Java

Java:

- Java is a popular programming language.
- It's used to create software and apps.
- Known for its "write once, run anywhere" feature.
- Java is not a fully object-oriented language as it supports primitive data types like int, byte, long, short, etc., which are not objects.

Basic Structure:

- Code is written in classes.
- Each program has a **main** method to start from.
- Statements end with a semicolon **;**.

Variables:

- Hold data like numbers or text.
- A variable is nothing but a memory location name for the data.
- Declare with a type and a name.
- Example: **int age = 25;**

- In Java, variables can be classified into three main types: local variables, instance variables, and static variables

difference between object and variable in Java?

Simple variable can only hold one value (string, number, boolean etc). Object can hold pairs of variables and values.

Data Types:

A category that specifies the kind of data a variable can hold in a programming language.

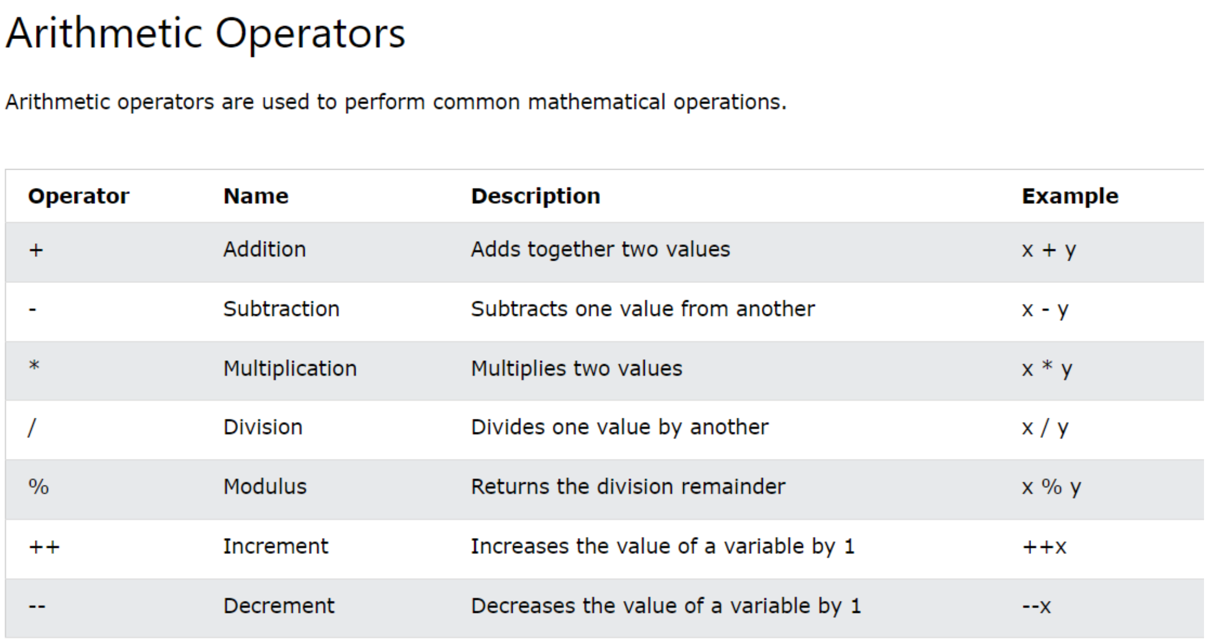
- **int** for whole numbers.
- **double** for decimal numbers.
- **boolean** for true/false values.
- **String** for text.

Operators:

Operators are used to perform operations on variables and values.

