

JAVA FUNDAMENTALS COURSE

VARIABLES SCOPE IN JAVA



By: Ubaldo Acosta



JAVA FUNDAMENTALS COURSE

www.globalmentoring.com.mx

Hello, Ubaldo Acosta greets you again. I hope you're ready to start with this lesson.

We are going to study the subject of methods in Java.

Are you ready? Come on!

RESERVED WORD "THIS"

Variable that point to an object of Arithmetic type

obj = 0x333

Constructor example (Class):

```
//Constructor with 2 arguments
//We use the "this" operator
Arithmetic( int a , int b){
    this.a = a;
    this.b = b;
}
```

Object created

Arithmetic
(0x333)

a = 2

b = 1

⋮

this

Abstract Test Class:

```
// Creation of the obj object
public static void main(String args[]){
    Arithmetic obj = new Arithmetic(2,1);
    obj.add();
}
```

Sometimes a method needs to refer to the object with which we are currently working. For this task Java added the reserved word `this`. The word `this` is an operator which allows us to access the current object (the class with which we are working), and it will help us, among other things, to access the attributes or methods of a class. With this we can make a difference between the arguments received in a method and the attributes of a class.

As we can see in the picture, we have a code that has an attribute called `a` and `b`, and also the constructor of the class receives two arguments called `a` and `b`. To make a difference between these two variables (attributes of the class and the arguments received in the method) we can use the word `this` as follows:

```
// Constructor with 2 arguments. We use the operator this
Arithmetic (int a, int b) {
    this.a = a;
    this.b = b;
}
```

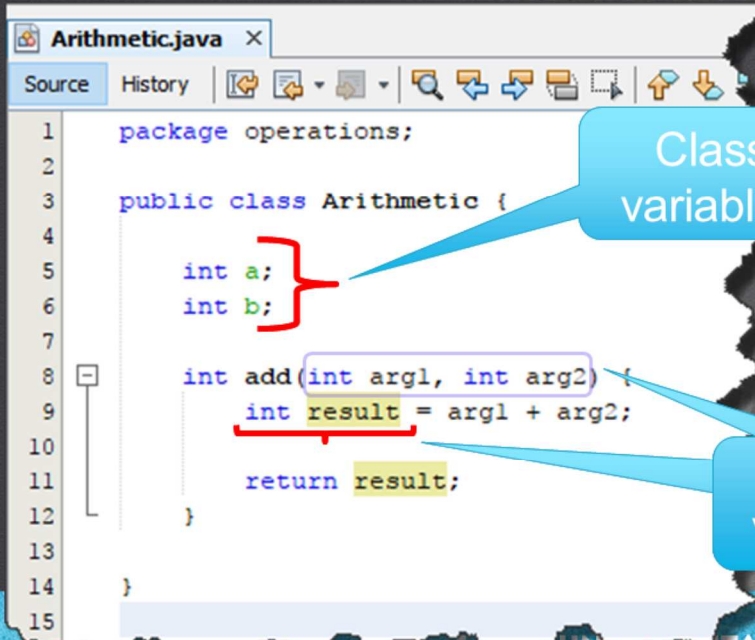
As we can see, to make a difference between the values received in the Constructor and the attributes of the class, we can use the `this` operator. This is just an example of the use of the `this` operator, but basically it allows us to access the attributes and methods of the current object with which we are working.

This can be confusing at first, since we are working with the code of our class, that is, the template, and this code will not be executed until we have created an object and let's call this constructor, it will be at this moment when we send the arguments and the constructor receives them and processes them. So we must get used to thinking about two moments, the creation of our classes or templates, and the creation of our objects, which will be when the code we have programmed in our classes or templates is actually executed.

Although the use of the `this` operator will sometimes seem redundant, it is a good practice to use it to refer to the attributes of the class in which we are working, since by reading our code we will quickly recognize which variables are attributes of a class and which are not. they are.

Within the constructors or methods of a class, the `this` operator will always reference the object that was invoked. If we look at the code, both the received argument and the class attribute are called exactly the same, so the argument about the class attribute takes precedence, this is known as concealment of the class attribute. And to solve this problem, it is enough to use the `this` operator before the variable, just as if we were accessing the attribute of a class by means of the dot operator.

VARIABLE SCOPE



The screenshot shows a Java IDE with a file named `Arithmetic.java`. The code is as follows:

```
1 package operations;
2
3 public class Arithmetic {
4     int a;
5     int b;
6
7
8     int add(int arg1, int arg2) {
9         int result = arg1 + arg2;
10        return result;
11    }
12
13
14 }
15
```

Annotations in the image:

- A red bracket groups `int a;` and `int b;` with a callout box labeled **Class variables**.
- A red bracket groups `int result` in the method with a callout box labeled **Local Variables**.

Class Variables :

- They can be used in any method of the class
- They are initialized with default values

Local Variables :

- They can be used only in the method they are defined
- They must be initialized

JAVA FUNDAMENTALS COURSE
www.globalmentoring.com.mx

In Java we have different types of variables, such as Class variables and Local variables.

Depending on where the variable is defined, it will be the duration of the variable, and this is known as the Scope of a Variable.

If we define a variable as an attribute of a class, this variable is known as a Class variable, and it will exist for as long as the object exists in memory. These variables are initialized with their default value automatically, for example, an integer type is initialized with the value of 0, a variable type `bool` with the value of `false`, and an Object type with the value `null`.

On the other hand, local variables are any variable defined within a method, including the arguments that a function receives. These variables have a shorter life time, since they are created when the method is executed and they are eliminated from the memory as soon as the method has been executed. In addition, local variables need to be initialized with some value, otherwise the compiler will mark an error due to the lack of initialization of this type of variables.

The local variables hide the class variables, and if we want to use the class variables in a method that has defined local variables with the same name, then we must use the prefix `this` in order to access the class variables instead of the variables. local attributes.

ONLINE COURSE

JAVA FUNDAMENTALS

Author: Ubaldo Acosta



JAVA FUNDAMENTALS COURSE

www.globalmentoring.com.mx

