Java Programming Training Module

A Comprehensive Guide for Beginners

Prepared by xAI Training Team

May 07, 2025

Contents

1 Introduction to Java

Java is a versatile, object-oriented programming language designed for portability across platforms. This training module introduces Java fundamentals, providing hands-on examples to build programming skills.

1.1 Why Learn Java?

- **Portability**: Runs on any device with a Java Virtual Machine (JVM).
- **Versatility**: Used in web, mobile, and enterprise applications.
- **Community**: Extensive libraries and frameworks.

1.2 Setting Up the Environment

Install the Java Development Kit (JDK) from Oracle's website. Use an IDE like IntelliJ IDEA or Eclipse for coding. Verify installation with:

```
java -version
```

2 Basic Java Syntax

Java programs are structured around classes and methods. Below is a simple "Hello, World!" program.

2.1 First Java Program

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

Explanation:

- `public class HelloWorld`: Defines a class named `HelloWorld`.
- `public static void main`: Entry point of the program.
- `System.out.println`: Prints text to the console.

2.2 Variables and Data Types

Java supports primitive data types like `int`, `double`, `boolean`, and reference types like `String`.

```java
public class VariablesExample {
    public static void main(String[] args) {
        int age = 25; // Integer
        double salary = 50000.50; // Floating-point
        String name = "Alice"; // String
        boolean isEmployed = true; // Boolean
        System.out.println(name + " is " + age + " years old.");
    }
}
```

3 Control Structures

Control structures manage the flow of a program.

3.1 Conditional Statements

Use `if-else` for decision-making.

```java
public class GradeCalculator {
    public static void main(String[] args) {
        int score = 85;
        if (score >= 90) {
            System.out.println("Grade: A");
        } else if (score >= 80) {
            System.out.println("Grade: B");
        } else {
            System.out.println("Grade: C");
        }
    }
}
```

3.2 Loops

Loops repeat code execution. Below is a `for` loop example.

```java
public class NumberPrinter {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Number: " + i);
        }
    }
}
```

4 Object-Oriented Programming

Java is built around OOP principles: encapsulation, inheritance, and polymorphism.

4.1 Classes and Objects

A class is a blueprint for objects.

```java
public class Car {
    String model;
    int year;

    public Car(String model, int year) {
        this.model = model;
        this.year = year;
    }

    public void displayInfo() {
        System.out.println("Model: " + model + ", Year: " + year);
    }

    public static void main(String[] args) {
        Car myCar = new Car("Toyota Camry", 2020);
        myCar.displayInfo();
    }
}
```

4.2 Inheritance

Classes can inherit properties and methods.

```java
class Vehicle {
    String brand;
    Vehicle(String brand) {
        this.brand = brand;
    }
    void honk() {
        System.out.println("Beep!");
    }
}

class Car extends Vehicle {
    int wheels;
    Car(String brand, int wheels) {
        super(brand);
        this.wheels = wheels;
    }
}

public class InheritanceExample {
    public static void main(String[] args) {
        Car myCar = new Car("Ford", 4);
        myCar.honk();
        System.out.println("Brand: " + myCar.brand);
    }
}
```

5 Exception Handling

Handle errors gracefully using `try-catch`.

```java
public class ExceptionExample {
    public static void main(String[] args) {
        try {
            int[] numbers = {1, 2, 3};
            System.out.println(numbers[5]); // Out-of-bounds
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Invalid index!");
        }
    }
}
```

6 Working with Collections

The Java Collections Framework simplifies data management.

6.1 ArrayList Example

```java
import java.util.ArrayList;

public class ArrayListExample {
    public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<>();
        names.add("Alice");
        names.add("Bob");
        for (String name : names) {
            System.out.println("Name: " + name);
        }
    }
}
```

7 Conclusion

This module covers Java fundamentals, from syntax to OOP and collections. Practice these examples and explore advanced topics like multithreading and networking.

8 References

- Oracle Java Documentation: `https://docs.oracle.com/javase`

- Java Tutorials by Oracle: `https://docs.oracle.com/javase/tutorial`