Types of Operators:

- Arithmetic operators

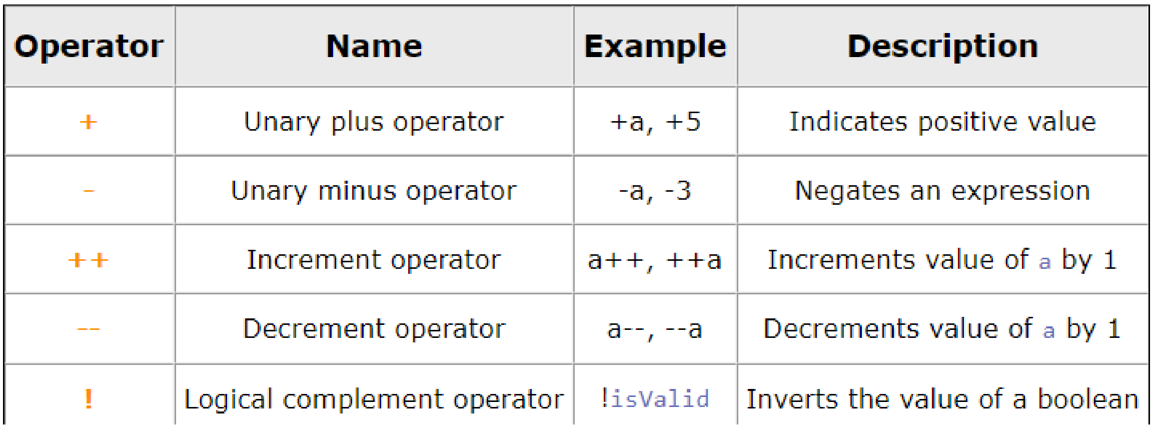


Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

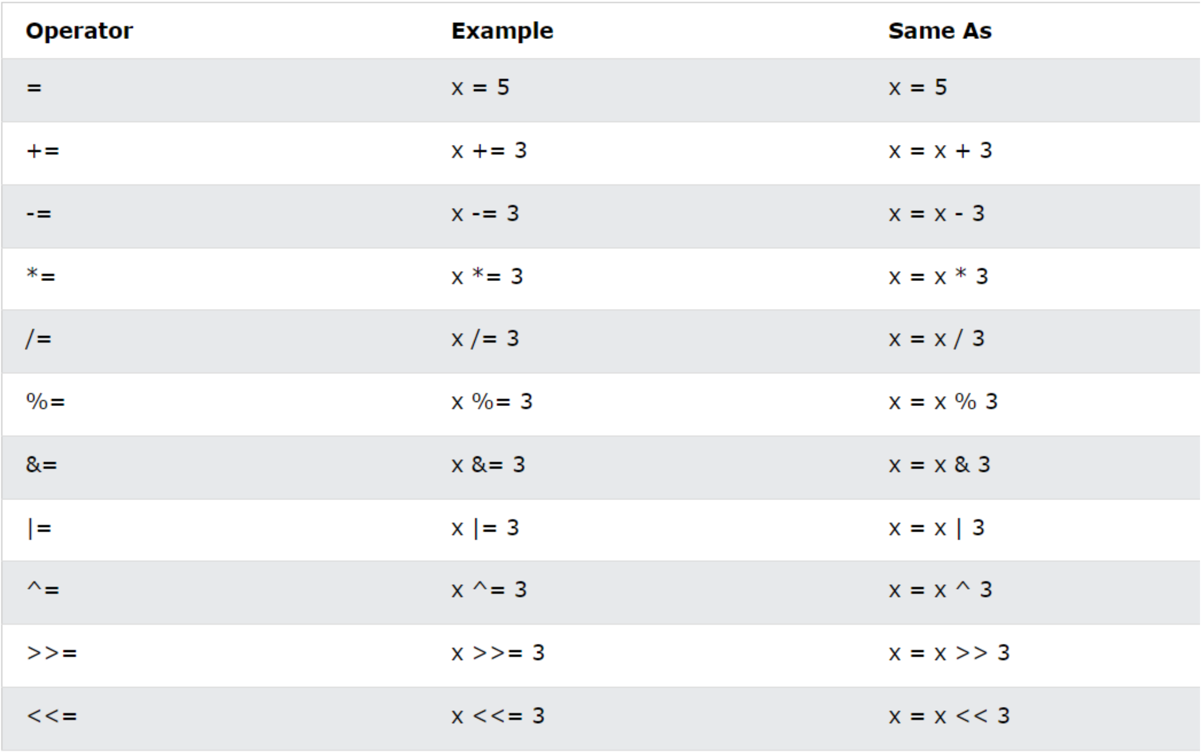
- ## Unary Operators:



Operator	Name	Example	Description
+	Unary plus operator	+a, +5	Indicates positive value
-	Unary minus operator	-a, -3	Negates an expression
++	Increment operator	a++, ++a	Increments value of a by 1
--	Decrement operator	a--, --a	Decrements value of a by 1
!	Logical complement operator	!isValid	Inverts the value of a boolean

- ## Assignment operators

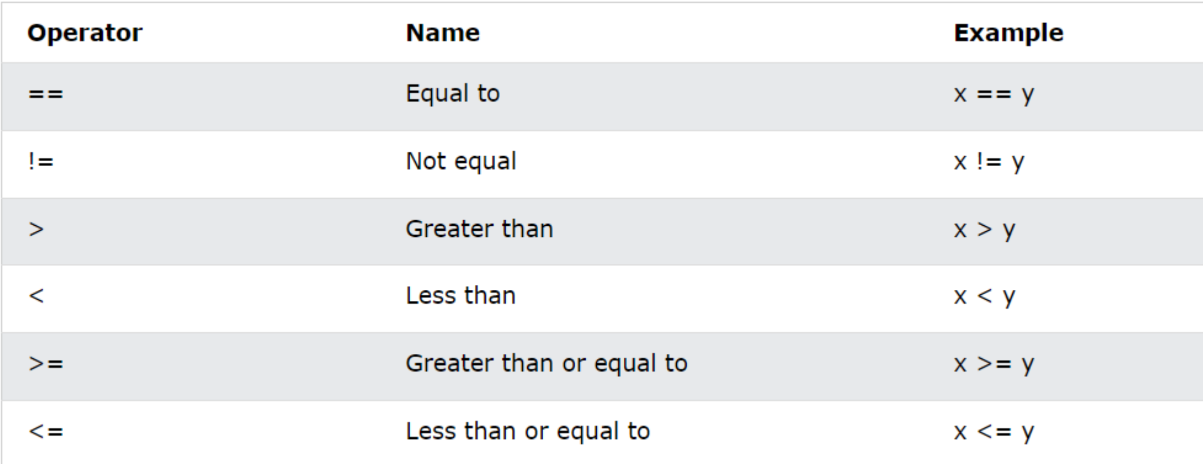
Assignment operators are used to assign values to variables.



Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

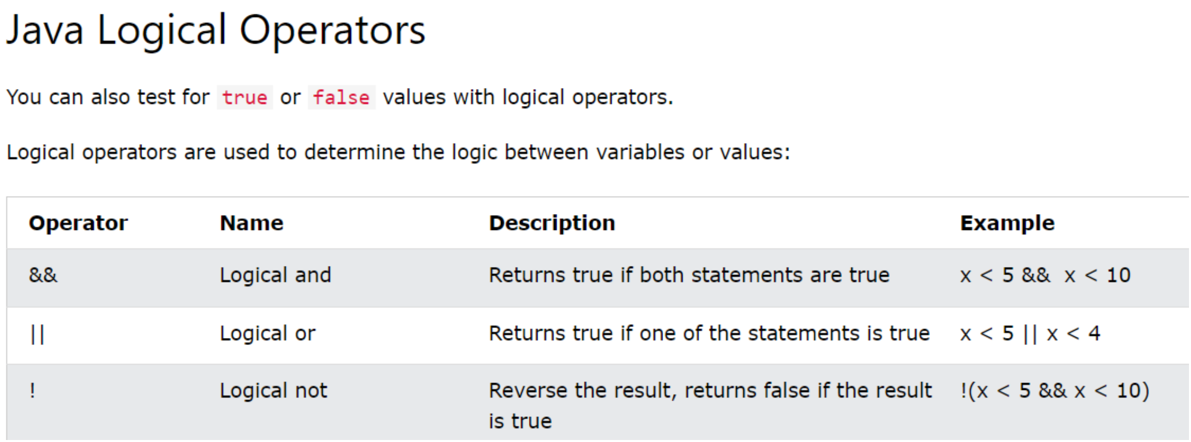
- Comparison operators

Comparison operators are used to compare two values (or variables). This is important in programming, because it helps us to find answers and make decisions.



Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

- Logical operators



Java Logical Operators

You can also test for `true` or `false` values with logical operators.

Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

- Bitwise operators

Comments:

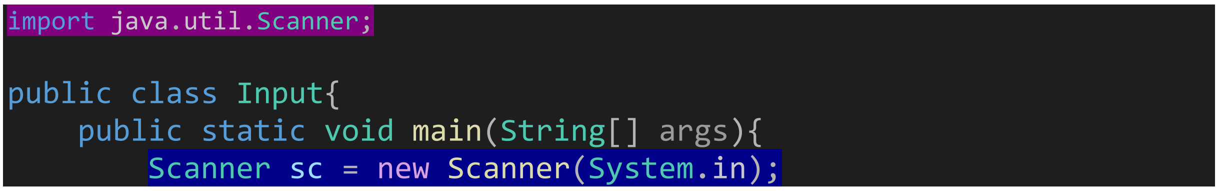
Comments in Java are the statements that are not executed by the compiler and interpreter.

Single line comment: //

Multi line comment: /* */

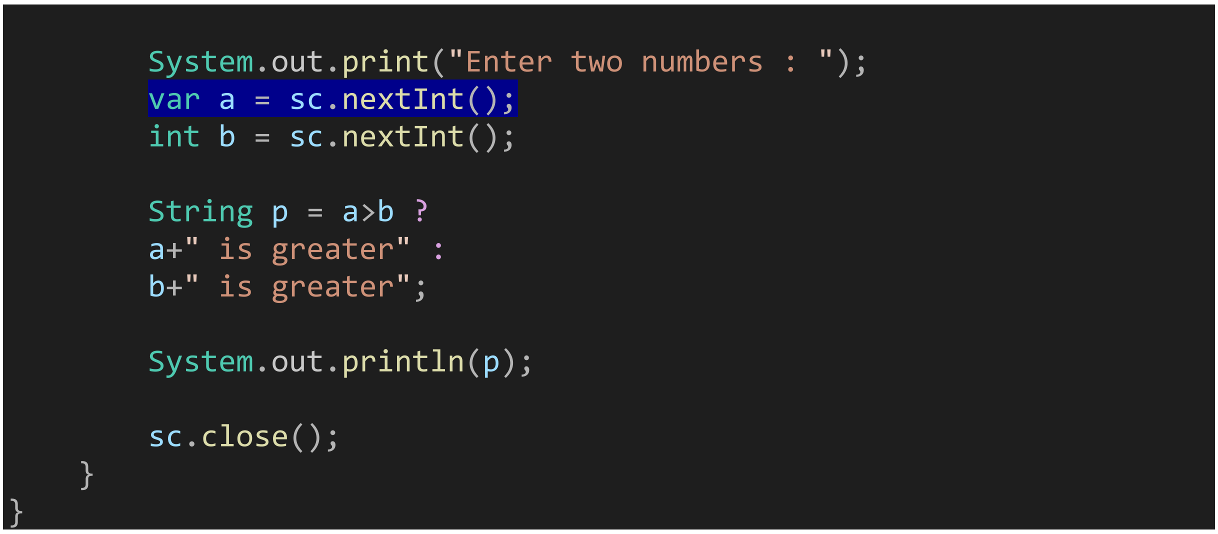
Input in java:

The input is the data that we give to the program.



```
import java.util.Scanner;

public class Input{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
```



```
        System.out.print("Enter two numbers : ");
        var a = sc.nextInt();
        int b = sc.nextInt();

        String p = a>b ?
        a+" is greater" :
        b+" is greater";

        System.out.println(p);

        sc.close();
    }
}
```

Closing the **Scanner** in Java is necessary to release system resources it uses for reading input. This helps prevent memory waste and potential program issues.

