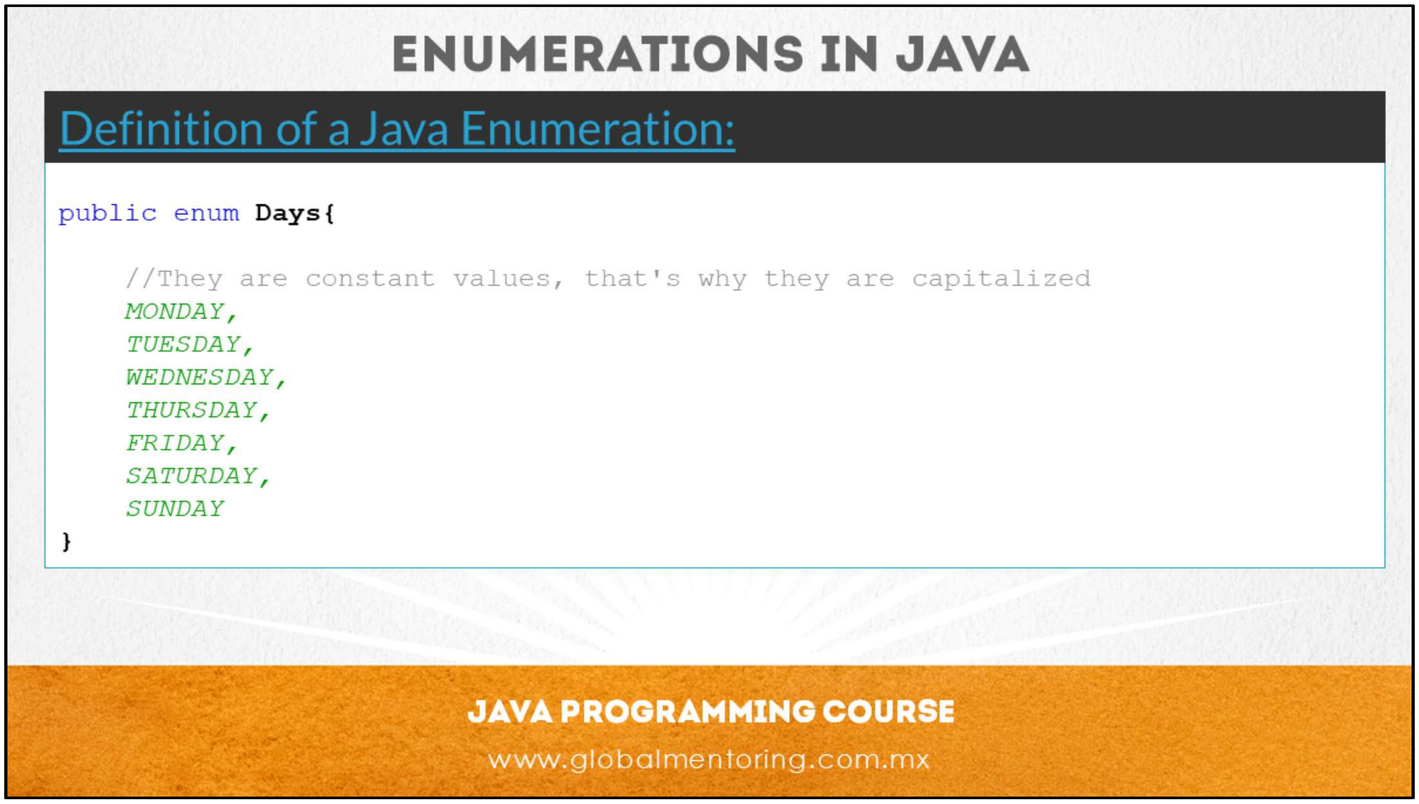
Hello, Ubaldo Acosta greets you. Welcome again. I hope you're ready to start with this lesson.

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

Are you ready? OK let's go!



