Index

Java Architecture and Memory Management..... 1
Java Fundamentals..... 2
Object Oriented Programming Concepts..... 7
Java 8 Basics..... 16
Collection Framework..... 17
Design Patterns and Principles Basics..... 21
Concurrency and multi-threading..... 22
Miscellaneous questions..... 23

Java Architecture and Memory Management

Can you tell me the difference between JVM, JRE, and JDK?
The JVM is the engine that runs Java bytecode and making Java platform-independent.
The JRE contains the JVM and the standard libraries that Java programs need to run.
The JDK is development kit for developers that contains everything in the JRE plus tools like compilers and debuggers to create Java applications.

What are the key components of JVM Architecture?
JVM has three components, the ClassLoader, the runtime data areas and the execution engine.
The Class Loader loads class files into the JVM. The Runtime Data Areas store data needed while the program runs, like memory for variables and code. The Execution Engine actually runs the instructions in the class files.

Can a Java application be run without installing the JRE?
We can't run a Java application without having the JRE (Java Runtime Environment) because it has the essential tools and libraries the application needs to work. But, there's a cool tool called jlink in newer Java versions that lets us bundle our Java application with its own little version of the JRE

Is it possible to have the JDK installed without having the JRE?
No, the JDK contains the JRE. It's not possible to have a JDK without a JRE, as the JRE contains essential components for running Java applications, which the JDK also uses for development.

What are Memory storages available with JVM?
VM memory is divided into Heap Space, Stack Memory, Method Area (Metaspace in Java 8 and above), and Native Method Stacks.
Heap space in Java is where the program stores objects and data that it creates and shares.

Stack memory is used for keeping track of what happens inside each function call, including variable values.
The Method Area, or Metaspace in newer Java versions, stores information about the program's classes, like methods and constants.

Which is faster to access between heap and stack, and why?
The stack is faster to access because it stores method calls and local variables in a Last-In-First-Out (LIFO) structure. The heap, used for dynamic memory allocation (objects), is slower due to its more complex management.

How does garbage collection work in Java?
Garbage collection in Java automatically frees memory by removing objects that are no longer used. It frees the memory by unused objects, making space for new objects.

Whats the role of finalized() method in garbage collection?
The finalize() method is called by the garbage collector on an object when it determines that there are no more references to the object. It's meant to give the object a chance to clean up resources before it's collected, such as closing file streams or releasing network connections.

Can you tell me what algorithm JVM uses for garbage collection?
JVM uses multiple garbage collection algorithms such as Mark-Sweep, Mark-Compact, and Generational Copying, depending on the collector chosen

How can memory leaks occur in Java even we have automatic garbage collection?
Memory leaks in Java occur when objects are no longer needed but still referenced from other reachable objects, and hence preventing the garbage collector from reclaiming their memory.

Java Fundamentals

Is java 100% object oriented programming language ?
No, Java is not considered 100% object-oriented because it uses primitive types (like int, char, etc.) that are not objects. In a fully object-oriented language, everything is treated as an object.

What are the advantages of Java being partially object-oriented?

1. Using simple, non-object types like integers and booleans helps Java run faster and use less memory.
2. The mix of features allows Java to work well with other technologies and systems, which might not be fully object-oriented.

What is the use of object-oriented programming languages in the enterprise projects?

Object-oriented programming (OOP) is used in big projects to make coding easier to handle. It helps organize code better, makes it easier to update and scale, and lets programmers reuse code, saving time and effort.

Explain public static void main(string args[])?

In Java, public static void main(String[] args) is the entry point of any standalone Java application.

public makes this method accessible from anywhere, static means I don't need to create an object to call this method, void means it doesn't return any value, and main is the name of this method.

The String[] args part is an array that holds any command-line arguments passed to the program. So, when I run a Java program, this is the first method that gets called

What will happen if we declare don't declare the main as static?

If I don't declare the main method as static in a Java program, the JVM won't be able to launch the application.

As aresult, the program will compile, but it will fail to run, giving an error like "Main method is not static in class myClass, please define the main method as: public static void main(String[] args)."

Can we override the main method?

No, we cannot override main method of java because a static method cannot be overridden.

The static method in java is associated with class whereas the non-static method is associated with an object. Static belongs to the class area, static methods don't need an object to be called.

Can we oveload the main method?

Yes, We can overload the main method in java by just changing its argument

Can JVM execute our overloaded main method ?

No, JVM only calls the original main method, it will never call our overloaded main method.

Whats the difference between primitive data types and non primitive data types ?

Primitive data types in Java are the basic types of data predefined by the language and named by a keyword. They have a fixed size and are not objects. Examples include int, double, char, and boolean.

Non-primitive data types, on the other hand, are objects and classes that are not defined by Java itself but rather by the programmer or the Java API. They can be used to call methods

to perform certain operations, and their size is not fixed. Examples include String, arrays, and any class instances.

Can primitive data types be NULL ?
No, primitive data types in Java cannot be null. They have default values (e.g., 0 for int, false for boolean, 0.0 for double) and must always have a value.

Can we declare pointer in java ?
No, Java doesn't provide the support of Pointer. As Java needed to be more secure because which feature of the pointer is not provided in Java.

