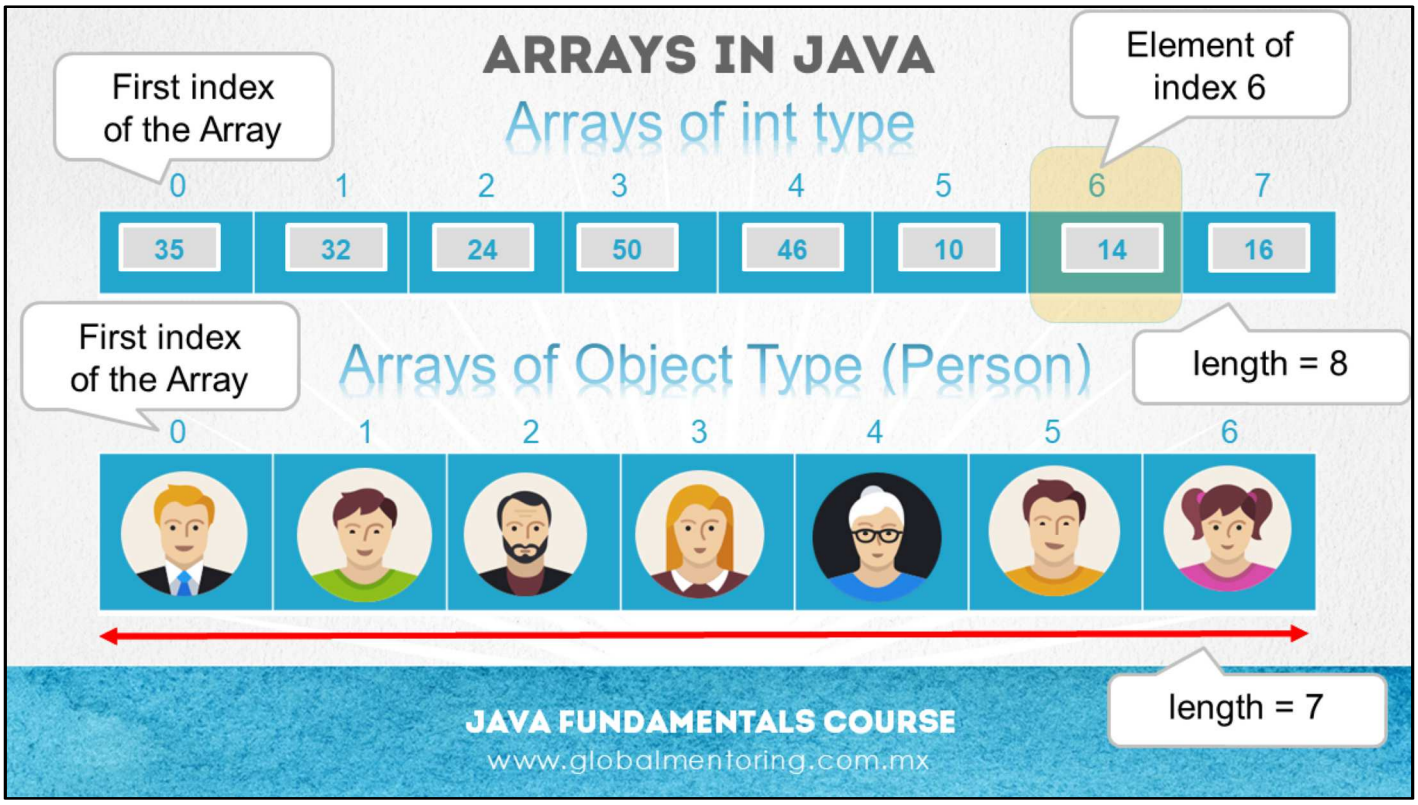

Hello, Ubaldo Acosta greets you. Welcome or welcome again. I hope you're ready to start with this lesson.

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

Are you ready? OK let's go!




In Java, the simplest data structure is the Array. An Array is basically a set of information, which allows us to group information normally of a particular type.

For example, in the figure we can observe two Arrays, one of the int primitive type and another of the Person Object type. That is, we can declare Arrays that contain any type of data, whether of primitive types or of type Object.

