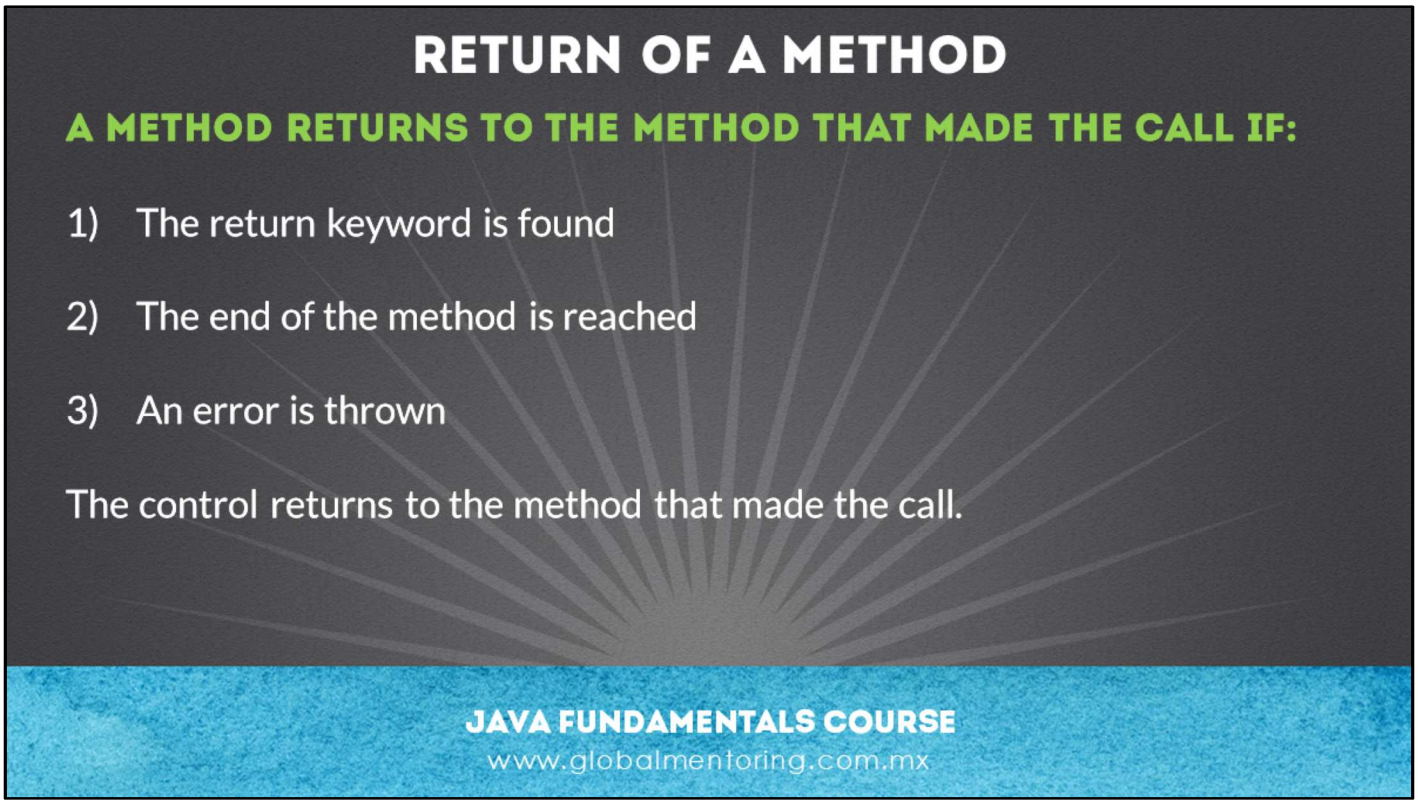
Hello, Ubaldo Acosta greets you again. I hope you're ready to start with this lesson.

We are going to study the topic of return of a methods in Java and the return keyword in Java.

Are you ready? Come on!



When we use methods in Java, there are 3 ways to exit or terminate the method that is running.

The first option is to use the return keyword, and once this instruction is executed, the control is returned to the method that made the previous call. The return keyword may not return any value, that is, only place the return keyword. The other option, which is how the return keyword is normally used, is adding a value, which corresponds to the type defined in the signature of the method in question. Later we will see examples of the use of the return keyword in more detail.

The second option is when we reach the end of a method, there is no need to put the return keyword in a mandatory way, but once the end of the method is reached, it is implicitly returned and with it, the same result is obtained as if we put the return keyword without arguments.

The third way out of a method is if an error occurs, this causes the method to terminate abnormally, and whether this exception is processed or not, this will cause the execution of the method to end. The handling of exceptions in Java is a subject that will be discussed in another course in detail.

Next we will see some examples of the use of the word return.

WORD RETURN AND PRIMITIVE TYPES

Test Class:

```
public class ReturnKeywordTest {

    public static void main(String[] args) {
        int result = add(3,6);
        System.out.println("Result: " + result);

    }

    public static int add(int a, int b){
        return a + b;
    }
}
```

www.globalmentoring.com.mx

As we can see in the code, we are creating a method called add, which receives two arguments. The static word will be shown later, in this case you just have to add it so that the code can work correctly, and then we will explain it in detail.

