);

}

System.out.printf("\n");

}

}

}

**16. What is anonymous array?**

**----------------------------------------------------------------------------------------------------------------------------------------------------------------🡪**

An anonymous array is an array that is created without explicitly storing it in a variable. This is often used when you need to pass an array to a method but don't need to reference the array elsewhere in your code. In Java, anonymous arrays are commonly used for this purpose.

Example with code:

public class AnonymousArrayExample {

public static void printArray(int[] arr) {

for (int i : arr) {

System.out.println(i);

}

}

public static void main(String[] args) {

// Passing an anonymous array to the method

printArray(new int[]{1, 2, 3, 4, 5});

}

}