What is the difference between == and .equals() in Java?
== compares object references (whether two references point to the same object), while equals() compares object content (whether two objects are logically equal).

What are wrapper classes?

In Java, a wrapper class is an object that encapsulates a primitive data type. It allows primitives to be treated as objects. Each primitive data type has a corresponding wrapper class (e.g., Integer for int, Double for double).

Why do we need wrapper classes?

1. Wrapper classes are final and immutable
2. Provides methods like valueOf(), parseInt(), etc.
3. It provides the feature of autoboxing and unboxing.

Why we use wrapper class in collections
Because Java collections, such as ArrayList, HashMap, and others in the Java Collections Framework, can only hold objects and not primitive types. Wrapper classes allow primitive values to be treated as objects, enabling them to be stored and managed within these collections.

Can you explain the difference between unboxing and autoboxing in Java?

Autoboxing automatically converts a primitive type (like int) to its corresponding wrapper class (Integer). Unboxing does the reverse, converting an Integer back to an int.

Can you provide an example where autoboxing could lead to unexpected behavior?
When comparing two Integer instances using ==, autoboxing might lead to false results because it compares object references, not values, for integers outside the cache range of -128 to 127.

Is there a scenario where autoboxing and unboxing could cause a NullPointerException?

A NullPointerException can occur if you unbox a null object; for example, assigning null to an Integer and then using it in a context where an int is expected.

Can you explain the role of each try, catch, and finally block in exception handling?

try block conatins code that might throw exceptions. catch handles those exceptions. finally executes code after try/catch, regardless of an exception, typically for cleanup.

What happens if a return statement is executed inside the try or catch block? Does the finally block still execute?

The finally block executes even if a return statement is used in the try or catch block, ensuring cleanup runs.

Is it possible to execute a program without a catch block? If so, how would you use try and finally together?

Yes, we can use try with finally without a catch block to ensure cleanup occurs even if we allow the exception to propagate up.

How does exception handling with try-catch-finally affect the performance of a Java application?

Using try-catch-finally can affect performance slightly due to overhead of managing exceptions but is generally minimal unless exceptions are thrown frequently.

Can you tell me a condition where the finally block will not be executed?

The finally block will not execute if the JVM exits via System.exit() during try or catch execution.

Can we write multiple finally blocks in Java?

No, each try can only have one finally block. Multiple finally blocks are not allowed within a single try-catch-finally structure.

What is the exception and the differences between checked and unchecked exceptions?

Exception is the unwanted even that occurs during the execution of program and disrupts the flow.
Checked exceptions must be declared or handled (IOException); unchecked do not need to be declared or caught (NullPointerException).

How would you handle multiple exceptions in a single catch block

Use a single catch block for multiple exceptions by separating them with a pipe (|), e.g., catch (IOException | SQLException e), to handle both exceptions with the same logic.

What is the difference between a Throwable and an Exception in Java?

Throwable is the superclass for all errors and exceptions. Exception is a subclass of Throwable representing recoverable conditions, while Error (another subclass) represents serious issues the application should not attempt to recover from.

Discuss the difference between finalize() and finally. Under what circumstances might finalize() not get called in a Java application?

finalize() is called by the garbage collector before an object is destroyed, while finally is used in a try-catch block to execute code regardless of exceptions. finalize() may not get called if the garbage collector doesn't run or the JVM shuts down.

What is string pool?

A Java String Pool is a place in heap memory where all the strings defined in the program are stored. Whenever we create a new string object, JVM checks for the presence of the object in the String pool, If String is available in the pool, the same object reference is shared with the variable, else a new object is created.

Are there any scenarios where using the string pool might not be beneficial?

It will not be beneficial when there are a lot of uique string because it will be complex situate to check each string.

Can you please tell me about String and string buffer?

'String; in Java is immutable, meaning once created, its value cannot be changed. 'StringBuffer' is mutable, allowing for modification of its contents and is thread-safe, making it suitable for use in multithreaded environments where strings need to be altered.

How does StringBuilder differ from StringBuffer, and when should each be used?

StringBuilder is similar to StringBuffer but is not thread-safe, making it faster for single-threaded scenarios.

Give a scenario where StringBuffer is better than the String?

A scenario where StringBuffer is more appropriate than String is in a multi-threaded server application where multiple threads modify a shared string, such as constructing a complex log entry concurrently from different threads.

What is the difference between a String literal and a String object?

A String literal is stored in the String pool for reusability. A String object, created using new String(), is stored in the heap, even if it has the same value as a literal.

Why is String immutable?

String is immutable to improve security, caching, and performance by ensuring that its value cannot be changed once created.

What are the packages in Java?

In Java, packages are namespaces that organize classes and interfaces into groups, preventing naming conflicts and managing access control. They provide a structured way to manage Java code, allowing related classes to be grouped together logically.

Why packages are used?

1. They help in organizing code
2. Packages prevent naming conflicts by providing a unique namespace
3. Packages support modularity by allowing developers to separate the program
4. Organizing classes into packages makes it easier to locate related classes

Object Oriented Programming Concepts

What are access modifies in java?

ava uses public, protected, default (no modifier), and private to control access to classes, methods, and fields, ensuring appropriate visibility and encapsulation.

