Java Programming Basics

Part 1: Introduction to Java

1. What is Java? Explain its significance in modern software development.

Java is a high-level, object-oriented programming language developed by Sun Microsystems in 1995 (now owned by Oracle). It is widely used for developing applications across various domains, including web, mobile, enterprise, and embedded systems. Java’s significance in modern software development lies in its platform independence, robustness, security, and vast ecosystem.

2. Key Features of Java

- **Platform Independent**: Write once, run anywhere (WORA) due to the Java Virtual Machine (JVM).
- **Object-Oriented**: Follows object-oriented programming principles like inheritance, encapsulation, and polymorphism.
- **Robust & Secure**: Provides memory management, exception handling, and security features.
- **Multithreaded**: Supports concurrent execution of multiple threads.
- **Automatic Memory Management**: Uses garbage collection to manage memory.
- **High Performance**: Uses Just-In-Time (JIT) compilation for optimized performance.
- **Distributed Computing**: Supports remote method invocation (RMI) and networking.

3. Compiled vs Interpreted Languages & Java's Position

Compiled languages (e.g., C, C++) are converted to machine code before execution, while interpreted languages (e.g., Python) execute code line-by-line. Java is both compiled and interpreted. Java code is compiled into bytecode, which is then interpreted by the JVM, making it platform-independent.

4. Platform Independence in Java

Java programs are compiled into an intermediate bytecode rather than platform-specific machine code. The JVM interprets this bytecode, allowing Java applications to run on any system with a compatible JVM.

5. Applications of Java

- Web Development (Spring, Java EE)
- Mobile Applications (Android development)
- Enterprise Applications
- Cloud Computing
- Big Data and Machine Learning
- Embedded Systems

Part 2: History of Java

1. Developer & Introduction

Java was developed by James Gosling and his team at Sun Microsystems and was released in 1995.

2. Initial Name & Change

Originally called "Oak," it was later renamed "Java" due to trademark issues and inspiration from Java coffee.

3. Evolution of Java Versions

- Java 1.0 (1995): Initial release
- Java 2 (1998): Introduced Swing, Collections Framework
- Java 5 (2004): Introduced Generics, Autoboxing
- Java 8 (2014): Introduced Lambda Expressions, Streams
- Java 11 (2018): Long-Term Support (LTS) version
- Java 17 (2021): Latest LTS version with performance improvements

4. Major Improvements in Recent Versions

- Records, Sealed Classes, Pattern Matching (Java 14-17)
- Virtual Threads for better concurrency (Java 19+)

5. Java vs Other Languages

- **C++**: Java has garbage collection, while C++ requires manual memory management.
- **Python**: Java is faster but has more boilerplate code.

Part 3: Data Types in Java

1. Importance of Data Types

Data types define the type of data a variable can hold, ensuring memory efficiency and correctness.

2. Primitive vs Non-Primitive Data Types

- **Primitive**: Predefined types (int, char, float, etc.)
- **Non-Primitive**: Objects and classes (String, Arrays, etc.)

3. Eight Primitive Data Types

1. **byte**: 1 byte (8-bit integer)
2. **short**: 2 bytes (16-bit integer)
3. **int**: 4 bytes (32-bit integer)
4. **long**: 8 bytes (64-bit integer)
5. **float**: 4 bytes (32-bit floating point)
6. **double**: 8 bytes (64-bit floating point)
7. **char**: 2 bytes (16-bit Unicode character)
8. **boolean**: 1 bit (true/false)

4. Declaration & Initialization Examples

```
int a = 10;
char letter = 'A';
double pi = 3.14159;
```

5. Type Casting in Java

Converting one data type into another.

```
int num = 10;
double converted = num; // Implicit casting
```

6. Wrapper Classes & Usage

Wrapper classes (Integer, Double, etc.) allow primitive types to be used as objects.

```
Integer obj = Integer.valueOf(100);
```

7. Static vs Dynamic Typing

Java is **statically typed**, meaning data types are checked at compile time.

Coding Questions

1. Program to print all primitive data types:

```
public class DataTypesExample {
   public static void main(String[] args) {
      int a = 10;
      double b = 20.5;
      char c = 'A';
      boolean d = true;
      System.out.println("Integer: " + a);
   }
}
```

2. Arithmetic operations on two integers:

```
Scanner sc = new Scanner(System.in);
int x = sc.nextInt(), y = sc.nextInt();
System.out.println("Sum: " + (x + y));
```

Part 4: Java Development Kit (JDK)

1. JDK, JRE & JVM Differences

- **JDK (Java Development Kit)**: Contains tools for Java development.
- **JRE (Java Runtime Environment)**: Runs Java applications.
- **JVM (Java Virtual Machine)**: Executes Java bytecode.

2. Components of JDK

- **javac**: Java compiler
- **java**: Java runtime
- **javadoc**: Documentation tool

3. Installing JDK

- Download from Oracle

- Configure environment variables

4. Hello World Program

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

5. PATH & CLASSPATH

- **PATH**: Points to JDK binaries.
- **CLASSPATH**: Defines location of Java classes.

6. OpenJDK vs Oracle JDK

- **OpenJDK**: Open-source, community-driven
- **Oracle JDK**: Licensed, commercial support

7. Java Compilation & Execution

1. Compile: `javac HelloWorld.java`
2. Run: `java HelloWorld`

8. JIT Compilation & JVM Role

- **JIT Compiler**: Converts bytecode into machine code at runtime for better performance.
- **JVM**: Executes Java programs, manages memory.
