TechGlobal

Java Interview Questions

What is JDK, JRE, JVM and JIT (basically talk about the Java architecture)?

- **JDK** stands for Java Development Kit and is used to create Java programs with help of its component like developments tools, libraries, JRE, JVM and JIT
- **JRE** stands for Java Runtime Environment and is a component of Java that comes with JDK. It holds all the Java libraries like lang, io, util and java classes. It has JVM inside too which runs the code
- **JVM** stands for Java Virtual Machine and used to run the programs. JVM makes java platform independent, which means, any machine with JVM can run Java programs but JVM itself is platform dependent
- **JIT** stands for Just in Time Compiler and it stands inside the JVM. It is used to increase the speed of execution

JDK = development tools + JRE + JVM
JRE = all the java libraries + java classes + JVM
JVM = Java Virtual Machine and it holds JIT as well

Why java is platform-independent?

- Java is platform independent as it has JVM. There is JVM available for every operating system and JVM can read byte code. So, different JVM is designed for different operating systems. That is why JVM is platform dependent, but it makes java platform independent.

What is source code, byte code and binary code and what are their relation in Java?

- Source code is the code written by using any programming language
- Byte code is the compiled Java code (Java source code)
- Binary code is the computer understanding code which is basically ones and zeros

  - Source code -> compiled by javac -> Byte Code -> JVM -> Binary Code
  - javac compiles .java file and returns .class file
  - .class file has the Byte Code

EX/ MyFirstProgram.java (Source code)-> javac -> MyFirstProgram.class (Byte code) -> java -> Result of the program on my screen

What is java and javac and how they work with each other?

- javac stands for java compiler and it used to compile source code
- java is used to run the compiled code by javac

- In java the flow of the execution will be as below

source code (actual code that we write) -> compiled by javac -> Byte Code -> JVM -> Binary Code

Explain main() method in java?

- It is one of the most important methods in java as it tells java where the programs start
- We cannot run our programs without main method and all our programs should be in main() method
- JVM always look for this method to start running an application

More to say about main() method

- main() method is public which makes it to be used in any class
- main() method is static which makes it be called without creating an object
- main() method is void as it does not return anything

What kind of data types we have in Java?

- We have primitive and non-primitive data types in Java
- Non-primitives are also known as reference types (Example: String)
- Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot

Explain primitive data types in Java?

- Java has 8 primitive data types to store data
- Those are simply boolean, byte, char, short, int, long, float, double
- byte, short, int and long are used for numeric data
- float and double can be used for numeric data but they have floating points
- char to store every single character
- boolean can store either true or false

What is variable declaration and assignment in Java?

- Variable in java is kind of a special container that stores data
- Variables are labels that describe a particular location in memory and associate it with its data type
- We can simply say variable is reserved area allocated in the memory

What are identifiers in Java?

- All java variables must be identified with unique names and these unique names are called as identifiers
- Identifiers can be short names like a, b or c or more descriptive names like age, balance, name
- NOTE: it is highly recommended to use more descriptive names in order to create readable and maintainable code

What is String in Java?

- String is a Java class
- We can create String objects to store a sequence of characters
- String is an immutable data type and value cannot be changed once it is initialized
- String class has many useful methods to manipulate data based on our need like charAt(), indexOf() and so on…

What is String concatenation in Java and how can we concatenate Strings?

- String concatenation is forming a new String that is the combination of multiple Strings
- There are 2 ways of String concatenation
    1. we can concatenate Strings by using + (plus) operator
    2. we can use concat() method to concatenate Strings

What is method chaining?

- Method chaining means repeatedly calling one method after another on an object, in one statement

What are different types of operators in Java?

- Firstly, operators in Java are used to perform operations on variables and values and there are so many different operators in java
- Assignment operators are used to assign values into variables (=, += , -=, *=, /=, %=)
- Arithmetic operators are used to perform common mathematical operations (+, -, *, /, %)
- Increment and decrement operators are also arithmetic operators (++, --)
- Comparison (Relational) operators are used to compare two values (<, >, <=, >=, ==, !=)
- Logical operators are used to determine the logic between variables or values (&&, ||, !)

What are if-else statements in Java and what they are used for?

- Java if-else statements are used to test specific conditions and they control the flow of the program based on the condition.
- It will execute a block of code if given condition is true, and another code of block if condition is false

What is ternary operator in Java?

- Java ternary operator is kind of a short-hand if else and it can be used instead of simple if else statements
- It is helpful to replace multiple lines of code with a single line
- *Syntax ->* *variable = (condition) ? expressionTrue : expressionFalse;*

What is switch statement in Java?

- switch statement can be used to select one of many code blocks to be executed
- switch statement can be used with byte, short, int, char, String and Enums in Java
- Java allows to have any number of cases within a switch statement
- *How it works:*
    - The switch expression is evaluated once
    - The value of the expression is compared with the values of each case
    - If there is a match, the associated block of code is executed.
    - The break keyword is optional and used to break the switch expression
    - The default keyword is also optional and usually used at the end of switch statement. It is used to perform a last block of code if none of the cases matched. No break is needed in the default case

What is break statement in Java?

- break is used for termination in Java
- It is used with switch statements and loops

What is type casting in Java?

- Type casting is known as assigning value of one primitive data type into another primitive
- In Java, there are two types of casting: implicit and explicit casting
- Implicit casting is known as converting a smaller data type to a larger data type size and performed automatically by the compiler
- Explicit casting is known as converting a larger data type to a smaller data type size and performed manually by users

    - Widening casting or up-casting (automatically) - converting a smaller type to a larger type size
      the flow of widening: byte -> short -> char -> int -> long -> float -> double

    - Narrowing casting or down-casting (manually) - converting a larger type to a smaller size type
      the flow of narrowing: double -> float -> long -> int -> char -> short -> byte

    NOTE: boolean is the only primitive data type that cannot be cast to any other primitive

What are autoboxing and unboxing in Java?

- Autoboxing is known as storing primitive type into object/wrapper class
- Unboxing is known as storing object/wrapper class into primitive type

What are wrapper classes in Java?

- Wrapper classes are object representations of primitives
- There is a wrapper class for each primitive type
- Wrapper class names are usually same as with primitive data types

Wrapper class	Primitive type
Boolean	boolean
Byte	byte
Short	short
Integer	int
Long	long
Float	float
Double	double
Character	char

What is the purpose of having wrapper classes in Java?