Can you provide examples of when to use each type of access modifier?

1. **Public:** Used when members should be accessible from any other class.
2. **Protected:** Ideal for members that should be accessible to subclasses and classes within the same package.
3. **Default:** Use when members should be accessible only within the same package.
4. **Private:** Best for members intended only for use within their own class.

Why do we use getters setter when we can make fields publick and setting getting directly?

Using getters and setters instead of public variables allows us to control how values are set and accessed, add validation, and keep the ability to change how data is stored without affecting other parts of your program.

Can a top-level class be private or protected in Java?

No, a top-level class cannot be private or protected because it restricts access, making it unusable from any other classes, contrary to the purpose of a top-level class.

Explain the concepts of classes and objects in Java.

Classes are blueprints for objects in Java, defining the state and behavior that the objects of the class can have. Objects are instances of classes, representing entities with states and behaviors defined by their class.

What are the ways to create an object?

1. Using the new Keyword, example: MyClass object = new MyClass();

2. Using Class Factory Methods, example: Calendar calendar = Calendar.getInstance();
3. Using the clone()

Can a class in Java be without any methods or fields?

Yes, a class in Java can be declared without any methods or fields. Such a class can still be used to create objects, although these objects would have no specific behavior or state

What are the methods available in the Object class, and how are they used?

The key methods are equals(), hashCode(), toString(), clone(), finalize(), wait(), notify(), and notifyAll(). These provide basic operations like equality checks, memory management, and thread coordination.

What are anonymous classes and their advantages?

Anonymous classes in Java are classes without a name, defined and instantiated in one place. They are useful when you need to create a subclass or implement an interface for a one-time use. The advantages include reduced boilerplate code, encapsulation of specific functionality, and the ability to override methods on the fly. This results in more compact and localized code, particularly in scenarios like event handling or passing behavior as an argument.

What is Singleton Class?

A singleton class in Java is a special class that can have only one instance (or object) at any time. It's like having only one key of the room. This is useful when we want to make sure there's just one shared resource, like a configuration setting or a connection to a database.

How can we create this singleton class?

In order to make singleton class, first we have a to make a constructor as private, next we have to create a private static instance of the class and finally we have to provide static method instance so that's how we can create the singleton class

Are these threads safe?

Singleton classes are not thread-safe by default. If multiple threads try to create an instance at the same time, it could result in multiple instances. To prevent this, we can synchronize the method that creates the instance or use a static initializer

What is a constructor in Java?

A constructor in Java is a special method used to initialize new objects. It has the same name as the class and may take arguments to set initial values for the object's attributes.

Can we use a private constructor?

Yes, we can use private constructors in Java. They are mostly used in classes that provide static methods or contain only static fields. A common use is in the Singleton design pattern, where the goal is to limit the class to only one object.

Can constructor be overloaded?
Yes, you can have multiple constructors in a Java class, each with a different set of parameters. This lets you create objects in various ways depending on what information you have at the time.

What is immutability mean in Java?

Immutability in Java means that once an object's state is created, it cannot be changed.

Why immutable objects are useful for concorrent programming?
These are useful in concurrent programming because they can be shared between threads without needing synchronization.

What are immutable classes?
Immutable classes in Java are classes whose objects cannot be modified after they are created. This means all their fields are final and set only once, typically through the constructor.

How can we create immutable class?

1. Declare the class as final so it can't be extended.
2. Make all of the fields final and private so that direct access is not allowed.
3. Don't provide setter methods for variables
4. Initialize all fields using a constructor method

What does Java's inheritance mean?

Inheritance in Java means a class can use the features of another class. This helps to reuse code and make things simpler.

Can a class extends on its own?
No, a class in Java cannot extend itself. If it tries, it will cause an error

Why multiple inheritance is not possible in java?
Java avoids using multiple inheritance because it can make things complicated, such as when two parent classes have methods that conflict.

What is the difference between inheritance and composition?
Inheritance is when one class gets its features from another class. Composition is when a class is made using parts from other classes, which can be more flexible.

Discuss the principle of "composition over inheritance". Provide an example where this principle should be applied in Java application design.
"Composition over inheritance" means using objects within other objects (composition) instead of inheriting from a parent class. It's applied when classes

have a "has-a" relationship. For example, a Car class can have an Engine class as a field rather than inheriting from an Engine.

What is the difference between association, aggregation, and composition in Java?
Association is a general relationship between two classes. Aggregation is a weak association (has-a) where the child can exist independently of the parent. Composition is a strong association where the child cannot exist without the parent.

Explain the IS-A (inheritance) and Has-A (composition) relationships in Java.
IS-A refers to inheritance, where a subclass is a type of the superclass. Has-A refers to composition, where a class contains references to other classes as fields.

What does mean by polymorphism in Java?
Polymorphism in Java means that the same piece of code can do different things depending on what kind of object it's dealing with. For example, if you have a method called "draw," it might make a circle for a Circle object and a square for a Square object.

How does method overloading relate to polymorphism?
Method overloading is using the same method name with different inputs in the same class. It's a simple way to use polymorphism when you're writing your code.

What is dynamic method dispatch in Java?
Dynamic method dispatch is a way Java decides which method to use at runtime when methods are overridden in subclasses. It ensures the correct method is used based on the type of object.

Can constructors be polymorphic?
No, constructors cannot be polymorphic. We can have many constructors in a class with different inputs, but they don’t behave differently based on the object type like methods do.

What does mean by abstraction in java?
Abstraction in Java means focusing on what needs to be done, not how to do it. You create a kind of blueprint that tells other parts of the program what actions they can perform without explaining the details.

Can you provide examples of where abstraction is effectively used in Java libraries?
Java uses abstraction in its collection tools. For example, when you use a List, you don't need to know how it stores data, whether as an ArrayList or a LinkedList.

What happens if a class includes an abstract method?
A class with an abstract method must itself be abstract. We can't create objects directly from an abstract class; it's meant to be a blueprint for other classes.

How does abstraction help in achieving loose coupling in software applications?

Abstraction lets us hide complex details and only show what's necessary. This makes it easier to change parts of your program without affecting others, keeping different parts independent and easier to manage.

What is interface in Java?

interface is like a blueprint for a class. It defines a set of methods that the class must implement, without specifying how these methods should work

What is the difference between an interface and an abstract class in Java?

abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction. Abstract class can **have abstract and non-abstract** methods whereas Interface can have **only abstract** methods. (Since Java 8, it can have **default and static methods** also.)

Can you provide examples of when to use an interface versus when to extend a class?

Use an interface when we want to list the methods a class should have, without detailing how they work. Use class extension when we want a new class to inherit features and behaviors from an existing class and possibly modify them.

How do you use multiple inheritance in Java using interfaces?

In Java, we can't inherit features from multiple classes directly, but we can use interfaces for a similar effect. A class can follow the guidelines of many interfaces at once, which lets it combine many sets of capabilities.

Can an interface in Java contain static methods, and if so, how can they be used?

Yes, interfaces in Java can have static methods, which you can use without creating an instance of the class.

When would you use an interface, and when would you use an abstract class?

Use an interface when you need multiple classes to share a contract without implementation. Use an abstract class when you need shared behavior (method implementations) along with method declarations.

Explain the difference between Comparable and Comparator interfaces. When would you use one over the other?

Comparable is used for natural ordering and is implemented by the class itself, while Comparator is used for custom ordering and can be implemented externally. Use Comparable when objects have a single logical ordering; use Comparator when you need multiple ways to order objects.

What is a static method in an Interface, and how is it different from a default method in an interface?

A static method in an interface belongs to the interface itself and cannot be overridden. A default method provides a default implementation for classes that implement the interface, and it can be overridden.

What is the diamond problem in Java and how does Java address it?
The diamond problem occurs in multiple inheritance where a class inherits from two classes with a common ancestor. Java resolves this by not allowing multiple inheritance with classes, but interfaces can use default methods to avoid this issue.

How does the concept of default methods in interfaces help resolve the diamond problem?
Default methods allow interfaces to provide method implementations, and in case of conflicts (multiple interfaces with the same default method), the implementing class must override the method, resolving ambiguity.

What does mean by encapsulation in java?
Encapsulation in Java is like putting important information into a safe. We store data and the methods inside a class, and we control who can access or change the data by using specific methods.

How Encapsulation Enhances Software Security and Integrity:
Encapsulation keeps important data hidden and safe. It only lets certain parts of our program use this data, which helps prevent mistakes and keeps the data secure from unwanted changes.

What is the concept of Serialization in Java?
Serialization is the process of converting an object into a byte stream for storage or transmission. It allows objects to be saved and restored later or transferred over a network.

What is the purpose of the serialVersionUID in Java serialization?
The serialVersionUID is a unique identifier for Serializable classes. It ensures that the serialized and deserialized objects are compatible by checking version consistency. If the serialVersionUID of the class doesn't match during deserialization, an InvalidClassException is thrown, preventing incompatible class versions from being used.

What happens if the serialVersionUID of a class changes during deserialization?
If the serialVersionUID changes between serialization and deserialization, the JVM considers the class as incompatible with the serialized object. This results in an InvalidClassException, as the runtime expects the version of the serialized class to match with the version defined in the deserialized class.

How can you prevent certain fields from being serialized in Java?
You can prevent specific fields from being serialized by marking them with the transient keyword. When a field is declared as transient, it is excluded from the serialization process, meaning its value will not be saved when the object is serialized.

Can a class be serialized if one of its member fields is not serializable?

A class can still be serialized even if one of its member fields is not serializable. However, you must mark the non-serializable field as transient. If the field is not transient and is not serializable, attempting to serialize the object will result in a NotSerializableException.

What is the difference between writeObject() and readObject() methods in Java serialization?

The writeObject() and readObject() methods allow customization of the serialization and deserialization processes. writeObject() is used to customize how an object is serialized, while readObject() customizes how it is deserialized. These methods can be overridden to handle complex scenarios, such as serializing transient fields or managing class versioning.

