

1. Features of Java

Java is a high-level, object-oriented programming language developed by Sun Microsystems. One of the most important features of Java is platform independence, which means Java programs can run on any operating system using the Java Virtual Machine (JVM). Java follows the principle of "Write Once, Run Anywhere".

Java is object-oriented, which allows programmers to create modular, reusable, and secure code using concepts such as classes, objects, inheritance, polymorphism, abstraction, and encapsulation. This makes Java suitable for large-scale enterprise applications.

Java is robust because it has strong memory management, automatic garbage collection, and exception handling. It also provides high security through bytecode verification, security manager, and no explicit use of pointers. Java supports multithreading, enabling efficient use of CPU resources. Due to all these features, Java is widely used in web applications, mobile applications, and enterprise systems.

2. Structure of a Java Program

The structure of a Java program consists of six sections. The first section is the documentation section, which contains comments that describe the purpose of the program. The second section is the package statement, which groups related classes into a package.

The third section is the import statement, which allows access to predefined classes. The fourth section is the interface section, which defines interfaces if required. The fifth section is the class definition, where variables and methods are declared. The final section is the main method, which acts as the entry point of the program.

Each section has its own importance and together they help in writing organized, readable, and maintainable Java programs.

3. Declaration of Variables in Java

Declaration of a variable in Java involves specifying the data type followed by the variable name. The data type defines the type of value the variable can store and the amount of memory allocated.

Java is a strongly typed language, which means every variable must be declared before use. This helps in early detection of errors and ensures type safety. Variables can be local, instance, or static depending on their scope.

Proper declaration of variables improves program clarity, reduces runtime errors, and ensures efficient memory usage in Java programs.

4. Primitive Variables vs Reference Variables

Primitive variables store actual data values such as integers, characters, and floating-point numbers. Examples include int, float, double, and char. These variables directly store values in memory.

Reference variables do not store actual values; instead, they store memory addresses of objects. Examples include objects, arrays, and strings.

Primitive variables are faster and require less memory, while reference variables provide flexibility and support object-oriented programming. Hence, both are fundamentally different in memory

allocation and usage.

5. Use of Different Data Types in Java

In every Java program, different types of data are used such as numeric data, character data, boolean data, and objects. Java provides primitive data types for simple values and non-primitive data types for complex data.

Using appropriate data types improves performance, memory efficiency, and program reliability. Java also supports user-defined data types using classes and interfaces.

Proper handling of data types plays a crucial role in building efficient and error-free Java applications.

6. Increment and Decrement Operators

Increment and decrement operators are used to increase or decrease the value of a variable by one. The increment operator (++) increases the value, while the decrement operator (--) decreases it.

These operators can be used in prefix and postfix forms. They are widely used in loops, conditional statements, and counters.

Efficient use of increment and decrement operators helps in simplifying code and improving readability.

7. Running Java Programs Using Command Prompt

Java programs can be compiled and executed using the command prompt. First, the `javac` command is used to compile the source code into bytecode. Then, the `java` command is used to run the compiled program.

This method helps programmers understand the compilation and execution process clearly. Using the command prompt is especially useful for debugging and learning purposes.

8. Nested Classes in Java

Nested classes are classes defined within another class. They help in logically grouping related classes together, improving code readability and maintainability.

Nested classes increase encapsulation and help in accessing private members of the outer class. They are commonly used in event handling and complex applications.

Thus, nested classes play an important role in organizing large Java programs.

9. Wrapper Classes in Java

A wrapper class is a class that encapsulates a primitive data type into an object. Java provides wrapper classes such as `Integer`, `Float`, `Double`, and `Character`.

Wrapper classes are useful in collections, frameworks, and APIs that work with objects instead of primitives. They also provide utility methods for data conversion.

Hence, wrapper classes bridge the gap between primitive data types and object-oriented programming in Java.