- Wrapper class object can store all data that primitives store and can be used instead of primitives
- But wrapper classes are mostly used with their useful methods to manipulate the data

What are different loops in Java and the reasons to use loops?

- Loops in Java are used to execute a block of code repeatedly
- Loops will run until a termination condition is met
- Loops are helpful to eliminate duplicating codes and time saving
- There are 4 types of loops in Java
  1. for loop (traditional i loop) -> is used when start and end point is exact
  2. for each loop -> similar to fori loop but preferred with collections
  3. while loop -> is used when termination is based on a condition
  4. do while loop -> similar to while loop but preferred if block of code executes at least once
- for loop vs while loop
  a. For loop is used to execute a block of code and preferred when we know the exact number of iterations
  b. While loop is used to execute a block of based on a condition and total number of iterations is not certain (dynamic)
- Loop control statements
  a. break is used to terminate the loop
  b. continue is used to skip only current iteration if a specified condition occurs, and continues with the next iteration in the loop

What is Array and ArrayList and their differences in Java?

- Array is one of the collection types in java
- Collection is storing multiple similar items in one place
- By using an Array, we can store many similar items/elements in one variable
- Arrays in Java have fixed size and once the size of an array is declared, it's not possible to change it
- To handle this issue, we can use the ArrayList class. It allows us to create dynamic sized arrays
- Unlike Arrays, ArrayLists can automatically adjust its capacity when we add or remove elements from it and hence, ArrayLists are also known as dynamic arrays
- Array members are accessed using [index] while ArrayList has a set of methods to access elements and modify them like get(), add(), set() and so on…
- Array can store both primitive data types as well as objects while ArrayList only supports object entries, not the primitive data types

What are differences between ArrayList and LinkedList?

- LinkedList is very similar to ArrayList and can be used for the same purposes
- ArrayList stores the value in a single position while LinkedList stores 3 values (previous address, data, and next address) in a single position
- Whenever an element is added or removed to an ArrayList in the middle, all elements after that position are shifted but Whenever an element is added or removed to a LinkedList, only the address of previous and next elements are changed
- LinkedList allocates more memory compared to ArrayList but more proper to use if we will add or remove data frequently but ArrayList allocates less memory and preferred when there is not too much data change
- ArrayList is slower compared to LinkedList when we add or remove new data

What is Class and Object in Java?

- Class in Java is a blueprint or template from which objects are created
- We can think of the class as a sketch (prototype) of a car. It contains all the details about the engine, doors, windows, etc. Based on these descriptions we build the car. Car is the object as well
- Since many cars can be made from the same description, we can create many objects from the car class
- Object is instance of a class
- Class is a template or blueprint from which objects are created.
- So, object is the instance(result) of a class
- EXAMPLE: if car is a class, then BMW or Audi can be an object of that class. BMW or Audi is known as instance of car class

What are the class members in Java?

- Instance variables – fields
  - They are used to define all the attributes of the objects of that class
  - Fields are used to store data
  - Fields can be accessed by objects of that class
  - ***For example:*** If car is a class, fields can be year, make, model, trim, color, etc.
  - NOTE: There could be final attributes of a class and those can be declared as final. Final attribute values cannot be changed anywhere in the project
  - NOTE: There could be static attributes of a class and this means, these static attributes belong to class itself and can be called with className. There is no need to create object to call static fields
- Methods – functions
  - A method is a block of code that performs a specific task
  - Methods are functions or behaviors of the class
  - There are void and return types of methods
  - returnType - It specifies what type of value a method returns. For example, if a method has an int return type then it returns an integer value. If, the method does not return a value, its return type is void
  - methodName - It is an identifier that is used to refer to the method in a program, could be any name but should be meaningful to increase readability of your code
  - method body - It includes the programming statements that are used to perform some tasks. The method body is enclosed inside the curly braces { }
  - method parameters – Arguments that we pass inside the method parentheses
  - A Java method may or may not return a value
  - We use the return statement to return any value for return type methods
  - Other methods are known as void methods and these methods do not return any value
  - Both void and return type methods can be final and static
  - NOTE: There could be final methods of a class and those can be declared as final. Final methods cannot be overriden
  - NOTE: There could be static methods of a class and this means, these static methods belong to class itself and can be called with className. There is no need to create object to call static methods

- Constructors
  - A constructor in Java is a **special method** that is used to initialize objects
  - Constructors have always same name as a ClassName they belong to
  - The constructor is called when an object of a class is created
  - It can be used to set initial values for object attributes
  - Object cannot get created without constructors
  - Constructors can be used only next to new keyword
  - Java will provide no argument constructor if we don't create one. That constructor is called ***Default Constructor***
  - Default constructor is no-argument. It doesn't take anything
  - Constructors can be overloaded
  - We can have multiple constructors. Overloading constructors follow the same rules as method overloading
  - ***Why do we create custom constructor?*** We may give initial values for every object that is getting created
- Blocks
  - A block in Java is a set of code enclosed within curly braces { } within any class, method, or constructor
  - It begins with an opening brace { and ends with a closing braces }
  - Between the opening and closing braces, we can write codes which may be a group of one or more statements
  - There are three types of blocks in Java. They are as follows:
  - Local block -> regular block used with methods, if else , switch, loops statements
  - Instance initialization block (Non-static initialization block) -> used to initialize non-static instance variables of a class
  - Static initialization block -> used to initialize static instance variables of a class

What is local variable and instance variable and their differences?

- ***Local variables:*** variables that are created within the method. Scope for local variables: it stays within the method
- ***Instance variables:*** variables that belongs to an object. The variables that are created right under the class. It can be used within the class
- When we have instance variable and local variables which have same variable name, java accepts local variable as a priority
- this keyword is a reference to the current object of its type
- this keyword is useful when we have instance variable and method parameter with same name and helps us to differentiate instance variables from local variables

What does null keyword mean in Java?

- **null** means an object reference is not pointing an object
- Primitives cannot be **null**
- If object variable holds null and programmer tries to execute some methods over null value, then **NullPointerException** will be thrown as a run-time error

What are the advantages of using methods in Java?

- The main advantage is **code reusability**. We can write a method once and use it multiple times. We do not have to rewrite the entire code each time
- Methods make code more **readable and easier** to debug

What is method overloading and method overriding and their differences?