Is it possible to serialize static fields in Java? Why or why not?

No, static fields are not serialized in Java because they belong to the class, not to individual instances. Serialization is intended to capture the state of an object, and static fields are part of the class's state, not the object's state.

How do you serialize an object with circular references in Java?

Java handles circular references during serialization by keeping track of references that have already been serialized. When the same object reference appears again, Java writes a reference to the already serialized object rather than serializing it again. This prevents infinite recursion and maintains the object graph structure.

What is method overloading in Java?

Polymorphism in Java means that the same piece of code can do different things depending on what kind of object it's dealing with. For example, if you have a method called "draw," it might make a circle for a Circle object and a square for a Square object.

How does the Java compiler determine which overloaded method to call?

When we call an overloaded method, the Java compiler looks at the number and type of arguments you've provided and picks the method that matches these arguments best.

Is it possible to overload methods that differ only by their return type in Java?

In Java, we cannot overload methods just by changing their return type. The methods must differ by their parameters for overloading to be valid.

What are the rules for method overloading in Java?

The parameters must differ in how many there are, what type they are, or the order they are in.

What is method overriding in Java?

To override a method, the new method in the subclass must have the same name, return type, and parameters as the method in the parent class. Also, the new method should not be less accessible than the original.

What are the rules and conditions for method overriding in Java?
In Java, method overriding occurs when a subclass has a method with the same name, return type, and parameters as one in its parent class. The method in the subclass replaces the one in the parent class when called.

How does the @Override annotation influence method overriding?
The @Override annotation tells the compiler that the method is supposed to replace one from its superclass. It's useful because it helps find mistakes if the method does not actually override an existing method from the parent class.

What happens if a superclass method is overridden by more than one subclass in Java?
If different subclasses override the same method from a superclass, each subclass will have its own version of that method.

What is 'this' and 'super' keyword in java?

'this' is used to refer current class's instance as well as static members.
'super' keyword is used to access methods of the parent class.

Can 'this' keyword be assigned a new value in Java?
No, this keyword cannot be assigned a new value in Java. It is a read-only reference that always points to the current object.

What happens if you attempt to use the "super" keyword in a class that doesn't have a superclass?
If we attempt to use the "super" keyword in a class that doesn't have a superclass, a compilation error occurs. The "super" keyword is only applicable within subclasses to refer to members of the superclass.

Can the this or super keyword be used in a static method?
No, the this and super keyword cannot be used in static methods. Static methods belong to the class, not instances, and super refers to the superclass's object context, which does not exist in a static context.

How does 'super' play a role in polymorphism ?
In Java, the super keyword lets a subclass use methods from its parent class, helping it behave in different ways and that is nothing but a polymorphic behavior

What is the static keyword in Java?

The static keyword in Java is used to indicate that a particular member (variable or method) belongs to the class, rather than any instance of the class. This means that the static member can be accessed without creating an instance of the class.

Can a static block throw an exception?
Yes, a static block can throw an exception, but if it does, the exception must be handled within the block itself or declared using a throws clause in the class.

Can we override static methods in Java?
No, static methods cannot be overridden in Java because method overriding is based on dynamic binding at runtime and static methods are bound at compile time.

Is it possible to access non-static members from within a static method?
No, it's not possible to access non-static members (instance variables or methods) directly from within a static method. This is because static methods belong to the class itself, not to any specific instance. To access non-static members, you need to create an instance of the class and use that object to reference the non-static members.

What is static block?
To initialize static variables, the statements inside static block are executed only once, when the class is loaded in the memory.

Can we print something on console without main method in java?
Prior to Java 8, yes, we can print something without main method buts its not possible from java 8 onwards

What is final keyword in java?
the 'final' keyword is used to declare constants, making variables unchangeable once assigned, or to prevent method overriding or class inheritance

What are some common use cases for using final variables in Java programming?
Common use cases for using final variables in Java programming include defining constants, parameters passed to methods, and local variables in lambdas or anonymous inner classes.

How does the "final" keyword contribute to immutability and thread safety in Java?
The "final" keyword contributes to immutability and thread safety in Java by ensuring that the value of a variable cannot be changed once assigned, preventing unintended modifications and potential concurrency issues.

Can you describe any performance considerations related to using final?
The final keyword improves the performance by reducing call overhead?

What is functional interfaces?

Functional interfaces in Java are interfaces with just one abstract method. They are used to create lambda expressions and instances of these interfaces can be created with lambdas, method references, or constructor references.

Can functional interface extend another interface?
No, as functional interface allows to have only single abstract method. However functional interface can inherit another interface if it contains only static and default methods in it

Advantages of using a functional interface.
Functional interfaces, which contain only one abstract method, are key to enabling functional programming in Java. They offer concise and readable code through lambda expressions and method references, improving code simplicity. Functional interfaces allow easy parallel processing, better abstraction, and reusability, especially in scenarios like streams and event handling, promoting a cleaner and more expressive programming style.

