Tab 1

### **JAVA**

DAY - 01

[**What is a Program? 1**](#_nswu539u376n)

[**What is Programming? 1**](#_7uf35qt2jy6i)

[**Types of Programming Languages: 1**](#_1nhkeh8dod4z)

[**What is Java? 1**](#_7v1i7z6ksmbo)

[**Java Classifications 1**](#_w7rfjom6nkan)

[1. Based on Platforms 1](#_ae8al040heid)

[2. Based on Application Types 2](#_x5pqgv9tao7e)

[**Best Features of Java 2**](#_tjt8g0eega08)

[**What is a Keyword in Java? 2**](#_e94uam4o4yl9)

[**Examples of Keywords: 2**](#_qujmy5pg6wd2)

[**What are Identifiers in Java? 2**](#_ci3uskng8t6e)

[**Rules for Identifiers: 2**](#_icu5ngdaxak4)

[**Java Naming Conventions 3**](#_y8o6xstg79a7)

[**Data Types in Java 3**](#_jvrmuydamw3n)

[1. Primitive Data Types 3](#_8emu874xfqe6)

[2. Reference Data Types 3](#_81g6d8w5ce7k)

[**Operators in Java 4**](#_9ban7jc7w0g5)

[**Typecasting in Java 4**](#_hcmp28kt0uq1)

### 

### **What is a Program?**

A program is a set of instructions that tells a computer what to do.

### **What is Programming?**

Process of designing the computer programs is known as programming

### **Types of Programming Languages:**

1. **Low-Level Languages:** Close to machine language (e.g., Assembly).
2. **High-Level Languages:** Easy to read and write (e.g., Python, Java).
3. **Procedural Languages:** Focus on step-by-step tasks (e.g., C).
4. **Object-Oriented Languages:** Based on objects and classes (e.g., Java, C++).
5. **Scripting Languages:** Used for automation and tasks (e.g., JavaScript, Python).

### **What is Java?**

Java is a **high-level, object-oriented programming language** designed for building platform-independent applications. Created by **Sun Microsystems** in 1995, it allows developers to write code once and run it anywhere using the Java Virtual Machine (JVM).

### **Java Classifications**

#### **1. Based on Platforms**

**Java is divided into 4 main platforms:**

1. **Java SE (Standard Edition):**
   * For desktop and basic applications.
   * Includes tools like file handling, networking, and collections.
2. **Java EE (Enterprise Edition):**
   * For large business and web applications.
   * Example: Banking or e-commerce apps.
3. **Java ME (Micro Edition):**
   * For mobile devices and small gadgets.
   * Example: Apps for IoT devices.
4. **Java FX:**
   * For creating rich graphical user interfaces (GUIs).

#### **2. Based on Application Types**

1. **Standalone Applications:**
   * Desktop apps (e.g., calculator, media player).
2. **Web Applications:**
   * Apps accessed via browsers (e.g., websites, online forms).
3. **Enterprise Applications:**
   * Large-scale apps for businesses (e.g., banking systems).
4. **Mobile Applications:**
   * Apps for phones (e.g., Android apps).
5. **Distributed Applications:**
   * Apps running across multiple computers (e.g., chat systems).

### **Best Features of Java**

1. **Simple: Easy to learn and write.**
2. **Platform Independent: Runs anywhere using JVM.**
3. **Object-Oriented: Based on objects and classes.**
4. **Secure: Built-in security features.**
5. **Robust: Handles errors effectively.**
6. **Multithreading: Supports multiple tasks simultaneously.**
7. **Portable: Runs on any platform.**

### **What is a Keyword in Java?**

**A keyword is a reserved word in Java with a predefined meaning. It cannot be used as names for variables, methods, or classes.**

### **Examples of Keywords:**

* **Data Types: int, float, boolean**
* **Control Statements: if, else, for**
* **Others: class, public, return, static**

**Example:**

**int number = 10; // 'int' is a keyword**

### **What are Identifiers in Java?**

**Identifiers are names given to elements like variables, methods, and classes in Java.**

### **Rules for Identifiers:**

1. **Start with a letter, underscore (\_), or dollar sign ($).**
2. **Can contain letters, digits, underscores, or dollar signs.**
3. **Cannot be a Java keyword.**
4. **Case-sensitive.**
5. **No spaces.**

### **Examples:**

* **Valid: age, totalAmount, MyClass**
* **Invalid: 123abc, class, my variable**

### **Java Naming Conventions**

1. **Class Names: Use PascalCase.**
   * **Example: MyClass.**
2. **Method Names: Use camelCase.**
   * **Example: calculateTotal().**
3. **Variable Names: Use camelCase.**
   * **Example: totalAmount.**
4. **Constant Names: Use UPPER\_SNAKE\_CASE.**
   * **Example: MAX\_VALUE.**
5. **Package Names: Use all lowercase.**
   * **Example: com.company.project.**
6. **Interface Names: Use PascalCase.**
   * **Example: Readable.**

### **Data Types in Java**

What type of data should be stored in the variables will be decided by datatypes.

Java has two main types of data types: **Primitive** and **Reference** types.

#### **1. Primitive Data Types**

These are the basic types built into the language:-

* **byte**: 1 byte, values from -128 to 127.
* **short**: 2 bytes, values from -32,768 to 32,767.
* **int**: 4 bytes, values from -2^31 to 2^31-1.
* **long**: 8 bytes, values from -2^63 to 2^63-1.
* **float**: 4 bytes, for decimal values ( 6-7 decimal digits).
* **double**: 8 bytes, for decimal values ( 15-16 decimal digits).
* **char**: 2 bytes, a single Unicode character (e.g., 'A').
* **boolean**: 1 bit, values true or false.

#### **2. Reference Data Types**

These refer to objects and arrays:

* **Objects**: Instances of classes (e.g., String, Scanner).
* **Arrays**: Stores multiple values of the same type.

Example of declaring variables:

int age = 25; // Primitive

String name = "John"; // Reference

### **Operators in Java**

It is a symbol that acts upon operands and response back to user with output

1. **Arithmetic Operators**:
   * +, -, \*, /, % (Addition, Subtraction, Multiplication, Division, Modulus)
2. **Relational Operators**:
   * ==, !=, >, <, >=, <= (Equal, Not equal, Greater than, Less than, etc.)
3. **Logical Operators**:
   * &&, ||, ! (AND, OR, NOT)
4. **Assignment Operators**:
   * =, +=, -=, \*=, /= (Assign and modify)
5. **Unary Operators**:
   * +, -, ++, --, ! (Unary plus, minus, increment, decrement, NOT)
6. **Ternary Operator**:
   * condition ? expr1 : expr2 (Shorthand for if-else)

Example:

int result = (x > y) ? x : y; // result = x if x > y, else y

### **Typecasting in Java**

1. **Implicit Typecasting (Widening)**
   * **Automatically converts a smaller type to a larger one (e.g., int to long).**
   * **No data loss.**
   * **int a = 10;**
   * **long b = a; // Implicit casting**
2. **Explicit Typecasting (Narrowing)**
   * **Manually converts a larger type to a smaller one (e.g., double to int).**
   * **May cause data loss.**
   * **double x = 9.99;**
   * **int y = (int) x; // Explicit casting**