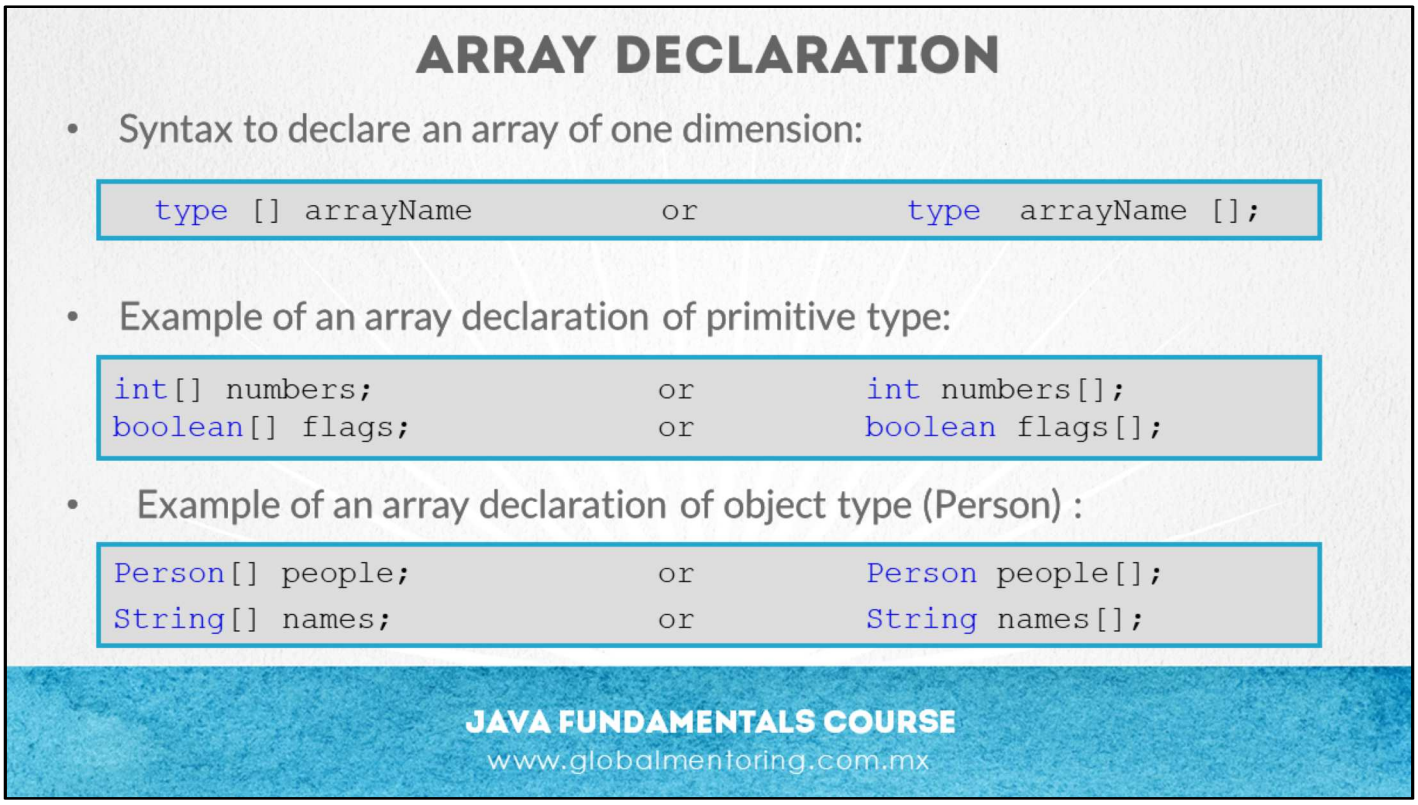
In the first case, the array contains 8 elements (length), which are numbered from 0 element to 7 element. This means that arrays in Java start at 0 index, and the last element would have the index number of elements less one.

We have on the other hand the array of type Person, which contains 7 elements (length), and therefore the index goes from 0 to 6, as seen in the figure.

Not all elements of the array should contain values. For example, if the integer array were 10 elements, but only had 7 values, the last 3 values would have their default value of the declared type, in this case, since it is of type int, the default value for the int type is 0

In the case of the object type array if it were of 10 elements, and it had only 5 objects of type person defined, then the remaining 5 its serial value null, since that is the default type for the type Object.

Next we will see the syntax for managing arrays.