Java 8 Basics

Can you tell me some new features that were introduced in Java 8?
Lambda Expressions, Stream API, Method References , Default Methods , Optional Class, New Date-Time API are the new features that were introduced in java 8

Why optional class, lambda expressions and stream API were introduced in java 8?
Optional class was introduced in Java 8 as a way to address the problem of null references
Lambda expressions were introduced in Java 8 to make it easier to write code for interfaces that have only one method, using a simpler and more direct style.
The Stream API was introduced in Java 8 to help developers process collections of data in a more straightforward and efficient way, especially for bulk operations like filtering or sorting.

Difference between filter and map function of stream API?
filter() eliminates elements of collection where the condition is not satisfied whereas map() is used to perform operation on all elements hence, it returns all elements of collection

Can you tell me some new features that were introduced in Java 11?
HTTP Client, Epsilon Garbage Collector, Z Garbage Collector, Local-Variable Syntax for Lambda Parameters are some of the new features and along with these new features, isBlank(), strip(), stripLeading(), stripTrailing(), and repeat() were also introduced for strings

Can you tell me some new features that were introduced in Java 17?
Sealed Classes, Pattern Matching for switch, Foreign Function and Memory API are some of the examples

Can you tell me some new features that were introduced in Java 21?
Virtual Threads, Structured Concurrency, Scoped Values, Sequenced Collections, Record Pattern are some of the examples

Which is faster, traditional for loop or Streams?
Traditional for loops are generally faster due to less overhead, but Streams provide better readability and are optimized for parallel processing in large datasets.

In which scenarios would you prefer traditional for loops and streams?
Use traditional loops for simple, small datasets requiring maximum performance. Use Streams for more complex data transformations or when working with large datasets where readability, maintainability, and potential parallelism are prioritized.

Explain intermediate and terminal operations in streams.
Intermediate operations (e.g., filter(), map()) return another stream and are lazy (executed only when a terminal operation is called). Terminal operations (e.g., forEach(), collect()) trigger the actual processing of the stream and produce a result or side effect.

Differences in Interface from Java 7 to Java 8.
In Java 7, interfaces could only have abstract methods. Java 8 introduced default and static methods, allowing interfaces to have method implementations.

Use of String.join(….) in Java 8?
String.join() concatenates a sequence of strings with a specified delimiter, simplifying string joining operations.

Collection Framework

What is collection framework in java?
The Java Collection Framework is a set of tools that helps us organize, store, and manage groups of data easily. It includes various types of collections like lists, sets, and maps.

What are the main interfaces of the Java Collection Framework?
The main parts of the Java Collection Framework are interfaces like Collection, List, Set, Queue, and Map. Each one helps manage data in different ways.

Can you explain how Iterator works within the Java Collection Framework?
An Iterator is a tool in the Collection Framework that lets you go through a collection's elements one by one.

What are some common methods available in all Collection types?
Some common methods all collection types have are add, remove, clear, size, and isEmpty. These methods let us add and remove items, check the size, and see if the collection is empty.

How does Java Collection Framework handle concurrency?
The Collection Framework deals with multiple threads using special collection classes like ConcurrentHashMap and CopyOnWriteArrayList, which let different parts of our program modify the collection at the same time safely.

How do you choose the right collection type for a specific problem?
To pick the right collection type, think about what we need: List if you want an ordered collection that can include duplicates, Set if you need unique elements, Queue for processing elements in order, and Map for storing pairs of keys and values.

What enhancements were made to the Java Collection Framework in Java 8?
Java 8 made improvements to the Collection Framework by adding Streams, which make it easier to handle collections in bulk, and lambda expressions, which simplify writing code for operations on collections.

What is the difference between Iterator and listIterator?
Iterator allows forward traversal of a collection, while ListIterator extends Iterator functionality to allow bidirectional traversal of lists and also supports element modification.

Name of algorithm used by Arrays.sort(..) and Collections.sort(..)?
Arrays.sort() uses a Dual-Pivot Quicksort algorithm for primitive types and TimSort for object arrays. Collections.sort() uses TimSort, a hybrid sorting algorithm combining merge sort and insertion sort.

How do you store elements in a set to preserve insertion order?
Use a LinkedHashSet, which preserves the insertion order of elements.

How do you store elements in a way that they are sorted?
Use a TreeSet or a TreeMap, which automatically sorts elements based on their natural ordering or a specified comparator.

Whats the use case of arrayList, linkedList and Hashset?
We use arrayList where we need efficient random access to elements via indices, like retrieving elements frequently from a list without altering it.

We use LinkedList where you frequently add and remove elements from the beginning or middle of the list, such as implementing queues or stacks.

We use HashSet where we need to ensure that there are no duplicates and we require fast lookups, additions, and deletions. It is ideal for scenarios like checking membership existence, such as in a set of unique items or keys.

How does a HashSet ensure that there are no duplicates?

A HashSet in Java uses a HashMap under the hood. Each element you add is treated as a key in this HashMap. Since keys in a HashMap are unique, HashSet automatically prevents any duplicate entries.

Can you describe how hashCode() and equals() work together in a collection
hashCode() determines which bucket an object goes into, while equals() checks equality between objects in the same bucket to handle collisions, ensuring that each key is unique.

