# JAVA PROGRAMMING

## JAVA :

Java is a high-level, object-oriented programming language that was first developed by Sun Microsystems in the mid-1990s and is now owned by Oracle Corporation. It is one of the most widely used programming languages in the world, known for its portability, scalability, and robustness.

- **Object-Oriented**: Java is an object-oriented language, which means it is based on concepts such as classes, objects, inheritance, polymorphism, encapsulation, and abstraction.
- **Syntax**: Java's syntax is largely influenced by C and C++, which makes it familiar to developers with experience in those languages. The syntax emphasizes clarity and simplicity, making it relatively easy to learn and use.
- **Memory Management**: Java handles memory management automatically through garbage collection, which helps manage memory by cleaning up unused objects.

### **Applications of Java**:

→ **Web development** (building websites and web apps)

→ **Mobile apps** (especially Android apps)

→ **Enterprise software** (large business applications)

→ **Embedded systems** (software for devices like TVs and smart gadgets)

→**Big data processing** (handling large data sets)

<u>Here's a simple example of a Java program that prints "Hello, World!"</u>

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

**Output:**

```
Hello, World!
```

**Explanation:**

- public class HelloWorld: This defines a class named HelloWorld. In Java, every application must have at least one class.
- public static void main(String[] args): This is the entry point of the program. The main method is where the program starts execution.
- System.out.println("Hello, World!"); : This prints the text Hello, World! to the console.

## Comments in Java:

There are three types of comments in Java:

1. **Single-Line Comment:**

   A single-line comment starts with //. Everything after // on that line is considered a comment.

   Example:

```
// This is a single-line comment
int number = 10;  // This is an inline comment
```

2. **Multi-Line Comment:**

   A multi-line comment starts with /* and ends with */. It can span multiple lines.

   Example:

```
  /*
 This is a multi-line comment.
 It can span several lines.
 Useful for commenting out large sections of code or providing detailed explanations.
*/
int x = 5;
```

3. **Javadoc Comment:**

   Javadoc comments start with /** and end with */. These are special comments used to generate API documentation for your Java code. They are often used to describe classes, methods, or fields.

   Example:

```
 /**
* This is a Javadoc comment.
```

```
 * It is used to describe the purpose of a class or method.
 *
 * @param x The input value
 * @return The square of the input
 */
public int square(int x) {
   return x * x;
}
```

## Basic Topics in Java:

### 1. Variables and Data Types

- Variables store data that can be used later.
- Data Types define the type of data a variable can hold (e.g., int, String, boolean, double).

  Java has two categories of data types:

  **→Primitive Data Types**

  1. byte: 1 byte, integer
  2. short: 2 bytes, integer
  3. int: 4 bytes, integer
  4. long: 8 bytes, integer
  5. float: 4 bytes, floating-point
  6. double: 8 bytes, floating-point
  7. char: 2 bytes, single character
  8. boolean: 1 bit, true or false

  **→Non-Primitive Data Types (Reference Data Types)**

  1. String: Sequence of characters
  2. Arrays: Collections of elements of the same type
  3. Classes and Interfaces: Custom data types that define objects and behavior.

Examples:

```
int age = 25;          // Integer data type

double price = 19.99;    // Double (floating-point) data type

String name = "John";    // String data type
boolean isActive = true; // Boolean data type
```

### 2. **Control Flow Statements:**

- **if, else if, else**: Used for conditional checks.
- **switch**: Used for multiple choices based on a value.
- **while, for, do-while**: Looping structures for repeating code.
-

## 3.Methods:

Methods (also called functions) are blocks of code that perform a specific task and can be called from other parts of the program.Methods are blocks of code that perform a specific task, and they are used to execute actions or compute values. A method is defined within a class and can be called to perform its operation whenever needed.

→ **Method Declaration**: A method is declared with a specific signature, including a return type, method name, and parameters.

→ **Method Call**: Once a method is defined, you can invoke it (call it) from other parts of your program.

<u>Example:</u>

```
public static void greet(String name) {
   System.out.println("Hello, " + name);
}
// Calling the method
greet("Alice"); // Prints "Hello, Alice"
```

## 4. Arrays:

An array in Java is a container that holds a fixed number of values of the same type. It allows you to store multiple values in a single variable, making it easier to manage and process related data.

Example :

```
int[] numbers = {1, 2, 3, 4, 5};  // Array of integers
System.out.println(numbers[2]);
```

**<u>Output:</u>** 3

## 5. Object-Oriented Programming (OOP) Concepts:

- **Classes**: Templates for creating objects.
- **Objects**: Instances of classes.
- **Inheritance**: A mechanism to create a new class based on an existing class.
- **Polymorphism**: Ability to take many forms (method overloading/overriding).
- **Encapsulation**: Wrapping data and methods into a single unit (class).
- **Abstraction**: Hiding complex details and showing only the necessary parts.