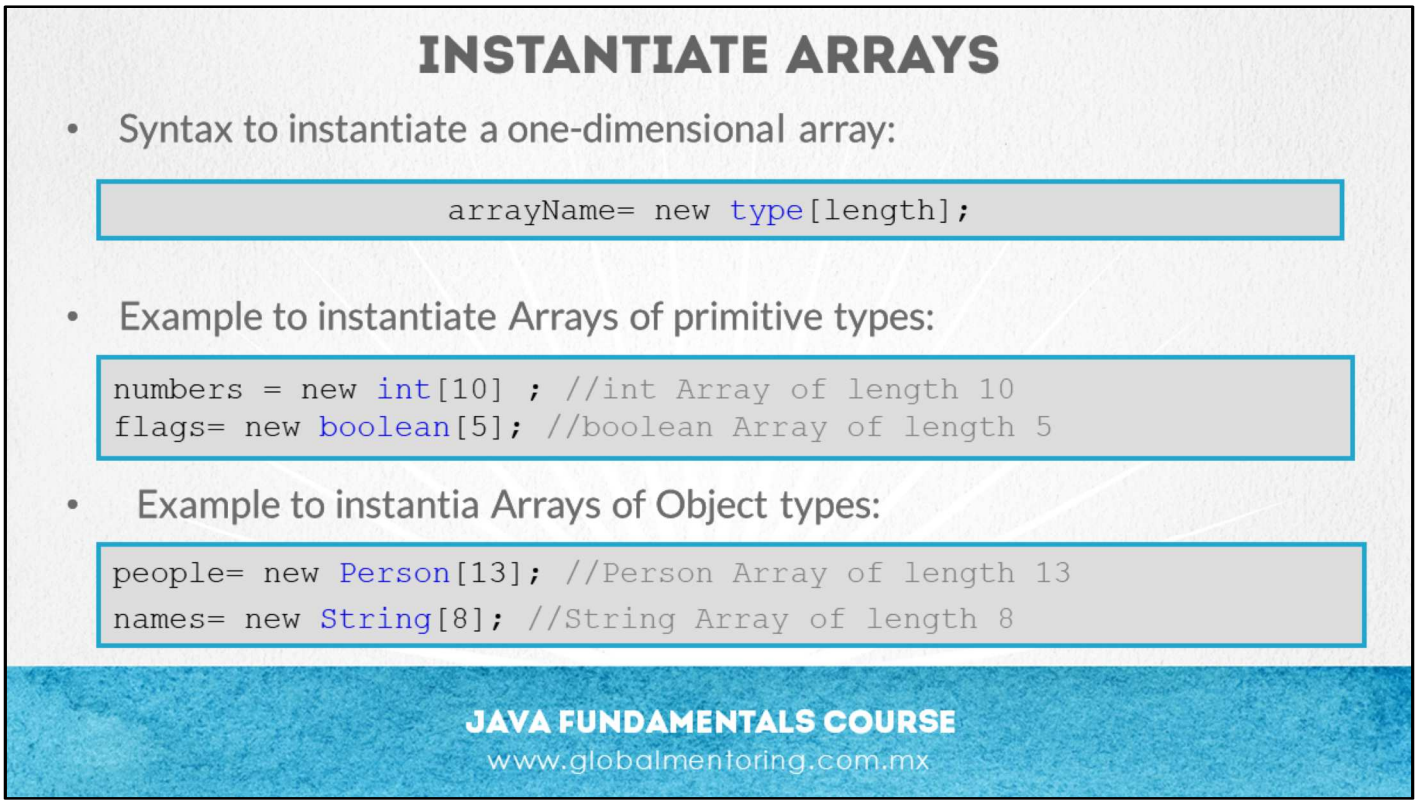

In this slide we can observe the syntax for the declaration of a one-dimensional array. In the next lesson we will see the declaration of a two-dimensional arrangement.

Basically it is like declaring a variable except that we are going to use the brackets [], which can be used in two parts, either before the name of the variable or after the name of the variable, that's why the two options are shown in each declaration.

Declaring an array is the same as declaring variables, we can declare arrays that store primitive types or store object references. In the slide we show both cases, first we show two examples of primitive type, one of type int and another of type boolean. Later we show the declaration of two arrays that will store references of objects of type Person and of type String.

Because arrays are a collection of data, normally the name of an array is plural, so that we can easily read the variable and recognize that it is a collection of data, and in this case an array, although later We will see that it can be not only about arrays, but about other data structures.

The code shown in the slide is only the declaration of the variable, we will see below how to initialize a one-dimensional array, since up to now, by only declaring a variable of type, the JVM does not know how long this array is, for this we must initialize it, let's see how.




Starting from the variables defined in the previous slide, in this slide we see the syntax to instantiate arrays of a dimension according to the type of data we are using.

The syntax is very similar to instantiating a variable of type object, and in fact this is one of the features of Java, including arrays or any type in Java that stores a reference inherits from the Object class directly or indirectly, so both arrays also descend from the Object class.

