In Java, **operators** are special symbols or keywords used to perform operations on variables or values. They can be categorized into several types based on their functionality. Let's go through them in detail:

### **1. Arithmetic Operators (Used for mathematical operations)**

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| + | Addition | a + b (adds a and b) |
| - | Subtraction | a - b (subtracts b from a) |
| \* | Multiplication | a \* b (multiplies a and b) |
| / | Division | a / b (divides a by b) |
| % | Modulus (remainder) | a % b (remainder of a divided by b) |

### **2. Relational (Comparison) Operators (Used to compare two values)**

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| == | Equal to | a == b (true if a equals b) |
| != | Not equal to | a != b (true if a is not equal to b) |
| > | Greater than | a > b (true if a is greater than b) |
| < | Less than | a < b (true if a is less than b) |
| >= | Greater than or equal to | a >= b (true if a is greater than or equal to `b) |
| <= | Less than or equal to | a <= b (true if a is less than or equal to `b) |

### **3. Logical Operators (Used for logical operations, mainly in conditionals)**

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| && | Logical AND | a > 5 && b < 10 (true if both conditions are true) |
| ` |  | ` |
| ! | Logical NOT | !(a > 5) (true if a is not greater than 5) |

### **4. Assignment Operators (Used to assign values to variables)**

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| = | Simple assignment | a = 5 (assigns 5 to a) |
| += | Add and assign | a += 5 (same as a = a + 5) |
| -= | Subtract and assign | a -= 5 (same as a = a - 5) |
| \*= | Multiply and assign | a \*= 5 (same as a = a \* 5) |
| /= | Divide and assign | a /= 5 (same as a = a / 5) |
| %= | Modulus and assign | a %= 5 (same as a = a % 5) |

### **5. Unary Operators (Used to perform operations on a single operand)**

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| + | Unary plus | +a (does nothing but indicates a positive value) |
| - | Unary minus | -a (negates the value of a) |
| ++ | Increment | a++ (increments a by 1, post-increment) |
| -- | Decrement | a-- (decrements a by 1, post-decrement) |
| ! | Logical NOT | !a (true if a is false, false if a is true) |

### **6. Bitwise Operators (Used for operations on bits)**

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| & | Bitwise AND | a & b (bits set to 1 where both a and b are 1) |
| ` | ` | Bitwise OR |
| ^ | Bitwise XOR | a ^ b (bits set to 1 where a and b are different) |
| ~ | Bitwise NOT | ~a (inverts the bits of a) |
| << | Left shift | a << 2 (shifts bits of a left by 2 positions) |
| >> | Right shift | a >> 2 (shifts bits of a right by 2 positions) |
| >>> | Unsigned right shift | a >>> 2 (shifts bits of a right by 2 positions, fills with 0) |

### **7. Ternary Operator (Shorthand for if-else statement)**

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| ?: | Ternary conditional | result = (a > b) ? "a is greater" : "b is greater"; (If a > b, assign "a is greater", else assign "b is greater") |

### **8. Instanceof Operator (Used to test whether an object is an instance of a particular class or subclass)**

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| instanceof | Checks if an object is an instance of a specific class or subclass | a instanceof String (checks if a is an instance of String class) |

### **9. Type Cast Operator (Used to convert one data type to another)**

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| (type) | Type casting | int a = (int) 3.14; (casts the double value 3.14 to an int, resulting in 3) |

### **💡 Summary of Operator Categories:**

* **Arithmetic Operators**: +, -, \*, /, %
* **Relational Operators**: ==, !=, >, <, >=, <=
* **Logical Operators**: &&, ||, !
* **Assignment Operators**: =, +=, -=, \*=, /=, %=
* **Unary Operators**: +, -, ++, --, !
* **Bitwise Operators**: &, |, ^, ~, <<, >>, >>>
* **Ternary Operator**: ?:
* **Instanceof Operator**: instanceof
* **Type Cast Operator**: (type)

Rules for Java identifiers:

* An identifier must start with a letter, an underscore (\_), or a dollar sign ($).
* Subsequent characters can be letters, digits, underscores, or dollar signs.
* Identifiers are case-sensitive.
* Identifiers cannot be keywords.

| Names can contain letters, digits, underscores, and dollar signs |
| --- |
| Names must begin with a letter |
| Names should start with a lowercase letter and it cannot contain whitespace |
| Names can also begin with $ and \_ |
| Names are case sensitive ("myVar" and "myvar" are different variables) |
| Reserved words (like Java keywords, such as int or boolean) cannot be used as names |

**Examples of valid Java identifiers:**

age\_var\_1

sum

Sum

totalVolume

int\_variable

float\_variable

**Examples of invalid Java identifiers:**

123\_variable (identifiers cannot start with a number)

my-variable (identifiers cannot contain hyphens)

class (class is a keyword)

It is generally considered good practice to use descriptive names for identifiers, and to avoid using abbreviations or acronyms unless they are well-known.