JAVA Interview Questions

1. Why java is platform independent?

Java language was developed so that it does not depend on hardware or software because the Compiler compiles the code and converts it to platform independent byte code which can be run on multiple systems.

The only cond^n is machine should have JRE Installed in it.

2. Why java is not a pure object oriented programming?

Java supports primitive data types - bytes, boolean, int ... and hence It is not pure OOPL

3. Difference b/w stack and heap memory?

Heap Memory:

purpose: Heap memory is used for dynamic memory allocation. Whenever you create an object in java using new keyword, the object is allocated memory on the heap. This memory is managed by the Java Virtual Machine (JVM) and is shared among all threads.

-> The size of heap is much larger than the stack. Objects stored in the heap remain there until they are no longer referenced by any part of program, at which point they are eligible for garbage collection

Stack Memory

stack memory is used for storing temporary variables created by methods. Each time a method is called, a new block (frame) is created on the stack. The stack is thread specific, meaning each thread has its own stack.

-> The stack is typically smaller than the heap. Memory allocation in the stack follows LIFO. The variables stored in the stack have a short life span, existing only for the duration of the method execution

Stack memory is used to store:

1) primitive data types (eg: int, float) defined inside methods.
2) References to objects (the object itself is on heap, but the reference is on the stack)
3) Method call history and return address.

Heap memory is used to store:

1) All objects created using 'new'
2) Instance variables of classes.
3) Arrays.

→ Also garbage collector in stack as we have LIFO where as soon as the method is finished, the stack frame is popped.

How Java utilizes heap and stack:

→ when a method is called, its local variables and references to objects are stored in a stack frame. If an object is created inside a method, the reference to that object is stored on the stack, while the actual object is stored on the heap.

→ Even though object references are stored in the stack, the object themselves are stored in heap. This allows objects to persist beyond the scope of the method that created them, as long as other references to the object exists.

→ Garbage Collection: Java's automatic garbage collection helps manage heap memory by removing objects that are no longer accessible. This prevent memory leak (A memory leak refers to a situation where a program allocates memory but fails to release it back hence lead to gradual increase in memory usuage overtime).

How is Java different from C++?

→ C++ is only a Compiled language whereas Java is Compiled and Interpreted.

→ Java programs are machine independent whereas C++ program can only run in the machine in which it is compiled.

>C++ allows user to use pointers in the program. whereas Java doesn't allow it. Java Internally uses pointers.

>C++ Support the Concept of multiple Inheritance whereas Java doesn't Support this. And it is due to avoiding the Complexity of name ambiguity that causes the diamond problem.

why Java not make use of pointers?

pointers are quite complicated and unsafe to use by beginner programmer. Java focuses on Code Simplicity, and the use of pointers can make it challenging. Moreover, security is also compromised if the pointers are used the user can access memory with the help of pointers

What do you understand by Instance variable and local variable?

Instance variables are those variables that are accessible by all the methods of class. They are declared outside the methods and inside the class. All the object of the class have their copy of the variable for utilization. If any modification is done on these variable, then only the Instance will be Impacted by it.

Local variables are those present inside or within a block.

what do you mean by data encapsulation?

Data encapsulation is a fundamental Concept in object-or. pr. that involves bundling the data (attributes or properties) and the methods (function) that operate on that data into single unit, typically a class. Encapsulation also Includes restricting direct access to some of the object's Components, which is known as data hiding.

we make variable private so that no one can access it. In order to access those private variables, we use getter and setters.

☆☆ Tell us something about JIT Compiler. ☆☆ Explaination is Imp.

Compilation of Bytecode!

→ when you write java code, the Java Compiler (Javac) translates the source code into bytecode. This bytecode is a low-level, platform Independent representation of your code. it is not machine code and cannot be executed by CPU.

2. Interpretation by the JVM.

The JVM (Java Virtual Machine) is responsible for executing the bytecode. Initially, the JVM interprets the bytecode, translating it into machine code that the underlying hardware can execute.

This Interpretation allows Java bytecode to be platform-Independent. You can run the same bytecode on any device that has JVM.

3. JIT Compiler (Just-In-time):

JIT Compiler within the JVM further compiles the bytecode into native machine code at runtime. This native code is cached and reused to Improve performanle, reducing the need for repeated interpretation.

4. Why Java is both Compiled and Interpreted?

Compiled: the source code (.java) is first Compiled into bytecode (.class), which is a lower-level Intermediate form. This Compilation step is similar to tradition-al Compiled lang.

Interpreted: Java is also considered an Interpreted language because JVM Initially Interprets the bytecode at runtime, converting it into machine code that can be executed.

Define Copy Constructor in Java?

Copy Constructor is used when we want to initialize the value

to the new object from the old object of the same class.

```
Animal (Animal obj)
{
    this.character = obj.character;
}
```

Can the main method be Overloaded?

Ans Yes, it is possible to overload the main method. We can create as many overloaded main methods we want. However, JVM has a predefined calling Method that JVM will only call the main method with the defination of

public static void main(String[] args)

Overloaded functions:

psvm(int[] args)
psvm(char[] args)
psvm(double[] args).

A single try block and multiple catch blocks can co-exist in Java program? Explain.

Yes, mutiple catch blocks can exists but specific approach (specific exception like Arithmatic) should come first or prior to the general approach because only first ~~try~~ catch cond^n is executed. general approach like (Exception) which carries all exception

Explain the use of final keyword in variable, method and class.

final
1. ^variable: variable is declared as final in java, the value can't be modified once it is assigned.

~~2. method~~:

-> If any value has not been assigned to that variable, then it can be assigned only by the constructor of the class (~~If you try to manually assign the variable like~~)

final int n; // correct - only rule is final var is allowed one initialization
n = 10; // ~~compilation error~~

Java enforces the rule that a final variable must be assigned exactly once. If you don't assign it

at the point of declaration, the only places where you can assign it are in the Constructor(s) or initializer blocks

example of intializer blocks:

```
Public class myclass{
    private int x;
    private String y;
    {
        x=10;
        y="Hello";
    }  // Initializer block.
}
```

Initializer blocks are used to Initialize instance variables of a class. They are executed when an Instance of the class is Created, before the Constructor is called.

