**1. Different operators in java.**

\*\*Java Operators\*\*

### 1. Arithmetic Operators

- \*\*Addition (`+`)\*\*: Adds two operands (e.g., `a + b`)

- \*\*Subtraction (`-`)\*\*: Subtracts the second operand from the first (e.g., `a - b`)

- \*\*Multiplication (`\*`)\*\*: Multiplies two operands (e.g., `a \* b`)

- \*\*Division (`/`)\*\*: Divides the numerator by the denominator (e.g., `a / b`)

- \*\*Modulus (`%`)\*\*: Returns the remainder of a division operation (e.g., `a % b`)

### 2. Relational Operators

- \*\*Equal to (`==`)\*\*: Checks if two operands are equal (e.g., `a == b`)

- \*\*Not equal to (`!=`)\*\*: Checks if two operands are not equal (e.g., `a != b`)

- \*\*Greater than (`>`)\*\*: Checks if the left operand is greater than the right (e.g., `a > b`)

- \*\*Less than (`<`)\*\*: Checks if the left operand is less than the right (e.g., `a < b`)

- \*\*Greater than or equal to (`>=`)\*\*: Checks if the left operand is greater than or equal to the right (e.g., `a >= b`)

- \*\*Less than or equal to (`<=`)\*\*: Checks if the left operand is less than or equal to the right (e.g., `a <= b`)

### 3. Logical Operators

- \*\*Logical AND (`&&`)\*\*: True if both operands are true (e.g., `a && b`)

- \*\*Logical OR (`||`)\*\*: True if at least one operand is true (e.g., `a || b`)

- \*\*Logical NOT (`!`)\*\*: Reverses the logical state of its operand (e.g., `!a`)

### 4. Bitwise Operators

- \*\*Bitwise AND (`&`)\*\*: Performs AND operation on bits (e.g., `a & b`)

- \*\*Bitwise OR (`|`)\*\*: Performs OR operation on bits (e.g., `a | b`)

- \*\*Bitwise XOR (`^`)\*\*: Performs XOR operation on bits (e.g., `a ^ b`)

- \*\*Bitwise Complement (`~`)\*\*: Inverts all bits (e.g., `~a`)

- \*\*Left Shift (`<<`)\*\*: Shifts bits to the left (e.g., `a << 2`)

- \*\*Right Shift (`>>`)\*\*: Shifts bits to the right (e.g., `a >> 2`)

- \*\*Unsigned Right Shift (`>>>`)\*\*: Shifts bits to the right and fills 0 from the left (e.g., `a >>> 2`)

### 5. Assignment Operators

- \*\*Simple Assignment (`=`)\*\*: Assigns the right operand to the left (e.g., `a = b`)

- \*\*Add and Assign (`+=`)\*\*: Adds and assigns (e.g., `a += b`)

- \*\*Subtract and Assign (`-=`)\*\*: Subtracts and assigns (e.g., `a -= b`)

- \*\*Multiply and Assign (`\*=`)\*\*: Multiplies and assigns (e.g., `a \*= b`)

- \*\*Divide and Assign (`/=`)\*\*: Divides and assigns (e.g., `a /= b`)

- \*\*Modulus and Assign (`%=`)\*\*: Modulus and assigns (e.g., `a %= b`)

- \*\*Bitwise AND and Assign (`&=`)\*\*: Performs bitwise AND and assigns (e.g., `a &= b`)

- \*\*Bitwise OR and Assign (`|=`)\*\*: Performs bitwise OR and assigns (e.g., `a |= b`)

- \*\*Bitwise XOR and Assign (`^=`)\*\*: Performs bitwise XOR and assigns (e.g., `a ^= b`)

- \*\*Left Shift and Assign (`<<=`)\*\*: Left shifts and assigns (e.g., `a <<= 2`)

- \*\*Right Shift and Assign (`>>=`)\*\*: Right shifts and assigns (e.g., `a >>= 2`)

- \*\*Unsigned Right Shift and Assign (`>>>=`)\*\*: Unsigned right shifts and assigns (e.g., `a >>>= 2`)

### 6. Unary Operators

- \*\*Unary Plus (`+`)\*\*: Indicates a positive value (e.g., `+a`)

- \*\*Unary Minus (`-`)\*\*: Negates the value (e.g., `-a`)

- \*\*Increment (`++`)\*\*: Increases the value by one (e.g., `++a` or `a++`)

- \*\*Decrement (`--`)\*\*: Decreases the value by one (e.g., `--a` or `a--`)

- \*\*Logical Complement (`!`)\*\*: Inverts a boolean value (e.g., `!true`)

### 7. Ternary Operator (Conditional Operator)

- \*\*Syntax\*\*: `condition ? expression1 : expression2;`

- \*\*Example\*\*: `int result = (a > b) ? a : b;` // assigns the greater of a or b to result.

### 8. Instanceof Operator

- This operator checks whether an object is an instance of a specific class or subclass.

- \*\*Syntax\*\*: `object instanceof ClassName`

- \*\*Example\*\*: `if (obj instanceof MyClass) { ... }`

### 9. Type Comparison Operator

- Introduced in Java 14, the type comparison operator can help to validate the presence of a specific type within pattern matching.

-----------------------------------------------------------------------------------------------------------------------------------------