Operators in Java

Operators are special **symbols** in Java used to perform specific **operations** on **operands**.

- Example: `10 + 20`
  - `+` → operator
  - `10`, `20` → operands

Java operators are classified into **three types**, based on the number of operands:

1. **Unary Operators** – operate on **one operand**
2. **Binary Operators** – operate on **two operands**
3. **Ternary Operators** – operate on **three operands**

---

1. Unary Operators

Unary operators apply **only to variables**, not to literal values or expressions.

- Invalid: `++10`, `++(x++)` (as both are **values**, not variables)
- All primitive types support unary operations **except** `boolean`.

Increment Operator (`++`)

Increments the value of a variable by 1.

Types:

- **Pre-increment:** `++x` → Increments before assignment
- **Post-increment:** `x++` → Increments after assignment

Example – Pre-Increment

```java
class PreIncrement {
    public static void main(String[] args) {
        int x = 12;
        int y = ++x;
        System.out.println(x + " " + y); // 13 13
    }
}
```

> `INCREMENT` → `ASSIGN` → `INITIALIZE`

Example – Post-Increment

```java
class PostIncrement {
    public static void main(String[] args) {
        int x = 12;
        int y = x++;
        System.out.println(x + " " + y); // 13 12
    }
}
```

> `ASSIGN` → `INCREMENT` → `INITIALIZE`

Decrement Operator (`--`)

Decreases the value of a variable by 1.

Types:

- **Pre-decrement:** `--x`
- **Post-decrement:** `x--`

Example – Pre-Decrement

```java
class PreDecrement {
    public static void main(String[] args) {
        int x = 12;
        int y = --x;
        System.out.println(x + " " + y); // 11 11
    }
}
```

Example – Post-Decrement

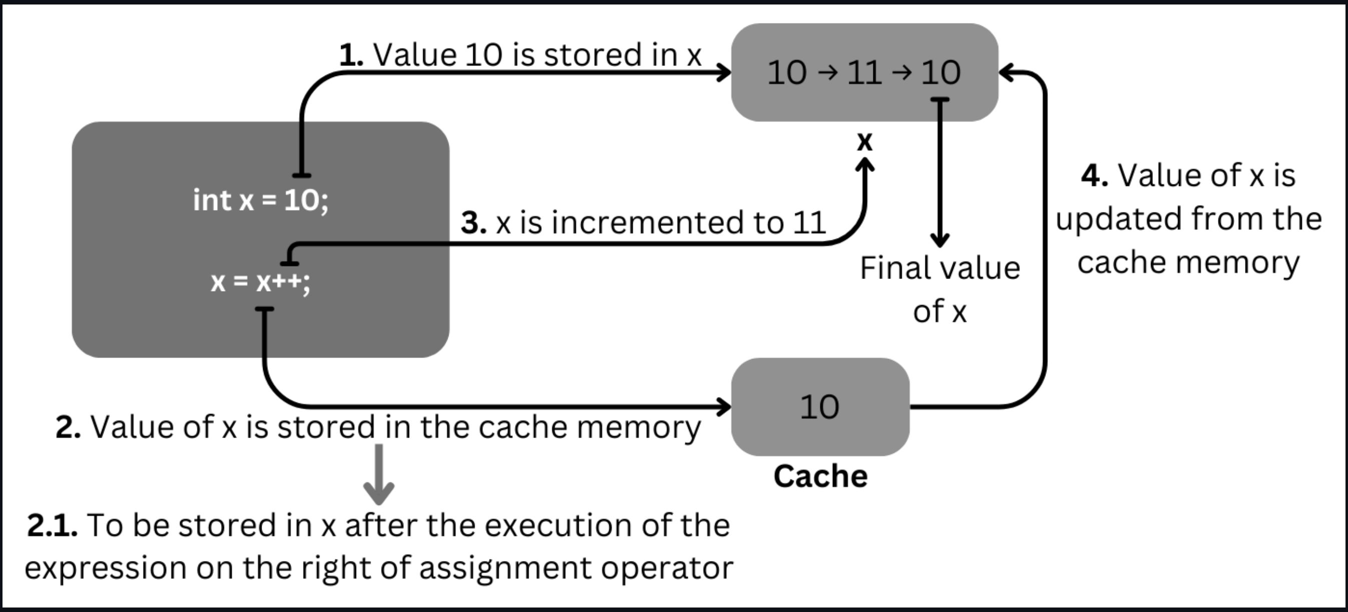
```java
class PostDecrement {
    public static void main(String[] args) {
        int x = 12;
        int y = x--;
        System.out.println(x + " " + y); // 11 12
    }
}
```

