What Are Tokens?

In Java, a **token** is the smallest unit of code that is meaningful to the compiler. The Java compiler breaks a program into tokens during compilation.

There are **five types** of tokens in Java:

1. **Keywords**
2. **Identifiers**
3. **Literals**
4. **Operators**
5. **Separators (Delimiters)**

---

1. Keywords

Definition: Keywords are reserved words in Java that have predefined meanings. They **cannot be used as identifiers** (variable names, method names, etc.).

List of Keywords in Java (53 total)

Control Statements	Data Types	Modifiers	Exception Handling	Class & Object	Other
`if`, `else`	`int`, `char`, `double`, `float`, `boolean`, `byte`, `long`, `short`	`public`, `private`, `protected`, `static`, `final`, `abstract`, `synchronized`, `transient`, `volatile`	`try`, `catch`, `finally`, `throw`, `throws`	`class`, `interface`, `extends`, `implements`, `this`, `super`	`new`, `return`, `break`, `continue`, `instanceof`, `assert`
`switch`, `case`, `default`	`void`	`strictfp`, `native`	`throws`	`enum`, `package`, `import`	`goto` (not used), `const` (not used)
`for`, `while`, `do`				`this`, `super`	

Example Usage of Keywords

```
class Example {
    public static void main(String[] args) {
        int number = 10; // 'int' is a keyword
        if (number > 5) { // 'if' is a keyword
            System.out.println("Number is greater than 5");
        }
    }
}
```

---

2. Identifiers

Definition: Identifiers are the **names** given to variables, methods, classes, and objects.

Rules for Identifiers:

✅ Can contain letters (A-Z, a-z), digits (0-9), underscore (_) and dollar sign ($).
✅ Must **start with a letter**, underscore _, or dollar sign `$`.
✅ **Cannot be a keyword.**
✅ **Case-sensitive** (`myVar` and `myvar` are different).
✅ **No special characters** (`! @ # % & *` etc. are not allowed).

Example of Identifiers

```
class MyClass { // MyClass is an identifier
    int myVariable = 10; // myVariable is an identifier

    void myMethod() { // myMethod is an identifier
        System.out.println(myVariable);
    }
}
```

---

3. Literals

Definition: A **literal** is a **fixed value** assigned to a variable.

Types of Literals in Java

1. **Integer Literals** → `int a = 100;`
2. **Floating-Point Literals** → `double pi = 3.14;`
3. **Character Literals** → `char letter = 'A';`
4. **String Literals** → `String msg = "Hello";`
5. **Boolean Literals** → `boolean flag = true;`
6. **Null Literal** → `String name = null;`

Example of Literals

```
class Example {
    public static void main(String[] args) {
        int age = 25;       // Integer literal
        double pi = 3.1416; // Floating-point literal
        char letter = 'A';  // Character literal
        String text = "Java"; // String literal
        boolean isJavaFun = true; // Boolean literal

        System.out.println(age + ", " + pi + ", " + letter + ", " + text +
", " + isJavaFun);
    }
}
```

---

4. Operators

Definition: Operators **perform operations** on variables and values.

Types of Operators in Java

A. Arithmetic Operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Remainder)

B. Relational (Comparison) Operators

Operator	Meaning
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

C. Logical Operators

Operator	Meaning
&&	Logical AND
!	Logical NOT

D. Assignment Operators

Operator	Meaning
=	Assign value
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign

E. Bitwise Operators

Operator	Meaning
&	Bitwise AND
^	Bitwise XOR
~	Bitwise Complement

Operator	Meaning
<<	Left shift
>>	Right shift

Example of Operators

```
class OperatorsExample {
    public static void main(String[] args) {
        int a = 10, b = 5;
        System.out.println("Addition: " + (a + b)); // Arithmetic
        System.out.println("Comparison: " + (a > b)); // Relational
        System.out.println("Logical AND: " + (a > 5 && b < 10)); // Logical
    }
}
```

---

5. Separators (Delimiters)

Definition: Separators **help structure the code** by defining blocks, parameters, and expressions.

List of Separators

Symbol	Purpose
`()`	Parentheses (used in methods, control statements)
`{}`	Curly braces (used for class, methods, loops)
`[]`	Square brackets (used for arrays)
`;`	Semicolon (used to terminate statements)
`,`	Comma (used in variable declarations and method parameters)
`.`	Dot (used for object and package access)

Example of Separators

```
class Example {
    public static void main(String[] args) { // () used in method
definition
        int numbers[] = {1, 2, 3}; // [] used for array
        System.out.println(numbers[0]); // . used for method call
    }
}
```

- **Tokens** are the building blocks of Java programs.
- **Keywords** are reserved words.
- **Identifiers** are user-defined names.
- **Literals** represent constant values.
- **Operators** perform computations.
- **Separators** structure the program