Java Programming Basics

Part 1: Introduction to Java

1. **What is Java?** Java is a high-level, object-oriented programming language known for its platform independence, robustness, and wide adoption in software development.
2. **Key Features of Java:**
   * Platform Independence
   * Object-Oriented
   * Simple and Secure
   * Multi-threaded
   * Automatic Memory Management (Garbage Collection)
   * Rich API
   * Robust and Portable
   * High Performance
3. **Compiled vs Interpreted Languages:**
   * Compiled: Translated into machine code before execution (e.g., C, C++).
   * Interpreted: Executed line by line by an interpreter (e.g., Python, JavaScript).
   * **Java fits as a hybrid language**: It is compiled into bytecode and then interpreted by the JVM.
4. **Platform Independence:** Java programs are compiled into bytecode, which can run on any platform with a Java Virtual Machine (JVM), making Java platform-independent.
5. **Applications of Java:**
   * Web Development (Spring, JSP, Servlets)
   * Mobile Applications (Android)
   * Enterprise Applications
   * Game Development
   * Embedded Systems
   * Cloud Computing
   * Big Data Processing

Part 2: History of Java

1. **Who developed Java and when?** Java was developed by James Gosling and his team at Sun Microsystems in 1995.
2. **Original Name and Reason for Change:** Initially called "Oak," it was renamed "Java" due to trademark issues.
3. **Evolution of Java Versions:**
   * Java 1.0 (1996) – Initial release
   * Java 2 (J2SE 1.2, 1998) – Introduced Swing and Collections
   * Java 5 (2004) – Generics, enhanced for-loop
   * Java 8 (2014) – Lambda expressions, Streams API
   * Java 11 (2018) – LTS release, improved APIs
   * Java 17 (2021) – LTS release, pattern matching
4. **Major Improvements in Recent Versions:**
   * Lambda Expressions
   * Records and Sealed Classes
   * Improved Garbage Collection
   * Pattern Matching for Switch
   * Jigsaw (Modularization)
5. **Comparison with C++ and Python:**
   * Java is more secure and platform-independent than C++.
   * Java has better performance than Python but lacks its simplicity.

Part 3: Data Types in Java

1. **Importance of Data Types:** They define the type and size of data stored in variables.
2. **Primitive vs Non-Primitive:**
   * Primitive: Built-in types (int, double, char, etc.).
   * Non-Primitive: Objects, Arrays, Strings, etc.
3. **Eight Primitive Data Types:**
   * byte, short, int, long, float, double, char, boolean
4. **Examples:**
5. int age = 25;
6. double price = 99.99;
7. char grade = 'A';
8. boolean isJavaFun = true;
9. **Type Casting:**
10. int num = 10;
11. double converted = num; // Implicit Casting
12. double d = 9.5;
13. int i = (int) d; // Explicit Casting
14. **Wrapper Classes:** Provide object representation of primitive types (Integer, Double, etc.).
15. **Static vs Dynamic Typing:**
    * Static: Type checked at compile-time (Java, C++)
    * Dynamic: Type checked at runtime (Python, JavaScript)
    * **Java is statically typed.**

**Coding Questions on Data Types:**

1. **Declare and initialize all primitive types:**

public class DataTypes {

public static void main(String[] args) {

byte b = 100;

short s = 5000;

int i = 100000;

long l = 150000L;

float f = 5.75f;

double d = 19.99;

char c = 'A';

boolean bool = true;

System.out.println("Values: " + b + ", " + s + ", " + i);

}

}

1. **Arithmetic Operations:**

import java.util.Scanner;

public class Arithmetic {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int a = sc.nextInt(), b = sc.nextInt();

System.out.println("Sum: " + (a + b));

}

}

1. **Type Casting Demonstration:**

int num = 10;

double d = num; // Implicit

int x = (int) 10.5; // Explicit

1. **Wrapper Class Conversion:**

Integer i = 100;

Double d = i.doubleValue();

System.out.println(d);

1. **Swap Numbers:**

int a = 5, b = 10;

a = a + b - (b = a);

1. **Vowel or Consonant:**

char ch = 'a';

if("aeiouAEIOU".indexOf(ch) != -1) System.out.println("Vowel");

else System.out.println("Consonant");

1. **Even or Odd using Command-Line Arguments:**

int num = Integer.parseInt(args[0]);

System.out.println(num % 2 == 0 ? "Even" : "Odd");

Part 4: Java Development Kit (JDK)

1. **JDK vs JRE vs JVM:**
   * JDK: Development Kit with compiler and tools
   * JRE: Runtime Environment
   * JVM: Runs Java bytecode
2. **Main Components of JDK:**
   * Compiler (javac)
   * Java Runtime (JRE)
   * Debugging Tools
3. **JDK Installation Steps:**
   * Download from Oracle/OpenJDK
   * Install and configure PATH
4. **Hello World Program:**

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

1. **PATH vs CLASSPATH:**
   * PATH: Locates executables
   * CLASSPATH: Locates Java classes
2. **OpenJDK vs Oracle JDK:**
   * OpenJDK: Open-source
   * Oracle JDK: Proprietary with additional features
3. **Java Compilation & Execution:**

javac HelloWorld.java

java HelloWorld

1. **JIT Compilation:**
   * Converts bytecode to machine code at runtime for speed.
2. **Role of JVM:**
   * Executes Java bytecode, manages memory (GC), and ensures security.