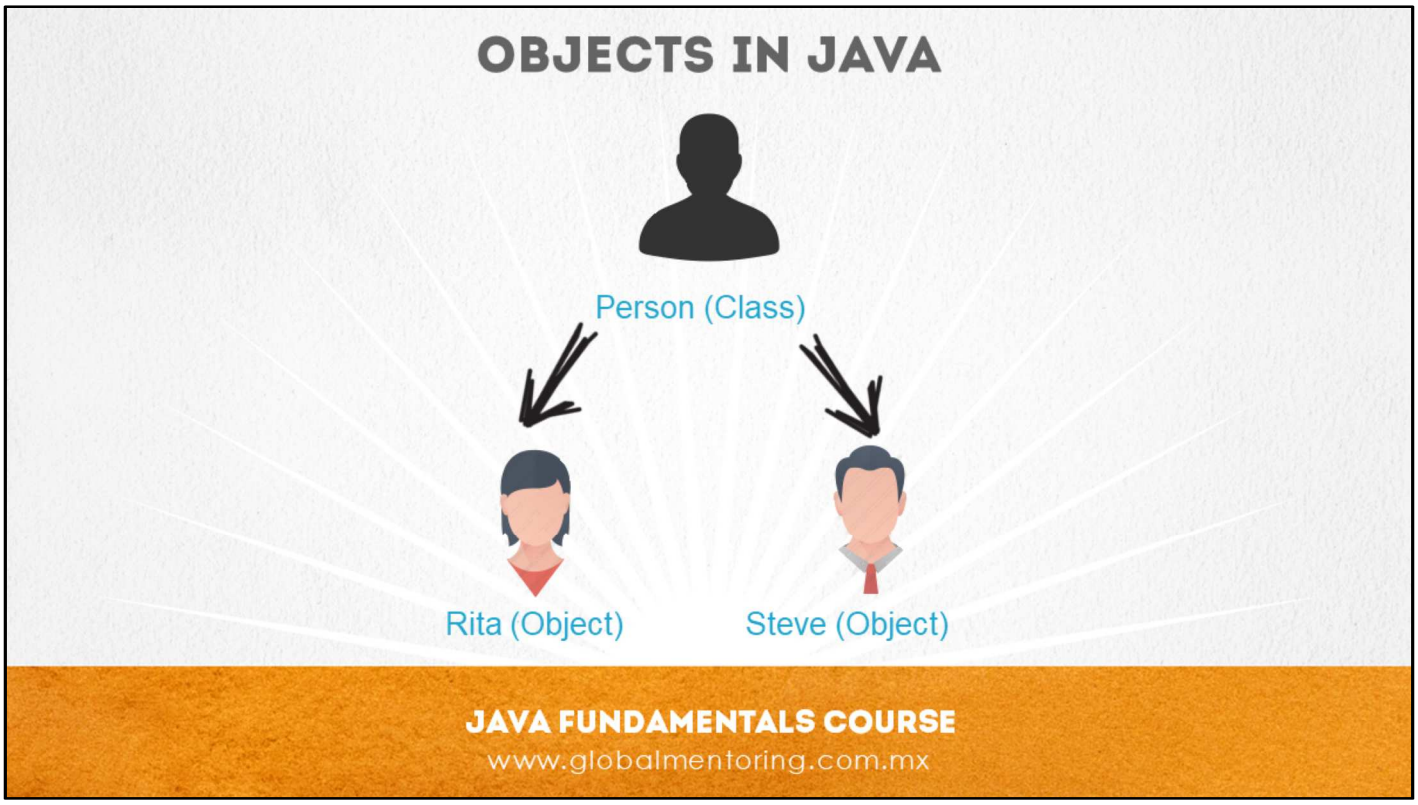
Hello, Ubaldo Acosta greets you. Welcome or welcome again. I hope you're ready to start with this lesson.

We are going to study the subject of Objects in Java. This is the basis of object-oriented programming.

Are you ready? OK let's go!