* If we simply write
  final int n;
then It must be assigned a value before it is used. using Constructors or initailizer blocks.

final method:

when the final keyword is used with method, it prevents the method from being overridden in any subclass. This is useful when you want to ensure that the Implementation of a method remains unchanged accross all subclasses.

* A Constructor Cannot be marked as final because whenever a class is inherited, the Constructors are not inherited. Hence making it final doesn't make Sense. Java throws Compilation error Saying- modifier final not allowed here

Final Class

when the final keyword is used with class, it prevents the class from being subclassed. This means that no other class can extend or inherit from a final class. This is useful when you want to prevent inheritence for Security, design reason

Do final, finally and finalize keyword in Java have the same function?

Ans No.

Final: used to define constants, prevent method overriding and prevent inheritence

finally: used in Exception handling to define a block of code that will always execute after 'try' block regardless of whether an Exception was thrown or not.

The finally block is typically used for cleanup tasks, like closing files or releasing resources, ensuring these actions occur regardless of whether an exception was thrown.

finalize: used to perform cleanup before an Object is garbage-collected

This method is defined in the 'Object' Class and can be overriden to release resources or perform other cleanup tasks

```
try {
}
finally {
    super.finalize()
}
```

Is it possible that the 'finally' block will not be executed?

Ans. Yes, It is possible that the 'finally' block will not be executed. The cases are.

- suppose we use System.exit() in the above statements.
- If there are fatal errors like stack overflow, memory access error etc.
- Infinite loop or deadlock - If try or catch block Contain an Infinite loop or if the thread is deadlock, the finally block may neves be reached.

when Can You Use a Super Keyword?

1. Accessing Superclass Method

→ we can use super.MethodName() to call the method of Super Class method until it is private because private method are not accessible.

2. Accessing Superclass Constructor

→ we can use super() to call the Constructor of parent class.

3. Accessing Superclass properties

we can use Super.NameOfProperty to get the value of variable of parent class

4. Accessing SuperClass Static Method

we cannot access static methods coz It can be accessed using class name.

Can static Method be overloaded?

Ans Yes, There can be two or more static methods in a class with the same name but different I/P parameters.

Can static methods be overriden?

Ans: No. Since static methods belong to the class itself and not to any specific Instance, they are not subject to overriding. Instead, If a Subclass defines a static method with the same Signature, It hides the Superclass's static method.

overriding or dynamic polymorphism occurs during the runtime, but the static methods are loaded and looked up at the Compile time statically. Hence, these methods can't be overriden

In Order to call the static method of parent class, we use parentName.methodName()

Difference Between Method Hiding and method overriding?

Method hiding and method overriding are concepts in java related to how methods are defined and called in class hirarchy.

Method Overriding:

Overriding occurs when a subclass provide a specific implementation of a method that is already present or defined in its Super Class.

Method overriding Supports runtime polymorphism. [dynamic method

dispatch.

Rules:

1. The overriden method cannot have a more restrictive access modifier than the method in Superclass.
2. The overriden method can throw the same or fewer exceptions compared to the method in the superclass.
3. The @override annotation can be used to indicate that a method is intended to override a method in the Superclass.

Method hiding

Method hiding occurs when a subclass defines a static method with same name and parameters list as a static method in the Superclass. The method in subclass hides the method in the Superclass.

use-case:

This method deal with static method. As overriding doesn't work for Static methods, method hiding is introduced. To give a specific defination for subclass (like overriding) we use the concept of method hidding.

→ It is Compile time Binding.

Difference b/w static methods, static variables, static classes in java?

static Method, Static Variables:

are those methods and variable that belong to the class of the java program, not the object of the class. This gets memory where the class is loaded. And these can directly be called with the help of class names. eg: we have mathematical functions in the java program like-max(), min(), sqrt(), pow() etc. And If we notice that, then we will find that we call it directly with the class name. (like. Math.max(), Math.min() etc. so this is a static method.

Static classes -

A static class in Java is nested (inner) class that is declared with the static keyword.

What is the main objective of garbage collection?

The main objective of the garbage collector is to automatically manage memory by reclaiming memory that is no longer in use by the program. when the object is no longer reachable or referenced by any part of program garbage collection identifies this unused memory and frees it, making it available for future allocation.

What is class loader?

A class loader in java is part of the JRE responsible for dynamically loading java classes into java virtual machine during runtime.

Key responsibilities of a class loader:

1- Loading class:

The primary responsibility of a classloader is to load class files into memory. it reads the bytecode of the class - from its source (e.g., a .class file) and converts it into a class object that the JVM can use. This converted into class object because The class object in java is representing a class in JVM. It provides a way for the JVM to interact with and manipulate the class at runtime. The class object contains metadata about the class Including information about its methods, fields, Constructors and other attributes.

2. Linking: After loading the class, the classloader performs linking, which involves

1. Verification: Ensuring the bytecode which is stored in adheres to Jvm Specifications. .class files.

Types of class Loader:

1. Boot Strap ClassLoader:

The root classloader, responsible for loading core Java classes (java.lang, java.util etc) from the 'rt.jar' file.

->Java allows developers to Create Custom Classloaders by extending the classloader Class.

About JDK, JRE and JVM.

JDK (Java Development kit).

JDK is a Software development kit that provides the tools necessary to develop Compile, debug and execute java applications.

Components:

Compiler ('javac'): Translate Java Source Code into bytecode.

JRE: The JDK includes the JRE, allowing developers to run Java programs during development.

Usuage: The JDK is used when you need to develop Java applications, as it provides all the necessary tools for Coding, Compiling and testing.