Continuing with the definition of the add method, the signature of this method is: public static int add(int, int)

What we are interested in here is to identify which type of return is indicating this method, which is int type.

In Java we can only return a simple value, however this value can be a primitive type or object type. And as part of the object types can be an array, a list or a collection of elements. In this way Java can return only one type, this type can be a collection and thus return the reference to an object of type Collection. The subject of arrays will be studied later, and the theme of collections in another level.

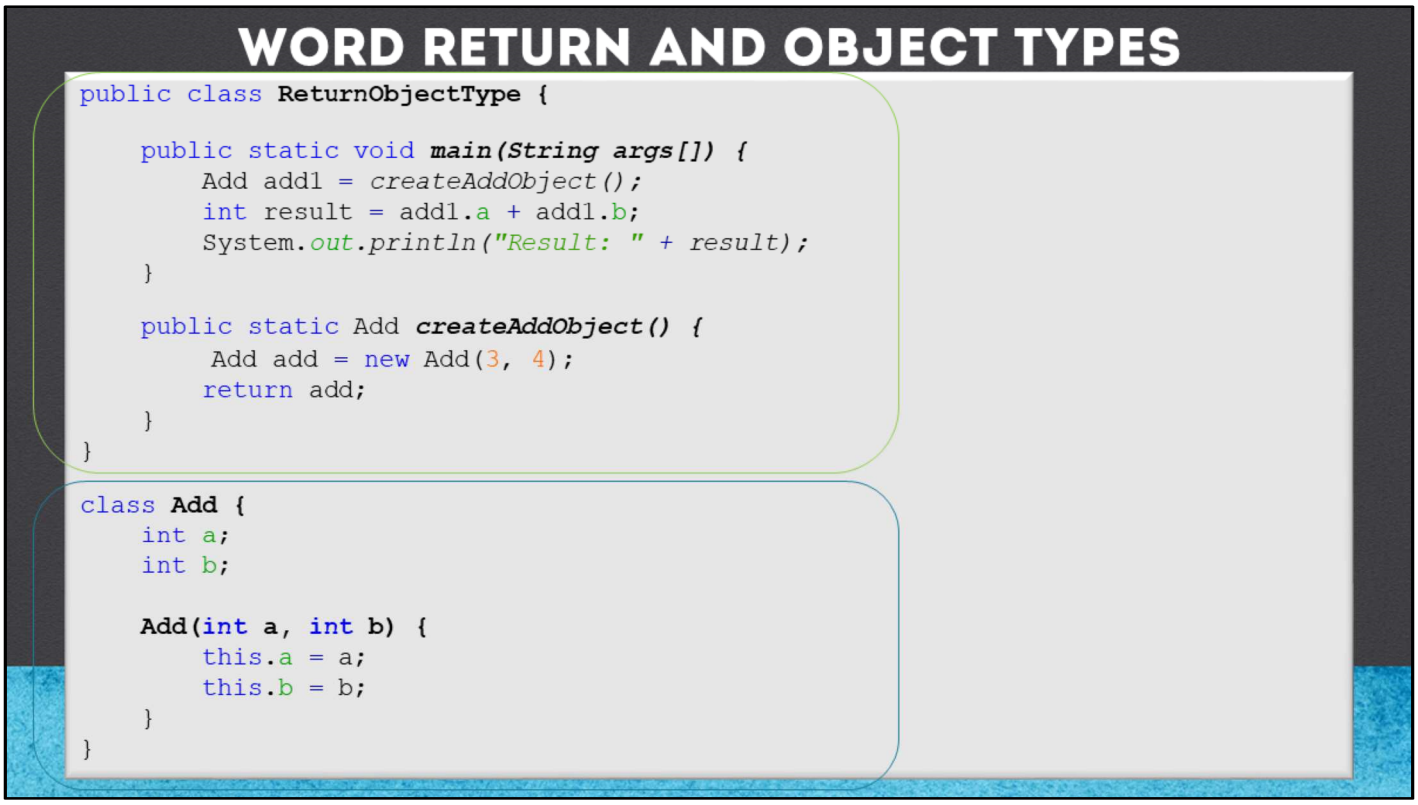
Once we have defined the add method, we can observe that when it receives the values a and b, we use the word return to return the result of the add of a plus b.

We can use parentheses or not, this is optional, but what we should know is that what is returned is the result of the expression a + b, not the values separately, because as we discussed Java only returns a simple value, and this particular method is indicating that what will return is a value of type int, which is a primitive type.

Once this method is finished, the control is returned to the method that made the initial call, which is the main method in this case, and the value of the add is assigned to the result variable.

This result variable must be of the same type that the add method returns, therefore the result variable is of type int.

Later we will perform an exercise using the word return with primitive types.

