# **Java**

Question: Is JVM Platform dependent?

Ans: Yes, JVM is catered to individual OS. So that it can convert Byte code i.e. class files to machine code for each OS. That is why we have individual installer for each type of OS

Question: Can we run Java Program if we only have JRE?

Ans: yes, if we have class files we can run with JRE, but JRE doesn’t have necessary tools to develop or run .java files

Question: How does JIT improve performance?

Ans: JIT converts frequently used bytecode ("hotspots") into native machine code, allowing faster execution. When a Java program starts, the JVM interprets bytecode line by line (slow due to interpretation overhead). The JVM monitors execution and identifies frequently run code (hotspots). The JIT compiler compiles only these hotspots into native machine code. Future calls to these hotspots run directly as machine code, skipping interpretation — much faster.

Question: What is cold start in Java?

Ans: A **cold start** happens when an application (especially in serverless platforms like AWS Lambda) has to **start from scratch** and longer time to start

Java has **notoriously longer cold starts** compared to languages like Python or Node.js. Here's why:

| **Reason** | **Description** |
| --- | --- |
| Heavy JVM startup | Starting the Java Virtual Machine (JVM) takes time — it's a full-blown runtime environment. |
| Class loading | Java loads classes dynamically, and the process takes time during startup. |
| Static initialization | Java apps often have lots of static initialization or dependency injection (like Spring Boot) that slows down startup. |
| Memory usage | JVM-based apps consume more memory, causing more time in constrained environments like serverless. |
| JIT Compilation | The JVM may use Just-In-Time (JIT) compilation to optimize code at runtime, which adds a startup penalty. |

Question: For the below Box class

class Box {  
 int height, width, length;  
  
 int area() {  
 return this.height \* this.length \* this.width;  
 }  
  
 public Box(int height, int width, int length) {  
 this.height = height;  
 this.width = width;  
 this.length = length;  
 }  
}

if we do

Box b1 = new Box(1,1,1);  
Box b2 = b1;  
b2.length =2;  
System.out.println(b1.length);

What will be the value of b1.length? Will it change?

Ans: Yes Both b1 and b2 point to same object. Changing one will change other objects so b1.length will also be 2

Question: For the below Box class

class Box {  
 int height, width, length;  
  
 int area() {  
 return this.height \* this.length \* this.width;  
 }  
  
 public Box(int height, int width, int length) {  
 this.height = height;  
 this.width = width;  
 this.length = length;  
 }  
}

if we do

Box b1 = new Box(1,1,1);  
Box b2 = b1;  
b1 = null;  
System.out.println(b2.length);

What will be the value of b2.length?

Ans: when we do b1= null. It change the reference of b1 to null, but b2 keeps pointing to same memory location and will hold the same value of 1

Question: What is the type of k below?

public static void main(String[] args) {  
 byte i =4 ;  
 byte j = 6;  
 var k = i+j;  
}

Answer: int ,