## Operators

- Operators are predefined symbol which is used to perform specific task on the given data.
- The data given as an input to the operator is known as operand.
- Based on the number of operand, operators are classified into following:

1. Unary Operator
2. Binary Operator
3. Ternary Operator

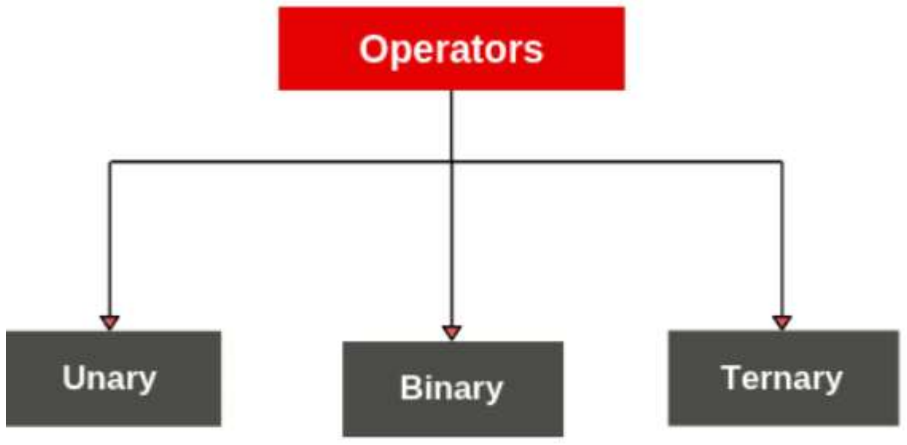


Fig: Classification of java operators based on number of operands

1. **Unary Operator:**

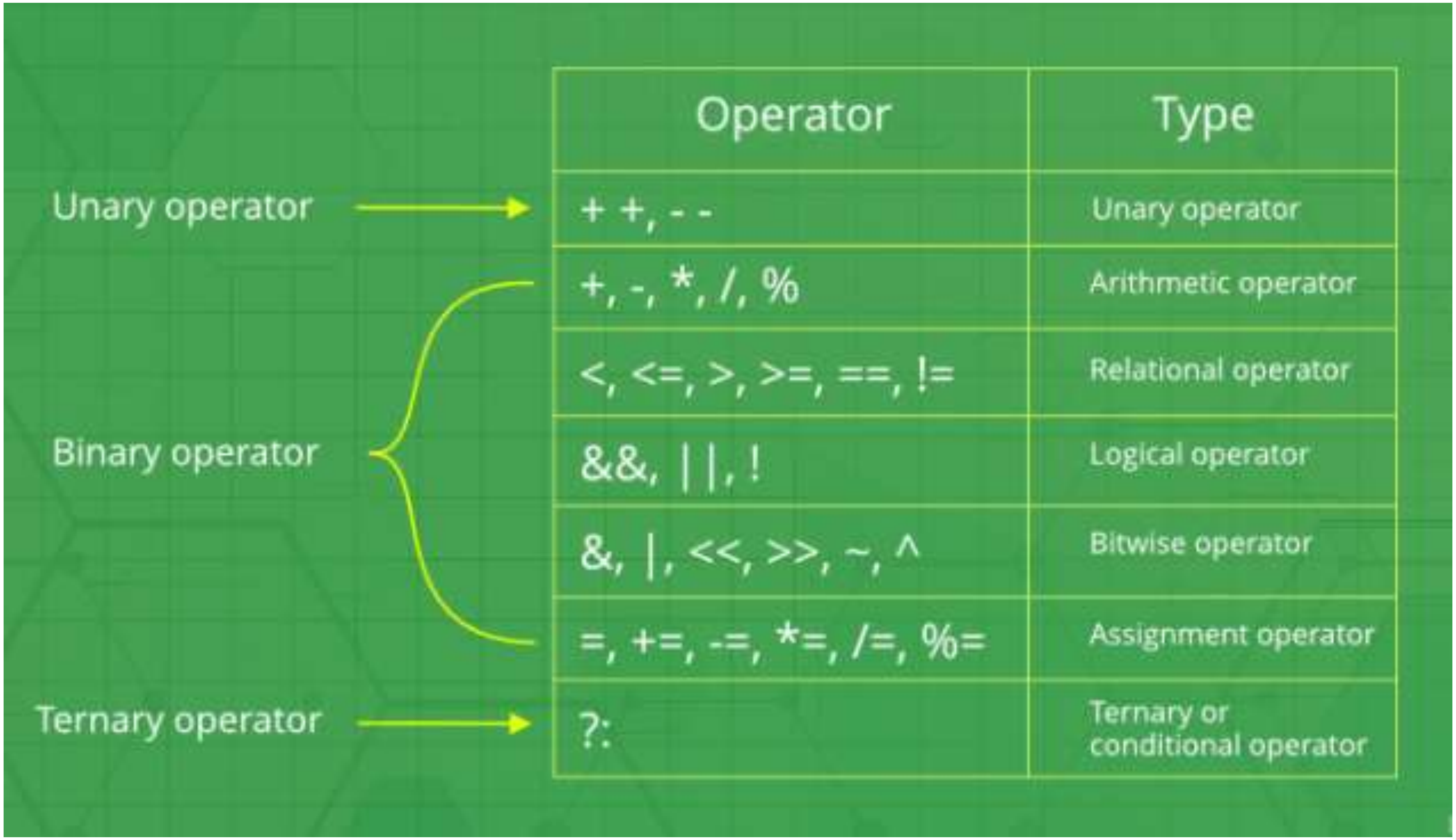
- The operator which can accept only one operand is known as unary operator.