Why is it important to override the hashCode method when you override equals? What would be the consequence if we don't?
Overriding hashCode() is crucial because hash-based collections like HashMap and HashSet use the hashcode to locate objects. Without consistent hashCode() and equals(), objects may not be found or stored correctly.

Can you give an example where a TreeSet is more appropriate than HashSet?
A TreeSet is more appropriate than a HashSet when you need to maintain the elements in a sorted order. For example, if we are managing a list of customer names that must be displayed alphabetically, using a TreeSet would be ideal.

What is the internal implementation of ArrayList and LinkedList?
ArrayList is backed by a dynamic array, which provides O(1) access time but requires resizing. LinkedList is implemented as a doubly-linked list, providing O(1) insertion and deletion at both ends but O(n) access time.

Can you explain internal working of HashMap in Java?

A HashMap in Java stores key-value pairs in an array where each element is a bucket. It uses a hash function to determine which bucket a key should go into for efficient data retrieval. If two keys end up in the same bucket, a Collison happened then the HashMap manages these collisions by maintaining a linked list or a balanced tree depend upon the java version in each bucket.

What happens when two keys have the same hash code? How would you handle this scenario?
When two different Java objects have the same hashcode, it's called a hash collision. In this case, Java handles it by storing both objects in the same bucket in a hash-based collection, like a HashMap. It then compares the objects using the equals() method to differentiate them.

How does a HashMap handle collisions in Java?
In Java, when a HashMap encounters a collision (two keys with the same hashcode), it stores both entries in the same bucket. Prior to Java 8, it linked them in a simple list structure. In Java 8, if the number of entries in a bucket grows large, the list is converted to a balanced tree for faster lookups.

Can you please tell me what changes were done for the HashMap in Java 8 because before java 8 hashMap behaved differently ?
Before Java 8, HashMap dealt with collisions by using a simple linked list. Starting from Java 8, when too many items end up in the same bucket, the list turns into a balanced tree, which helps speed up searching.

Can we include class as a key in hashmap?
No, as functional interface allows to have only single abstract method. However functional interface can inherit another interface if it contains only static and default methods in it

Can you please explain ConcurrentHashMap
ConcurrentHashMap is a version of HashMap that's safe to use by many threads at once without needing to lock the entire map. It divides the map into parts that can be locked separately, allowing better performance.

How does it(ConcurrentHashMap) improve performance in a multi-threaded environment?
ConcurrentHashMap boosts performance in multi-threaded settings by letting different threads access and modify different parts of the map simultaneously, reducing waiting times and improving efficiency.

What is time complexities insertions, deletion and retrieval of hashSet and HashMap?

1. **Insertion**:
2. Average: O(1)
3. Worst case: O(n) when rehashing occurs
4. **Deletion**:
5. Average: O(1)
6. Worst case: O(n) when rehashing occurs
7. **Retrieval**:
8. Average: O(1)
9. Worst case: O(n) when rehashing occurs (due to hash collisions)

NOTE: HashSet and HashMap are not internally sorted

What is time complexities insertions, deletion and retrieval of TreeSet and TreeMap?
O(log n) for operations like insertions, deletion and retrieval
NOTE: HashSet and HashMap are not internally sorted

What techniques did hashMap, treeMap, hashSet and TreeSet uses internally for performing operations?
HashMap uses an array of nodes, where each node is a linked list or Tree depend upon the collisions and java versions (From Java 8 onwards, if there is high hash collisons then linkedList gets converted to Balanced Tree).

TreeMap uses a Red-Black tree, which is a type of self-balancing binary search tree. Each node in the Red-Black tree stores a key-value pair.
HashSet internally uses a HashMap whereas **TreeSet** internally uses TreeMap

Design Patterns and Principles Basics

What is a design pattern in Java and why do we use this?
Design patterns are proven solutions for common software design problems. They provide standardized approaches to organize code in a way that is maintainable, scalable, and understandable.

Can you list and explain a few common design patterns used in Java programming?
Common design patterns in Java:

1. **Singleton:** Ensures a class has only one instance, with a global access point.
2. **Observer:** Allows objects to notify others about changes in their state.
3. **Factory Method:** Delegates the creation of objects to subclasses, promoting flexibility.

How can design patterns affect the performance of a Java application?
Design patterns can impact performance by adding complexity, but they improve system architecture and maintainability. The long-term benefits often outweigh the initial performance cost.

Which design pattern would you use to manage database connections efficiently in a Java application?
The **Singleton** pattern is commonly used to manage database connections, ensuring a single shared connection instance is reused efficiently.

How do you choose the appropriate design pattern for a particular problem in Java?
Understand the problem fully, identify similar problems solved by design patterns, and consider the implications of each pattern on the application's design and performance.

Difference between HashMap and TreeMap.
HashMap stores key-value pairs without ordering, while TreeMap sorts the entries by keys. TreeMap has O(log n) operations due to its tree structure, whereas HashMap has O(1) operations under ideal conditions.

In what scenarios would you prefer to use a TreeMap over a HashMap?
Use a TreeMap when you need to maintain a sorted order of keys, such as when iterating over sorted data. A HashMap is preferable for fast lookups without concern for ordering.

Can we add objects as a key in TreeMap?