Self-Assignment Case

When using `x = x++`, the value of `x` remains unchanged.

```java
class SelfAssignment {
    public static void main(String[] args) {
        int x = 10;
        for (int i = 0; i < 5; i++) {
            x = x++;
            System.out.println(x); // Always prints 10
        }
    }
}
```

- `x = x++` stores the old value in `x`, not the incremented value.
- To ensure the value updates, use `x = ++x`.


1. Value 10 is stored in x
10 → 11 → 10
x
int x = 10;
x = x++;
3. x is incremented to 11
Final value of x
4. Value of x is updated from the cache memory
2. Value of x is stored in the cache memory
10
Cache
2.1. To be stored in x after the execution of the expression on the right of assignment operator

Bitwise Complement (`~`)

- Only for **integer** types
- Not applicable to `boolean`

```java
class BitwiseComplement {
    public static void main(String[] args) {
        System.out.println(~(-7));     // 6
        System.out.println(~4);        // -5
        System.out.println(~22);       // -23
        System.out.println(~'a');      // -98
    }
}
```

Boolean Complement (`!`)

- Only for **boolean** type

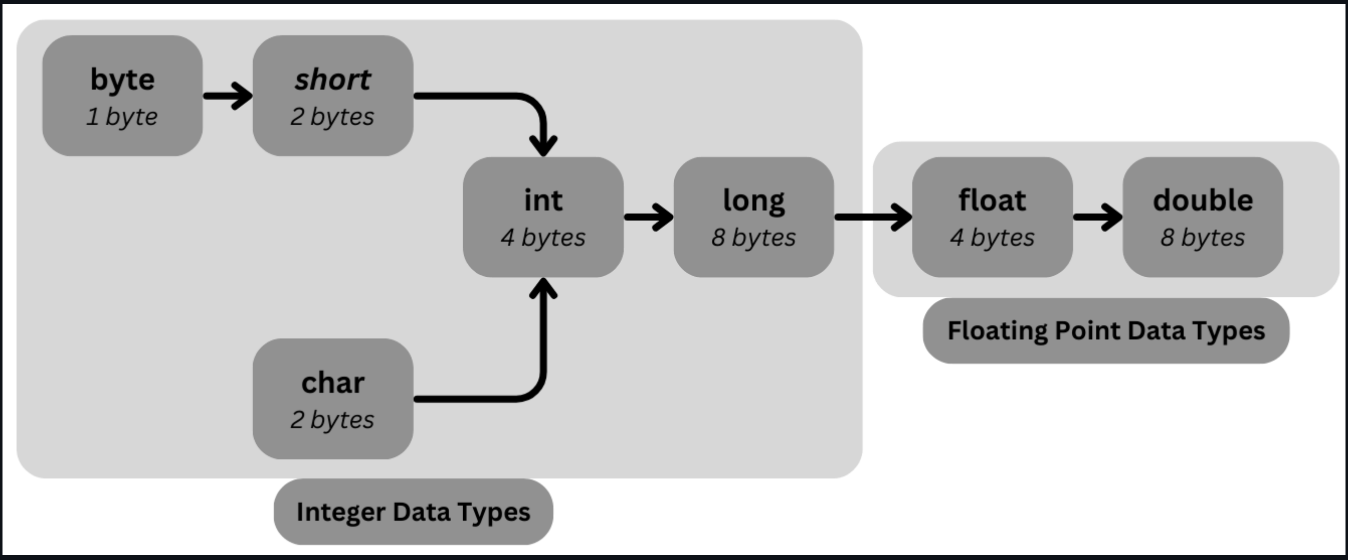
```java
class BooleanComplement {
    public static void main(String[] args) {
        boolean b = true;
        System.out.println(!b);       // false
        System.out.println(!true);    // false
        System.out.println(!false);   // true
    }
}
```