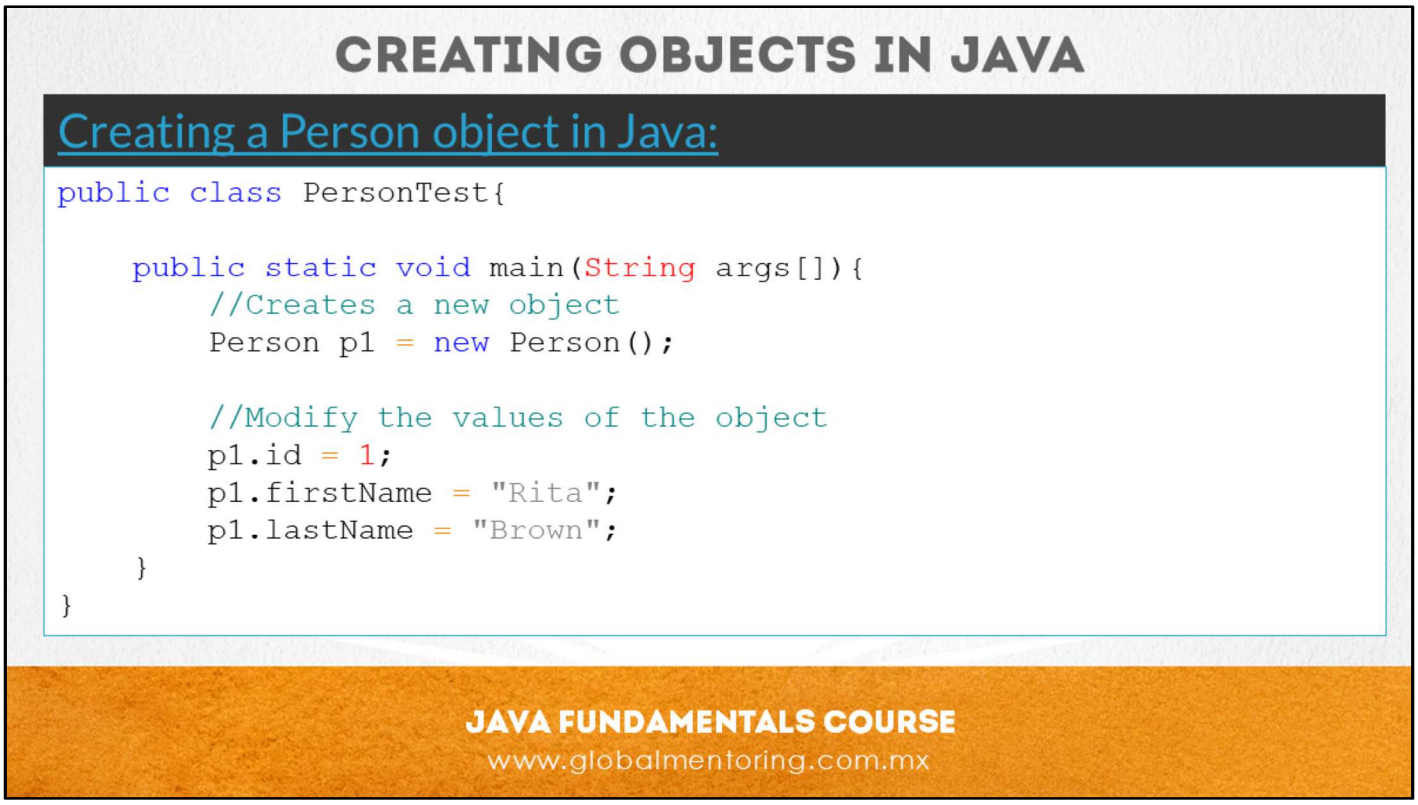
An object is the instance of a class, that is to say that from our template, it is possible to create concrete objects with which we can work, that is, change its values and call its methods.

Later we will see that there are cases in which we can work directly with the class, this is called static context, but for now we will be studying the dynamic context, that is, we need to create an instance of a class to be able to work with them.

As we can see in the figure, we have the Person class, and from this template we can create concrete objects to be able to work with their characteristics and attributes. For example we have created two objects of type Person, and each one has similar characteristics but with different values.

The characteristics are for example: Name, Surname, Color, Gender, schooling, etc.

And each one of the objects is assigned these characteristics, in this way the class serves us only to indicate what characteristics the objects we are going to create will have. Let's see now the syntax to create an object.



To define an object we are going to use the reserved word new. This word in Java means in simple terms that we are going to create a new variable of type Person. To create an object we will use a line of code as follows:

Person p1 = new Person( );

After this line is executed, variable p1 will be able to access an object of type Persona. Each time that the word new is called, a new object of type Person will be created, and it will contain its own values. For example:

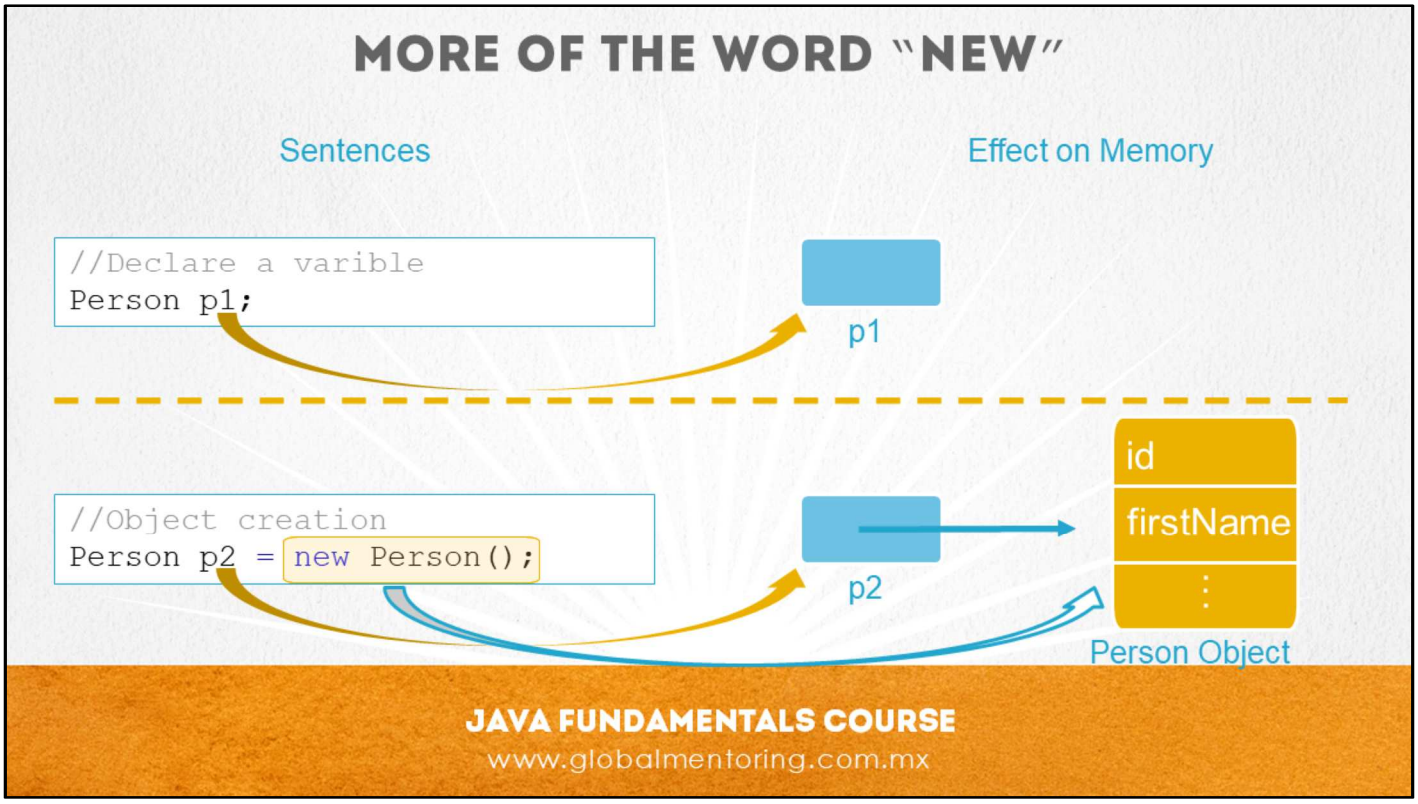
Person p2 = new Person( );

This new variable p2, points to a new object of type Person, and therefore will contain values other than the variable p1. Thus, each new object will contain different values in the attributes of id, firstName, and lastName. The syntax new Person( ) refers to the name of the class from which we want to create a new instance, that is, an object. In addition, we are making use of the concept of constructor of the Class, this concept we will study later, here we should only notice that to call the constructor of the class the name of the class is put and later parentheses are used, in this case a constructor without arguments, but we will see later that we can send arguments when creating an object.

The name of the variable can be any that we choose, but it must not have special characters, or spaces. To access the attributes of our classes we will use the operator. Through this operator the name of the object is linked to the name of the attribute of the class. For example, to assign the Armando value to the attribute name of object p1, it would be as follows:

p1.firstName = “Rita";

In later lessons we will see in more technical terms what the word new does, but for this lesson it is enough to know that it allows us to create new objects.



The word new requests memory space dynamically in order to store an object. With the syntax generally as follows:

variableThatStorageReference = new TypeClassToInstantiate( );

In the figure we can see that first we declare a variable of type of the Person class, called p1. We can see that the effect this has on Java memory is that space is created for this variable called p1. But this variable does not yet have any assigned value, since only the variable was declared.