Use **scanner.close()** after you're done to keep your program efficient and error-free.

Type Conversion:

Type conversion means we convert one type of data to the other type.

e.g., int to float, float to int

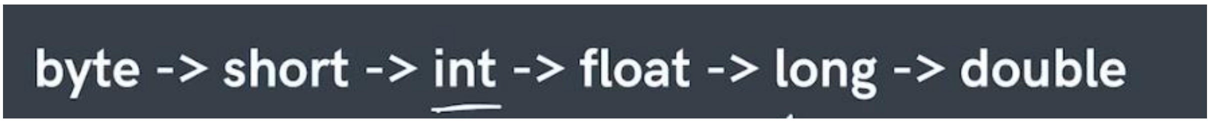
Also known as widening conversion and implicit conversion.

We can convert int to float but we cannot convert int to bool because they are different data types with incompatible value ranges and meanings.(It is not possible for the compiler)

At the time of conversion destination data type size must be greater than source data type.

e.g., int -> long (is possible)

long -> int (not possible) because long size 8 and int is 4 so 8 byte data cannot fit in 4 byte.



byte -> short -> int -> float -> long -> double

// Reverse of this is not possible

If we do this then this error will come:

```
error: incompatible types: possible lossy conversion from
long to int
        int b = a;
                ^
```

type conversion vs type casting in java?

In type casting, a data type is converted into another data type by a programmer using casting operator. Whereas in type conversion, a data type is converted into another data type by a compiler.

Float value assigning:

When we try to assign the value 4.5 to a float variable, it can lead to an error. This happens because 4.5 is considered a "double" by default, and we can't put a double into a float directly.

```
float a = 4.5;
```

For a float assignment, we must show the compiler that it's a float value. To do this, we add 'f' to the end of the value:

```
float a = 4.5f; // Note the 'f' suffix to indicate a float literal
```

This way, the compiler understands that 4.5 should be treated as a float value and assigns it correctly to the float variable "a".

Type Casting:

Type casting is a way of converting data from one data type to another data type.

Type casting in Java is performed by the programmer forcefully to convert one data type to another.

However, it's important to note that during this process, data may be lost if the target type can't **handle** the full range of the source type.

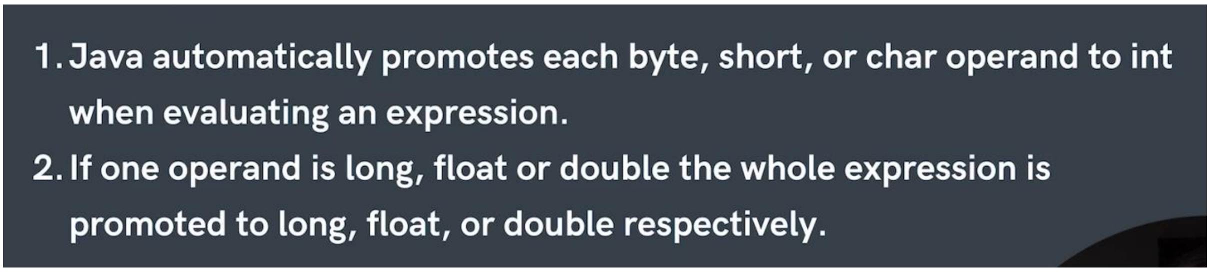
Type Casting Method:

```
float a = 4.598f;
int b =(int) a;

System.out.println(b);  // output = 4
```

It is also known as narrowing and explicit conversion.

Type promotion In Expression:



1. Java automatically promotes each byte, short, or char operand to int when evaluating an expression.
2. If one operand is long, float or double the whole expression is promoted to long, float, or double respectively.

1) In this Example, during the expression evaluation, byte, char, and short values are automatically converted to int. This precaution prevents errors from occurring.

```
byte a = 15;
char b = 'b';
short c = 45;

int result = a+b+c;
System.out.println(result);
```

2) At the time of expression evaluation all data types automatically converted into biggest data type in expression.