2. Binary Operators

Arithmetic Operators: `+` `-` `*` `/` `%`

```java
class ArithmeticOperators {
    public static void main(String[] args) {
        int a = 10, b = 2;
        System.out.println(a + b); // 12
        System.out.println(a - b); // 8
        System.out.println(a * b); // 20
        System.out.println(a / b); // 5
        System.out.println(a % b); // 0
    }
}
```

> Result data type = `max(int, type of operand1, type of operand2)`



String Concatenation using `+`

```java
class StringConcatenation {
    public static void main(String[] args) {
        int i = 4;
        boolean b = true;
        char c = '2';
        System.out.println("Result: " + i);       // Result: 4
        System.out.println("Boolean: " + b);      // Boolean: true
        System.out.println("Char: " + c);         // Char: 2
    }
}
```

Relational Operators: `<` `<=` `>` `>=`

```java
class RelationalOperators {
    public static void main(String[] args) {
        System.out.println(10 < 20);      // true
        System.out.println(12 <= 12.65);  // true
        System.out.println(97 > 'a');     // false
        System.out.println(97 >= 'a');    // true
    }
}
```

Equality Operator: `==`

```java
class EqualityOperator {
    public static void main(String[] args) {
        int x = 10, y = 10;
        EqualityOperator ob1 = new EqualityOperator();
        EqualityOperator ob2 = new EqualityOperator();
        EqualityOperator ob3 = ob1;
        System.out.println(x == y);   // true
        System.out.println(ob1 == ob2); // false
        System.out.println(ob1 == ob3); // true
    }
}
```

Assignment Operators: `=`, `+=`, `-=`, etc.

Chain Assignment

```java
class ChainAssignment {
    public static void main(String[] args) {
        int a, b, c;
        a = b = c = 10;
        System.out.println(a + " " + b + " " + c); // 10 10 10
    }
}
```

Compound Assignment

```java
class CompoundAssignment {
    public static void main(String[] args) {
        int a = 10;
        a += 5;
        a *= 2;
        a -= 4;
        System.out.println(a); // 26
    }
}
```

Bitwise Operators: `&`, `|`, `^`

```java
class BitwiseOperators {
    public static void main(String[] args) {
        System.out.println(4 & 5); // 4
        System.out.println(4 | 5); // 5
        System.out.println(4 ^ 5); // 1
    }
}
```

Bit	8	4	2	1
4	0	1	0	0

Bit	8	4	2	1
5	0	1	0	1
`&`	0	1	0	0 = 4
`|`	0	1	0	
`^`	0	0	0	1 = 1

Logical Operators: `&&`, `||`

```java
class LogicalOperators {
    public static void main(String[] args) {
        boolean result;

        result = (10 > 2) && (12 == 12);
        System.out.println(result); // true

        result = (10 < 2) || (12 != 12);
        System.out.println(result); // false
    }
}
```

---

3. Ternary Operator: `? :`

Conditional Expression

```java
class TernaryOperator {
    public static void main(String[] args) {
        int result = (1 > 0) ? 1000 : 2000;
        String name = (false) ? "Sagar" : "Sambit";
        System.out.println(result + " " + name); // 1000 Sambit
    }
}
```