2. **Binary Operator:**

- The operator which can accept two operand is known as Binary operator.

3. **Ternary Operator:**

- The operator which can accept three operands is known as Ternary operator.



Unary operator

Binary operator

Ternary operator

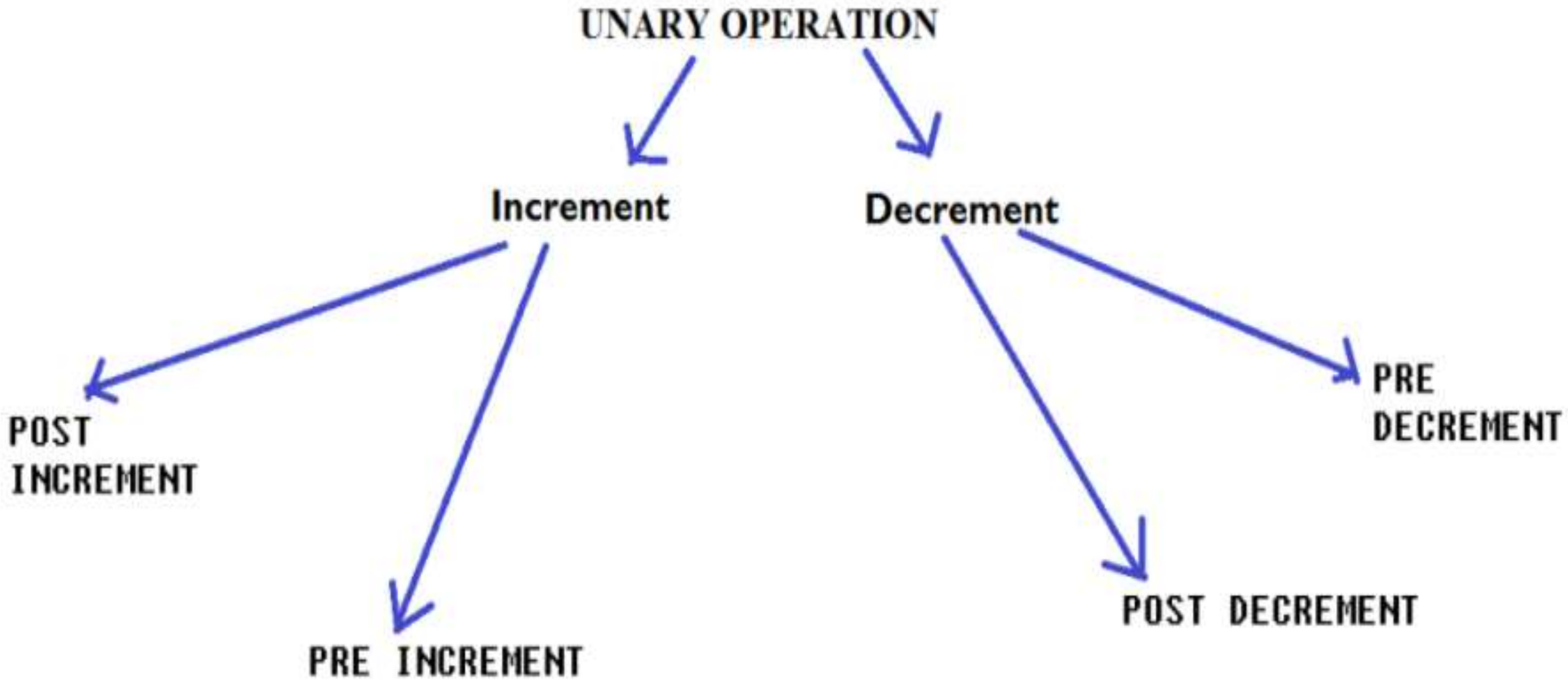
| Operator | Type |
|---|---|
| + +, - - | Unary operator |
| +, -, *, /, % | Arithmetic operator |
| <, <=, >, >=, ==, != | Relational operator |
| &&, \|\|, ! | Logical operator |
| &, \|, <<, >>, ~, ^ | Bitwise operator |
| =, +=, -=, *=, /=, %= | Assignment operator |
| ?: | Ternary or conditional operator |