ENUMERATIONS IN JAVA

Definition of a Java Enumeration:

```
public enum Days{

    //They are constant values, that's why they are capitalized
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
}
```

JAVA PROGRAMMING COURSE

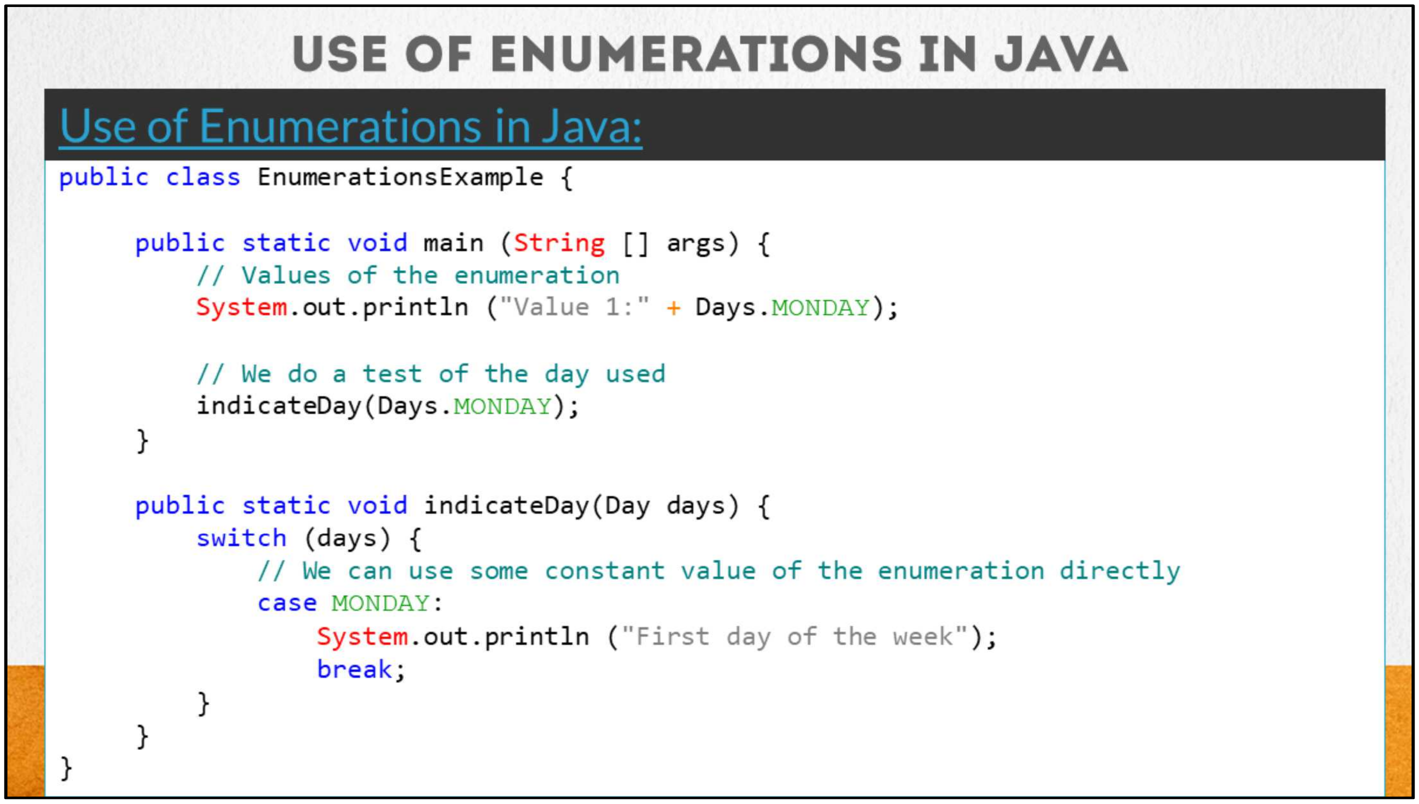
www.globalmentoring.com.mx

The types listed are a special type of class, which allows us to associate a series of values of final type, that is, they are constant. For example we can observe the constants of the days of the week.

We see that it is like creating a Java class, only that instead of the word class, we use the reserved word enum.

Because the elements of an enumeration are constant, they are written in capital letters, as we have said it is a good practice when we use constants in Java.