However, the difference between instantiating a normal object and an array is that in an array we indicate the maximum number of elements that the array will contain. Previously, in the definition of the variable, it was already indicated which type is to be stored, and now we must indicate that we will create an object of a certain type and that it will contain a number of elements as indicated.

We can see in the slide several examples of initialization of arrays according to the type of data we choose. Next we will see how to initialize the elements of an array.

## INITIALIZE THE ELEMENTS OF AN ARRAY

- Syntax to initialize the elements of a one-dimensional array :

```
arrayName[index] = value;
```

- Example to initialize the elements of an int type array:

```
numbers[0] = 15; // The value of 15 is assigned at index 0
numbers[1] = 13; //The value of 13 is assigned at index 1
```

- Example to initialize the elements of an Object type array:

```
people[0] = new Persona(); //The person object is assigned at index 0
people[1] = new Persona("Charly", 33);//Object is assigned at index 1
names[0] = new String("John"); //The String is assigned at index 0
names[1] = new String("Rita"); //The String is assigned in index 1
```

In the slide we can observe the initialization of the elements of a one-dimensional arrangement. What we must do is to add elements to an array, it is to select the indexes that we want to initialize one by one.

Therefore, it is important to know that the zero index is the first element of the array, and the last element of an array can be obtained with the length property minus an element, for example if we write: numbers.length -1 will return the last index of the array that we can occupy. If we exceed the maximum index and we want to add an element out of the maximum number of elements, it will throw an error, so we must know what is the maximum number of elements with the mentioned code.

We can see in the slide several examples of how to add elements to our arrangement. We can add them manually, that is, one by one each element, or we can add the elements more dynamically using a counter of elements that have been added, so that we can know if we have reached the limit of added elements or not.

In this slide we can see more clearly that not all the elements of an array will be always full, for example if the integer array is 10 elements, then we have only filled 2 of the 10 available elements, this means that 8 elements will have the value by default, in this case the value of 0. Therefore, in many cases it will be convenient to have a counter in order to know how many elements have been initialized in our array, which is different from the maximum number of elements that our array supports.

In the exercise of this lesson we will see how to initialize the elements of our arrays.

## EXTRACT ELEMENTS FROM AN ARRAY

- Syntax to extract the elements of a one-dimensional array :

```
receivingVariable = arrayName[index];
```

- Example to extract the elements of an int type array:

```
int i = numbers[0]; //Extract the value stored at index 0
int j = numbers[1]; //Extract the value stored at index 1
```