- Method overloading
  - In Java, two or more methods may have the same name
  - This is allowed if they differ in parameters (different number of parameters, different types of parameters, or both).
  - These methods are called overloaded methods and this feature is called **method overloading**
  - **Note:** Multiple methods can have the same name if the number and/or type of parameters are different
- How to achieve method overloading
  - Method overloading is achieved by either:
    - changing the number of arguments
    - or changing the data type of arguments
- Method overriding
  - It is a concept of implementing a new body to an inherited method
  - It is common to use when we want to specify body in the child classes method. So, we override the parent class' methods in child
  - @Override annotation is used to override a parent class' method
- Rules of Method Overriding
  - Method signature should be same (parameters and name)
  - Return type should be same
  - Access modifier should be same or can be more accessible
  - @Override annotation is highly recommended to use in case of overriding but it is not mandatory
- Differences
  - **Method overriding** is implementing a new body to an inherited method from parent class
  - Overriding methods can happen only when a child class inheriting a parent class
  - Overriding is writing same method from parent class with a different implementation in the child class
  - **Method overloading** is creating a method which has same name with another method in the class
  - Overloading happens in the same class with different parameters
  - Overloaded methods are sharing same name but different signature
  - Method overloading is not related to inheritance

What is this and this() in Java?

- this is a reserved keyword in Java and used as reference to the current object of its type in a class
- this() is used to call a custom constructor in another custom constructor and provides code reusability
- this() must be the first statement in the constructor block it is called

What is super and super() in Java?

- super is a reserved keyword in Java and used as reference to the parent class object
- super() is used to call a custom constructor of parent class in to be used in child class constructor and provides code reusability
- super() must be the first statement in the constructor block it is called

What is constructor chaining?

- It is calling a custom constructor in another custom constructor

It can be achieved by using this() or super()

What are the differences between constructors and methods?

- Constructors cannot be return type while methods can be
- Constructor name should always be same as the class name while method name can be any identifier based on need
- Constructor is called automatically (implicitly) when a new object is created while methods can be called with class name or with an object
- Java compiler provides a default constructor if custom constructor is not created but methods may or may not be created based on need
- Both methods and constructor can be overloaded
- Constructor cannot be overridden but method can be overridden

What is mutable/immutable in Java?

- Immutable: it means once an object is created, value for the object cannot be changed like String
- Mutable: it means when an object is created, value for it can be changed like StringBuffer and StringBuilder

What is StringBuilder and StringBuffer and their differences?

- These are 2 Java classes located in java.lang package
- They can be considered as mutable type of String
- Both has many useful methods to manipulate data like append(), insert(), delete(), reverse() and so on…
- Both can be converted to String by using toString() method
- DIFFERENCE: StringBuffer is thread safe, that is why it is slower compared to StringBuilder and StringBuilder is faster compared to StringBuffer

What is Stack and Heap in Java?

- These terms are coming from Java Memory Management System
- Stack is where methods, local variables and object references are stored
- When you run a program, different layers will be created in stack and they will be executed in the order of program (Last in, first out)
- Heap is where objects (instances) are stored
- Accessing object is slower compared to accessing an instance variable

What is Garbage Collection?

- Garbage means unreferenced objects (objects that lost their references)
- Garbage Collection is a process of destroying objects that lost references
- Garbage Collection runs automatically (implicitly) for better memory management in Java
- NOTE: Garbage Collection applies only to reference types data (objects)

What does System.gc() or Runtime.getRunTime().gc() methods do in Java?

- In Java, Garbage Collection happens automatically (implicitly)
- However, it can be done explicitly as well by calling System.gc() method or Runtime.getRunTime().gc()

What is finalize() method and why it is used for?

- finalize() method can be used to run a block of code before object reference is garbage collected
- Purpose of using this method is to do proper clean up before removing the object
- By default, finalize method is empty. However, you can write your own finalize method to take certain action
- This method is located in Object class and can be overridden wherever it is needed

What are access modifiers in Java?

- Access modifiers are used to define the scope of accessibility for the members of the class
  - Members of a class = constructors, instance variables, methods, blocks
- There are 4 access modifiers -> public, protected, default and private

- *A class itself in a Java can only be public or default*
  - public class -> can be accessed by any other class in the project
  - default class -> can be accessed by classes in the same package
- More accessible to less accessible: public > protected > default > private
- We can give access class methods, fields and constructors at 3 access levels

  1. Only within the class:

  private keyword is used to make fields and methods to be accessible only within the class

  2. Only within the package

  protected and default keywords are used to make fields and methods to be accessible only within the same package

  HOWEVER, protected can be used outside of the package as well in case of inheritance

  3. Anywhere within the project

  public keyword is used to make fields and methods to be accessible anywhere in the project

- NOTE: Access modifiers cannot be applied to local variables

What is final keyword in Java and where to use it?

- final keyword: it is kind of a non-access modifier and do not control the access level for class members, but it provides some other functionalities
    - A field can be final -> final field must be initialized, and this value cannot be changed
    - A method can be final -> this method cannot be overridden (it can still be overloaded)
    - A class can be final -> this class cannot be extended
    - Constructors cannot be final

What is static keyword in Java and where to use it?

- static keyword: it is used define some class members to be called by class name
    - A field can be static -> static field means, it belongs to class can be called with className
    - A method can be static -> static method means, it belongs to class can be called with className
    - A block can be static -> static blocks are executed before everything else once an object of that class is created
    - A class cannot be static -> It is better to say outer class cannot be static (There can be static inner classes)
    - Constructors cannot be static

What are Object-Oriented Programming Principles?

