Keywords and Identifiers in Java

1. Keywords in Java

Definition:

A **keyword** is a **predefined, reserved word** in Java that has a **special meaning** to the compiler.
You **cannot use keywords** as **variable names, class names, or identifiers**.

Java keywords are **case-sensitive**, and all of them are written in **lowercase**.

```
int number = 10;
public class Demo { }
if (number > 5) {
    System.out.println("Greater than 5");
}
```

`int`, `public`, `class`, `if` are **keywords**
`Demo`, `number` are **identifiers**

🔷 Rules for Keywords

1. All keywords are **in lowercase**.
2. They are **reserved** by Java — cannot be used as identifiers.
3. Used to define syntax and structure of Java programs.
4. There are **53 keywords** in Java (as of Java 17).

📄 List of Java Keywords

Category	Keywords
Access Control	`public`, `private`, `protected`
Class, Object & Interface	`class`, `interface`, `extends`, `implements`, `new`
Flow Control	`if`, `else`, `switch`, `case`, `default`, `while`, `do`, `for`, `break`, `continue`, `return`
Exception Handling	`try`, `catch`, `finally`, `throw`, `throws`
Modifiers	`static`, `final`, `abstract`, `native`, `synchronized`, `transient`, `v` `strictfp`
Data Types	`int`, `float`, `double`, `char`, `byte`, `short`, `long`, `boolean`, `void`
Package & Imports	`package`, `import`

Inheritance & Superclass	`super`, `this`
Object Reference	`instanceof`, `new`
Others	`enum`, `assert`, `const*`, `goto*` (*not used, reserved for future use*)

2. Identifiers in Java

Definition:

An **identifier** is the **name** given to a variable, class, method, package, or interface in a Java program.

Identifiers help **uniquely identify** elements in the program.

```
class StudentRecord {  // 'StudentRecord' is a class identifier
    int age;           // 'age' is a variable identifier

    void showDetails() {  // 'showDetails' is a method identifier
        System.out.println("Age: " + age);
    }

    public static void main(String[] args) {  // 'main' and 'args' are identifiers
        StudentRecord s1 = new StudentRecord();  // 's1' is an object identifier
        s1.age = 20;
        s1.showDetails();
    }
}
```

Rule	Description	Example
1	Can contain **letters (A–Z, a–z)**, **digits (0–9)**, **underscore (_)**, and **dollar sign ($)**	`total`, `marks_1`, `$amount`
2	**Must not start with a digit**	`123name` (invalid) `name123` (valid)

3️⃣	**Cannot use keywords** as identifiers	`int class = 10;`
4️⃣	**Case-sensitive** — `Name` and `name` are different	Valid but distinct
5️⃣	**No special characters or spaces**	`first name` (invalid)
6️⃣	**Can be of any length**	But should be meaningful and readable
7️⃣	**Should follow naming conventions** (camelCase for variables, PascalCase for classes)	`studentName`, `StudentRecord`

Examples: Valid vs Invalid Identifiers

Valid Identifiers	Invalid Identifiers	Reason
`age`	`1age`	Cannot start with number
`total_marks`	`total-marks`	Hyphen not allowed
`$value`	`@rate`	@ symbol not allowed
`_count`	`class`	class' is a keyword
`StudentName`	`student name`	Space not allowed

Naming Conventions (Best Practices)

Type	Convention	Example
Class	PascalCase	`StudentDetails`, `BankAccount`
Variable	camelCase	`studentName`, `totalMarks`
Method	camelCase (verb first)	`showDetails()`, `calculateSum()`
Constant	UPPERCASE with underscores	`MAX_SPEED`, `PI_VALUE`

Feature	Keywords	Identifiers
Meaning	Reserved words with predefined meaning	Names given to variables, classes, methods, etc.
Defined By	Java language itself	Programmer
Case-Sensitive	Yes	Yes
Allowed Characters	Only lowercase predefined words	Letters, digits, _, $

Example	`public`, `class`, `static`,	`Student`, `age`, `showDetails()`
Can we change meaning?	No	Yes, we define them
Total Count	53 (Java 17)	Unlimited (user-defined)