# Java Programming Tutorial

@ Coding Scape

July 27, 2024

# 1 Introduction to Java

## 1.1 Definition

Java is a high-level, class-based, object-oriented programming language designed to have as few implementation dependencies as possible. It is a general-purpose language used for developing a wide range of applications, from web to mobile and desktop applications.

## 1.2 Uses

Java is widely used for:

- Web applications
- Mobile applications (Android)
- Desktop applications
- Enterprise-level applications
- Scientific applications

# 2 Basic Syntax

## 2.1 Definition

The basic syntax of Java is the set of rules that defines the structure of Java programs. It includes the way keywords, identifiers, and other elements are used to construct valid Java statements and expressions.

## 2.2 Uses

Understanding Java syntax is fundamental to writing correct and efficient Java code.

### 2.3 Example 1

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

### 2.4 Example 2

```
public class BasicMath {
    public static void main(String[] args) {
        int a = 5;
        int b = 10;
        int sum = a + b;
        System.out.println("Sum: " + sum);
    }
}
```

### 2.5 Unsolved Question

Write a Java program to calculate the area of a rectangle. The program should prompt the user for the length and width and then display the area.

## 3 Variables and Data Types

### 3.1 Definition

Variables in Java are used to store data values. Each variable in Java must be declared with a data type, which determines the type of data it can hold.

### 3.2 Uses

Variables and data types are crucial for data manipulation and storage in Java programs.

### 3.3 Example 1

```
public class VariableExample {
    public static void main(String[] args) {
        int age = 25;
        double salary = 55000.50;
        char grade = 'A';
        boolean isEmployed = true;

        System.out.println("Age: " + age);
        System.out.println("Salary: " + salary);
        System.out.println("Grade: " + grade);
        System.out.println("Employed: " + isEmployed);
    }
}
```

### 3.4 Example 2

```
public class DataTypesExample {
    public static void main(String[] args) {
        byte smallNumber = 10;
        short mediumNumber = 1000;
        int largeNumber = 100000;
        long veryLargeNumber = 1000000000L;

        System.out.println("Byte: " + smallNumber);
        System.out.println("Short: " + mediumNumber);
        System.out.println("Int: " + largeNumber);
        System.out.println("Long: " + veryLargeNumber);
    }
}
```

### 3.5 Unsolved Question

Create a Java program that defines a variable of each primitive data type (byte, short, int, long, float, double, char, boolean) and prints their values.

## 4 Control Flow Statements

### 4.1 Definition

Control flow statements in Java control the flow of execution in a program based on conditions. These include if statements, switch statements, loops (for, while, do-while).

### 4.2 Uses

Control flow statements are essential for making decisions and repeating actions in a program.

### 4.3 Example 1

```
public class IfElseExample {
    public static void main(String[] args) {
        int number = 10;
        if (number > 0) {
            System.out.println("Number is positive.");
        } else if (number < 0) {
            System.out.println("Number is negative.");
        } else {
            System.out.println("Number is zero.");
        }
    }
}
```

### 4.4 Example 2

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

### 4.5 Unsolved Question

Write a Java program that prints the multiplication table of a given number using a for loop.

## 5 Methods

### 5.1 Definition

Methods in Java are blocks of code that perform a specific task. They help in code reusability and organization.

### 5.2 Uses

Methods are used to execute code in a modular way, which enhances readability and maintainability.

### 5.3 Example 1

```
public class MethodExample {
    public static void main(String[] args) {
        greetUser("Alice");
    }

    public static void greetUser(String name) {
        System.out.println("Hello, " + name + "!");
    }
}
```

### 5.4 Example 2

```
public class Calculator {
    public static void main(String[] args) {
        int result = add(5, 10);
        System.out.println("Sum: " + result);
    }

    public static int add(int a, int b) {
        return a + b;
    }
}
```

### 5.5 Unsolved Question

Write a Java method that takes an integer as input and returns whether it is a prime number or not.

