
# Object Oriented Programming with Java (OOPJ)

Session 2: Programming concepts

Kiran Waghmare

# Java Tokens

- Tokens - The smallest individual unit of program are known as Tokens.

- Java Program – It is a collection of Tokens , comments and white spaces. It contains 5 types of tokens:

1. **Reserved words – keywords**
    - 50 keywords
    - Having specific meaning – we cannot use them as names for variables ,class name etc
    - Always lower case letters, case sensitive
    - E.g., abstract, case, short, super etc
2. **Identifiers – a**
    - Programmer designed tokens
    - Used for naming classes, methods, variables, labels, packages, interfaces in a program
    - Rules-
        1. Have alphabets,digits and _ and $
        2. Not begin with digit
        3. Uppercase & lowercase letters are distinct
        4. Can be of any length

1. **Literals –**
    - Sequence of character
    - Represents constant value to be stored in variable
    - 5 – types- Integer, Floating-point, Character, String and Boolean
2. **Operators –**
    - Symbol that takes one / more arguments & operates on them to produce a result.
3. **Separators –**
    - Group of code are divided & arranged
    - i.e., ( ), { }, [ ], ;, , ,& .

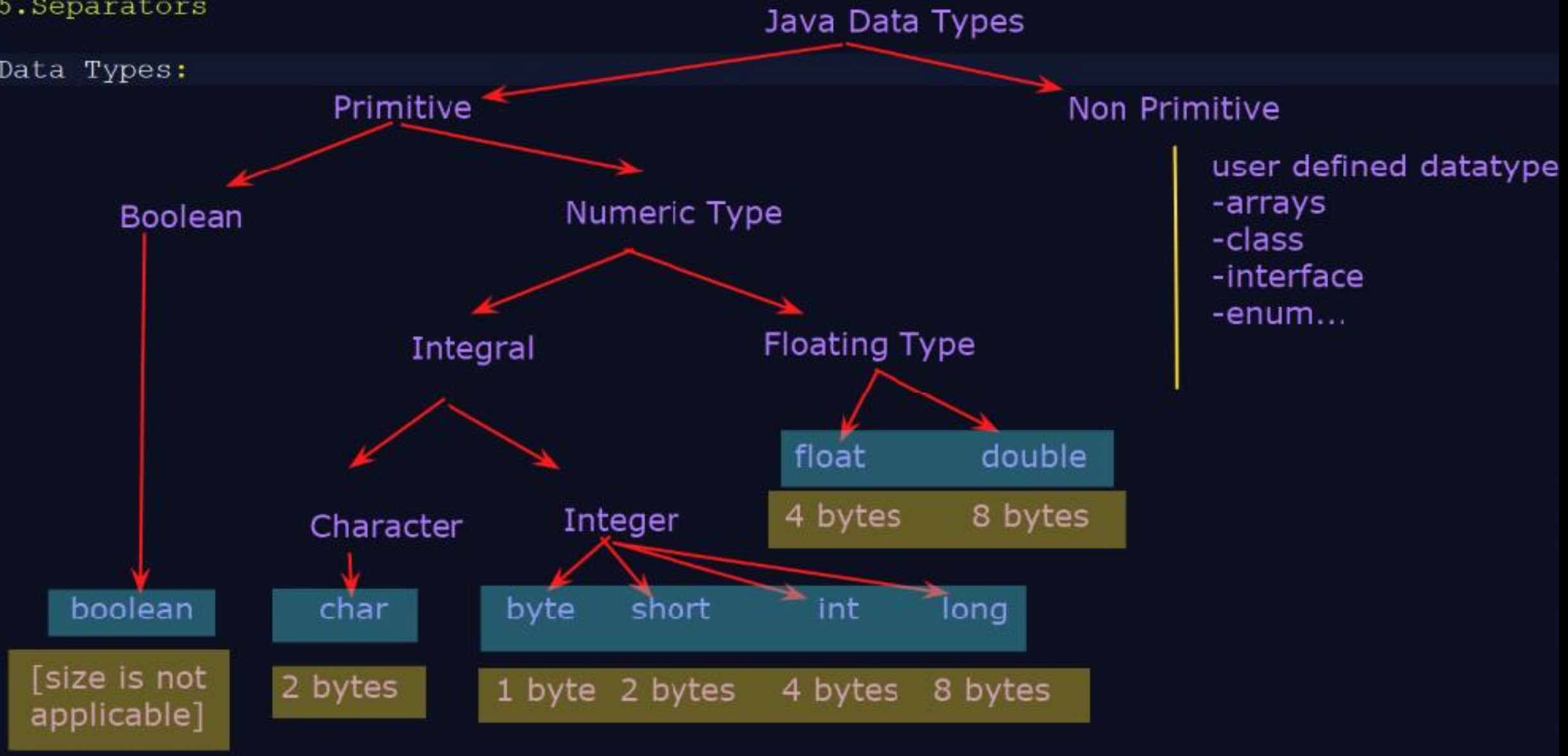
1. `studentName`
2. `123abc`
3. `_myVariable`
4. `$salary`
5. `class`
6. `user-age`
7. `mainMethod`
8. `int`
9. `String`
10. `myVariable123`

## Table: Java Reserved Words

| Category | Keywords |
| --- | --- |
| Data Types | `byte`, `short`, `int`, `long`, `float`, `double`, `char`, `boolean` |
| Control Flow | `if`, `else`, `switch`, `case`, `default`, `while`, `do`, `for` |
| Loop Control | `break`, `continue`, `return` |
| Access Modifiers | `public`, `private`, `protected` |
| Non-Access Modifiers | `static`, `final`, `abstract`, `synchronized`, `transient`, `volatile` |
| Class & Object Handling | `class`, `interface`, `extends`, `implements`, `new`, `this`, `super` |
| Exception Handling | `try`, `catch`, `finally`, `throw`, `throws` |
| Miscellaneous | `void`, `native`, `strictfp`, `instanceof`, `package`, `import` |
| Unused Reserved Words | `goto`, `const` *(reserved but not used in Java)* |
| Reserved Literals | `true`, `false`, `null` *(These are literals, not keywords but reserved)* |

4.Operators
5.Separators

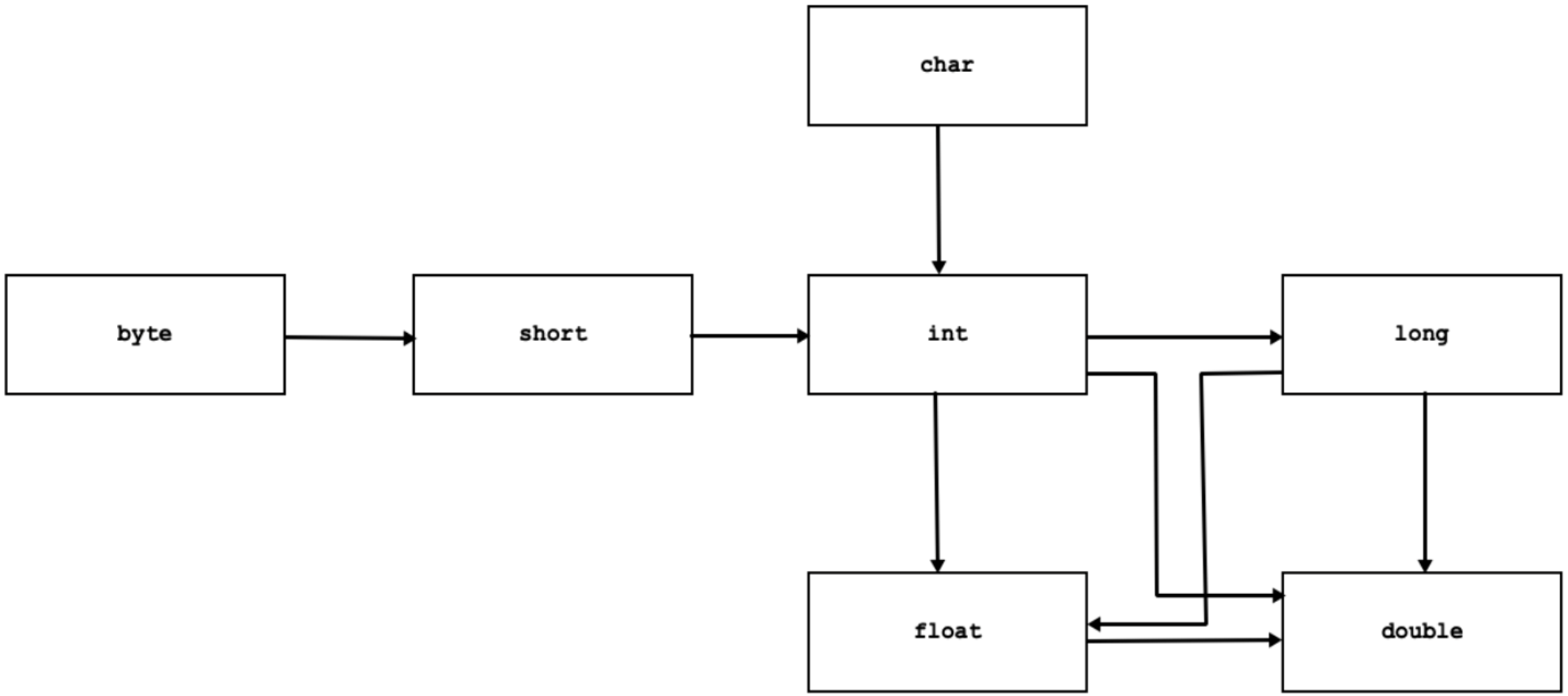
Data Types:



