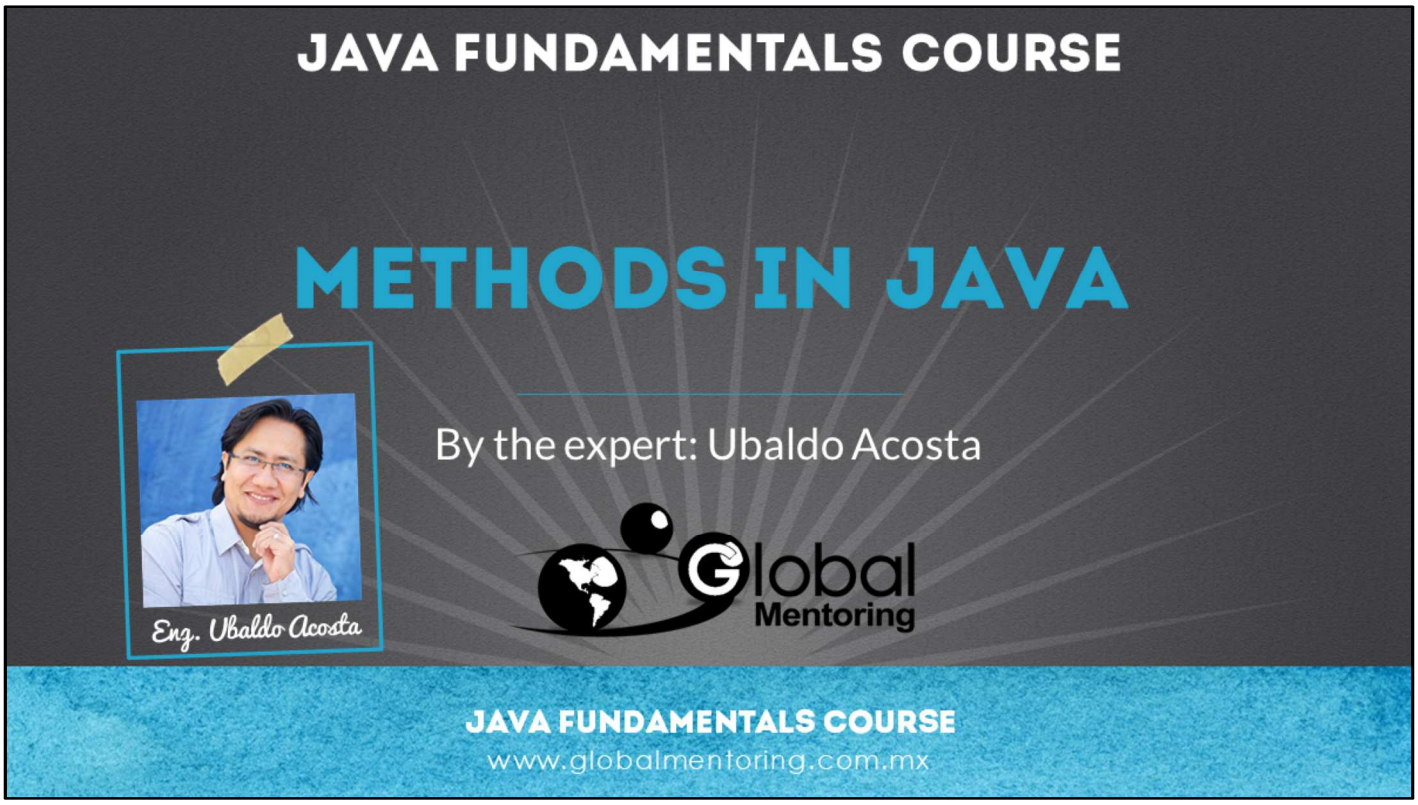

Hello, Ubaldo Acosta greets you again. I hope you're ready to start with this lesson.

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

Are you ready? Come on!

METHODS IN JAVA

General definition of a Java method:

```
class ClassName{

    returnType methodName(arguments_list){
        //Body of the method
    }
}
```

Example of a Java method:

```
public class Arithmetic{

    int add(int a, int b){
        // make the addition and return the result as an int type
        return a + b;
    }
}
```

As we mentioned earlier, a class consists of attributes and methods. The subject of methods is very broad, however in this first introduction we will see how we can create and use the methods we declare.

In Java, it is necessary to create a method it must be inside a class. This is because Java is an object-oriented language, and therefore one of the practices is that the methods must be part of a class to be correctly defined.

On the other hand, we can observe that the general definition of a method. First of all we can observe the type that will return this method. Some methods do not return any particular value, so in Java we will use the reserved word void for this case where no value is returned. We can specify primitive types, or of type Object, including those that we believe, like the type Person that we created previously.

After the type, we specify the name of the method, we must remember that Java is a case sensitive language, so we must respect the name of the method and use it as it is written. It is common for the name of the method to be an action, for example insert Person, delete Student, etc. It is also common in Java to use the camel notation to write the names of both variables and methods, and in general of the names we write. This notation consists of writing the first letter in lowercase and then the first letter of each word must be uppercase. As we progress, we will observe examples of this type of notation throughout the course.

After the name, we specify the list of arguments, specifying first the type and then the name of the argument. For each argument that we want to add, we must separate them by a comma. If the method does not have a list of arguments, we will only see that parentheses are opened and closed ().

Finally we will write the body of the method in braces {}. Everything we write inside these keys is what will execute the method. We can see the example of the sum method. In this case, because the method returns a value, we use the reserved word return followed by the value to return to the method that made the call to the sum method.

An important concept when defining methods in Java, is known as the signature of the method. In the case of the sum method, the signature of the method would be: int add (int, int); // signature of the method.

That is, the method returns an integer value, its name is sum, and it receives two arguments of integer type. So the signature of this method tells us how we can use it, what value to expect and also whether we should send arguments or not, and what type these arguments are. Later we will put this example into practice.

CALLING A METHOD

General call of a method in Java:

```
// Create an object of the class to call its method
TipoClase objeto = new TipoClase();

//We call the method, sending arguments if required
//If the method returns a value we can receive it according to the type
returnedType result = object.methodName(arg1, arg2, etc);
```

Example of calling a Java method:

```
//We create an object of the Arithmetic class
Arithmetic a = new Arithmetic();

//We call the sum method and receive the returned value
int result = a.add(5, 3);
```

Once we have added a method to a class, we can make use of it by doing the following:

1) We must declare an object of the type of the class that has the method that we want to use.

2) By means of the declared object, we use the dot operator (.) And after this operator we write the name of the method that we want to call. Recall that in Java, the names are case sensitive, so we must make the call to the method as we have written, respecting the uppercase and lowercase letters that we have used to define the method.

3) After the name of the method, we open and close parentheses. If the method does not receive any argument, we only open and close parentheses, however if the method was written to receive arguments, we must provide the arguments of the expected type, separating each of them by commas.

4) Finally, if the method was written to return a value, it is optional to receive it with a variable of the type that returns the method. We should note that the type returned by a method does not have to be the same as the types of arguments received. For this we must observe the signature of the method that we are going to use. If we remember, the signature of the sum method would be:

int add (int, int); // signature of the method

So we observed that the method received two arguments of integer type, but it could be any other type, it is called adding and returning an integer type, but it could also be any other type. Later we will put this example and several others into practice so that the declaration and use of methods becomes clearer and clearer.