## 6 Classes and Objects

### 6.1 Definition

Java is an object-oriented language, and classes and objects are fundamental to this paradigm. A class is a blueprint for creating objects, and an object is an instance of a class.

### 6.2 Uses

Classes and objects are used to model real-world entities and their interactions.

### 6.3 Example 1

```
public class Person {
    String name;
    int age;

    public void introduce() {
        System.out.println("Hello, my name is " + name + " and I am " + age + " years ol
    }
}

public class Main {
    public static void main(String[] args) {
        Person person = new Person();
        person.name = "John";
        person.age = 30;
        person.introduce();
    }
}
```

### 6.4 Example 2

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

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

public class Main {
    public static void main(String[] args) {
        Car car = new Car();
        car.model = "Toyota";
        car.year = 2020;
        car.displayInfo();
    }
}
```

### 6.5 Unsolved Question

Create a Java class called ‘Book‘ with attributes ‘title‘ and ‘author‘. Write a method to display the book’s details and a main method to create and display a ‘Book‘ object.

## 7 Inheritance

### 7.1 Definition

Inheritance is a mechanism in Java where one class acquires the properties and behaviors of another class. It promotes code reuse and establishes a natural hierarchy between classes.

### 7.2 Uses

Inheritance is used to create a new class based on an existing class, enhancing code reusability and organization.

## 7.3 Example 1

```
public class Animal {
    public void makeSound() {
        System.out.println("Animal sound");
    }
}

public class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Woof");
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.makeSound();
    }
}
```

## 7.4 Example 2

```
public class Shape {
    public void draw() {
        System.out.println("Drawing shape");
    }
}

public class Circle extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing circle");
    }
}

public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle();
        circle.draw();
    }
}
```

### 7.5 Unsolved Question

Create a base class ‘Employee‘ with attributes ‘name‘ and ‘salary‘. Create a derived class ‘Manager‘ that inherits from ‘Employee‘ and adds an attribute ‘department‘. Implement a method to display the manager’s details.

# 8 Polymorphism

## 8.1 Definition

Polymorphism in Java allows objects to be treated as instances of their parent class rather than their actual class. It enables one interface to be used for a general class of actions.

## 8.2 Uses

Polymorphism enhances flexibility and reusability by allowing objects to be manipulated in a general way.

## 8.3 Example 1

```
public class Animal {
    public void makeSound() {
        System.out.println("Animal sound");
    }
}

public class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Meow");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal myCat = new Cat();
        myCat.makeSound();
    }
}
```

## 8.4 Example 2

```
public class Shape {
    public void draw() {
        System.out.println("Drawing shape");
    }
}

public class Rectangle extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing rectangle");
    }
}

public class Main {
    public static void main(String[] args) {
        Shape shape = new Rectangle();
        shape.draw();
    }
}
```

### 8.5 Unsolved Question

Create a base class ‘Vehicle‘ with a method ‘start()‘. Create subclasses ‘Car‘ and ‘Bike‘ that override the ‘start()‘ method. Demonstrate polymorphism by calling the ‘start()‘ method on an array of ‘Vehicle‘ objects.

# 9 Exception Handling

### 9.1 Definition

Exception handling in Java is a powerful mechanism to handle runtime errors and manage exceptions gracefully. It involves using try, catch, and finally blocks to handle exceptions.

### 9.2 Uses

Exception handling is used to manage and recover from runtime errors, ensuring that the program can continue to run or fail gracefully.

### 9.3 Example 1

```
public class ExceptionExample {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero!");
        } finally {
            System.out.println("Execution finished.");
        }
    }
}
```

### 9.4 Example 2

```
public class ArrayException {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3};
        try {
            System.out.println(numbers[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index out of bounds!");
        }
    }
}
```

### 9.5 Unsolved Question

Write a Java program that handles a ‘NumberFormatException‘ when converting a string to an integer. Ensure the program handles the exception gracefully and provides a user-friendly message.