- Java is an object-oriented programming
- The core concept of the object-oriented approach is to break complex problems into smaller objects. So, Object-oriented simply known as an approach to programming that breaks a programming problem into objects
- Everything in Java is associated with classes and objects
- Different Java objects can interact with each other
- There are 4 main principles of Object-Oriented Programming Languages
    - Encapsulation
    - Inheritance
    - Abstraction (Abstract Classes vs Interfaces)
    - Polymorphism
- Encapsulation is making sure that "sensitive" data is hidden from users
    - We need encapsulation to protect instance variables from being accessed directly
    - HOW ACHIEVE ENCAPSULATION?
        - Make instance variables private
        - Provide public setters and getters methods
    - Note: These public getters and setters' method will help us to access and update private fields
    - WHY WE NEED ENCAPSULATION?
        - Better control of class attributes and methods
        - Class attributes can be made read-only (if you only use the get method), or write-only (if you only use the set method)
        - Flexible: the programmer can change one part of the code without affecting other parts
        - Increased security of data

- Inheritance in Java is inheriting attributes and methods from one class to another. This will create "inheritance concept" into two classes
  - subclass (child) - the class that inherits from super (parent) class
  - superclass (parent) - the class being inherited from sub (child) class
  - To inherit from a class, use the extends keyword
  - WHAT CAN BE INHERITED FROM SUPER-PARENT-BASE CLASS?
    - Methods and variables of parent class can be inherited
      However,
    - Access modifiers decide which methods and variables will be inherited
    - private methods and variables cannot be inherited as they are private to the class
    - default methods and variables can only be inherited if parent and child classes are in the same package
    - protected and public members can be inherited by any class within the project
  - RULES OF INHERITANCE:
    - A child class can only 1 parent class
    - A parent class can be parent of multiple child classes
    - Making a class final means this class cannot be extended (inherited)
    - By default, each class will be extended to Object class
  - WHY WE NEED INHERITANCE?
    - It is useful for ***code reusability***: with the help of inheritance, we can reuse attributes and methods of a parent class when we create a new child class
    - In most of the case, most of the methods and attributes are already created in parent and we don't need to create those again in child classes
    - By this way, we will be using common code with parent while we will create only some specific methods and attributes for the child classes

- Abstraction is a process of hiding the implementation details and showing only functionality to the user
  - Abstraction can be achieved in 2 ways
    - abstract classes & Interfaces
  - NOTE: Abstract classes or Interfaces are created to be used by other classes
  - Abstract Class: A class which is declared as abstract is known as an abstract class
    - An abstract class must be declared with an abstract keyword
    - It can have both abstract, non-abstract methods and static methods
    - It cannot be instantiated (object cannot be created)
    - It can have constructors
    - It can have final methods which cannot be overridden
    - Abstract method: can only be used in an abstract class, and it does not have a body
    - The body is provided by the subclass (inherited from) by using @Override annotation
  - MORE TO KNOW ABOUT ABSTRACT CLASSES:
    - An abstract class can be inherited by many subclasses same as non-abstract classes
    - Any class that extends to an abstract class MUST implement body for the abstract methods
    - An abstract class can be inherited by another abstract subclass and in this case:
      - abstract methods may not be implemented in the subclass
      - BUT, any third class that extends to abstract subclass MUST implement all abstract methods in both abstract classes

- Interface: another way of achieving abstraction in Java and we can define additional features of objects by using interfaces
  - To access the interface methods, the interface must be "implemented" (like inherited) by another class with the implements keyword (instead of extends).
  - The body for the interface abstract methods is provided by the class that implements the interface
- RULES TO KNOW:
  - Like **abstract classes**, interfaces **cannot** be used to create objects
  - Interface methods do not have a body - the body is provided by the class that implements interface
  - On implementation of an interface, you must override all its methods
  - **Interface methods are by default abstract and public**
  - **Interface attributes are by default public, static and final**
  - An interface cannot have a constructor
- INTERFACE METHODS:
  - **Abstract methods:** Interfaces usually have ONLY abstract methods and they are public abstract by default
  - **Default methods:** Interfaces can have methods that has body by using default keyword
    NOTE: default keyword is not defining access modifier here!!!
  - **Static method:** Interfaces can have regular static methods that has body
    NOTE: These methods are always public as well and can be called with interface name
- MORE TO KNOW ABOUT INTERFACE:
  - Interfaces are created to define additional behavior of the object
  - One class can implement multiple interfaces
  - Interfaces can extend to another interface but cannot extend to a class
  - One Interface can extend to many Interfaces. In this case:
    - A class implementing the interface that extends many other should override all the abstract methods of all extended interfaces
  - Interfaces are usually named with <able> keyword. Rollable, Reliable, Scalable
- WHY WE NEED ABSTRACTION – ABSTRACT CLASSES AND INTERFACES?
  - Abstraction in Java can be achieved by abstract classes and interfaces
  - Abstract classes and interfaces are created to be used by many similar subclasses and provides some common fields and methods to those subclasses and increases code reusability
  - A class can extend to only one class in Java (or one abstract class)
  - In this case, interfaces can be used to provide multiple inheritance (biggest difference between abstract class and interface)

- Polymorphism is known as one object's being able get into many different shapes and it can be achieved only through multiple inheritance. (Parent-Child class concept, abstract classes and interfaces)
  - Polymorphism is also closely related to inheritance
  - When we create an object, reference of object can be all below:
    - 1. Itself
    - 2. Higher level super classes that the class is extending
    - 3. Interfaces that the class implementing

- Polymorphism is not useful when creating very few Parent and Child classes in Java
- It is meaningful when used with many parents and many child classes
- One of the most important purpose of Polymorphism is using Polymorphic arrays
- Polymorphic arrays: A collection which can store many different types in one array/collection
- Up-casting vs down-casting in polymorphism
  - Up-casting: when reference is type of parent, but object is child
  - Up-casting happens implicitly
  - Down-casting: when reference is type of child, but object is parent
  - Down-casting happens explicitly

What are the similarities and differences between Abstract classes and Interfaces?

- SIMILARITIES:
  - Both abstract class and interface are used to achieve abstraction where we can declare the abstract methods
  - Both abstract class and interface cannot be instantiated – meaning – we cannot create object from them

- DIFFERENCES:

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
7) An **abstract class** can be extended using keyword "extends".	An **interface** can be implemented using keyword "implements".
8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

What is Dynamic Polymorphism vs Static Polymorphism in Java?

- Dynamic Polymorphism - Runtime polymorphism -> it is method overriding
- Java doesn't know which version of method implementation gets executed until we run the code
- Static Polymorphism - Compile-time polymorphism -> it is method overloading
- Java knows exactly which method it will pick based on the parameter you passed for a method

What is IS-A and HAS-A relationship in Java?

- IS-A RELATIONSHIP is inheritance. Example/ Mercedes is a Car (Mercedes class extends to car class)
- HAS-A-RELATIONSHIP is one class' having reference to an instance of another class. Example/ Author has a Book (Book class instance is referenced by Author class)

What is Java Collections framework?