So we can use an enumeration-type class whenever we need to declare a series of constant values that we know previously, such as the days of the week, the cardinal points, the set of planets, continents, and any data set that can be considered as constants.



USE OF ENUMERATIONS IN JAVA

Use of Enumerations in Java:

```java
public class EnumerationsExample {

    public static void main (String [] args) {
        // Values of the enumeration
        System.out.println ("Value 1:" + Days.MONDAY);

        // We do a test of the day used
        indicateDay(Days.MONDAY);
    }

    public static void indicateDay(Day days) {
        switch (days) {
            // We can use some constant value of the enumeration directly
            case MONDAY:
                System.out.println ("First day of the week");
                break;
        }
    }
}
```

Once we have defined our enumeration type we can use it as seen in the code.

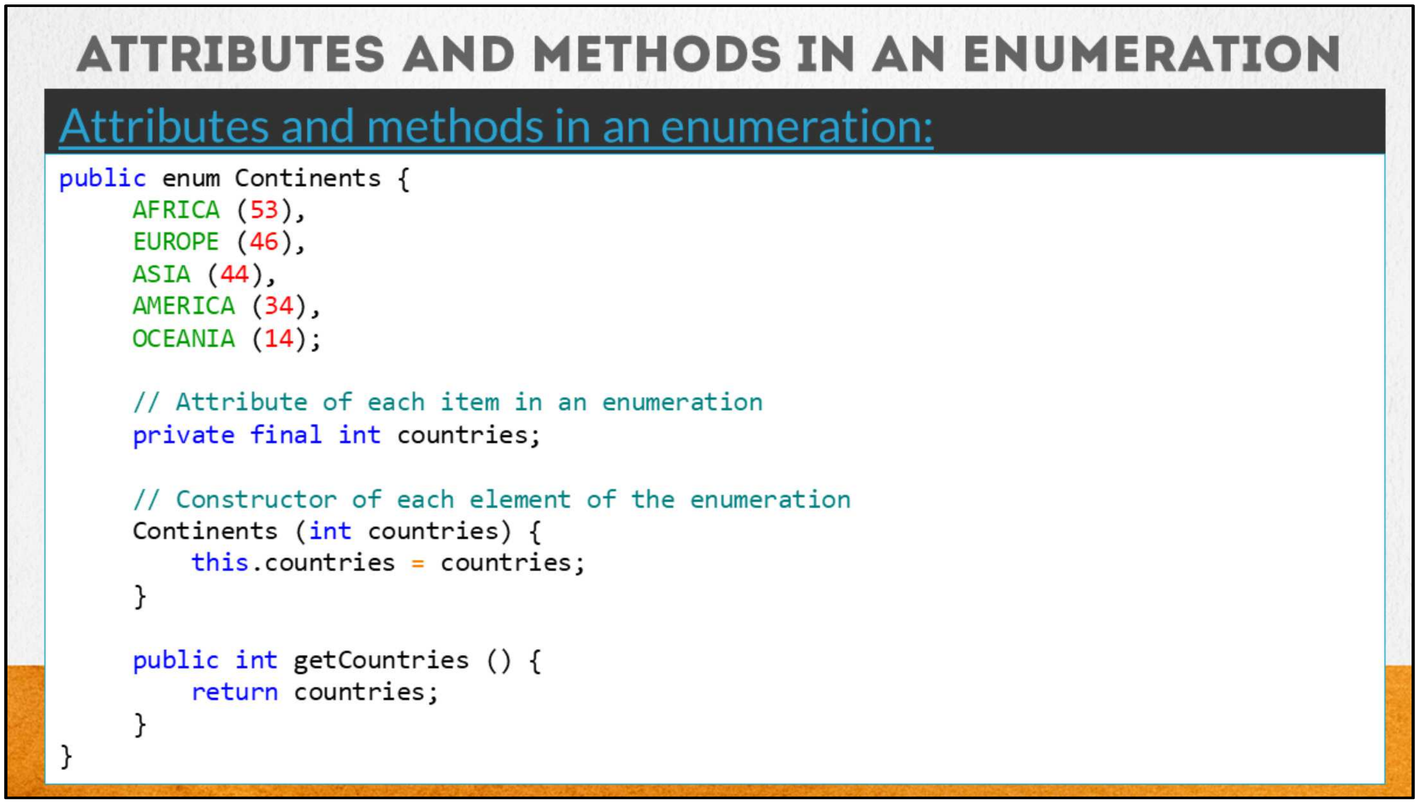
What we do is indicate the name of the enumeration, in this case Days, and then we use some of the constant values of the enumeration, for example: Days.MONDAY, with that we will be using some of the enumeration values as we need.

