How JAVA works internally ?

Java works on the principle of Write Once, Run Anywhere.
When we compile a Java program, the .java source file is first compiled by the Java Compiler (javac) into bytecode (.class file).
This bytecode is platform-independent and runs on the Java Virtual Machine (JVM).

The JVM then uses the Class Loader to load classes, the Bytecode Verifier to check security and validity, and the JIT (Just-In-Time) Compiler to convert bytecode into native machine code at runtime for faster execution.

What is Compiler ?

A compiler is a program that converts high-level source code into bytecode or machine code. In Java, the javac compiler translates .java files into .class files (bytecode), which are later executed by the JVM.

What is Interpreter ?

An interpreter executes the program line by line instead of compiling it all at once.
In Java, the JVM acts as the interpreter — it reads and executes the bytecode produced by the compiler.
This makes Java platform-independent, as the same bytecode can run on any system with a JVM.

What is JDK & JRE?

JDK stands for Java Development Kit — it contains tools to develop, compile, and run Java applications.
It includes the JRE, compiler (javac), and other development utilities.

JRE stands for Java Runtime Environment — it provides the libraries and JVM needed to run Java applications, but it doesn't include development tools like the compiler.

What is JVM ?

JVM is the engine that runs Java programs.
It takes the compiled bytecode and converts it into machine code that the OS can execute.
It provides platform independence, memory management, and garbage collection.

What is Class? why it requires?

A class in Java is a blueprint or template used to create objects.
It defines the properties (variables) and behaviour's (methods) that the objects created from it will have.

Why Class is Required?

1. ✅ Reusability – Write code once and create multiple objects.
2. ✅ Encapsulation – Combine data (variables) and behavior (methods) together.
3. ✅ Organization – Helps structure code logically and modularly.
4. ✅ Object-Oriented Programming – Classes make OOP possible by defining objects.
5. ✅ Maintainability – Easier to debug and maintain as logic is well-structured.

What is Object? why it requires?

An object is a runtime instance of a class.
It represents a real-world entity that has state (data/variables) and behavior (methods).

Why Object is Required?

Objects are essential in Object-Oriented Programming (OOP) because they:

1. ✅ Represent real-world entities — e.g., Car, Employee, Account.
2. ✅ Encapsulate data and behavior together — keeps data secure.
3. ✅ Enable reusability — one class can create multiple independent objects.
4. ✅ Improve modularity — each object works independently.
5. ✅ Simplify maintenance — easy to modify and extend.

What is Reference Variable? why it requires?

A reference variable is a variable that stores the memory address (reference) of an object — not the actual object itself.

It acts like a pointer to the object in Java's heap memory, allowing you to access the object's fields and methods.

Why Reference Variables are Required:

Purpose	Description
Access object members	We use the reference to access variables and methods of the object.
Memory efficiency	Multiple references can point to the same object — no need to duplicate data.
🔄 Object management	Helps JVM handle objects and garbage collection.
Flexibility	You can assign null, reassign to another object, or pass it as a method argument.