- Java Collections Framework is a set of pre-written classes and interfaces that helps us to organize and manipulate groups of objects
- By using Java Collections Framework, we can organize our objects in different ways like lists, sets, or maps
- The reason we need to use these types that Array is very useful, but it has some restriction
    - Array is fixed in terms of size (we cannot add elements or remove)
    - And Array does not have methods to manipulate data
- Java Collections Framework interfaces and classes comes from java.util package (Iterable is in java.lang only)
- Most important interfaces that are implemented by collection classes are List, Set and Map

What is List in Java?

- List is an interface that specifies the methods required to process an ordered list of objects
- There are 2 implementations provided for the List interface: ArrayList and LinkedList
- Both ArrayList and LinkedList implements List interface
- Both ArrayList and LinkedList allows us to store duplicated objects
- Both ArrayList and LinkedList allows us to have unlimited null elements
- LinkedList also implements Queue and Deque interfaces and this adds more methods to LinkedList for us to use to process or manipulate data like getFirst(), getLast(), poll(), peek()

What is Set in Java?

- Set is an interface that defines the methods required to process an unordered collection of objects in which there is no duplication
- HashSet implements Set interface and LinkedHashSet extends to HashSet
  LinkedHashSet extends HashSet | HashSet implements Set
- Both HashSet and LinkedHashSet eventually implements the Set interface, and they allow ONLY 1 null element
- HashSet is an unordered collection of the data set, whereas the LinkedHashSet is an ordered collection of HashSet. HashSet does not provide any method to maintain the insertion order, but LinkedHashSet maintains the insertion order of the elements
- We can use TreeSet to sort the elements, and TreeSet does not allow any null elements
- HashSet is faster compared to LinkedHashSet and TreeSet and allocates less memory as it does not store insertion point or sorting the elements

What is Map in Java?

- Map is an interface
- Map is a data structure that helps us to store multiple items in a single unit as KEY and VALUE
- It maps from key to value
- Keys are always unique in a map
    - a single key only appears once in the Map
    - a key can map to only one value
- Values do not have to be unique
- We can use put(key, value) method to add elements to the map and get(key) method to get element
- keySet() method returns all the keys as a Set (as they are all unique)
- values() method returns all the values as a Collection
- entrySet() method returns all key-value pairs
- containsKey() method is used to check whether a particular key is being mapped into the HashMap or not and returns a boolean
- There are 2 implementations provided for the Map interface: HashMap and HashTable
- Both HashMap and HashTable implements the Map interface
- LinkedHashMap extends to HashMap, and it maintains insertion order and TreeMap sorts the elements
- HashMap allows only 1 null key and multiple null values
- HashTable does not quarantee any kind of order and does not allow any null keys or values
- TreeMap does not allow any null key, while it allows multiple null values
- HashTable is synchronized and that is why thread safe while HashMap is not synchronized and not thread safe

What is Iterator in Java?

- An Iterator is an object that can be used to loop through collections, like ArrayList, HashSet, etc.
- We can use iterator to loop through all collections that extends to Iterable interface
- Iterator provides 3 useful methods to use
    - hasNext() -> returns true if there are more elements in the collection or false otherwise
    - next() -> gets next element from the collection
    - remove() -> removes the element from collection that is currently retrieved

NOTE: Trying to remove items using a for loop or a for-each loop would not work correctly because the collection is changing size while the code is trying to loop. It is better to use iterator if we need to remove elements from collection while looping it.

What are the differences between List, Set and Map?

- List, Set and Map are very commonly used interfaces and important in Java Collection Framework
- List is an ordered and indexed collection and allows duplication.
    - It contains index-based methods as it stores elements with their indexes and those methods can be used to insert, update, delete and search elements in the collection.
    - It allows to store countless null elements
- Set is an unordered collection of unique objects and it does not allow duplication.
    - There is no insertion point in Set collection which means we cannot use any methods that uses indexes to insert, update, delete or even get.
    - Since Set does not allow duplication, you can have only one null element.

- Map is a data structure that helps us to store multiple items in a single unit as KEY and VALUE
    - Each key is linked to a specific value.
    - Each key MUST be unique in a Map.
    - Value for the keys can be duplicate.
    - Since keys are always unique, there can be only one null key but many null values.

1) Duplicity:

- List allows duplicate elements. List can have any number of duplicate elements.
- Set does not allow duplicates. Set and all of the classes which implements Set interface should have unique elements.
- Map stores the elements as key & value pairs and Map does not allow duplicate keys while it allows duplicate values.

2) Null values:

- List allows any number of null values.
- Set allows only one null value.
- Map can have only one null key and any number of null values.

3) Order:

- List and all of its implementation classes maintains the insertion order.
- Set does not maintain any order but few of its classes sort the elements. For example, LinkedHashSet maintains the elements' in insertion order.
- Similar to Set Map also does not stores the elements in an order, however few of its classes does the same. For example, TreeMap sorts the map in the ascending order of keys and LinkedHashMap sorts the elements in the insertion order, the order in which the elements got added to the LinkedHashMap.

4) Commonly used classes:

- List: ArrayList, LinkedList etc.
- Set: HashSet, LinkedHashSet, TreeSet, SortedSet etc.
- Map: HashMap, TreeMap, WeakHashMap, LinkedHashMap, IdentityHashMap etc.

When to use List, Set and Map in Java?

- If you do not want to have duplicate values in the collection, then Set should be your first choice as all of its classes do not allow duplicates.
- If there is a need of frequent search operations based on the indexes, then List (ArrayList) is a better choice.
- If there is a need of maintaining the insertion order, then also the List is a preferred collection interface.
- If the requirement is to have the key & value mappings in the collection, then Map is best to use.

Differences	LIST	SET	MAP
DUPLICATION	The list interface allows duplicate elements	Set does not allow duplicate elements	The map does not allow duplicate elements
ORDER	The list maintains insertion order	Set do not maintain any insertion order.	The map also does not maintain any insertion order
NULL VALUES	We can add any number of null values	But in set almost only one null value	The map allows a single null key at most and any number of null values
IMPLEMENTATION	List implementation classes are Array List, LinkedList	Set implementation classes are HashSet, LinkedHashSet, and TreeSet.	Map implementation classes are HashMap, HashTable, TreeMap, ConcurrentHashMap, and LinkedHashMap.
INDEX	The list provides get() method to get the element at a specified index.	Set does not provide get method to get the elements at a specified index	The map does not provide get method to get the elements at a specified index
USE	If you need to access the elements frequently by using the index then we can use the list	If you want to create a collection of unique elements, then we can use set	If you want to store the data in the form of key/value pairs, then we can use the map.