Yes, objects can be used as keys in a TreeMap if they implement the Comparable interface or a Comparator is provided for sorting the keys.

What are SOLID Principles?

'S' stands for Single Responsibility Principle: It means a class should only have one reason to change, meaning it should handle just one part of the functionality.
For Example: A class VehicleRegistration should only handle vehicle registration details. If it also takes care of vehicle insurance, then it will violates this.
'O' stands for Open/Closed Principle: It means Classes should be open for extension but closed for modification.
For Example: We have a VehicleService class that provides maintenance services. Later, we need to add a new service type for electric vehicles and if without modifying VehicleService, we are able to extend it from a subclass ElectricVehicleService then it follows this priciple.
'L' stands for Liskov Substitution Principle: It means Objects of a superclass should be replaceable with objects of its subclasses without affecting the program's correctness.
For Example: If we have a superclass Vehicle with a method startEngine(), and subclasses like Car and ElectricCar, we should be able to replace Vehicle with Car or ElectricCar in our system without any functionality breaking. If ElectricCar can't implement startEngine() because it doesn't have a traditional engine, it should still work with the interface to not break the system.
'I' for Interface Segregation Principle: It means do not force any client to depend on methods it does not use; split large interfaces into smaller ones.
For Example: Instead of one large interface VehicleOperations with methods like drive, refuel, charge, and navigate, split it into focused interfaces like Drivable, Refuelable, and Navigable. An ElectricCar wouldn't need to implement Refuelable, just Chargeable and Navigable.
'D' stands for Dependency Inversion Principle: It means High-level modules should not depend directly on low-level modules but should communicate through abstractions like interfaces.
For Example: If a VehicleTracker class needs to log vehicle positions, it shouldn't depend directly on a specific GPS device model. Instead, it should interact through a GPSDevice interface, allowing any GPS device that implements this interface to be used without changing the VehicleTracker class.

Concurrency and multi-threading

What is a thread in Java and how can we create it?
A thread in Java is a pathway of execution within a program. You can create a thread by extending the Thread class or implementing the Runnable interface.

Can you explain the lifecycle of a Java thread?
A Java thread lifecycle includes states: New, Runnable, Blocked, Waiting, Timed Waiting, and Terminated.

How would you handle a scenario where two threads need to update the same data structure?
Use synchronized blocks or methods to ensure that only one thread can access the data structure at a time, preventing concurrent modification issues.

Can we strat thread twice?
No, a thread in Java cannot be started more than once. Attempting to restart a thread that has already run will throw an IllegalThreadStateException.

What is the difference between Thread class and Runnable interface in Java?
The Thread class defines a thread of execution, whereas the Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.

How can you ensure a method is thread-safe in Java?
To ensure thread safety, use synchronization mechanisms like synchronized blocks, volatile variables, or concurrent data structures.

What are volatile variables?
Volatile variables in Java are used to indicate that a variable's value will be modified by different threads, ensuring that the value read is always the latest written.

What is thread synchronization and why is it important?
Thread synchronization controls the access of multiple threads to shared resources to prevent data inconsistency and ensure thread safety.

Can you describe a scenario where you would use wait() and notify() methods in thread communication?
Use wait() and notify() for inter-thread communication, like when one thread needs to wait for another to complete a task before proceeding.

What challenges might you face with multithreaded programs in Java?
In Java, multithreaded programming can lead to issues like deadlocks, race conditions, and resource contention, which complicate debugging and affect performance. Managing thread safety and synchronization efficiently is also a significant challenge.

What is Java memory model and how it is linked to threads?
The Java Memory Model (JMM) defines the rules by which Java programs achieve consistency when reading and writing variables across multiple threads, ensuring all threads have a consistent view of memory.

Miscellaneous questions (Not too much important)

GenZ Career on YouTube
Subscribe for Interview Preparation

what is transient?
The transient keyword in Java is used to indicate that a field should not be serialized. This means it will be ignored when objects are serialized and deserialized.

Can we create a server in java application without creating spring or any other framework?
Yes, you can create a server in a Java application using only Java SE APIs, such as by utilizing the ServerSocket class for a simple TCP server or the HttpServer class for HTTP services.

What is exchanger class
The Exchanger class in Java is a synchronization point at which threads can pair and swap elements within pairs. Each thread presents some object on exchange and receives another object in return from another thread.

What is reflection in java?
Reflection in Java is a capability to inspect and modify the runtime behavior of applications. It allows programs to manipulate internal properties of classes, methods, interfaces, and dynamically call them at runtime.

What is the weak reference and soft reference in java?
Weak references in Java are garbage collected when no strong references exist. Soft references are only cleared at the discretion of the garbage collector, typically when memory is low.

What is Java Flight Recorder?
Java Flight Recorder (JFR) is a tool for collecting diagnostic and profiling data about a running Java application without significant performance overhead.

Discuss Java Generics.
Generics provide type safety by allowing classes and methods to operate on objects of specific types, preventing runtime ClassCastException and reducing code duplication

What is the difference between Young Generation and Old Generation memory spaces?
The Young Generation stores newly created objects. The Old Generationholds objects that have survived several garbage collection cycles in the Young Generation