Java Programming Basics

Nishat Tasnim

December 2023

1 Introduction

Welcome to the exciting realm of Java programming!

Within these pages, our goal is to furnish you with a thorough understanding of the fundamental concepts and distinctive features that define the Java programming language.

As you delve into this document, you'll gain insights into the core elements that make Java a powerful and versatile programming tool.

Whether you are a novice or an experienced developer, this comprehensive overview will serve as a valuable resource on your journey through Java.

Explore the intricacies of Java and unlock the potential to create robust and sophisticated applications that propel you to new heights in the world of programming.

2 Data Types

Java offers a diverse range of data types to handle various kinds of values.

We categorize them into two primary groups:

Primitive data types and Non-primitive data types.

2.1 Primitive Data Types

Primitive data types serve as the fundamental building blocks for representing simple values. Two commonly used primitive data types are:

2.1.1 Integer

Integers represent whole numbers without any decimal points. Examples include positive integers like 1, negative integers like -5, and large numbers like 100.

2.1.2 String

Strings are sequences of characters. They play a crucial role in representing textual data, such as names, sentences, and alphanumeric combinations.

2.2 Floating-point

Floating-point numbers are used to represent real numbers with decimal points. Examples include constants like 3.14, negative values like -0.5, and whole numbers with decimal points like 2.0.

2.3 Boolean

The Boolean data type represents true or false values. It is frequently employed in conditions and logical expressions to control the flow of a program.

2.4 Character

The character data type is used to represent individual characters, such as letters, digits, or special symbols like '%'.

2.5 Non-Primitive Data Types

Non-primitive data types, also known as reference types, are more sophisticated and can represent a combination of values or objects.

2.5.1 Array

Arrays enable the storage of multiple values of the same type in a single variable, providing a convenient way to manage and manipulate data.

2.5.2 Class

Classes serve as the foundational units of object-oriented programming in Java. They encapsulate both data and behavior, facilitating modular and reusable code.

3 Operators

Operators in Java are symbolic entities that perform operations on variables or values, allowing for the manipulation of data and the execution of calculations.

3.1 Arithmetic Operators

Arithmetic operators carry out fundamental mathematical operations, including addition, subtraction, multiplication, and division.

3.2 Comparison Operators

Comparison operators facilitate the comparison of values, yielding a Boolean result that indicates whether the comparison is true or false.

3.3 Logical Operators

Logical operators perform logical operations such as AND, OR, and NOT. They are integral in constructing conditional statements and controlling program flow.

4 Control Flow

Control flow statements govern the sequence of execution in a Java program.

4.1 if-else Statements

The if-else statement provides a mechanism for making decisions in your code based on specified conditions, allowing for branching and alternate execution paths.

4.2 Switch Statements

Switch statements offer an alternative approach to decision-making, relying on the value of an expression to determine the appropriate course of action.

5 Loops

Loops facilitate the repeated execution of a block of code, enabling efficient handling of tasks that involve iteration.

5.1 for Loop

The for loop is utilized to iterate over a specified range of values, making it a powerful construct for repetitive tasks.

5.2 while Loop

The while loop repeats a block of code as long as a specified condition remains true, providing flexibility in handling varying situations.

5.3 do-while Loop

The do-while loop is similar to the while loop but ensures that the block of code is executed at least once, offering specific advantages in certain scenarios.

6 Methods

Methods in Java encapsulate a set of instructions, promoting code modularity, reusability, and maintainability.

6.1 Defining Methods

Learn how to define methods to encapsulate functionality and promote code organization.

6.2 Calling Methods

Understand the process of calling methods, allowing you to leverage existing functionality in your programs.

6.3 Method Overloading

Explore the concept of method overloading, enabling the definition of multiple methods with the same name but different parameters.

7 Object-Oriented Programming (OOP)

Object-oriented programming is a paradigm that organizes code around objects, fostering modularity, encapsulation, inheritance, and polymorphism.

7.1 Classes and Objects

Understand the concept of classes and objects as the core building blocks in object-oriented programming.

7.2 Inheritance

Explore inheritance, a mechanism that allows a class to inherit properties and behaviors from another class, promoting code reuse.

7.3 Polymorphism

Learn about polymorphism, which allows objects to be treated as instances of their parent class, facilitating flexibility and extensibility.

7.4 Encapsulation

Discover the benefits of encapsulation in OOP, where data and methods are bundled together, providing better control and security.