JRE (Java Runtime Enviornment)

The JRE provides the enviornment necessary to run java applicati-on. It containg everything needed to execute java programs but does not include development tools like the compiler.

Components:

1. JVM (Java Virtual Machine):

The core component that executes java bytecode.

Java class Libraries: A set of pre-written classes that provide commonly used functionality (eg: for data structur, I/O, networking etc).

usage: The JRE is typically installed on machines where you only need to run Java programs, not develop them. End users of Java applications typically only need the JRE.

JVM (Java Virtual Machine).

The JVM is an abstract computing machine that enables a computer to run java programs. It Interprets the compiled Java bytecode and translate it into machine code that can be executed by the host machine's CPU.

Component:

Classloader: loads Java class file into the JVM for execution.

JIT (just-in-time) Compiler & Garbage Collector

Shallow Copy & Deep Copy (already done).

Java Intermediate Questions

Apart from Security aspect, what are the reasons behind making strings immutable in java?

Ans: 1. performance Optimization ~~using~~ with String pooling:

In java, strings are stored in a special area of memory called the "String pool" (part of heap memory). Immutability ensures that strings in the pool remain unchanged and can be reused safely as If Same object is Created, the reference in pointed to the existing rather than Creating new object hence optimize memory usage and improves performance.

where JVM stores string literals

JVM checks all the stuff of string

2. Threat Safety: Immutability makes strings inherently thread safe. Since the state of the string cannot change after it is created, multiple threads can safely share and use the same string instance without the risk of one thread modifying the value while another using it. This simplifies multi-threaded programming and avoid Sychronization issues.

3. HashCode Caching:

If a string's value could change, its hashCode would have to be recalculated each time, making the retrieval of objects from hash-based collections less efficient. with Immutable strings, the hashCode remains constant, ensuring faster access time.

Compatibility with Security and network libraries:

Many Java libraries, espicially those dealing with Security (like Cryptography) and network communication rely on the immutability of String to function correctly. for example classloader use strings to define class name and to identify paths to resources.

How would you differentiate b/w String, StringBuilder, StringBuffer?

String- Immutable

- Thread Safety.

StringBuffer: mutable

: multiple thread cannot access the same StringBuffer instance simultaneously.

—> Synchronized meaning one thread can access the the StringBuffer at the same time makes it bit slower.

→ Suitable when working with strings that are modified frequently and thread safety is a concern.

StringBuilder:

- mutable
- Not Synchronized, makes 'StringBuilder" fastes than "StringBuffer".
- prefered over 'StringBuffer' when String manipulation is needed in a Single-threaded enviornment.

Difference Between interfaces and abstract class.

In java, both Interface & abstraction is used to achieve abstraction, which means hiding implementation details and showing only functionality.

Interface defines methods which doesn't have defination but has declaration. It tells what to do but not how to do. Its subclass must Implement the defination of the methods.

Abstract class can have both abstract method & Concrete methods we can't Instantiate the abstract class directly neither interface because they contain the method which doesn't have defination. If we do so, we get an error: class is abstract Cannot be Instantiated.

Methods

Interface: All methods are abstract by default

abstract: can have both abstract & Concrete.

Inheritance:

Interface: A class can Implement multiple ~~Inherits~~ Interfaces. This allows for multiple inheritance of type Sin Java does not multiple Inheritance with classes.

Abstract Class: A class can extend only one abstract class due to java Single inheritance.

Access Modifiers:

Interface: The methods in an interface

Implicitly 'public' variables in an interface are Implicitly 'public' 'static' 'final'.

Abstract class: Can have any access modifiers.

Constructor:

Interface: Cannot have Constructors because they cannot be instantiated directly.

Abstract class: AC can have constructors which are called when a subclass is instantiated.

Default and Static Method (Java 8+)

Interface: Since java 8, interfaces can have default methods (method with body with default accessibility) and static methods. static meth can be called Interface-name.MethName()

Abstract class: It can have all types of methods: abstract, concrete, static, and final.

variables are public static final by default because it can be access by any subclass and once the value is assigned, It can't be changed.

Why concrete methods (default & static) are Introduced with java 8. in Interface?

1. prior to java 8, adding a new method to an inface causes changes or updation in all the existing subclasses. Default method allows developers to add new methods to interface with a default Implementation, avoiding the need for existing classes to be modified immediately.

what is Comparator in java?

The Comparator interface is used to provide an external Comparison mechanism for sorting object. It's useful when you want to sort objects in a way different from their natural ordering

It contains compare(T O1, T O2) (Type) method which returns

A neg. Int. if O1 less than O2

+ve if $O_1 > O_2$

0 if $O_1 = O_2$.

```
int compare (Person P1 , Person P2)
    ret Integer. Compare(P1.getAge(),
                         P2.getAge());
}
```

Difference between HashSet and TreeSet?

Both Implements Set interface in java.

Data Structure:

HashSet: hash table to store elem.

TreeSet: red-black tree (self balancing binary search tree).

Ordering:

HashS: Doesn't maintain any order

TS: Maintains a sorted order.

Performance:

HS: offers O(1) for basic operations like 'add', 'remove'

TS: offers O(log n) for basic opprs.

Null Element:

HS: Allows the insertion of single null elemen t (if not al. present t)

TS: Doesn't allow null elements. Attempting to add a 'null' element will result in a 'NullPointerEx..'

why character array is often preferred over a string?

1. Mutability:

Char. Arr. is mutable, meaning the contents of the array can be modified after it is created

2. Memory Visibility:

using char[] arr, the developer has explicit control over when and how the contents of the array

are overwritten or clear.

