**✅ 1. Java Keywords**

**Java Keywords** are reserved words that have **predefined meanings in the Java language**. These **cannot be used as variable names, class names, or identifiers**.

There are **50 keywords** in Java (as of Java SE 17), used to define **data types, control flow, exception handling, class declarations, and more.**

**🔹 Categories of Java Keywords with Use and Examples**

| **Category** | **Keywords** | **Use & Importance** | **Example** |
| --- | --- | --- | --- |
| **Data Types** | int, float, double, char, boolean, byte, short, long | Define variable types | int age = 25; |
| **Access Modifiers** | public, private, protected | Control visibility | public class MyClass {} |
| **Control Flow** | if, else, switch, case, default, while, do, for, break, continue, return | Direct program execution flow | if (x > 10) {} |
| **Class/Object Control** | class, interface, extends, implements, new, this, super, abstract, final | Define class behavior and relationships | class Car extends Vehicle {} |
| **Exception Handling** | try, catch, finally, throw, throws | Handle runtime errors safely | try { ... } catch(Exception e) {} |
| **Package & Import** | package, import | Organize and reuse code | import java.util.Scanner; |
| **OOP Related** | static, void, enum, instanceof | Used in OOP structure | static int count = 0; |
| **Threading & Synchronization** | synchronized, volatile | Multi-threading control | synchronized void run() {} |
| **Others** | boolean, const (unused), goto (unused), assert, native, transient, strictfp | Special purpose | assert x > 0; |
| **Literals/Values** | true, false, null | Boolean/null values | boolean isActive = true; |

**🔸 Example in Code:**

public class Example {

private int number = 10; // 'private' keyword for encapsulation

public static void main(String[] args) {

Example obj = new Example(); // 'new' keyword creates object

if (obj.number > 5) { // 'if' keyword for decision

System.out.println("Number is greater than 5");

}

}

}

**✅ 2. Identifiers in Java**

**Identifiers** are **names** given to various **elements** in a Java program like:

* Classes
* Variables
* Methods
* Objects
* Packages

You create **Identifiers** – they are not predefined like keywords.

**🔹 Rules for Writing Identifiers**

| **Rule** | **Explanation** | **Example** |
| --- | --- | --- |
| Must start with a **letter (A-Z/a-z)**, **underscore (\_) or dollar ($)** | Cannot start with number | int age;, String \_name; |
| Can be followed by letters, digits, underscores | int rollNo123; is valid |  |
| **Cannot use keywords** | int class; ❌ (Invalid) |  |
| Case-sensitive | RollNo and rollno are different |  |
| No spaces or special characters except \_ or $ | int first name; ❌ (Invalid) |  |

**🔹 Types of Identifiers**

| **Identifier Type** | **Example** | **Purpose** |
| --- | --- | --- |
| **Class Name** | public class Student {} | Naming the class |
| **Variable Name** | int marks = 90; | To store data |
| **Method Name** | void display() {} | To perform task |
| **Object Name** | Student s1 = new Student(); | Refers to instance of class |

**🔸 Example:**

public class Student { // 'Student' is a class identifier

int rollNo; // 'rollNo' is a variable identifier

String name; // 'name' is a variable identifier

void displayInfo() { // 'displayInfo' is a method identifier

System.out.println(rollNo + ": " + name);

}

public static void main(String[] args) {

Student s1 = new Student(); // 's1' is an object identifier

s1.rollNo = 101;

s1.name = "Omkar";

s1.displayInfo();

}

}

**✅ Summary Table**

| **Element** | **Keyword or Identifier?** | **Example** | **Notes** |
| --- | --- | --- | --- |
| public | Keyword | public class | Access modifier |
| Student | Identifier | class Student | Class name |
| main | Identifier | public static void main() | Method name |
| if | Keyword | if (x > 0) | Conditional check |
| marks | Identifier | int marks = 90; | Variable |
| new | Keyword | new Student() | Creates object |

**📘 Final Tip for Students:**

Use **meaningful identifiers** and **avoid using keywords as names** to make code readable and maintainable.