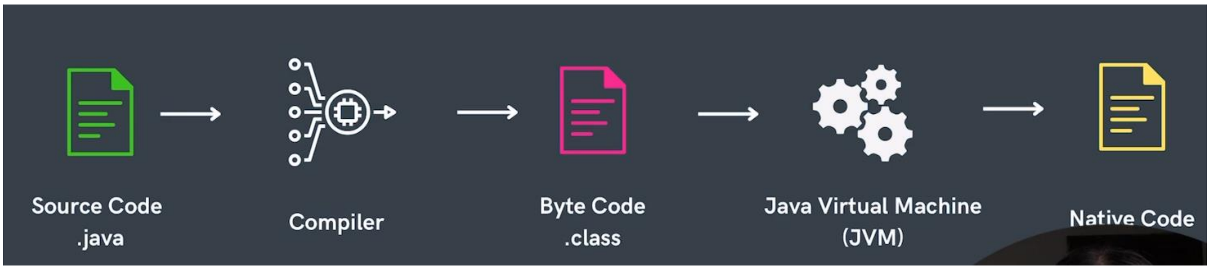
```
        int a = 45;
        byte b = 1;
        long l = 56;
        double d = 45.22;

    /*  long result = a+b+l+d; -> it gives error because
the result is double
         and we cannot store it in long */

        double result = a+b+l+d;
        System.out.println(result);
```

The system ensures compatibility by promoting all values to a common data type to ensure that the expression is evaluated correctly.

How is our code running?



1. **Writing Code:** You write instructions in Java to tell the computer what you want it to do. This code is like a recipe.
2. **Compiling:** Before the computer can understand your code, it needs to be translated into a language it can read. This process is called compilation.
3. **Creating Files:** After compilation, your code becomes a file that the computer can understand. This file has a special name with ".class" at the end.
4. **Java Virtual Machine (JVM):** When you want to run your program, a special program called the Java Virtual Machine (JVM) comes into play. It reads and executes your ".class" files.
5. **Interpreting Code:** The JVM reads the instructions in your ".class" files one by one, like following the steps in a recipe. It knows the Java language and can understand what each instruction means.
6. **Output and Actions:** As the JVM reads your instructions, it performs actions like calculations, displaying text, or interacting with users.
7. **Results:** Your code's instructions are carried out step by step, and you see the results on your screen or get other outcomes depending on what your code does.

In short, you write code, the compiler makes it understandable for the computer, and the JVM interprets and executes the instructions, making your program run and produce results.

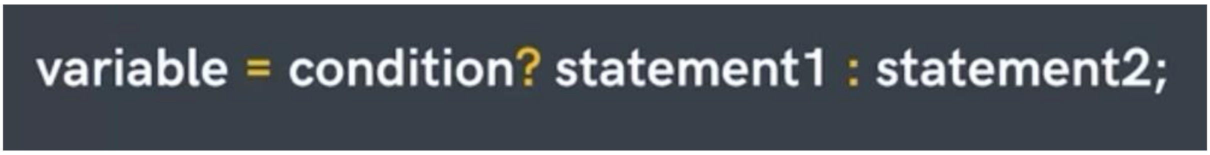
Note: If there is errors in the code then it is detected during the compilation stage, and the compiler informs us about them.

why java is called portable language?

Java is called a portable language because code written in Java can run on different computer systems without modification.

Ternary Operators:

variable = condition ? (if true) : else;



```
variable = condition? statement1 : statement2;
```

```
boolean c = (5>2) ? true : false;
```

Function:

A block of code that performs a specific task, making it easier to organize, reuse, and manage code in a program.

```
public class Basic {
    static void greet(String name) {  // Function with parameter
'name'
        System.out.println("Good Morning, " + name + "!!");  //
Printing a greeting
    }
    public static void main(String[] args) {
        greet("Sumit");  // Calling greet function with argument
"Sumit"
        greet("Saurabh");  // Calling greet function with argument
"Saurabh"
    }
}
```

