Introduction to Java Programming

A Step-by-Step Guide for Learners

1. Introduction to Java

Java is one of the most resilient, powerful, and widely-used programming languages in the world today. Since its debut in 1995, it has set the standard for enterprise computing.

What is Java?

Java is an object-oriented programming language that allows developers to write code that runs on any platform capable of running the Java Virtual Machine. This is the famous **WORA** slogan: Write Once, Run Anywhere.

A Brief History

In 1991, James Gosling at Sun Microsystems began working on a project called "Oak". It was originally intended for consumer electronic devices like cable boxes. By 1995, the project was renamed to **Java**, inspired by the coffee consumed by the team! It was soon adopted by Netscape and became the language of the internet.

2. Key Features of Java

Why do millions of developers choose Java? Here are the core reasons:

- **Simple:** Java is designed to be easy to write, compile, and debug.
- **Object-Oriented:** It uses the concept of 'objects' to model real-world entities, making code reusable and easier to maintain.
- **Platform Independent:** Unlike C or C++, Java does not compile to machine-specific code. It compiles to Bytecode, which is interpreted by the Java Virtual Machine (JVM).
- **Robust:** Java emphasizes error checking and strict memory management, reducing the chances of program crashes.

3. Applications of Java

Java's versatility allows it to exist in many environments:

- **Web Applications:** Back-end systems for large companies use Java frameworks like Spring.
- **Mobile Apps:** Android was built on Java (though Kotlin is now popular, Java remains foundational).
- **Enterprise Software:** Banking, HR, and inventory systems rely on Java's stability.
- **Scientific Computing:** Java is used in natural language processing and advanced mathematics projects.

4. Installing Java: A Quick Setup

Before writing code, you need two things on your system:

1. **JDK (Java Development Kit):** This is the toolkit for developers. It includes the compiler (javac) and tools needed to run programs.
2. **JRE (Java Runtime Environment):** Included in the JDK, this allows your computer to run the compiled code.

Step 1: Download & Install

Visit the official Oracle website or adopt a distribution like OpenJDK for Linux/MacOS users. Run the installer as you would with any software.

Step 2: Environment Variables

This is crucial! You need to tell your operating system where the Java binary files are stored. On Windows, you add the path to the `bin` folder to your System PATH variable. This allows you to run `javac` from any terminal.

Step 3: Verification

Open a terminal or command prompt and type:
`java -version`
If you see a version number return, you are ready to code!

5. Structure of a Java Program

Every Java program must follow a specific blueprint. Java is strictly class-based.

```
public class MyProgram {
    public static void main(String[] args) {
        // Your code goes here
    }
}
```

- **class:** A keyword used to define a class. Everything in Java exists inside a class.
- **public:** An access specifier meaning anyone can access this code.
- **static:** Allows the main method to run without needing to create an object of the class first.
- **void:** The return type; it means this method returns nothing.
- **main():** This is the "start" button of your program!

6. Your First Program

Let's print "Hello, Universe!" to the screen.

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

Explanation: The `System.out.println()` statement is how we tell the computer to write text to the output console followed by a new line.

7. Understanding Java Syntax

Java follows strict rules for syntax:

- **Statements:** Every line that performs an action MUST end with a semicolon (`;`).
- **Case Sensitivity:** Names are significant. `MyVar` is not the same as `myvar`.
- **White Space:** Java ignores extra spaces, but using them properly makes your code "readable" and professional.
- **Comments:** Use `//` for single-line comments and `/* ... */` for multi-line comments.

8. Variables and Data Types

Variables are containers for data. In Java, you must declare the type of data a variable holds.

Primitive Types

- `int`: For integers (e.g., 5, -100).
- `float`: For decimal numbers (e.g., 3.14).
- `char`: For single characters (e.g., 'z').
- `boolean`: For true/false values.

> **Naming Tip:** Use "CamelCase" for variables (e.g., `userAge`, `totalScore`).

9. Operators in Java

Operators perform operations on variables and values.

Arithmetic Operators

- `+` (Addition), `-` (Subtraction), `*` (Multiplication), `/` (Division).
- `%` (Modulus): Returns the remainder of division (e.g., 10 % 3 = 1).