However, it is not its only use, we can use an enumeration to check some of the cases of a switch structure, which we know will select one of the cases according to the input of the switch statement.

To prove any of the cases of the switch structure, we must already provide a value of the enumeration, such as Days.MONDAY or Days.TUESDAY, etc.

And within the switch in each case we can specify the values already without specifying the name of the enumeration, but simply by indicating the constant of the enumeration, for example case MONDAY, or case TUESDAY, etc.

This is just an example of how we can use the enumerations.



ATTRIBUTES AND METHODS IN AN ENUMERATION

Attributes and methods in an enumeration:

```java
public enum Continents {
     AFRICA (53),
     EUROPE (46),
     ASIA (44),
     AMERICA (34),
     OCEANIA (14);

     // Attribute of each item in an enumeration
     private final int countries;

     // Constructor of each element of the enumeration
     Continents (int countries) {
         this.countries = countries;
     }

     public int getCountries () {
         return countries;
     }
}
```

However, the type enumeration is more complex than that. We can also define simple constant values, values for each of the constants.

To do this we must create a constructor to initialize each of the values associated with each constant.

For example, in the code we can see that we are creating an enumeration of the existing continents, however we would be interested to indicate the number of countries existing for each continent.

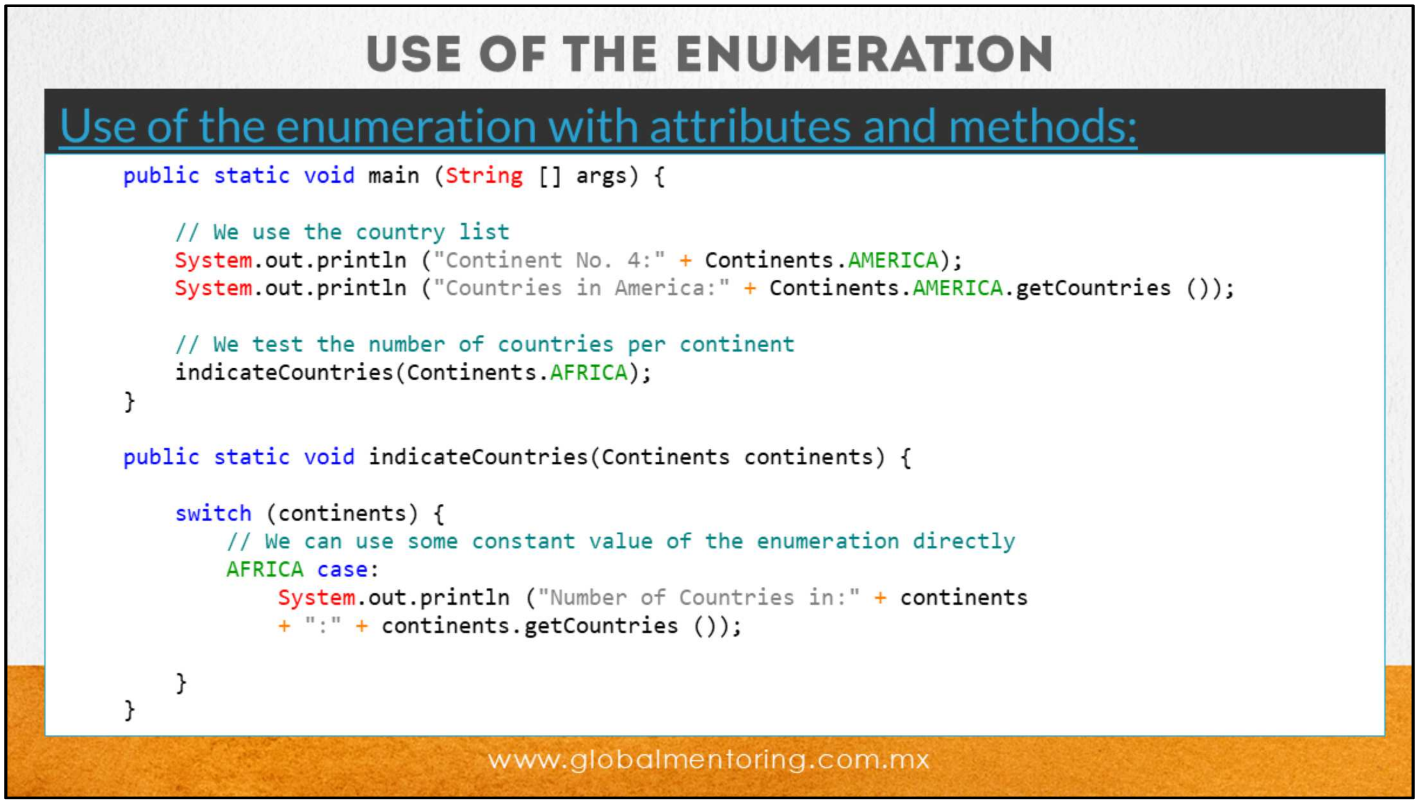
When we define each constant of the enumeration, we can provide the values we want separated by commas, in this case it is only an integer value that we are provided, which means the number of countries for each continent that we declare.