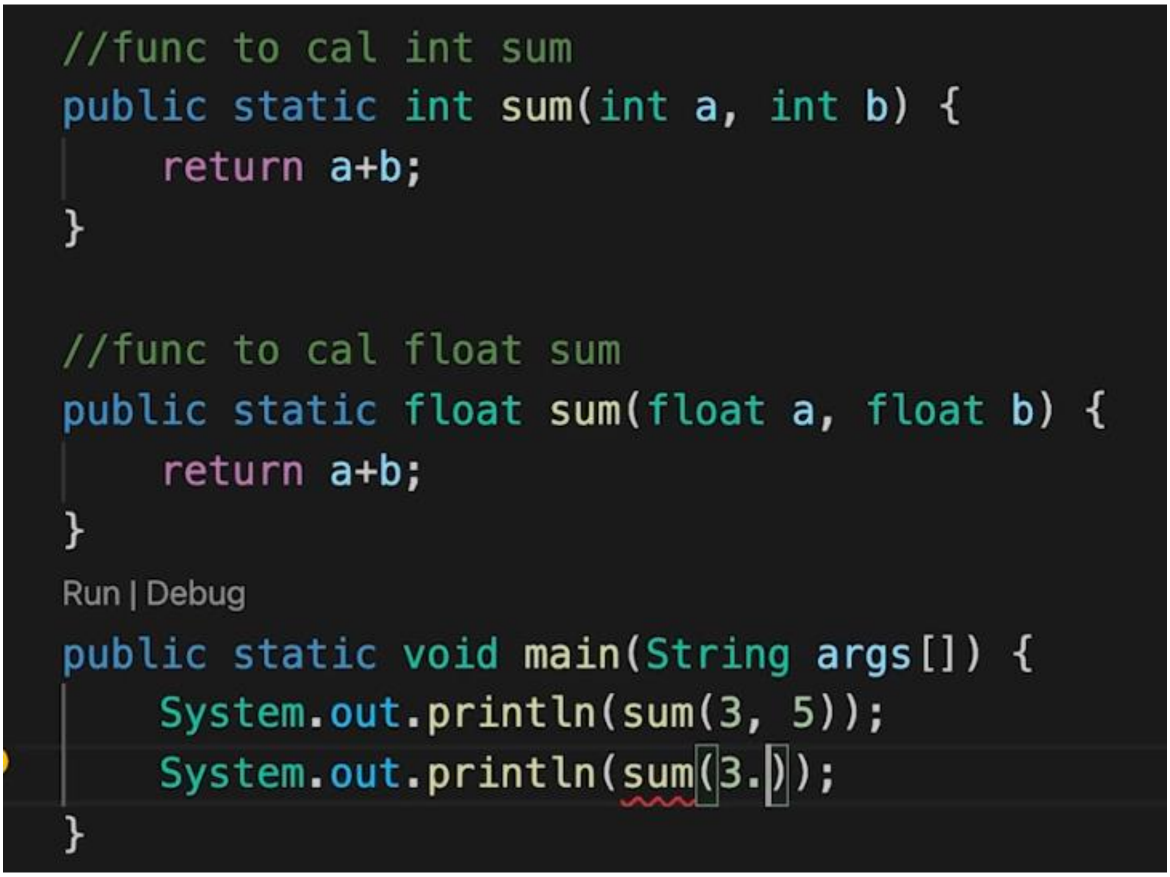
Function Overloading:

Function overloading is a feature in programming where multiple functions with the same name exist in a class, but they differ in the number or types of their parameters (Same Name, Different Parameters).

```
public class FunctionOverloading {
    static void greet(String name) {
        System.out.println("Good Morning, " + name + "!!");
    }

    static void greet(String name , String name2) {
        System.out.println("Good Morning, " + name + " and
"+name2+"!!");
    }

    public static void main(String[] args) {
        greet("Sumit");
        greet("Sumit","Saurabh");
    }
}
```



