# **Java Basic**

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: Where are below items stored?

* Objects
* Local variables method calls
* Class definition static variables
* Native library

Ans:

* Objects - heap
* Local variables method calls - Stack
* Class definition static variables – Method area
* Native library – Native Heap

# Data Types Operators

Questions: Why is char given so much storage space? 2^16 or 65,536

Ans: So large to accommodate all special character, uses UNICODE

Question: How we use special character like ‘?’

Ans: We first get the Unicode value 003F and the represent it using char using \u003F

char i = '\u5632';  
System.out.println(String.valueOf(i));

Question: Can we store numbers in char?

Ans: Yes, they represent special characters

char c = 63;  
System.out.println(c); // Will print '?'

Question: Can we store -ve numbers in char?

Ans: No only +ve numbers from 0 till 2^16

Question: What is the type of k below?

byte i = 4;  
byte j = 6;  
var k = i + j;  
byte b = 3 + 4;

Answer: k 🡪 Int Java defaults to integer when we do operation of variables smaller than int like byte, short

b 🡪 byte it doesn’t happen inside an expression as it is calculated compile time.

Question: Which type of casting is implicit or automatic?

Answer: Widening Casting or converting a smaller type to a larger type size. It is done automatically & safe because there is no chance to lose data.

byte -> short -> char -> int -> long -> float -> double

Question: Here we are assigning a integer to a byte which is out of range of byte(- 128 to 127) with explicit casting. Will this compile and run?

int i = 128;  
byte b = (byte) i;  
System.out.println(b);

Ans: Yes, it will compile and run. If the number is out of range, it will go to negative side and the leftover will be calculated on the negative side. The process will continue till the total value is calculated

Byte range -128 till 127, range r = 127

Int I = 128 -r = 1

SO value will be -1

Question: With explicit casting we can assign out of range values. Can you do direct assignment also out of range?

byte b = 300;

Ans: No with direct assignment it will not compile

Question: What will be the value of b?

int i = 300;  
byte b = (byte) i;  
System.out.println(b);

Answer: byte range -128 to 127

So with 300,

1. 0 to 127 int value 127 final value 127
2. -128 to -1 int value 255 final value 0
3. 0 to 45 int value 300 final value 44

Question: What will be the value of 2 print statement below

Integer i1 = 127;  
Integer i2 = 127;  
System.out.println(i1 == i2);  
i1 = 128;  
i2 = 128;  
System.out.println(i1 == i2);

Answer: Wrapper classes do caching for small values to save memory.

ByteCache (-128 to 127)

ShortCache (-128 to 127)

LongCache (-128 to 127)

CharacterCache (0 to 127)

Which means when the value is within this value java will resue the values

So first one will be true as both will point to same location , 2nd one will be false as it will be not cached and point to different location

Question: What will be the value of j in 2 print statements?

int i = 5;  
int j = i++;  
System.out.println(j);  
j = ++i;  
System.out.println(j);

Ans 5 and 7

Question: What is the difference between & , && and |, ||d

Answer:

|  |  |  |
| --- | --- | --- |
| & | Both true | Works for integer and boolean |
| && | Both true but if 1st is false won’t evaluate 2nd | Works only for boolean, better performance |
| | | Any one true | Works for integer and boolean |
| || | if 1st is true won’t evaluate 2nd | Works only for boolean, better performance |

Question: What is the difference between the 2 below?

        String s1 = "Hello";

s1 = new String("hello");

Ans : 1 will use string pool it will not create a new object if “Hello” exists in string pool, 2 will go into heap and irrespective of the value, if the string “Hello” is already created it will still create a new object.

Question: Why equals method is used for String rather than ==?

Ans: Since String usually uses string pool using so multiple objects point to same location. So it makes more sense to check the values

Question: How is the format method used with String?

Answer :

String message = "Hello, %s ! You have %d messages."; // %s for replacing String , %d for int  
System.out.println(String.format(message, "Amit", 3));  
System.out.printf((message) + "%n", "Amit", 3);  
  
String message1 = "The price is $%.2f"; // %.f to insert double or float 2 represent the no of decimal needed  
System.out.println(String.format(message1, 19.99));  
  
String message2 = "The number is %5d"; // %5d 5 for 5 spaces and d for int  
System.out.println(String.format(message2, 5));

Question: How can we break the String “Hello Java Developer Amit” in to “Hello Java” , “Developer” and “Amit” without using hardcoded index?

Answer:

String message = "Hello Java Developer Amit";  
 int middle = message.indexOf("Developer");  
 System.out.println(message.substring(0,middle));  
 System.out.println(message.substring(middle, middle + "Developer".length()));  
 System.out.println(message.substring(middle + "Developer".length()));  
}

Question: