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

We are going to study the topic of overriding methods in Java.

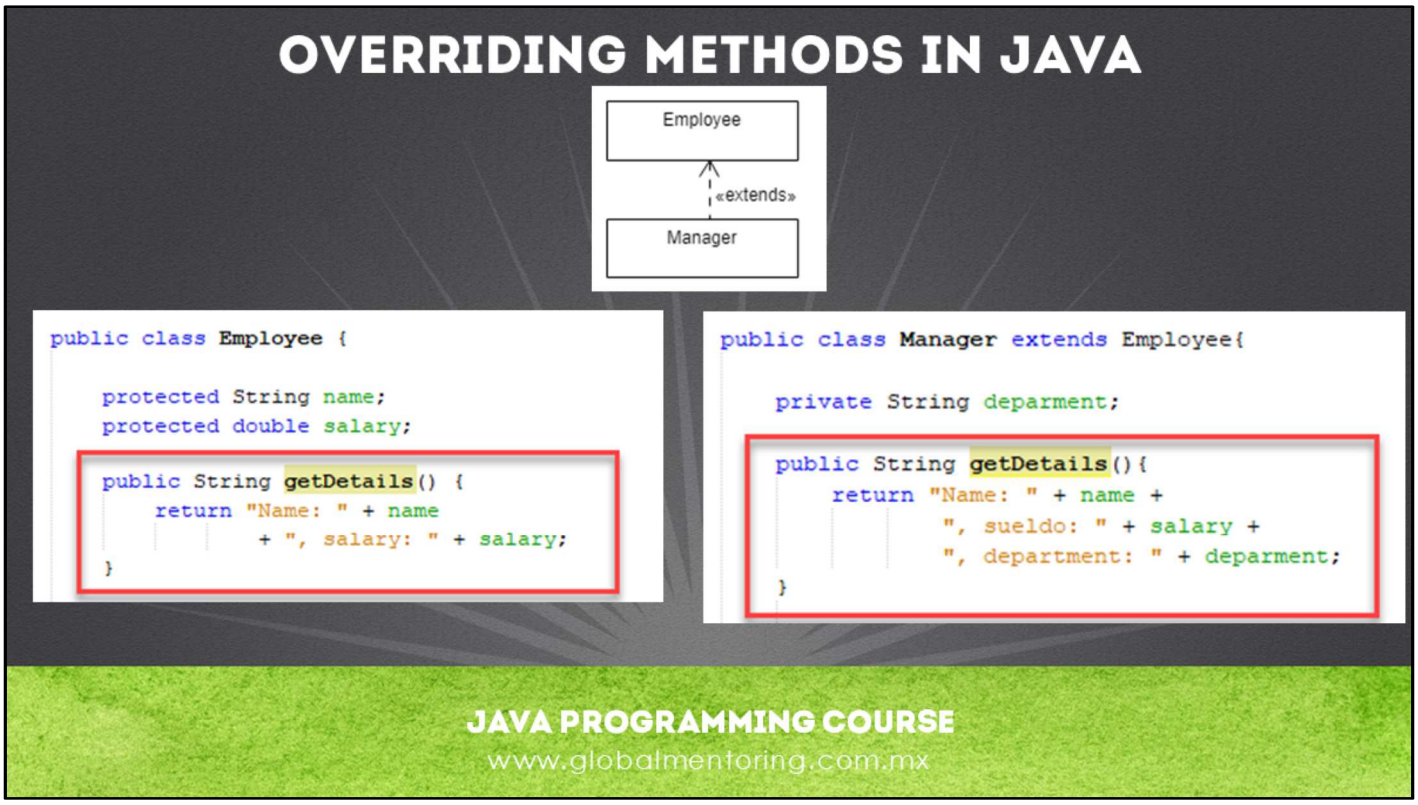
Are you ready? Come on!

Let's now study the concept of overriding methods in Java. Let's first see the concept as it happens in the real world and later we will explain how to apply this concept in Java code.

The overriding of methods in automatic has to do with the concept of inheritance, and it is basically the modification of a method from a daughter class to a method inherited by the parent class.

For example, in real life, we can observe that talent, for example singing, is inherited from mother to daughter, however, although the daughter may sing similar to the mother because she has inherited several characteristics of the mother, not the mother. It will do exactly the same, therefore, adapt the song to the daughter's form and her ability to exploit this talent.

In the same way the overriding of methods has to do with the modification of a method by the daughter class, of some method that has been inherited by the parent class, hence the name of overriding, since the son rewrites again the behavior of the legacy method to adapt it to their own needs.



As we have said, the overriding of methods applies if there is a parent class and a daughter class. As we see in the figure, we have two classes, a Parent class called Employee, and a daughter class called Manager. And we see an example of code of both classes.

On the one hand, the parent class declares 3 protected type attributes, that is, they are inherited and accessible directly by any daughter class. And also the Employee class defines a method called getDetails(). This method basically returns a string concatenating each of the attributes of the class, this is very similar to what the toString() method does, which we will study in detail later.

On the other hand we have the daughter class called Manager, and as we see it extends from the Employee class, therefore it inherits the protected type attributes and also the public method called getDetails(). However, the Manager class defines an attribute that is not defined in the parent class, called department, and therefore this attribute will not be displayed if we directly use the getDetails() method of the parent class. This is broadly the reason why in object-oriented programming it is necessary to overwrite methods, and in this case in particular the Manager class redefines or overrides the method getDetails(), and with that what it does is add the behavior for the method to include the department attribute, and thereby complete the desired operation from the Manager class.

So as we have seen, a subclass can modify the inherited behavior of a parent class.

The signature of the method overwritten by a subclass must be the same as the parent in: name, return type and list of arguments, this is a rule to enforce and respect the concept of overriding.

In the signature of the method the only thing that can vary is the access modifier, but an overwritten method can not be less accessible than the method that overwrites, for example, if the parent method is public, it can not be changed to protected in the child, and much less to private.