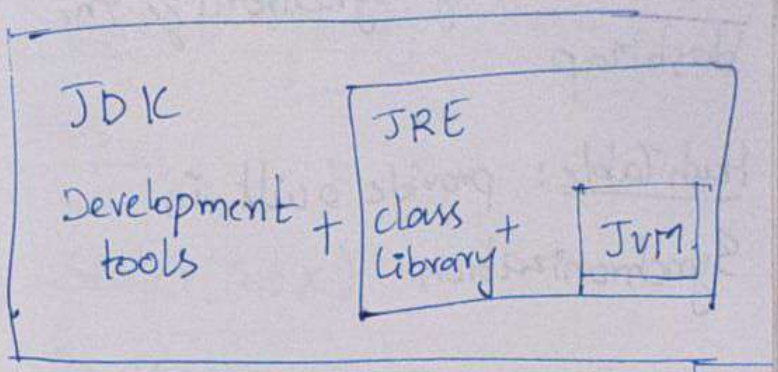
Due to Java's string pooling and optimization, the sensitive data may remain in memory longer than expected and could be potentially be visible in memory dumps (snapshot which a copy of current state of system).

3. Garbage Collection and Security:

→ After manually clearing the contents of a char[], even if a memory dump is performed, the sensitive Information is less likely to be exposed.

→ A String can be left in the memory until it is garbage collected. If memory dump is performed before garbage collection then sensitive data can be exposed.

What do we get in jdk file?



What are the difference between HashMap and HashTable in Java?

HashMap and HashTable are both data structures in java the implements the map interface, which means they store data in key-value pairs.

1. Thread Safety:

HashMap: Not thread-safe. Multiple thread can access a 'HashMap' simultaneously without proper sync.

HashTable: Thread Safe. All public methods in 'Hashtable' are Synchronised

2. Synchronization:

HashMap: Does not provide built-in Synchronization. if thread safety is required, the programmer

must manually Synchronize the HashMap.

Hash Table: provide built-in Synchronization.

Null keys and values

HM: Allows one 'null' key and multiple null values.

HT: Doesnot allow 'null' key or values.

Performance:

HM: fastes due to not Synchroni-zation. unsafe.

HT: Slower but safe.

what is the Importance of reflection in java?

It is a feature that allows a program to examine and manipulate the internal properties of classes, methods, and fields at runtime.

what are the different ways of thread usuage?

Threads in java allows multiple tasks perform simultaneously, by enabling Concurrent execution of code. They execute different parts of code simultaneously. for example one thread might be processing user input while another thread handles network Communication.

1. Extending the 'Thread' Class

The simplest way to create the thread is by extending the 'Thread' Class and overriding its 'run' method.

2. Implementing the 'Runnable' Interface.

Another common way to create a thread is by implementing the 'Runnable' interface. This is a more flexible approach because it allows the class to extend another class while still defining the thread's behaviour.

Different types of thread priorities in java?

* Thread priority helps the thread scheduler decide the order in which threads should be executed. The priority of a thread is represented by an integer value.

1. Thread. MIN_PRIORITY (value:1) minimum priority a thread can have
2. Thread. NORM_PRIORITY (value:5) This is the default priority assigned to a thread if no explicit priority is set.
3. Thread. MAX_PRIORITY (V:10)

What is the difference b/w program and the process?

* A program doesn't execute directly by the CPU. First, the resources are allocated to the program and when it is ready for execution then it is a process.

Important regading Super() & this

```
Scaler(int x)
{
  this();
  Super();
}
```

Compilation error. because Super() is used to call the parent class constructor and It should be first statement in the block.

If this() replaced with Super() then also it will throw Compilation error. Because this() also has to be the first statement in the block. so, in nutshell, we can say that we cannot use this() and Super() keywords in the same block.

Java works as "pass by value" or "pass by reference" phenomenon?

Java works as pass by

value. When object is send to the method, Java creates the copy of reference of the object and sent it to the function in the form of value.

What is the 'IS-A' relationship in OOPs Java?

Any IS-A relationship is another name for inheritance. when we inherit the base class from the derived class then it forms a relationship blw the classes. so that relationship is termed as 'IS-A' Relationship.

eg SMART-TV is a TV

What is Serialization In Java? How to not allow Serialization of attributes of a class?

Any It is a process of converting an objects's state into byte stream so that it can be easily saved to a file or transmitted over a network. It is useful for persisting objects or sending them between different parts of a program or different systems.

—> when a class Implements the 'Serialization' interface, all of its attributes are Serialized by default.

-> If you want to prevent specific attribute from being Serialized, we can use transient keyword.

eg: private transient String Password;

What happens if the static modifier is not Included in the main method Signature?

There wouldn't be any Compilation error, But then the program is run, since the JVM can't map the main method Signature, the code throws "No Such Method Error" at runtime.

Q) Find Output of the following code

Code:

```
public class Main {
public static void main (String[]
args)
{
    Sout ("Hello, main method");
}
public static void main (int[]
args)
{
    sout ("Hello, Main Method2");
}
}
```

Output: Hello, main method.

JVM will always call the main method based on the defination it already has. Doesn't matter how many methods we overload it will only execute one main method based on its declaration in JVM.

Concept of threading in java.

Threading in java allows a program to perform multiple task concurrently. Java's threading model is built in to the language.

1. Thread :-

→ A thread is a lightweight process. It is a single path of execution.

→ Each thread in java run independently and can perform tasks concurrently with other threads

User Threads vs Deamon Thread

User Thread :

→ These are regular threads that perform the core functions of Java app.

→ The JVM continues to run as long as there are user thread active. When all user threads have finished, the JVM can exit.

Daemon Threads:

These are background threads that do not prevent or interfere in java application's execution. They are typically used for background tasks such as garbage collection or monitoring.

Tasks that can be performed by Thread

1. perform Concurrent Tasks
2. Run Background Tasks
3. Handle user Interaction.
4. Manage Resource-Intensive Operations.

Thread Management Approaches:

1. Single Thread:

Single threaded application runs one thread of execution. This is the simplest approach, where only one task is executed at a time.