However, in order to create this type of more complex enumerations, we need to create a constructor to be able to create each of the elements of the enumeration, and also the attributes that will be associated with the values provided by each constant of the enumeration.

Therefore we see in the code, that it is not enough to create the constants of the enumeration, but we must also declare the attribute countries of the whole type, and later we declare a constructor, which will be called automatically by each one of the elements of the enumeration.

And finally, because each of the elements of the enumeration contains the number of countries defined, it is possible to declare a getCountries method to retrieve precisely the attribute countries declared in the type enumeration.

Next we will see how to use an enumeration of this type.



USE OF THE ENUMERATION

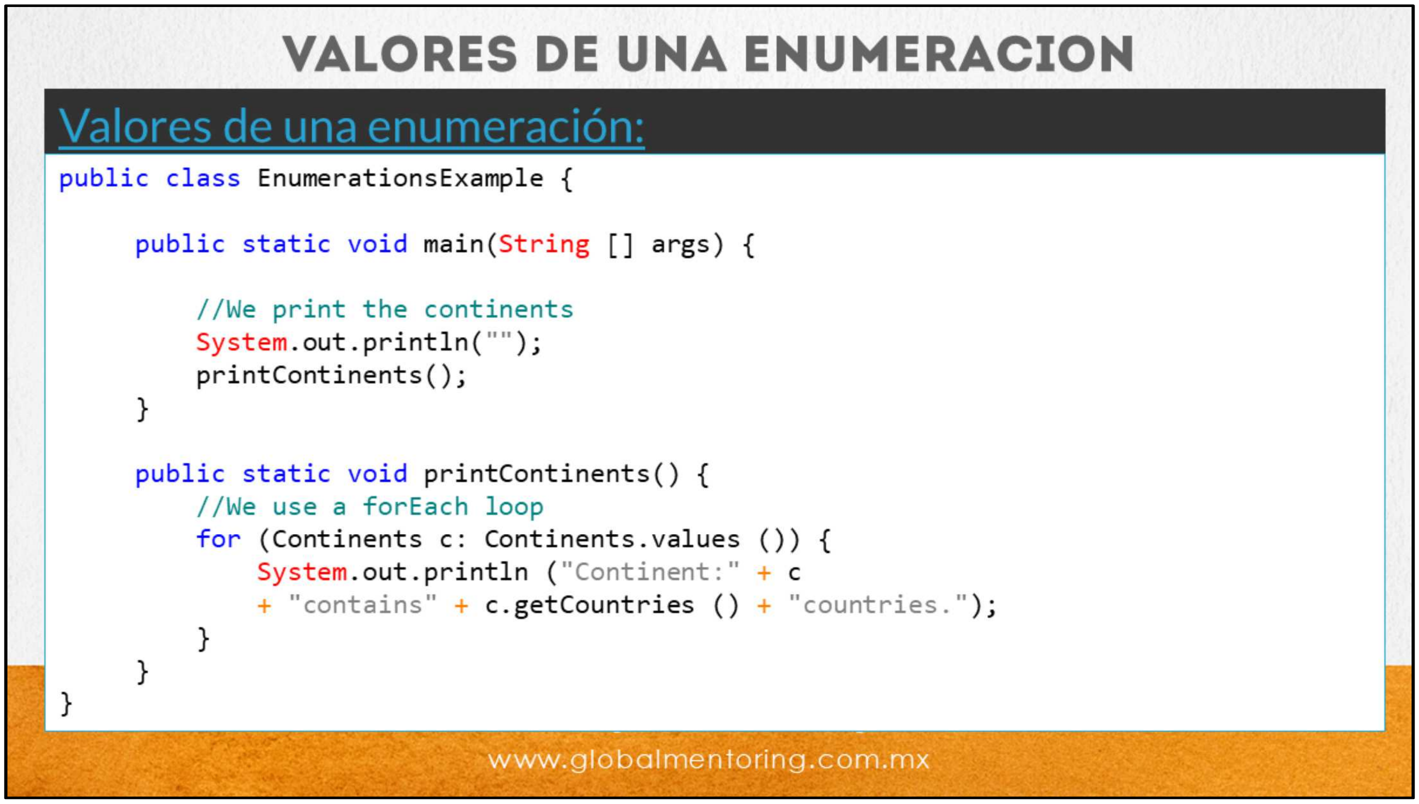
Use of the enumeration with attributes and methods:

```java
public static void main (String [] args) {

    // We use the country list
    System.out.println ("Continent No. 4:" + Continents.AMERICA);
    System.out.println ("Countries in America:" + Continents.AMERICA.getCountries ());

    // We test the number of countries per continent
    indicateCountries(Continents.AFRICA);
}

public static void indicateCountries(Continents continents) {

    switch (continents) {
        // We can use some constant value of the enumeration directly
        AFRICA case:
            System.out.println ("Number of Countries in:" + continents
            + ":" + continents.getCountries ());

    }
}
```

www.globalmentoring.com.mx

As we can see, the basic use is of the constants of an enumeration is the same as we have studied, for example, to access any of the constants we specify the name of the enumeration followed by the name of the constant, for example: Continents.AMERICA , and if we want to access the attributes of each defined constant, we can call the method to retrieve the attribute of each constant, for example: Continents.AMERICA.getCountries()