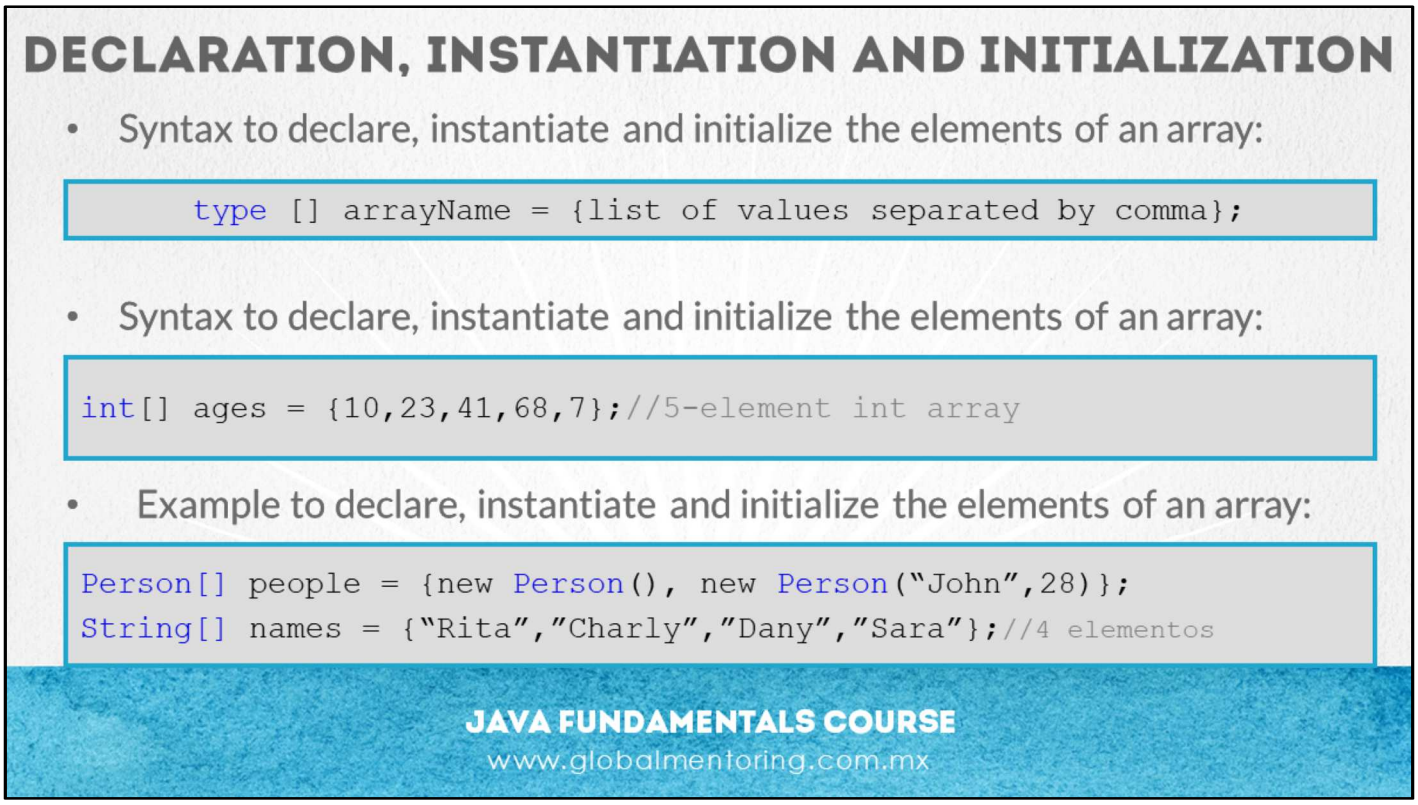
- Example to extract the elements of an Object type array :

```
Person p1 = people[0];//Extract the value stored at index 0
Person p2 = people[1];//Extract the value stored at index 1
String name1 = names[0];//Extract the value stored at index 0
String name2 = names[1];//Extract the value stored in index 1
```

To read or extract the elements stored in an array, simply indicate the name of the array and indicate the index of the element that we want to extract, this will return the indicated index element.

It is important not to exceed the maximum number of elements, otherwise an error will return.

We can see in the slide several examples with arrays that store primitive types or Object types, in both cases the syntax is the same, we should only have one variable that receives the value extracted from the respective array according to the specified index. Later we will see examples of how to extract the elements using a for or while loop to traverse each of the array elements.




There is another way to declare arrays, and at the same time instantiate and initialize each of its elements. This is a different syntax in the way in which the values are assigned.

We can see in the slide several examples, both with primitive types, as well as Object type. And what we can observe is that in the same line of code the variable is declared, the array is instantiated and each of its values is initialized.

However, this syntax is not always possible to use since we would need to know in advance all the elements that are going to be stored in the array, and in many cases we do not have this information from the beginning, but we do have this information before creating our array , then it is possible to use this simplified syntax.

# ARRAY MANAGEMENT EXAMPLE

## Example of one-dimensional arrays:

```
1  public class ArrayExample{
2
3      public static void main(String[] args) {
4          //1. Declare an array of int type
5          int ages[];
6          //2. instantiate the int array
7          ages = new int[3];
8          //3. Initialize the values of the integer array
9          age[0] = 30;
10         age[1] = 15;
11
12         //4. read the values of each element of the array
13         System.out.println("Array integers type, index 0: " + age[0]);
14         System.out.println("Array integers type, index 1: " + age[1]);
15
16         Person people[];
17         people = new Person[4];
18         people[0] = new Person("John");
19         people[1] = new Person("Rita");
20
21         System.out.println("Array people type, indice 0: " + personas[0]);
22         System.out.println("Array people type, indice 1: " + personas[1]);
23     }
24 }
```