## Classification of Operator

- **The operators can be classified based on the task.**

1. Assignment Operators. **Ex:** =.
2. Arithmetic Operators. **Ex**: +, -, *, /, etc.
3. Logical Operators. **Ex:**&&, ||, !.
4. Increment/Decrement Operators**. Ex:** + +, – –
5. Bitwise Operators. **Ex:** &, |, ^
6. Conditional Operators. **Ex:** ?:
7. Relational Operators**. Ex:**<**,** >, <=, >=, = =, !=.

# Increment and Decrement Operator



# Pre-increment operator

- The pre-increment operator is represented as the double plus (++a) symbol, appended before the variable's name.
- The pre-increment operator is used to increment the value of an operand by 1 before using it in the mathematical expression.
- In other words, the value of a variable is first incremented, and then the updated value is used in the expression.

**Syntax:**

```
x = ++a;
```

In the above syntax, the value of variable 'a' is first incremented by 1 before using in the expression

## Post-increment operator

- Post-increment is an increment operator, represented as the double plus (a++) symbol followed by an operator 'a'.
- It increments the value of the operand by 1 after using it in the mathematical expression.
- In other words, the variable's original value is used in the expression first, and then the post-increment operator updates the operand value by 1.

**Syntax:**

```
x = a++;
```

In the above syntax, the operand 'a' value is assigned to the variable x, and then the post increment operator increases or updates the value of 'a' by 1.

## Pre Decrement Operator

- The Pre Decrement Operator decreases the operand value by 1 before assigning it to the mathematical expression.
- In other words, the original value of the operand is first decreases, and then a new value is assigned to the other variable.

**Syntax:**

```
b=--a;
```

In the above syntax, the value of operand ‘a' is decreased by 1, and then a new value is assigned to the variable ‘b'.

## **Post decrement Operator:**

- Post decrement operator is used to decrease the original value of the operand by 1 after assigning to the expression.

**Syntax:**

```
        b = a--;
```

In the above syntax, the value of operand ‘a' is assigned to the variable ‘b', and then the value of a is decreased by 1.