Usuage: for those that do not require Concurrent execution like simple scripts.

Adv:

→ Simple to Implement and debug.
→ No need for Synchronization or Concurrency Control.

Disadv:

→ limited performance and responsiveness that require handling multiple tasks or user interaction.

2. Multi threading:

→ Using multiple threads to perform different tasks concurrently. This approach can improve performance.

Usuage: change content of

usage of single thread according to the multithreading.

Disadv:

-> Increased complexity in managing thread synchronization.

-> potentials for race-conditions, deadlocks, and other synchronization problems.

3. Thread pools:

Thread pool manages a collection of worker threads that execute tasks concurrently. This approach allows for efficient reuse of threads and control over the number of concurrent threads.

-> Suitable for managing a large number of short-lived tasks or when the overhead of creating and destroying threads is significant.

Adv:

-> Reduces the overhead of creating and destroying threads by reusing existing threads.

Can we make the main() thread a daemon thread?

In java multithreading, the main() threads are always non-daemon threads. And there is no way we can change the nature of the non-daemon thread to the daemon thread. Main() thread is a user thread.

What happens if there are multiple main methods in java

The program will compile successfully because java allows method overloading. Having multiple methods with the same name but different parameters is valid.

-> If there are multiple overloaded main methods, only the one with exact signature 'public static void main (String[] args)' will be recognized and executed by the JVM as the entry point.

What do you understand by object cloning and how do you achieve it in java?

Object cloning in java is the process of creating an exact copy of an object.

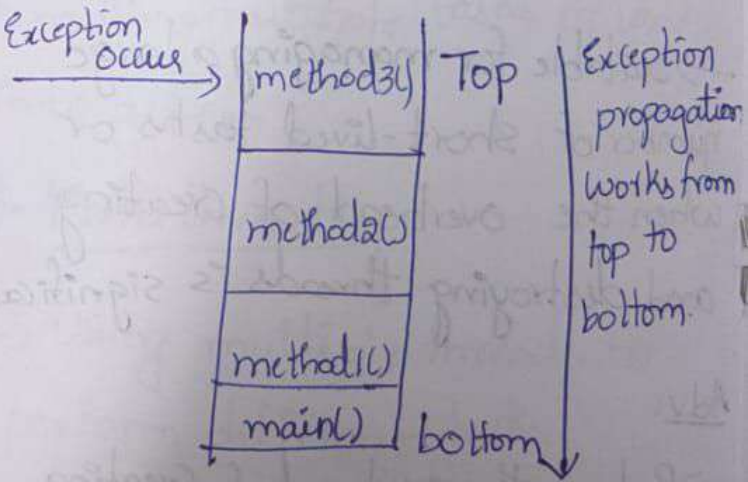
1. Shallow Cloning: primitive type's actual data is copied. Object's reference is copied not the obj.
2. Deep cloning: In deep cloning, a new object is created and all nested objects are recursively cloned.

Java provide a built-in mechanism for object cloning through 'cloneable' interface and the 'clone()' method defined in the 'object' class. The clone() method is defined in the Object class and is protected by default. To make it accessible, you must override it in your class and declare it public.

In shallow clone, we just copy reference, but In deep clone, we copy address of the object.

How does an exception propagate in the code?

When an exception occurs, first it searches to locate the matching catch block. In case, the matching catch block is located, then that block would be executed. Else, the exception propagates through the method call stack and goes into the caller method (previous method) where the process of matching the catch block is performed. This process repeats until a matching 'catch' block is found. If the match is not found, then the program gets terminated in the main method.



How do exceptions affect the program if it doesn't handle them?

Ans Exceptions are runtime errors. If they are not handled then it may cause a bad experience for user and maybe the application will crash. Suppose, you are making a application with java. And it works fine but there is an exceptional case when the application tries to get the file from the storage and file doesn't exist.

Is it mandatory for a catch block to be followed after a try block?

No, it is not necessary for a catch block to be present after a try block. A try block should be followed either by a catch block or by a finally block. If the exceptions likelihood is more, then they should be declared using the throws clause of the method.

Will the finally block executed when the statement is written at the end of try block and catch block

```
try {
  }
Catch(Exception e) {
  return 999;
  }
finally {
  }
```

Yes, the finally block will be executed. finally block is designed to execute after the try and catch blocks, ensuring that important cleanup code (such as closing resources) is executed. The only case where finally block is not executed is when it encounters 'System.exit()' method anywhere in try/catch block.

Can you call a Constructor of a class inside the another Constructor?

Ans Yes, we can using this(). This is called as Constructor chaining.

Contiguous memory locations are usually used for storing actual values in an array but not in arraylist. Explain:

In the case of arraylist, data storing in the form of primitive data types (like int, float, etc.) is not possible because it is designed to store object. The data members / object present in the Arraylist have references to the object which are located at various sites in the memory. Thus storing of objects or actual non-primitive data types (like Integer, Double) etc. take place in various memory locations.

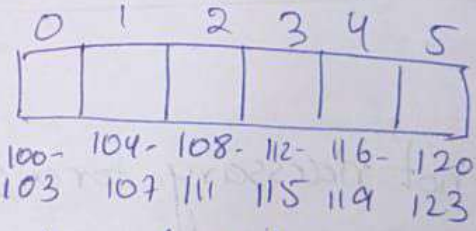
However, the same does not apply to the arrays. object or reference of obj primitive type values can be stored in arrays in contiguous memory locations, hence every element does not require any reference to the next element.

Why does the java index start with 0?

There are several reasons:

1. Because the 0index array avoids the extra arithmetic operations to calculate the memory address

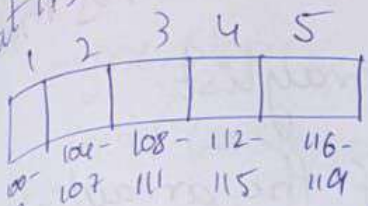
eg: Consider the array and assume



Each element takes 4-byte memory space.