With this we can see that the use of type enumerations can be more complete than simply associating some constants, but we can also define attributes and therefore add methods to retrieve the attributes for each of the defined constants of an enumeration.



VALORES DE UNA ENUMERACION

Valores de una enumeración:

```
public class EnumerationsExample {

    public static void main(String [] args) {

        //We print the continents
        System.out.println("");
        printContinents();
    }

    public static void printContinents() {
        //We use a forEach loop
        for (Continents c: Continents.values ()) {
            System.out.println ("Continent:" + c
            + "contains" + c.getCountries () + "countries.");
        }
    }
}
```

www.globalmentoring.com.mx

The values method is added by default every time we create an enumeration. The objective of this method is to return a list of each of the constants defined in an enumeration.

In the code we can see in the method printContinents, the use of this method, supported by a forEach loop, what we do is define a variable of type enumeration that you are using, in this case Continents c, and then indicate the values of the enumeration calling precisely the Continents.values () method.

Once we are iterating each of the elements, we have the variable c to retrieve the attribute of the constant that we are iterating, so we can request the c.getCountries method to retrieve the number of countries of the selected continent and if we want to print the continent that we are iterating we simply use the variable c defined in the forEach loop.

Next, let's see an example of the use of enumerations.

ONLINE COURSE

JAVA PROGRAMMING

By: Eng. Ubaldo Acosta

JAVA PROGRAMMING COURSE

www.globalmentoring.com.mx