Java Programming Training Module

A Comprehensive Guide for Beginners

Prepared by xAI Training Team May 07, 2025

Downloadable PDF version available for training purposes.

**Contents**

1. [Introduction to Java](#_bookmark0) 2
   1. [Why Learn Java?](#_bookmark1) 2
   2. [Setting Up the Environment](#_bookmark2) 2
2. [Basic Java Syntax](#_bookmark3) 2
   1. [First Java Program](#_bookmark4) 2
   2. [Variables and Data Types](#_bookmark5) 2
3. [Control Structures](#_bookmark6) 3
   1. [Conditional Statements](#_bookmark7) 3
   2. [Loops](#_bookmark8) 3
4. [Object-Oriented Programming](#_bookmark9) 3
   1. [Classes and Objects](#_bookmark10) 4
   2. [Inheritance](#_bookmark11) 4
5. [Exception Handling](#_bookmark12) 5
6. [Working with Collections](#_bookmark13) 5
   1. [ArrayList Example](#_bookmark14) 5
7. [Conclusion](#_bookmark15) 5
8. [References](#_bookmark16) 5

1

# Introduction to Java

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

## 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.

## 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

1

# Basic Java Syntax

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

## First Java Program

1

**public class** HelloWorld {

**public static void** main(String[] args) { System.out.println(”Hello, World!”);

}

}

2

3

4

5

**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.

## Variables and Data Types

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

1

**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.”);

}

}

2

3

4

5

6

7

8

9

# Control Structures

Control structures manage the flow of a program.

## Conditional Statements

Use if-else for decision-making.

**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”);

}

}

}

1

2

3

4

5

6

7

8

9

10

11

12

## Loops

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

**public class** NumberPrinter {

**public static void** main(String[] args) {

**for** (**int** i = 1; i <= 5; i++) { System.out.println(”Number: ” + i);

}

}

}

1

2

3

4

5

6

7

# Object-Oriented Programming

Java is built around OOP principles: encapsulation, inheritance, and polymor- phism.

## Classes and Objects

A class is a blueprint for objects.

**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();

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

## Inheritance

Classes can inherit properties and methods.

**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);

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

# Exception Handling

Handle errors gracefully using try-catch.

**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!”);

}

}

}

1

2

3

4

5

6

7

8

9

10

# Working with Collections

The Java Collections Framework simplifies data management.

## ArrayList Example

1

**import** java.util.ArrayList;

**public class** ArrayListExample {

}

2

3

|  |  |  |
| --- | --- | --- |
| 4 | **public static void** main(String[] args) { | |
| 5 |  | ArrayList<String> names = **new** ArrayList<>(); |
| 6 |  | names.add(”Alice”); |
| 7 |  | names.add(”Bob”); |
| 8 |  | **for** (String name : names) { |
| 9 |  | System.out.println(”Name: ” + name); |
| 10 |  | } |
| 11 | } |  |

12

# Conclusion

This module covers Java fundamentals, from syntax to OOP and collections. Prac- tice these examples and explore advanced topics like multithreading and net- working.

# References

* Oracle Java Documentation: <https://docs.oracle.com/javase>
* Java Tutorials by Oracle: <https://docs.oracle.com/javase/tutorial>