Now, if I want to access the index 4, Then java Internally calculates the address using the formula:

[Base Address + (index * no_of_bytes)]. So according to this, we can easily get the address (100 + (4 * 4)) = 116



and calculate address using the same formula

(100 + (4*4)) = 116

which is wrong address and hence need some extra calculation to find the exact address. And in case of calculating millions of address, this causes complexity.

2. Influenced from clang: Java was heavily influenced by c and C++ language which also uses Zero-based indexing. The original decision to use Zero-based indexing in c was influenced be early language like BCPL and assembly language.

Why remove method is faster in Linked List Compare to an array?

Ans: In linked list, we only need to adjust the references when we want to delete the element from either end or the front of the linked list. But in arrays, Index are used. In order to manage Index, when we delete a element of 0th Index, we need to switch the or shift the element ahead which makes it more complex Compare to linked list removal.

How many overloaded add() and addAll() methods are available in the List interface?

There are total of 4 overloaded methods for add() and addAll() methods available in List interface.

1. add(Element e): This method is used for adding the element at the end of the list. It returns the boolean indicating successfully inserted or not.

2. add(int idx, Element e): This method is overloaded version of add() method. In this, along with the element, an idx is passed.

return type is void.

3. addAll(Collection<extends ?Element> c)

This method helps to add all elements at the end of collections from the list received in the parameter. It contains an iterator that helps to iterate the list and add the elements to the collection.

4 - addAll(int idx, Collection<extends ? Element>c): overloaded method of addAll which has an idx.

How does the size of Arraylist grow dynamically? And also state how it is Implemented internally?

Ans Consider initially that there are 2 elements in the Arraylist. [2, 3].

If we add the element into this, Arraylist will allocate the new Arraylist of size (Current Size + half of the current size) grown by 50%.

And add the old elements into the new arraylist.

default size of the arraylist is 10 (if not specified).

Composition In Java.

Composition is an OOP Concept where a class is Composed of one or more objects from other classes. It describes has-a relationship. Instead of inheriting from a parent Class, a class holds reference to one or more objects of other class.

eg:

```
class Engine{
    Pub void start(){
        Sout("Engine started");
    }
    Pub voi Stop(){
        Sout("Engine Stop")
    }
}
class Car{
    private Engine engine;
pub Car(){
    this.engine=new Engine();
}
```

```
Public void stop(Car c) {
    engine.stop();
    sout("Car stopped");
}
public void start(Car c){
    engine.start();
    sout("Car started");
}
}
Public Class main {
    psvm(sarg[]) {
    Car car = new Car();
    car.start(car); // engine.start() is in start(car) Method. It is private, that's why can't directly access it.
    car.stop(car);
    }
}
```

Although Inheritence is a popular OOPs concept, it is less.

Inheritence: Inheritence is an OOP concept where a new class (child or Subclass) is derived from an existing class (parent or Super class)

Adv

Code Reusability: Allows the reuse of code across multiple classes,

Polymorphism: Supports polymorphism

dis

Tight Coupling: child class are Tightly Coupled with parent class. changes in parent class inadvertently affect the child class.

Adv & disadv of Composition

1. Easier to understand: Composition ~~can be~~ tends to result in more understandable and maintainable code since it avoids complex inheritance chain and fragile base class issue (changes made in parent class effects to child as well)
2. Encapsulation: classes using ~~encapsulation~~ Composition can better encapsulate and hide their internal details.
3. Loose Coupling: Since classes are composed of other classes rather than inheriting from them, there is less dependency.

dis

Can be less Intuitive Initially:

For developers new to OOP, Composition can be less intuitive than inheritance as it involves creating relationship b/w diff. obj's. rather than extending existing ones.

Why Composition is often preferred over inheritance?

1. Maintainability: Code that uses Composition is generally easier to main tain and modify. Since classes are loosely coupled, we can change one class without worrying about breaking others.

2. Scalability: Composition allows for better scalability, You can easily add new functionality by adding a new class and Composing them, rather than having to modify complex inheritance hirarchies.

what is the difference between >> and >>> operators in java?

1. >> (Arithemetic Right shift).

>> operator is an arithmetic (or signed) right shift operator. It shifts bits to the right while preserving the sign of the number.

eg: a= -8

11111111 11111111 11111111 11111000

a>>2

11111111 11111111 11111111 11111110.

a=-2 in decimal.

2. >>> (Logical Right shift).

>>> doesn't preserve the sign. It shifts the right most bit and fills the leftmost bits with '0', regardless of the original sign of the number.

eg a>>>8

00111111 11111111 11111111 11111110

a=1073741822 in decimal.

Aggregation

Aggregation in java is a design principle that represents a "has-a" relationship between two class.

unlike Composition, where the object of other class is created in a class, In aggregation object are created in other function. and hence we use a data structure (inside the class which stores the reference of other class) to store those references.

eg:

```
class Book{
    private String title;
    public Book(String title)
    {
        this.title = title;
    }
    public String getTitle(){
        return title;
    }
}

class Library{
    private List<Book> books; //Aggregation
    public Library(List<Book> books){
        this.books=books;
    }
    public void addBook(Book book)
    {
        books.add(book);
    }
    public List<Book> getBooks(){
        return books;
    }
}
```

Difference B/w aggregation and Composition

In Composition, as the objects of other classes are created inside a class (also called whole), if the object of whole is destroyed, we will also lose the object of other class as well. In aggregation, even the object of the class (which stores the reference of object of other class) is destroyed we can still access them. They are loosely coupled

When to use?

Composition:

1. Tight Coupling: You want a strong relationship where the child object's lifecycle is tightly coupled with the parent. If the parent object should control the creation and destruction of the child objects, Composition is better choice

Aggregation:

Loose Coupling: You want a weaker relationship where the child objects can exist independently of the parent. Aggregation is more flexible in scenarios where objects are not tightly dependent on each other.