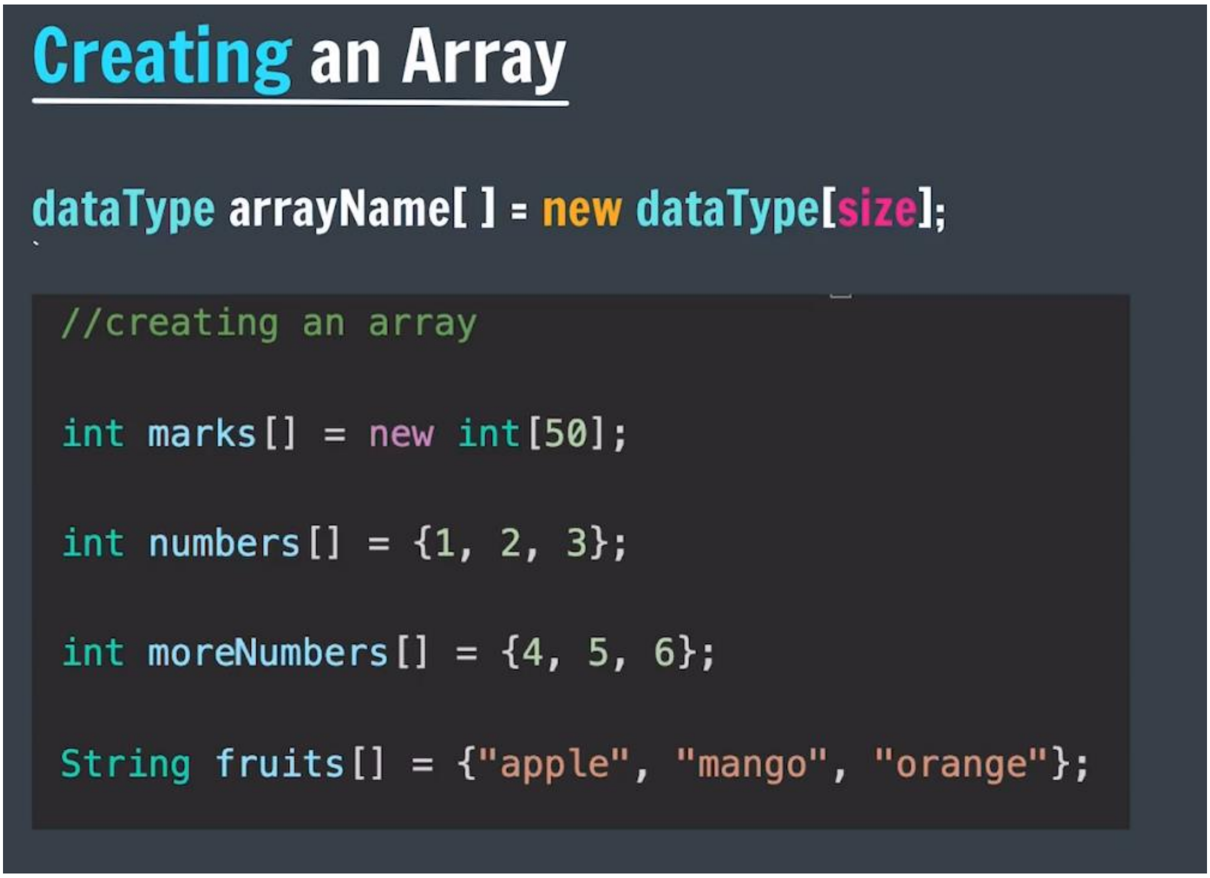
```java
//func to cal int sum
public static int sum(int a, int b) {
    return a+b;
}

//func to cal float sum
public static float sum(float a, float b) {
    return a+b;
}

public static void main(String args[]) {
    System.out.println(sum(3, 5));
    System.out.println(sum(3.));
}
```

Array:

Array is collection of similar elements.



Creating an Array

```
dataType arrayName[ ] = new dataType[size];
```

```
//creating an array

int marks[] = new int[50];

int numbers[] = {1, 2, 3};

int moreNumbers[] = {4, 5, 6};

String fruits[] = {"apple", "mango", "orange"};
```

Note:

-∞ == Integer.MIN_VALUE

∞ == Integer.MAX_VALUE

But to use that you have to include

```
import java.util.*;
```