What are the similarities and differences between HashMap and HashTable?

- Similarities:
  - Both HashMap and HashTable classes are implementing Map Interface.
  - Both HashMap and HashTable are used to store key/value pairs.
  - We specify an object that is used as a key, and the value that is linked to that key when use either HashMap or HashTable.
- Differences:
  - HashMap is not synchronized which means it is not thread safe and cannot be shared between many threads without proper synchronization whereas HashTable is synchronized. It is thread-safe and can be shared with many threads.
  - HashMap allows one null key and multiple null values whereas HashTable does not allow any null key or value.
  - HashMap is generally preferred over HashTable if thread synchronization is not needed. So, we don't have issues with synchronization, then we will prefer to go with HashMap as it is faster than HashTable considering performance.

Differences	HashMap	HashTable
SYNCHRONIZATION	HashMap is non synchronized. It is not thread safe and cannot be shared between many threads without proper synchronization code.	Hashtable is synchronized. It is thread-safe and can be shared with many threads.
NULL VALUES	HashMap allows one null key and multiple null values.	Hashtable does not allow any null key or value.
PERFORMANCE	HashMap is fast.	Hashtable is slow.
UN-SYNCHRONIZATION	We can make the HashMap as synchronized by calling this code Map m = Collections.synchronizedMap(hashMap);	Hashtable is internally synchronized and cannot be unsynchronized.
USE	HashMap is preferred to be used when we do not have issues with synchronization as it is faster than HashTable	HashTable is used when we need thread-safe execution and synchronization

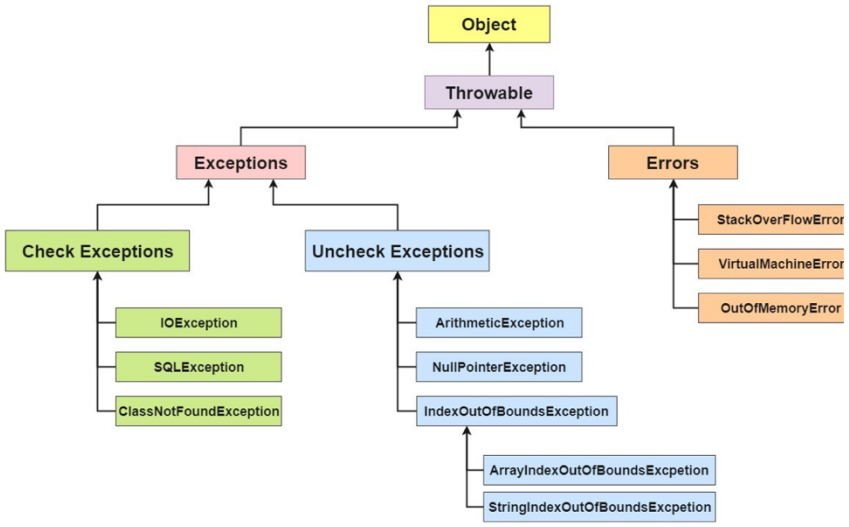
What does thread-safe mean in Java?

- Java has Multithreading feature, and this feature allows us to run multiple threads simultaneously.
- Imagine, few test scripts running at the same time in different threads. There might be possibility that these scripts are trying to use same data, and the value of that data might change due to other threads running. In this case, we might get un-expected results although our code is right. Issues we might get are because of changing data and threads not being synchronized. In such cases we are implementing multithreading, one thread should wait other in case of sensitive data and all threads must be synchronized in this way. This process is called Thread-Safety and easiest way to achieve it is synchronization.
- REMEMBER, in the past we discussed that if you need multithreading for your script, you should consider using thread-safe classes for your scripts like HashTable for key/value pairs instead of HashMap or StringBuffer to manipulate strings instead of StringBuilder.

What are errors and exceptions in Java?

- Error and exceptions are known as unexpected result that breaks flow of the program
- Both errors and exceptions are the subclasses of java.lang.Throwable class.
- ERRORS:
    - Errors are the conditions which cannot get recovered by any handling techniques.
    - Errors are unchecked types and mostly occur at runtime
    - These are serious and not programmer's responsibility like OutOfMemoryError
- EXCEPTIONS:
    - Exceptions can get recovered by try catch blocks or throw keyword
    - Exceptions are divided into two categories
    - Checked exceptions are known to the compiler at compile time like IOException
    - Unchecked exceptions are known to the compiler at runtime like ArrayIndexOutOfBoundException
    - It is mostly caused by the program written by the programmer. It is a BUG needs to be fixed

What is throwing exception in Java?



- Many different exceptions may be thrown when we execute a java program. It could be some coding errors made by the programmer or errors due to wrong input, or other unexpected things
- Execution of the code stops when an exception is thrown, and an error message is generated
- The technical term for this is throwing an exception in Java

What are checked and unchecked exceptions in Java?

- Checked exceptions are also known as compile-time exceptions
    - These kinds of exceptions must be handled before a program is run
    - We need to handle these with try-catch (sometimes finally) blocks before running program
    - Our program will not compile until checked exception is handled
    - EX: InterruptedException, FileNotFoundException, IOException
- Unchecked exceptions are also known as run-time exceptions
    - These kinds of exceptions will be thrown when program is run
    - So, program compiles and does not show any error until we run the program
    - EX: StringOutOfBoundException, ClassCastException, NullPointerException

How to handle exceptions in Java?

- We can handle checked exceptions by using try catch blocks
- The try statement allows you to define a block of code to be tested for errors while it is being executed
- The catch statement allows you to define a block of code to be executed, if an error occurs in the try block
- try block can be used with multiple catch blocks in some cases. This might be used when code executed in try block might throw multiple different exceptions
- NOTE: We cannot catch child exception under parent exception type. In this case catch block used for child exception will be unreachable and cause a compiler error
- NOTE: We can add exception to method signature to handle as well but this is not the proper way!

What is finally block and its relationship with try-catch blocks?

- The finally block allow us to execute a final code after try...catch, regardless of the result
- It is used to run a final block of code after try catch
- NOTE: try can ONLY be used either with catch or finally block
- BUT it is optional to have only catch or finally block or both after try block

What are final, finalize, finally keywords in Java and their differences?