How is the creation of a String using new() different from that of a literal?

In java, there are two main ways to create a string object using string literal and using the 'new' keyword.

1. String Creation Using a literal:

when we create string using literal

String str1 = "Hello";

-> Java maintains a special area in memory called the string pool where string literals are stored.

-> If a string literal with the same content "Hello" already exists in the string pool, JVM does not create a new 'String' object. Instead, It reuses the existing string object from the pool.

2. String Creation Using 'new' keyword.

String str1 = new String("hello");

the new keyword creates a new string object in heap memory, even if an identical String already exists in the string pool.

No reuse.

How is the 'new' operator different from the 'newInstance()' operator in java?

In java, the 'new' and the newInstance() method are both used to create new objects, but they have different purposes, use cases, and behaviours.

New:

Instantiates a new object of specified class and calls its constructor. Object is created in compile time.

→ Type safety is ensured, meaning the compiler checks that the class name is correct and the constructor is accessible.

newInstance()

It is the part of java.lang. class.

Creates objects of a class dynamically at runtime.

Throws several checked exception like 'InstantiationException'.

About ~~that~~ Error & Exception.

Diff b/w checked and unchecked exception. (ans is in further pages)

is exceeding the memory limit possible in a program despite having a garbage collector in java?

Ans Yes, garbage collector in java helps manage memory by automatically reclaiming memory that is no longer in use.

1. Memory Leaks Due to Retained References:

In java, a memory leak (ML occurs when a program fails to release memory that is no longer needed) occurs when objects are no longer needed but are still referenced. Since garbage collector only reclaim memory for objects that are no longer reachable.

2. Excessive Memory Allocation.

If a program allocates memory faster than the garbage collector can reclaim it, the JVM might run out of memory.

3. Large Object Creation.

If a program creates very large objects that exceed the java heap space the program can throw an 'outofMemoryError'.

4. Improper JVM HeapConfiguration

→ JVM uses heap to store objects. The size of the heap is also limited which depends on the yours RAM of your system. Also we can configure the size using JVM. If the maximum heapsize (-Xmx parameter) is set to too low for the application's requirement the program may run out of memory.

What is Error & Exception. Diff b/w checked and unchecked exception?

Error is a serious problem that occurs during the execution of a program that is usually not recoverable. Errors typically happen due to faults in the enviornment in which the application is running, such as hardware failure, running out of memory or a stackoverflow.

Exception is an event that disrupt the normal flow of program's instructions. Exceptions are typically due to logic error in code, such as divide by zero, accessing invalid Index etc..

Checked Exception:
exceptions that are checked at compile-time by the compiler. This means that the programmer must handle these exception using try-catch blocks or declare them using the throws keyword. Eg: IOException, ClassNotFound exception etc..

Unchecked Exception:
exception that are not checked at compile time but are checked at runtime. They are also known as runtime exception.
Eg: Nullpointer exception, Arithematic exception etc..

Why Synchronization is Important?
When multiple threads attempt to read and write shared data simultaneously, they may interfere with each other's operations, leading to inconsistent or corrupt data.
for example, you want to calculate the number of viewers without Synchronization, when multiple ~~threads access the~~ request the website, the first thread T1 and ~~checks~~ the Count by 1 (If Increments the total Count is 10) and hence total views becomes 11. Simultaneously If another thread Increments the value, It could read the value as 10 and hence It will also Increment it to 11 but the actual Count is 12. Hence this makes inconsistancy in data & Information.

What is the Significance of ... in below code?

```
public void fooBarMethod(String ... variables) { }
```

The 'String ... variables' Syntax in the method parameter list is called varargs (variable length arguement). It allows the method to accept Zero or more arguements of

Specified type.

eg:

```
Class Example{
  public void fooBarMethod(
  String... variables){
    for(String V : variables)
      {
        sout(v);
      }
  }

PSVM[S[]ar){   indicates the method call which is Constructor
  Example e=new Example()
  example - fooBarMethod();
  example.fooBarMethod("one");
  }
```

Internally, the varargs parameter is treated as an array. So, 'variables' inside method is actually a String[].

What is the use of Constructor other than initializing the property of class?

1. Enforcing Constraints or validation:

Constructors can be used to apply certain Constraints when object creating

eg:

```
Public class BankAccount{
  private double balance;
  public BankAccount(
  double intialDepo){
    if(initialDepo<0){
    throw new IllegalArgumen
    Exception("Initial depo
    Can't be -ve");
    }
    this.balance= InitialDepo;
  }.
```

we can add a Constraints

that the initial bankaccount balance should not be -ve.

2. Resource Allocation:

Constructors can be used to allocate resources such as memory, file handles or network connections. This is useful when the object requires certain resources to function correctly. for example, a constructor could open a file and prepare it for reading or writing.

what is polymorphism. why it is used?

polymorphism in programming is a concept that allows one piece of code (like function or method) to work in different ways depending on the object it is working with.

Eg:- Smartphone can be used to make calls, take photos, send messages, or browse the internet.

why use polymorphism?

1. Flexibity: You can use the same code for different types of objects

Eg:-

```
Class Animal {
    void makeSound() {
        Sout ("Animal make Sound");
    }
}

Class Dog extends Animal {
    @override
    void makeSound() { //Flexibity
        Sout ("Dog Barks");
    }
}

public Class Main {
    PSVM(S[] a) {
        Animal myAnimal = new Anu();
        Animal myDog = new Dog(); //
                                  diff. obj.
```

2. Simpler Code: You write less code because one function can handle different types of actions.

Execution order of Code. which static method, Instance Initializer block is given in a code snippet.

① Static blocks are executed first among all before any method. Executed from top to bottom.

② Identification of static members from top to bottom.

③ Execution of the main method

④ Identification of Instance members from top to bottom.