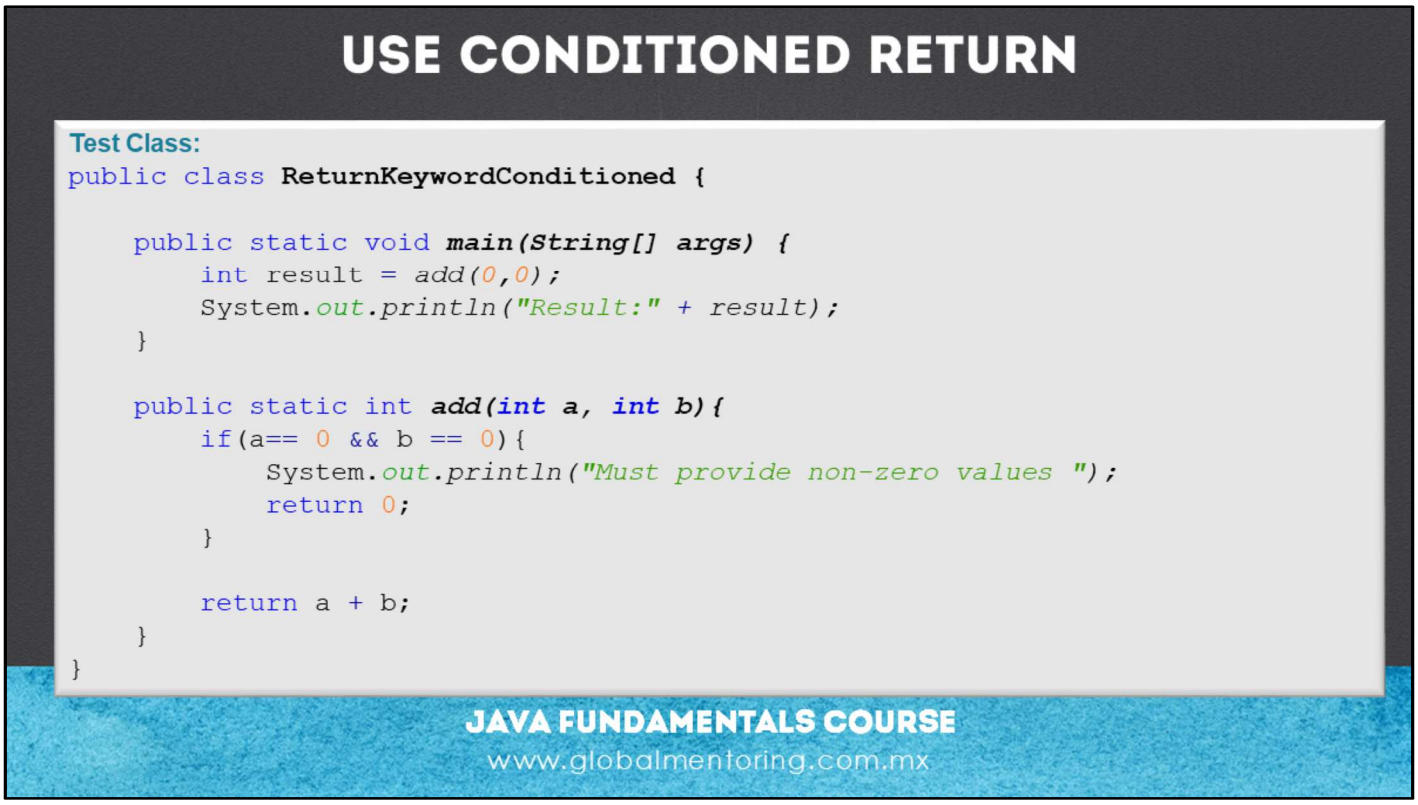


As we have said, the word return can only return a simple type, however it can be of primitive type or Object type (whatever the type). In Java all classes inherit from the Object class, so this allows us to return any Java type either from the libraries already created or some type (class) created by us.

In this example we are creating one class within another, in order to exemplify how to return an Object type. In Java we can declare several classes within a file, but only one of them can be public, and the public class corresponds to the name of the class file. Therefore, the Add class does not have the public modifier, unlike the ReturnObjectType class.

In the ReturnObjectType class we define the main method, and a method called createAddObject, this method is really not necessary, we have created it only in order to demonstrate how we can return an object type from a method, which can be directly in the line of code where the word return is found, or we can create a variable of type Add, and make the respective return of the variable of type Add created previously.

In other level we will see that when we say that the method must return the same type defined in the signature of the method, it is really a value of the same type or a value compatible with this type, since the issue of inheritance in Java, as well as the management of interfaces, allows us to declare a hierarchy of Classes, which generates compatible types among the class hierarchy, so we will see this topic in detail in the next level. However, it is important to know that the signature of the method defines a type and the word return must return a value of the same type or a compatible value according to the class hierarchy to which that type belongs.



Finally we can see that the word return can exist several times in the same method, since we can have code that is executed conditionally.

In the same way, the code shown is only to exemplify the use of the word return in case operand a and b are zero, in which case we can assure that the result is 0, and therefore we can make the return of the value 0 instead to perform the respective operation. We know that this code can be implemented in other ways, however it is to exemplify the use of the word return using a condition with the help of the if conditional block.

If any of the operands is non-zero then we perform the respective operation and return the value to the main method, very similar to the exercise previously created.