- final: it is kind of a non-access modifier and do not control the access level for class members, but it provides some other functionalities
- A field can be final-> final field must be initialized, and this value cannot be changed
- A method can be final-> this method cannot be overridden (it can still be overloaded)
- A class can be final-> this class cannot be extended

- finalize(): it is a method can be used to run a block of code before object reference is garbage collected
- Purpose of using this method is to do proper clean up before removing the object
- By default, finalize method is empty. However, you can write your own finalize method to take certain action
- This method is in Object class and can be overridden wherever it is needed

- finally: is a block used with try-catch block
- The **finally** block allow us to execute a final code after try...catch, regardless of the result
- It is used to run a final block of code after try catch

What are throw and throws keywords in Java and their differences?

- Both of these keywords are related to Exceptions, but they are different from each other.
- We can create our own exceptions in Java and throw keyword allows us to create a custom error/exception
- The throw statement is used together with an **exception type** within a method block
- *We can create both checked and unchecked exceptions*
- On the other hand, throws keyword is used to add exception to method signature

NOTE:

  - In method overriding concept, if parent class method throws an exception, then child class method should also throw an exception
  - This exception must be same or more specific exception, but it cannot be more general type of exception

TechGlobal

Recently Asked Java Interview Questions

What are the restrictions of Arrays?

- This question could be asked as "Why we need Collections in Java" as well
- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value
- By using an Array, we can store many similar items/elements in one variable
- Arrays can be used with both primitive and reference data types while all other collections can only be used with reference types (objects)
- HOWEVER, Array has some restriction and that's why we need to have a good knowledge of using collections based on the need
- Array does not provide methods to manipulate data as other collections

How to convert String to an int?

- This question could be asked for all of the primitives except char.
- We can convert String to related primitive by using their Wrapper classes
- valueOf() and parseType() methods can be used for the conversion

EXAMPLE:

```
String str = "235";
int number1 = Integer.valueOf(str);
or
int number2 = Integer.parseInt(str);

double d1 = Double.valueOf(str);
or
double d2 = Double.parseDouble(str);
```

NOTE: parseInt(), parseDouble() or parseBoolean() and other parse methods from wrapper classes are used to convert given String into the primitive data types and return primitive while valueOf() methods converts given object to wrapper class objects.

How to convert primitive int to a String?

- This could be asked for all other primitive data types
- We can convert any primitive data type to a String
- Using + (concatenation), will convert any data type to a String
- Or String.valueOf(variable) method can be used

EXAMPLE:

```
int age = 45;
String stringAge1 = age + "";      // "" is known as empty String
String stringAge2 = String.valueOf(age);
```

What is the difference between = and == operators?

- = is the assignment operator and used to assign values into variables.
  EXAMPLE:
  int age = 45; // 45 is assigned to the age variable
- == is one of the relational operators and it is used to find if both sides of the operator are equal to each other. It will return a boolean, either true or false based on the comparison
  EXAMPLE:
  5 == 5 // this comparison will return true and it can be assigned to a boolean as below
  boolean equal = (5 == 5); // in this statement, equal variable has value of true as 5 equals to 5

What is the difference between == operator and equals() method?

- == is one of the relational operators and it is used to find if both sides of the operator are equal to each other. It will return a boolean, either true or false based on the comparison
- == is mostly used with primitive data types and should not be used with reference types (objects)
- When == is used with reference types, it compares their location in the memory and each object created with new keyword will have a unique location. That's why, although the object variable values are same, == will return a false since it compares their location and locations are obviously different.
- equals() is mostly known to be used with String objects but String is not the only class that has equals() method. Wrapper classes, Arrays, ArrayList, Vector and other collections also has equals() method and this method is used to check 2 objects of same data type has same value or not and return a boolean as true or false

EXAMPLES

```
String str1 = "Tech";
String str2 = "Global";
System.out.println(str1.equals(str2)); // this statement will print false as 2 String object values are not equal
```

Or

```
Map<String, String> map1 = new HashTable<>();
map1.put("SchoolName", "TechGlobal");
map1.put("Address", "2860 S River Rd Suite 350, Des Plaines IL 60018");
Map<String, String> map2 = new HashTable<>(map1);
System.out.println(map1.equals(map2)); // this statement will print true as 2 HashTable object keys and values are equal
```

How can we use primitive data types as objects?

- Primitive data types like int can be handled as objects by the use of their respective wrapper classes
- For example, Integer is a wrapper class for primitive data type int
- We can apply different methods to a wrapper class, just like any other object

Is it possible to override and overload main, private, final, static or abstract methods?

- This question is frequently asked for final methods, abstract methods, static methods, private methods and main method. DO NOT MEMORIZE, LEARN LOGIC BELOW!

REMEMBER:

- Method overloading is creating a method which has same name with another method in the class but different signature
- Overloading happens in the same class with different parameters
- Overloaded methods are sharing same name but different signature
- Method overloading is not related to inheritance
- Method overriding is implementing a new body to an inherited method from parent class
- Overriding methods can happen only when a child class inheriting a parent class
- Overriding is writing same method from parent class with a different implementation in the child class
-

FROM INFORMATION GIVEN ABOVE,

- Private methods are specific to the class they are created and are not visible to child classes. So, overriding a private method is not possible but they can be overloaded in the same class with different method signature
- Static methods belong to the class they are created and cannot be overridden in the child class. On the other hand, they can be overloaded in the same class meaning that you can have 2 or more static methods with same name as long as they have different parameter types
- Abstract methods do not have body and it is left to be implemented by non-abstract child classes, so the purpose of abstract methods is to be overridden. And they can also be overloaded in the same class
- Final methods cannot be overridden and purpose of making a method final is to prevent overriding but they can be overloaded
- NOTE: abstract methods cannot be final. These two modifiers are completely conflicting with each other. Making a method final means the method cannot be overridden while making a method abstract means it must be overridden in the child class. So, a method can never be final and abstract at the same time.
- NOTE: abstract methods cannot be private. These two modifiers are also conflicting with each other. Making a method private means the method will be specific to the class it created and cannot be used in any other class while making a method abstract means it must be overridden in the child class. So, a method can never be private and abstract at the same time.
- main() method in Java is used to run a program and it tells JVM where the starting point of the program is. Remember, we said static methods cannot be overridden and since main() method is also static, it cannot be overridden. However, it can be overloaded. You should know that if you overload main() method, JVM will still only use original main() method to run your program and your overloaded main method can only be executed if you call it in the original main() method

Can we have abstract class without having any abstract method in it?