⑤ Execution of Instance variable assignment and Instance block from top to bottom

⑥ Execution of Constructor.

Instance Initializer block is executed before Constructor because:

When an object of a class is created (using the 'new' keyword), Java performs several steps in the following order:

1. Memory is allocated for the new object.

2. Instance variables are initialized to their default values.

3. Instance block are used to execute common code that should run every time an instance is created before Constructor is called.

Define System.out.println().

System.out.println() is a commonly used statement for printing text to the Console.

System: It is a final class in java.lang package. It provides several useful class fields and

methods that are related to the system environment, including standard input, output and error output streams.

2. out:

It is a static field in the System class. It is the Instance of 'print Stream' which is a class that provides various methods to print data. out field represent the standard output stream, which is typically console where program is running.

public static final printstream out;

Static represents that out belongs to the System Class but not the Instance of it.

println(): This is a method of the 'printstream' class. This method is used to print a message to the console.

Thread LifeCycle

New - when the Instance of thread is created, and start() method is invoked, the thread is considered to be alive and hence is in NEW state.

Runnable - once the start() method is invoked, before the run() method is called by the JVM, the thread is said to be Runnable (ready to run) state.

Running: when the run() method has been invoked and the thread start its execution, the thread is said to be in a RUNNING State.

Non-Runnable (Blocked/waiting) when the thread is not able to run despite the fact of its aliveness, the thread is said to be in a Non-Runnable state -

Blocked: A thread is said to be in a blocked state if it wants to enter Synchronization code but it is unable to as another thread is operating in that Synchronized block.

Waiting: A thread is said to be in waiting state if it is waiting for the signal to execute from another thread i.e., It waits for work until the signal is received.

Terminated: once the run() method execution is completed, the thread is said to enter the Terminated step and is considered to not be alive.

What could be the tradeoff between the usage of an unordered array versus the usuage of an ordered array.

Unordered array: ~~Array~~ elements are not sorted, they are store in the order in which they are inserted.

Ordered array: the array is sorted either in decreasing or increasing order.

How to declare them?

→ we use normal array and by default, the arrays are unsorted (unordered)
→ when we apply any sorting algorithm, then it becomes ordered array.

Trade-offs

1) Insertion is fast, but ~~deletion~~ searching & deletion is slow in unordered.
2) Insertion is slow, but searching is fast.

Main adv of having ordered array is the reduced search time complexity of O(log n) Binary search.

Is it possible to import the same class or package twice in java?

Ans: It is possible, but JVM only considers one import statement and ignores remaining all.

In case a package has sub ~~class~~ packages, will it sufficient to import only the main package?

No, Importing a package in java does not automatically import its sub package. In java, packages and sub-packages are considered distinct. Even though, they are logically nested, the package structure doesn't imply automatic inheritance of imports. we need to explicitly import child ~~class~~ or sub packages.

What do you understand by marker interfaces in java?

It is a Interface that has no methods or fields; it serves as a marker or tag to indicate something to the compiler or JVM.

what do you understand by marker interfaces in java?

{{}} is a convinient means of initializing any collections in java.

eg:

```
List<String> mylist = new
ArrayList<String>() {{
    add("Apple");
    add("Banana");
}};
Sout(mylist); //o/p: App Ban..
```

→ The first pair of braces
new ArrayList<String>() { }
Creates an anonymous subclass of Arraylist

→ Instance Initializer Block:
The second pair of braces {{... }}
is an Instance initializer block that adds elements to the list.

Advantages:
Its a more compact way to initialize collections or objects with value right after creation.

Disadvantages:
Creates an Anonymous Class:
It Creates an anonymous inner class, which can be less efficient because it leads to additional overhead, memory as well.

What are the possible ways for making object eligible for garbage-collection (GC)?

① Set the object references to null once the object creation purpose is saved.

```
PSVM(S[]ar) {
String S1= "Some String";
   S1=null
}
```

② point the reference variable to another object. Doing this, the object which the reference variable was referencing before becomes eligible for GC

```
PSVM(S [])
{
String S1= "To Garbage Collect";
String s2 = "Another Obj";
Sout(S1);
S1=S2.
}
```

what is the best way to inject dependencies?

Ans The best way to inject dependency is Constructor injection.

1. Immutability: With Constructor injection, dependencies are passed to the object when it's instanced, ensuring that they cannot be modified after creation. This promotes immutability, which leads to more stable and predictable objects.

2. Clear Intent: The Constructor explicitly shows what the class depends on. This Improve code readability.

eg:

```
Class Car {
    private Engine engine;
    public Car (Engine engine){
        this.engine= engine; // dependency is injected through Constructor.
    }
    public void drive(){
        engine.start();
    }
}
```

In this example, the car depends on Engine. Constructor injection ensures than an engine is provided when a car is created.

How can we Set the Spring bean Scope. And what Supported Scopes does it have?

A Scope can be set by an annotation such as @Scope. Spring Bean Supports the following five Steps:

— Singleton — Session.
— prototype — Global-Session.
— Request.

* learn about Java Design pattern. Self learn.

What is memory leak?

In memory leak, leak represents unintended or uncontrolled loss of memory. Just like water leaking from a container, memory leak

When a program allocates memory but fails to release it back to the system when it's no longer needed over time, these small leaks can accumulate, eventually leading to inefficient memory usuage and if serve enough, causing the system to run out of memory.

Common Causes of Memory Leaks:

1. unreleased Object Reference:

- when a program holds references to objects that are no longer needed, the garbage collector cannot free the memory occupied by these objects. The garbage collector is a part of JVM.

2. Failing to close Resources:

- failing to release resources such as files, database connections or network sockets can result in memory leaks.

3. Static References:

Static variables are class-level variables, meaning they belong to the class, not the instance of the class. They are initialized once when the class is loaded and live until the jvm shuts down.

Objects referenced by static variables persist as long as the class is loaded, making them ineligible for garbage collection.