# Increment and Decrement Operator

| Types | Operator | Operation/Meaning |
|---|---|---|
| | | |
| Pre-Increment | ++a | Increment a value by 1,then use the new value of a. |
| Post-Increment | a++ | Use the value of a, then increment value of a by 1 |
| Pre-Decrement | --a | Decrement a value by 1,then use the new value of a. |
| Post-Decrement | a-- | Use the value of a, then decrement value of a by 1. |
| | | |

## Limitations of increment and decrement operators

- We can apply Increment and decrement operators only for variables but not for constant values. If we apply, then we will get compile time error.
- We can’t apply the nesting on increment and decrement operators.
- We can’t apply increment and decrement operators on boolean types.
- We can’t apply increment and decrement operators on final variables.

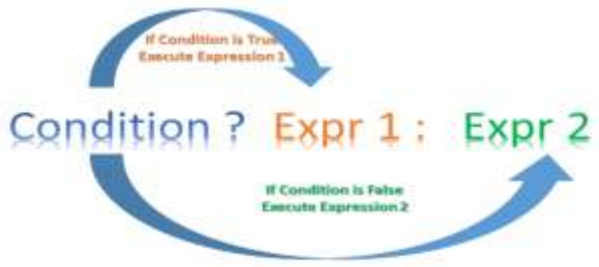
## <u>**Conditional Operator**</u>

- It is a ternary operator.
- **Syntax:**

    operand1?operand2:operand3

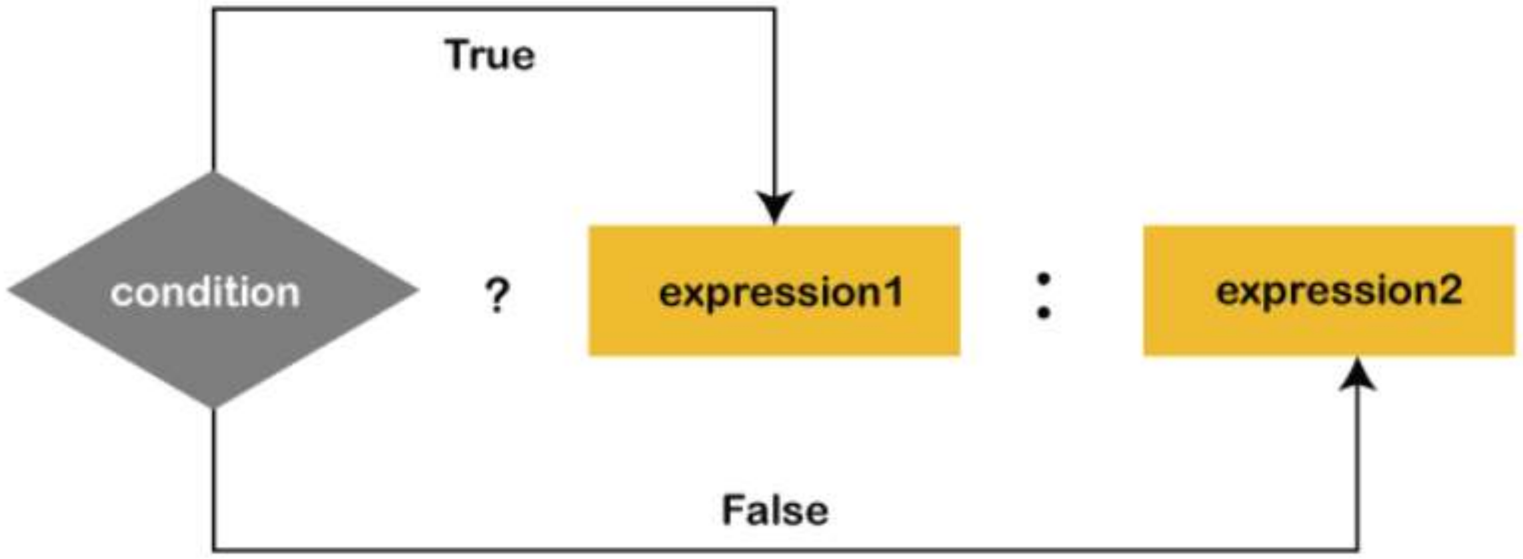
    condition?statement1:statement2

**Operation:**



- The return type of operand 1 must be a boolean.
- If condition is true, statement 1 is executed else statement 2 is executed.

# Conditional Operator


True
condition
?
expression1
:
expression2
False