- Yes, we can have abstract class without having any abstract method
- REMEMBER, abstract class can have both abstract and non-abstract methods and it may not have one of these

Can we have two methods in a class with the same name?

- Yes, we can define two methods in a class with the same name but with different number/type of parameters
- This is known as method overloading

Is it possible to have static methods in an Interface?

- Methods are by default public abstract in the Interface
- BUT we can have static methods in an Interface after Java 8
- We can also have default methods that has body in an Interface

Can we declare the main() method as private?

- main() method must be public static in order to run any application correctly
- If main method is declared as private, programmer will not get any compilation error but, it will not get executed

How we can execute any code even before main() method?

- We know that starting point of an application in Java is main() method and when we run a program JVM first finds the main() method and executes statements in it
- If we want to execute any statements before even creation of objects, we can use a static block of code in the class
- Any statements inside the static block of code will get executed once at the time of loading the class even before creation of objects in the main method

Can main() method return any data in Java?

- No, main() method cannot return any data as it is a void type method

Is it possible to make constructor final?

- No, we cannot declare constructor as final and we cannot declare constructor as static

How objects of a class are created if no constructor is defined in the class?

- REMEMBER: Java always provides a default constructor even if we do not create a custom one
- Objects are getting created successfully with default constructor
- This constructor has no parameters

Is it possible to call a constructor from another constructor's body?

Yes. If a class has multiple constructors, then it is possible to call one constructor from the body of another one using this()

Can we use a default constructor of a class even if a custom constructor is created?

- Java provides a default no argument constructor if we do not create any custom constructor
- However, once we create a custom constructor, default constructor cannot be used anymore
- If we still want to create objects with default constructor, we need to explicitly define it in the class

Is it possible to create an object of any Interface?

- They may ask same question for abstract classes as well
- NO, we cannot instantiate both abstract classes and interfaces meaning we cannot create objects for them
- REMEMBER:
    - Both abstract classes and interfaces are created to be used by other classes with inheritance
    - Abstract classes and interfaces are created to be used by many similar subclasses and provides some common fields and methods to those subclasses and increases code reusability
    - Abstract class can be parent to many other child classes but can extend to only one parent (single inheritance). To be able to have additional features, Java introduces interfaces and multiple inheritance is possible with interfaces. One interface can extend to many other interfaces while one class can implement many other interfaces

Can one interface implement another interface in java?

- No, one interface cannot implement another interface. It can extend it using extends keyword

Can a class be a super class and a sub-class at the same time?

- Yes, a class can be a super class for another class and a sub-class to another one at the same time

Can a class in Java be inherited from more than one class?

- Yes, one class can be base (parent) to multiple classes, but one child cannot extend to multiple base classes

How can we restrict inheritance for a class so that no class can be inherited from it?

- REMEMBER: when we make a class final, it means it cannot be inherited
- So, if we want a class not to be extended further by any class, we can use the keyword final with the class name, and this will restrict it to be a parent class

What is the base class for all the classes in Java?

- Base class means super class or parent class as well
- Object class the base class for all of the other classes in Java
- Object class is coming from java.lang package

Can you list down some of important method from object class?

- equals() - It compares the object references
- toString() - Provides String representation of the object
- finalize() - This method is called when object is being garbage collected

What are the types of comments in Java?

- Single line comment with //This is a single line comment
- Multiple line comments /*This is multiple line comments*/

Why String is immutable?

- String class is immutable in Java
- It means unable to be changed or unchanging over time, so String is unchangeable or unmodifiable in Java
- When we create a String as String s = "Text"; It will be stored in String Pool in the heap and this value might be referenced by some other variables as well. And if you were able to change the value, then it would change for all variables referencing to the same value. To prevent this, String is immutable
- Another reason is being thread-safe (synchronized). String and all other immutable object are thread-safe by default, so you don't need to explicitly implement synchronization for them
- It is also safe to use considering security as the value of String will not be changed

What are two different ways to call garbage collector explicitly?

- Garbage collection is destroying objects that lose references and it happens implicitly in Java
- BUT, we can do garbage collection explicitly using System.gc() or Runtime.getRuntime().gc() methods
- These methods can be used as a hint to the JVM, in order to start a garbage collection. However, this it is up to the Java Virtual Machine (JVM) to start the garbage collection immediately or later in time

Can we have try block without catch block in Java?

- Yes, we can have try block without catch block by using finally block
- You can use try with either catch or finally block or with both

How can you generate random numbers in Java?

- We can use Math.random() to generate random numbers
- Also, Random class in package java.util can be used to generate Random numbers
  NOTE: Some third-party libraries can be added to project and be used to generate random data including numbers

What are Java packages? What's the significance of packages?

- Package is a collection of classes and interfaces which are bundled together as they are related to each other
- Use of packages helps programmers to modularize the code and group the code for proper re-use
- Once code has been packaged in packages, it can be imported in other classes and used

What is the purpose of using break in each case of switch statement?

- break is used after each case (except the default case) in a switch, so that code breaks after the valid case and does not flow in the proceeding cases too
- If break is not used after each case, all cases after the valid case will get executed resulting in wrong results

Can a variable be local and static at the same time?

- No, a variable cannot be static as well as local at the same time
- Defining a local variable as static gives a compilation error
  NOTE: Instance variables can be static, and it means it shared by all objects of the class and belongs to the class anymore

Is Java %100 Object-Oriented Programming Language?

- Java is an Object-Oriented Programming Language
- BUT Java has primitive data types, and that is why Java is not %100 Object-Oriented

What is Enum in Java?

- An enum is a special "class" that represents a group of constants (unchangeable variables, like final variables).
- To create an enum, use the enum keyword (instead of class or interface), and separate the constants with a comma
- Note that they should be in uppercase letters
- Enum is useful to use for values that will not change like months, days, colors, etc.

What is multi-threading in Java?

- Multi-threading is a programming concept to run multiple tasks in a concurrent manner within a single program
- Threads are used for that and they share same process stack and running in parallel for different tasks
- It helps in performance improvement of any program