Example:

```
class Person {
   String name;
   int age;
   // Constructor
   public Person(String name, int age) {
      this.name = name;
      this.age = age;
   }
   // Method
   public void introduce() {
      System.out.println("My name is " + name + " and I am " + age + " years old.");
   }
}
// Creating an object of the Person class
Person p1 = new Person("Alice", 30);
p1.introduce();  // Prints "My name is Alice and I am 30 years old."
```

## 6. String Manipulation:

- **Strings** are used to handle text. Java provides various methods for manipulating strings, such as concatenation, comparison, and substring extraction.
  **Example:**

  ```
  String message = "Hello, World!";
  System.out.println(message.length());      // 13
  System.out.println(message.toUpperCase());  // "HELLO, WORLD!"
  ```

```
System.out.println(message.substring(7));   // "World!"
```

## 7. Exception Handling:

- **Exceptions** are errors that occur during the execution of a program. Java provides mechanisms like try, catch, and finally to handle exceptions.

Example:

```
try {
   int result = 10 / 0;  // Division by zero
} catch (ArithmeticException e) {
   System.out.println("Error: " + e.getMessage());
} finally {
   System.out.println("This block runs no matter what.");
}
```

## 8. Java Collections Framework:

- **Collections** are groups of objects (like lists, sets, and maps) that provide useful data structures.
- Common classes include ArrayList, HashSet, HashMap, and LinkedList.
  **Example:**

```
ArrayList<String> names = new ArrayList<>();
names.add("Alice");
names.add("Bob");
System.out.println(names.get(0));  // Prints "Alice"
```

## 9. File I/O:

- Java provides classes for reading and writing data from files using streams (e.g., FileReader, BufferedReader, FileWriter).

## 10. Java Packages:

- **Packages** are used to group related classes and avoid name conflicts.

# Basic programs on Java

1. **Hello, World! :-**

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

**2. Sum of Two Numbers**

Problem: Write a program that takes two numbers as input and prints their sum.

```
import java.util.Scanner;
public class SumOfTwoNumbers {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter first number: ");
      int num1 = scanner.nextInt();

      System.out.print("Enter second number: ");
      int num2 = scanner.nextInt();

      int sum = num1 + num2;
      System.out.println("Sum: " + sum);
   }
}
```

**3. Even or Odd Number**

Problem: Write a program that checks whether a given number is even or odd.

```
import java.util.Scanner;

public class EvenOdd {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter a number: ");
      int num = scanner.nextInt();
```

```
        if (num % 2 == 0) {
            System.out.println(num + " is even.");
        } else {
            System.out.println(num + " is odd.");
        }
    }
}
```

**4. Find the Largest of Two Numbers**

**Problem**: Write a program to find the largest of two numbers.

```
import java.util.Scanner;
public class LargestOfTwo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter first number: ");
        int num1 = scanner.nextInt();

        System.out.print("Enter second number: ");
        int num2 = scanner.nextInt();

        if (num1 > num2) {
            System.out.println("Largest number is: " + num1);
        } else {
            System.out.println("Largest number is: " + num2);
        }
    }
}
```

**5. Check if a Number is Prime**

**Problem**: Write a program to check if a given number is prime or not.

```
 import java.util.Scanner;
public class PrimeNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        boolean isPrime = true;
        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                isPrime = false;
                break;
            }
        }
        if (isPrime && num > 1) {
            System.out.println(num + " is a prime number.");
        } else {
            System.out.println(num + " is not a prime number.");
        }
    }
}
```

**6. Factorial of a Number**

**Problem**: Write a program to find the factorial of a number.

```
import java.util.Scanner;
    public class Factorial {
        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter a number: ");
            int num = scanner.nextInt();
            long factorial = 1;

            for (int i = 1; i <= num; i++) {
                factorial *= i;
            }
            System.out.println("Factorial of " + num + " is: " + factorial);
        }
    }
```

**7. Fibonacci Sequence**

**Problem**: Write a program to print the Fibonacci sequence up to n numbers.

```
 import java.util.Scanner;
public class Fibonacci {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of terms: ");
        int n = scanner.nextInt();
        int first = 0, second = 1;
        System.out.print("Fibonacci Series: " + first + " " + second);
        for (int i = 3; i <= n; i++) {
            int next = first + second;
            System.out.print(" " + next);
            first = second;
            second = next;
        }
    }
}
```

**8. Reverse a Number**

**Problem**: Write a program to reverse a given number.

```
 import java.util.Scanner;
public class ReverseNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        int reversed = 0;
        while (num != 0) {
            int digit = num % 10;
            reversed = reversed * 10 + digit;
            num /= 10;
        }
        System.out.println("Reversed number: " + reversed);
    }
}
```

**9. Count the Number of Digits in a Number**

**Problem**: Write a program to count the number of digits in a given number.

```
 import java.util.Scanner;
public class CountDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        int count = 0;

        while (num != 0) {
            num /= 10;
            count++;
        }
        System.out.println("Number of digits: " + count);
    }
}
```

**10. Sum of Digits in a Number**

**Problem**: Write a program to calculate the sum of digits in a given number.

```
import java.util.Scanner;
public class SumOfDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        int sum = 0;

        while (num != 0) {
            sum += num % 10;  // Add last digit to sum
            num /= 10;        // Remove last digit
        }
        System.out.println("Sum of digits: " + sum);
    }
}
```

**Note***:* **These programs are provided for reference purposes only. There may be some mistakes or minor issues in the code. Please ensure that you execute the programs in a compiler or an IDE to check for correctness and to make any necessary adjustments based on your specific requirements.**