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

We are going to study the topic of JavaBeans in Java.

Are you ready? OK let's go!

JAVABEANS IN JAVA

Example of a Java Bean definition in Java:

```
package javabean;

//1. Implement the Serializable interface of the java.io package
public class PersonBean implements java.io.Serializable{

    //2. Each property is of private type
    private String name;
    private int age;

    //3. Always have a Constructor without arguments
    //Other Constructors are optional
    public PersonBean() {
    }

    //4. For each property add a get and set
    //or just one of them if it does not require both
    public void setName(String name) {
        this.name = name;
    }

    //The other methods continue. Other methods are optional
}
```

A JavaBean is a pure Java class, which follows certain minimum rules to be able to be called JavaBean.

The idea is that these classes can be used by other classes without needing to know in detail the content of the class. That is why it must comply with certain rules, since frameworks or technologies or other classes will use our JavaBeans, and therefore must already comply with certain characteristics to be used.

The basic characteristics that a JavaBean must fulfill are:

1) You must have an empty constructor. Regardless of whether you define other constructors, you must have an empty constructor mandatory. This is because the other technologies that make use of our JavaBeans must be able to instantiate our classes, and for that they will not get into complications, but they will create an instance of our JavaBeans using the empty constructor.
2) The attributes must be private. With this we apply the concept of encapsulation that will be basic so that the class that makes use of our JavaBeans are forced to use the methods of type get / set and not directly the properties of our class.
3) Each property must have its get / set method. And in case of being of type boolean instead of get will be as we have already commented previously for the types boolean. It is not necessary to have both methods (get / set) but you can define only some of them as needed, for example a read-only attribute will only implement the get method, and an attribute that we want to be write-only will add the set method. This is how we guarantee in conjunction with point no. 2, that we will not directly access our attributes, but will use the get / set / is methods.
4) It will implement the Serializable interface of the java.io package. It should be mentioned that there are many interfaces called Serializable, so we must ensure that the java.io.Serializable interface is implemented. This interface does not have methods, so it does not force us to implement any method, however it will serve us for more advanced issues that we will see later, but basically it is to serialize our object if necessary. This broadly means that the technology or framework used by our class in case you need to transmit our class over the network, you can serialize our object, that is, convert it into ones and zeros, and then once we receive our object, apply the process Inverse, known as deserialization, which means to take those ones and zeros and become again the original JavaBean object. with this brief explanation we are ready to create JavaBeans classes and comply with the basic standard that Java requires.

It should be noted that we should not confuse JavaBeans with Enterprise JavaBeans. The latter are server-side technology and we will study them in the course of Java Enterprise Edition. JavaBeans can be used in any version of Java, so they are simpler and we do not require the enterprise version to be used, of course the Enterprise JavaBeans provide us with many more features than JavaBeans. Therefore, we should not confuse these two concepts.