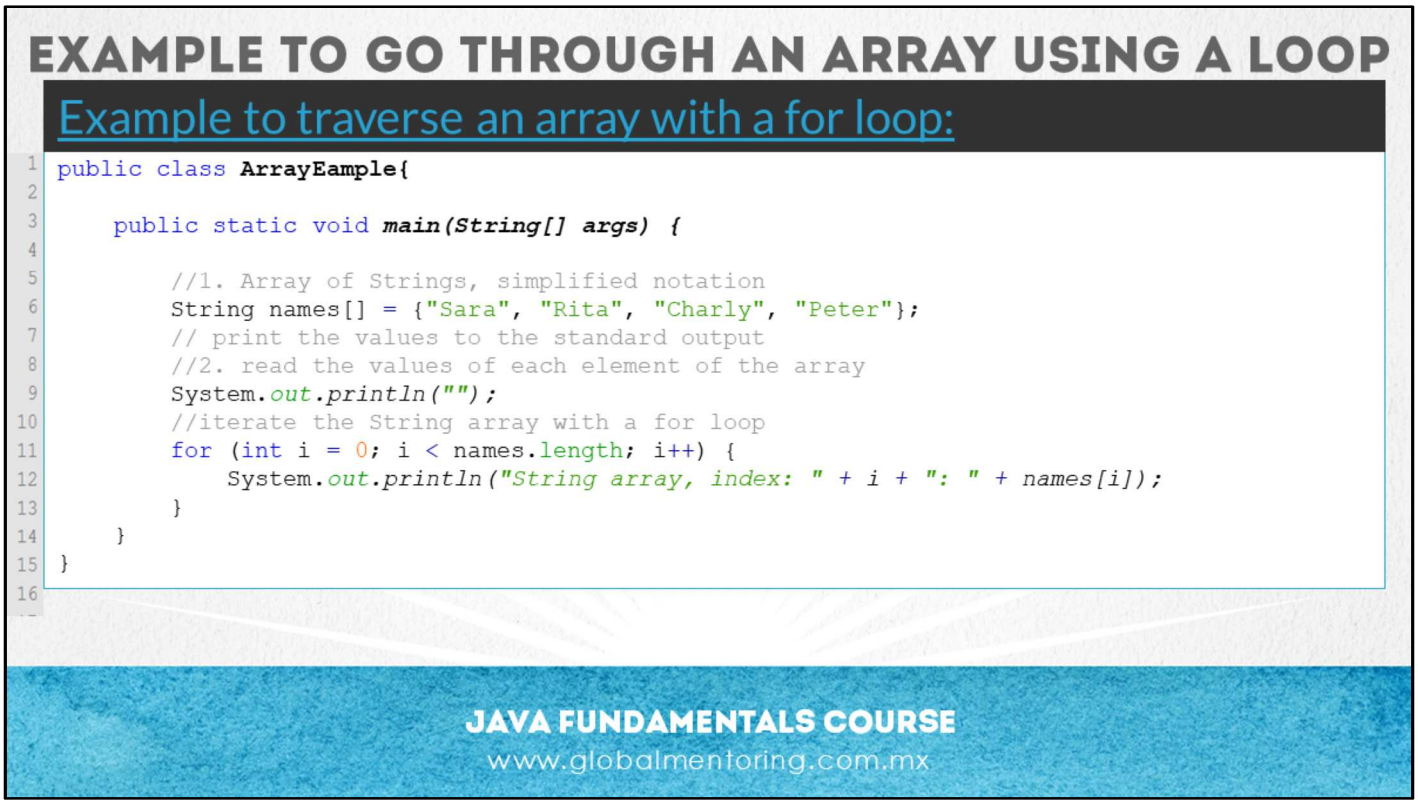


We can see in the slide some examples for the use of arrays.

From the declaration (lines 5 and 16), the instantiation (lines 7 and 17), the initialization of values (lines 9-10 and 18-19), and finally the reading of the values (lines 13-14 and 21-22 ).

We can see that not all the values of the array are initialized, so the values that have not been initialized will be the default value of the declared type, in the case of the array of type int the default value is 0 and in the case of the Person type object, its default value will be null.

Later we will carry out this exercise to put these concepts into practice.




Finally we will see an example of the simplified notation, and we will use it to show how to iterate an array with the help of a loop such as the for loop.

First of all we see an example of the use of simplified notation (line 6). In this case it is an array of type String, and in the same line we instantiate the array and initialize the array values. In this case it is not necessary to indicate the number of elements that the array will contain, this number will be obtained directly from the number of elements that are added in the initialization of the array. It should be noted that in this data structure it is not possible to make the array larger or smaller once declared or as in this case once initialized. However we will see in the next course the theme of collections, where we will see structures such as an ArrayList which are structures that can grow dynamically.

Once we have defined how many elements the array will have, the length property will return the maximum number of elements. It is therefore possible to combine a for loop, and use a counter, in this case the entire variable i, to iterate the number of elements of the array, and the end of this loop will be when this counter i is smaller than the length of the array . And this will end the journey of each of the elements of the arrangement.

We will see later the execution of this code to put these concepts into practice.