

Table of Contents



- [1. Introduction to Java](#)
- [2. Setup and Running Java](#)
- [3. Java Program Structure](#)
- [4. Variables and Data Types](#)
 - [Primitive Data Types](#)
 - [Non-Primitive Types](#)
- [5. Control Structures](#)
 - [1. Conditional Statements](#)
 - [2. Switch Statement](#)
 - [3. Loops](#)
- [6. Arrays](#)
- [7. Methods \(Functions\)](#)
- [8. Object-Oriented Programming \(OOP\)](#)
 - [1. Class and Object](#)
 - [2. Inheritance](#)
 - [3. Polymorphism](#)
 - [4. Encapsulation](#)
- [9. Exception Handling](#)
- [10. File I/O](#)
- [Useful Commands and Shortcuts](#)
 - [Tips for Java Development](#)

1. Introduction to Java

- Java is a high-level, object-oriented programming language.
 - Known for Write Once, Run Anywhere (WORA) – Java code can run on any platform with a Java Virtual Machine (JVM).
 - Used for web development, mobile apps (Android), desktop applications, and enterprise systems.
-

2. Setup and Running Java

1. Install Java Development Kit (JDK):
 - Download from [Oracle](#).
2. Verify Installation:

```
java -version
javac -version
```

3. Run a Java Program:

```
javac HelloWorld.java # Compile
java HelloWorld       # Run
```

3. Java Program Structure

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

- public class HelloWorld – Defines a class.
- public static void main(String[] args) – Main method, program entry point.
- System.out.println – Prints to the console.

4. Variables and Data Types

Primitive Data Types

Type	Size	Example
byte	8-bit	byte b = 100;
short	16-bit	short s = 32000;
int	32-bit	int i = 12345;

Type	Size	Example
long	64-bit	long l = 123456789L;
float	32-bit	float f = 5.75f;
double	64-bit	double d = 19.99;
char	16-bit	char c = 'A';
boolean	1-bit	boolean isTrue = true;

Non-Primitive Types

- String: String name = "Java";
 - Array: int[] arr = {1, 2, 3};
 - Object: MyClass obj = new MyClass();
-

5. Control Structures

1. Conditional Statements

```
int x = 10;
if (x > 0) {
    System.out.println("Positive");
} else if (x == 0) {
    System.out.println("Zero");
} else {
    System.out.println("Negative");
}
```

2. Switch Statement

```
int day = 3;
switch (day) {
    case 1 -> System.out.println("Monday");
    case 2 -> System.out.println("Tuesday");
    default -> System.out.println("Other Day");
}
```

3. Loops

For Loop:

```
for (int i = 1; i <= 5; i++) {  
    System.out.println(i);  
}
```

While Loop:

```
int i = 1;  
while (i <= 5) {  
    System.out.println(i);  
    i++;  
}
```

Do-While Loop:

```
int i = 1;  
do {  
    System.out.println(i);  
    i++;  
} while (i <= 5);
```

6. Arrays

```
int[] numbers = {10, 20, 30, 40};  
System.out.println(numbers[0]); // Access element  
numbers[1] = 25;                // Modify element
```

- Array Length: numbers.length
- Iterate Array:

```
for (int num : numbers) {  
    System.out.println(num);  
}
```

7. Methods (Functions)

```
public class Calculator {  
    public static int add(int a, int b) {  
        return a + b;  
    }  
  
    public static void main(String[] args) {  
        int result = add(5, 3);  
        System.out.println(result);  
    }  
}
```

- add – Custom method.
- Return Type: int (returns integer).
- void – No return value.

8. Object-Oriented Programming (OOP)

1. Class and Object

```
public class Car {  
    String brand;  
    int speed;  
  
    // Constructor  
    public Car(String brand, int speed) {  
        this.brand = brand;  
        this.speed = speed;  
    }  
  
    public void drive() {  
        System.out.println(brand + " is driving at " + speed + "  
km/h");  
    }  
}
```

```
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Car myCar = new Car("Toyota", 120);  
        myCar.drive();  
    }  
}
```

2. Inheritance

```
class Animal {  
    void sound() {  
        System.out.println("Animal makes a sound");  
    }  
}  
  
class Dog extends Animal {  
    void sound() {  
        System.out.println("Dog barks");  
    }  
}
```

3. Polymorphism

```
Animal myDog = new Dog();  
myDog.sound(); // Dog barks
```

4. Encapsulation

```
public class Person {  
    private String name;
```

```
public String getName() {  
    return name;  
}  
  
public void setName(String newName) {  
    this.name = newName;  
}  
}
```

9. Exception Handling

```
try {  
    int result = 10 / 0;  
} catch (ArithmeticException e) {  
    System.out.println("Cannot divide by zero");  
} finally {  
    System.out.println("End of try-catch");  
}
```

10. File I/O

```
import java.io.*;  
  
public class FileExample {  
    public static void main(String[] args) {  
        try {  
            FileWriter writer = new FileWriter("example.txt");  
            writer.write("Hello Java");  
            writer.close();  
        } catch (IOException e) {  
            System.out.println("Error occurred");  
        }  
    }  
}
```

Useful Commands and Shortcuts

Command	Description
<code>javac FileName.java</code>	Compile Java code
<code>java FileName</code>	Run compiled code
<code>System.out.println()</code>	Print output
<code>Ctrl + Shift + /</code>	Multi-line comment
<code>Ctrl + Space</code>	Auto-complete

Tips for Java Development

- Practice OOP Concepts – Master classes, inheritance, and polymorphism.
- Work with APIs – Java has vast libraries like Spring and Hibernate.
- Understand JVM – Knowing how the JVM works will improve debugging and performance tuning.