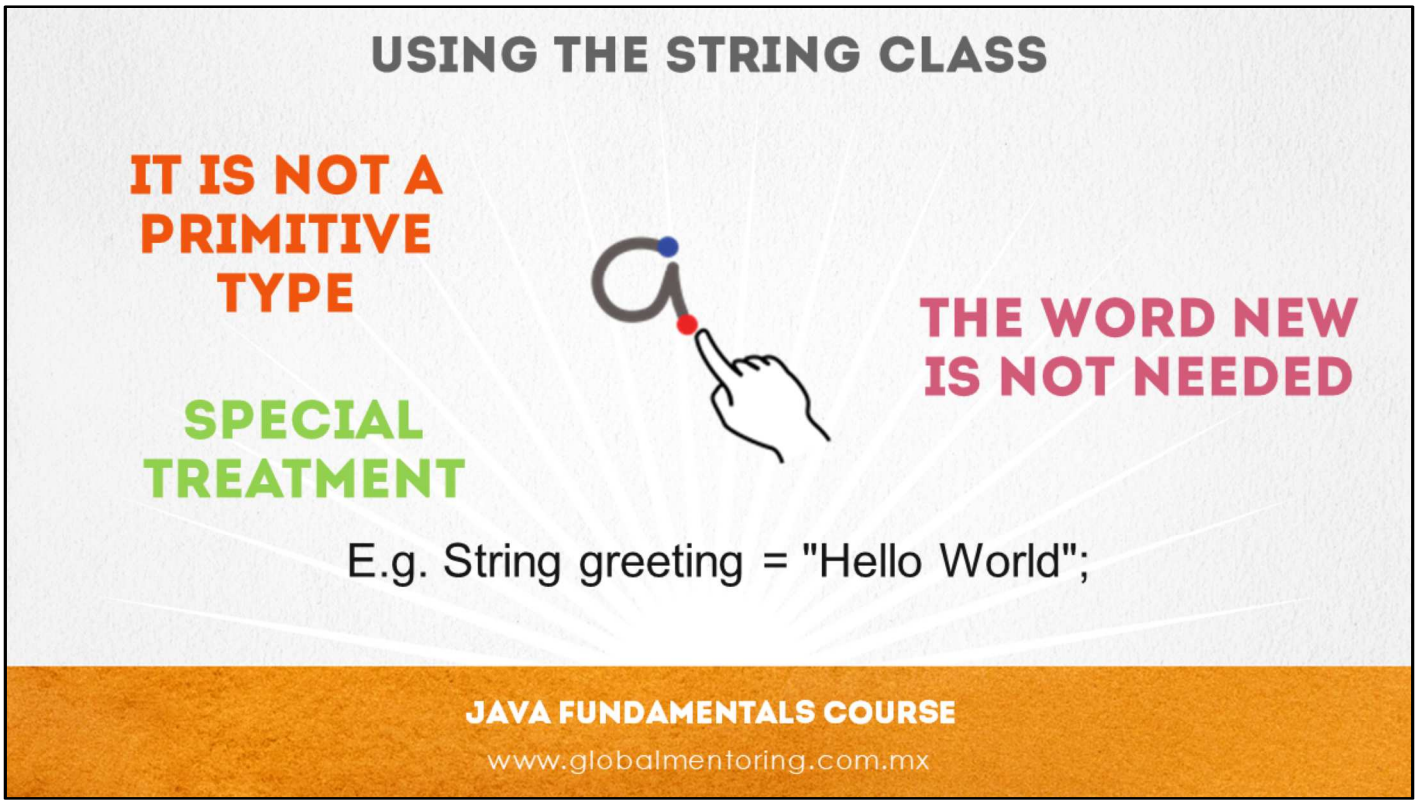
In the second sentence, we can observe, that in addition to declaring a variable of type Person called p2, it is assigned the reference of an object of type Person. For the creation of this person type object the word new is used, followed by the name of the class from which we want to create the object and finally the parentheses is the type of constructor that we are going to use to create this object.

The concept of constructor will be studied in detail later, but we only think of the constructor as the way in which we want to create our object in an initial way, we can create for example a person-type object already with values such as name, surname, etc, or as in this case, we can simply create the Person type object, without providing the first and last name values and then we can change and / or assign these values to the attributes of our Person type object.

We must clarify that the primitive types such as int, long, etc, are not objects, therefore to declare variables of these types you do not need to use the reserved word new.

Recall that the word new what it does internally is to request memory space for the creation of a new object, as many objects as we need, however, the memory is finite, and therefore we could receive an error when creating too many objects in execution time of our program. In later topics we will see the handling of errors and how to deal with the most common, such as the lack of memory in Java.

In conclusion, a class defines a new type, and an object is an instance of this new type that has already been created previously. We can create as many objects of the type created as we need, this with the help of the word new.



If we remember in previous lessons, we mentioned that the String type is a Java class, but with a special treatment. Because this type of data is used repeatedly in our programs, it is possible to omit the use of the word new, and therefore simplify the creation of a variable of type String.

Then we can create a String type object in the classic way:

String name = new String (“Rita");

But we will use it more commonly without the use of the word new, this is the simplified method:

String name = “Rita";

Both options are valid, but we will use the second option more because the code is simplified. Just do not forget that the String type is a class and not a primitive type. This type of data is the only exception to the rule for creating objects using the new operator. All other classes must use the new operator in order to create an object.

We will begin to use this type in our following examples, and thus quickly become familiar with the use of the String type in Java.