Relational Operators

Used to compare values. They return either `true` or `false`.

- `==` (Equal to), `!=` (Not equal), `>` (Greater than), `<` (Less than).

10. Input and Output in Java

Interactive programs need user input. Java uses the `java.util.Scanner` class.

```
import java.util.Scanner;

public class InputDemo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter your age: ");
        int age = input.nextInt();
        System.out.println("You are " + age + " years old.");
    }
}
```

11. Conditional Statements

Logic allows code to make decisions.

The if-else statement

```
if (num > 0) {
    System.out.println("Positive number");
} else {
    System.out.println("Negative number");
}
```

The switch statement

Use switch to select one of many code blocks to be executed.

```
switch (day) {
    case 1: System.out.println("Monday"); break;
    case 2: System.out.println("Tuesday"); break;
    default: System.out.println("Other day");
}
```

12. Loops

Loops run blocks of code repeatedly.

- **for:** Known number of iterations.
- **while:** Condition-based loop.
- **do-while:** Guarantees at least one execution.

13. Functions (Methods)

Methods prevent code duplication. You define them once and call them whenever needed.

```
public static int add(int a, int b) {
    return a + b;
}
// Calling it:
int result = add(5, 7);
```

14. Arrays in Java

An array is a collection of variables of the same type under a single name.

```
int[] numbers = {10, 20, 30, 40};
System.out.println(numbers[0]); // Output: 10
```

15. Common Errors & Debugging

- **Syntax Error:** Forgot a semicolon or a typo.
- **Runtime Error:** Division by zero or accessing an array index that doesn't exist (e.g., trying to access `arr[10]` when size is 5).
- **Logic Error:** The program runs, but the result is wrong (common in math formulas).

16. Important Viva Questions & Answers

1. **What is the JVM?** It is the Java Virtual Machine. It loads, verifies, and executes Java code. It makes Java platform-independent.
2. **What is JDK?** Java Development Kit. It contains the JRE, compiler (javac), and debugger.
3. **Why is main() static?** Because the main method is called by the JVM before any objects are created.
4. **What is the difference between Array and ArrayList?** An Array has a fixed size; an ArrayList is dynamic and resizable.

17. Beginner Practical Programs

1. Even or Odd

```
Scanner sc = new Scanner(System.in);
int num = sc.nextInt();
if(num % 2 == 0) System.out.println("Even");
else System.out.println("Odd");
```

2. Multiplication Table

```
int n = 5;
for(int i = 1; i <= 10; i++)
    System.out.println(n + " * " + i + " = " + (n * i));
```

3. Largest of Three Numbers

```
int a = 10, b = 25, c = 15;
if (a >= b && a >= c) System.out.println(a);
else if (b >= a && b >= c) System.out.println(b);
else System.out.println(c);
```

4. Factorial Calculation

```
int n = 5, fact = 1;
for(int i = 1; i <= n; i++) fact *= i;
System.out.println("Factorial: " + fact);
```

5. Check Prime Number

```
int num = 7;
boolean isPrime = true;
for(int i = 2; i <= num/2; i++) {
    if(num % i == 0) isPrime = false;
}
```

6. Swap Two Numbers

```
int a = 5, b = 10;
int temp = a;
a = b;
b = temp;
```

7. Reverse a String

```
String str = "Hello";
String rev = "";
for(int i = str.length()-1; i>=0; i--) rev += str.charAt(i);
```

8. Check Palindrome

```
// Compare original string with reversed string
if(str.equals(rev)) System.out.println("Palindrome");
```

9. Fibonacci Series

```
int t1 = 0, t2 = 1;
for(int i = 1; i <= 5; i++) {
    System.out.print(t1 + " ");
    int sum = t1 + t2;
    t1 = t2; t2 = sum;
}
```

10. Power of a Number (using math library)

```
double result = Math.pow(2, 3);
System.out.println(result);
```

18. Summary Revision Notes

Remember these key takeaways for your exams and projects:

- Java programs start with `public class ClassName`.
- Always use `Scanner` for user input.
- Learn the difference between `=` (assignment) and `==` (comparison).
- Methods are excellent for keeping code simple.
- Programming is about practice. Type out your code, don't just copy-paste!