Literals in Java 21

A **literal** is a fixed value written directly in your code. Instead of calculating or reading a value, you just type it in. Java 21 has several types of literals.

---

1. Integer Literals

Used for whole numbers. By default, they're of type int.

```
int age = 25;
int negativeValue = -10;
```

You can write them in different **bases**:

```
int decimal     = 255;        // base 10 (normal)
int hexadecimal = 0xFF;       // base 16 — prefix 0x
int octal       = 0377;       // base 8  — prefix 0
int binary      = 0b11111111; // base 2  — prefix 0b
```

All four lines above equal the same value: 255.

For long values, add an L suffix:

```
long bigNumber = 10_000_000_000L;
```

---

2. Underscores in Numeric Literals

Java lets you add underscores (_) inside numbers to make them easier to read. They have no effect on the value.

```
int million       = 1_000_000;
long creditCard   = 1234_5678_9012_3456L;
double pi         = 3.141_592_653;
int mask          = 0b1010_1010;
```

---

3. Floating-Point Literals

Used for decimal numbers. Default type is double.

```
double price  = 19.99;
double sci    = 1.5e3;   // scientific notation: 1500.0
```

For float, add an F suffix:

```
float temp = 98.6F;
```

---

4. Character Literals

A single character wrapped in **single quotes**:

```
char letter  = 'A';
char digit   = '7';
char space   = ' ';
```

You can also use **escape sequences**:

```
char newline = '\n';   // new line
char tab     = '\t';   // tab
char quote   = '\'';   // single quote
char unicode = '\u0041'; // Unicode for 'A'
```

---

5. String Literals

Text wrapped in **double quotes**:

```
String greeting = "Hello, World!";
String empty    = "";
```

Strings also support escape sequences:

```
String message = "She said \"hello\".";
String path    = "C:\\Users\\name";
```

---

6. Text Block Literals (Java 15+, still in Java 21)

A modern way to write multi-line strings using **triple double-quotes** ("""). Great for HTML, JSON, SQL, etc.

```
String json = """
        {
            "name": "Alice",
            "age": 30
        }
        """;
```

The indentation is automatically stripped based on the closing """. This is much cleaner than concatenating strings.

---

7. Boolean Literals

Only two possible values:

```
boolean isJavaFun  = true;
boolean isHard     = false;
```

---

8. Null Literal

Represents the absence of a value. Can be assigned to any reference type (objects, Strings, arrays — but **not** primitives):

```
String name = null;
int x = null; // ❌ compile error — null can't be used with primitives
```

Quick Reference Table

Type	Example	Suffix/Prefix
`int`	`42`, `0xFF`, `0b101`	none / `0x` / `0b`
`long`	`100L`	`L`
`float`	`3.14F`	`F`
`double`	`3.14`, `1.5e3`	none
`char`	`'A'`, `'\n'`	single quotes
`String`	`"hello"`	double quotes
Text block	`"""..."""`	triple double quotes
`boolean`	`true`, `false`	none
`null`	`null`	none

Things to Watch Out For

Integer overflow — an int maxes out at ~2.1 billion. Use long for bigger numbers.

```
int tooBig = 3_000_000_000; // ❌ compile error
long justRight = 3_000_000_000L; // ✅
```

Float vs Double precision — floats are less precise. Prefer double unless memory is a concern, and use BigDecimal for financial calculations.

```
System.out.println(0.1 + 0.2); // 0.30000000000000004 (floating point quirk)
```

L not l — always use uppercase L for long literals. Lowercase l looks too much like the number 1.

```
long val = 100l; // ❌ confusing
long val = 100L; // ✅ clear
```

That covers all the literal types in Java 21! Once you're comfortable with these, a good next step is learning about **variables**, **data types**, and **type casting**.