## Quick Summary of Data Types:

| Data Type | Size (Bytes) | Range | Default Value | Precision |
|---|---|---|---|---|
| `byte` | 1 | -128 to 127 | 0 | - |
| `short` | 2 | -32,768 to 32,767 | 0 | - |
| `int` | 4 | -2,147,483,648 to 2,147,483,647 | 0 | - |
| `long` | 8 | ±9.2E18 | 0L | - |
| `float` | 4 | ±3.4E38 | 0.0f | 5-6 decimals |
| `double` | 8 | ±1.7E308 | 0.0d | 14-15 decimals |
| `char` | 2 | 0 to 65,535 (Unicode) | '\u0000' | - |
| `boolean` | 1 bit | true / false | false | - |

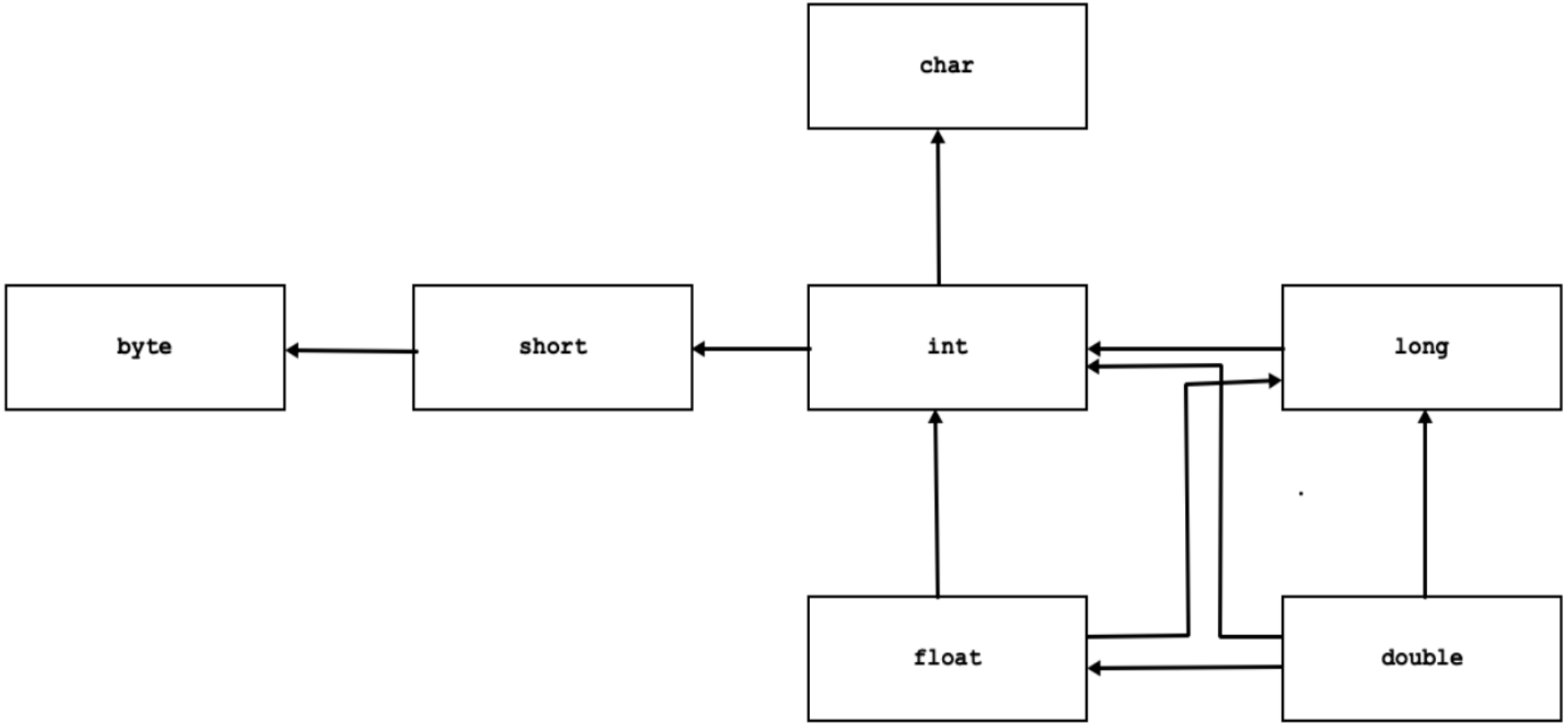
# Widening




- Widening is the process of converting value of variable of narrower type into wider type.

# Narrowing

- Narrowing is the process of converting value of variable of wider type into narrower type.



# Arithmetic Operators

| Operator | Description | Example | Output |
|---|---|---|---|
| + | Addition | 10 + 5 | 15 |
| - | Subtraction | 10 - 5 | 5 |
| * | Multiplication | 10 * 5 | 50 |
| / | Division | 10 / 5 | 2 |
| % | Modulus (Remainder) | 10 % 3 | 1 |

# Relational Operators

| Operator | Description | Example | Output |
|---|---|---|---|
| == | Equal to | 10 == 5 | false |
| != | Not equal to | 10 != 5 | true |
| > | Greater than | 10 > 5 | true |
| < | Less than | 10 < 5 | false |
| >= | Greater than or equal to | 10 >= 5 | true |
| <= | Less than or equal to | 10 <= 5 | false |

# Logical Operator

| Operator | Description | Example | Output |
|---|---|---|---|
| && | Logical AND | (10 > 5) && (5 < 10) | true |
| \|\| | Logical OR | (10 > 5) \|\| (5 < 10) | true |
| ! | Logical NOT | !(10 > 5) | false |

# Bitwise Operator

| Operator | Description | Example | Output |
|---|---|---|---|
| & | Bitwise AND | 5 & 3 (0101 & 0011) | 1 |
| \| | \| | Bitwise OR | `5 |
| ^ | Bitwise XOR | 5 ^ 3 (0101 ^ 0011) | 6 |
| ~ | Bitwise NOT | ~5 (~0101) | -6 |
| << | Left Shift | 5 << 1 | 10 |
| >> | Right Shift | 5 >> 1 | 2 |

# Assignment Operator

| Operator | Description | Example | Equivalent |
|---|---|---|---|
| = | Assign | x = 5 | x = 5 |
| += | Add and assign | x += 5 | x = x + 5 |
| -= | Subtract and assign | x -= 5 | x = x - 5 |
| *= | Multiply and assign | x *= 5 | x = x * 5 |
| /= | Divide and assign | x /= 5 | x = x / 5 |
| %= | Modulus and assign | x %= 5 | x = x % 5 |