by Nacho Iborra

Development Environments
Block 2
Unit 2: Control structures. Conditions and loops

In this section we are going to see the different selective and iterative structures that you can use in Java. In all of them you can/must use the curly braces `{ ... }` to group everything that must be contained in the corresponding clause, as you do in C#, for instance.

2.2.1. Selective structures

In Java we can use either `if` structures, `if..else` structures or `switch` structures to determine the path to follow depending on a given condition.

2.2.1.1. if .. else if .. else

You can use the `if` basic structure, and `if .. else if .. else if .. else` structures as in many other languages. The first one lets us run a piece of code if a given condition is true:

```
if (age >= 18)
{
    System.out.println("You are old enough");
}
```

The second structure lets us choose among several paths depending on the condition relative to each one. Only one path will be chosen.

```
if (number > 0)
{
    System.out.println("It is positive");
}
else if (number < -10)
{
    System.out.println("It is under -10");
}
else
{
    System.out.println("It is between -10 and 0");
}
```

2.2.1.2. switch

Besides, there's a `switch` clause, with their corresponding `case` clauses and a `default` case. The data managed in the `switch` clause must be a primitive type; strings are NOT allowed in early versions of Java (Java 6 and earlier). The `break` at the end of each case is not compulsory, but if you don't put it, you go to next case.

```
switch(number)
{
    case 0:     System.out.println("It is 0"); break;
    case 1: System.out.println("It is 1");
    case 2: System.out.println("It is 2"); break;
    default: System.out.println("Unknown number");
}
```

In previous example, if number is 1, it would output the messages "It is 1" and "It is 2", since there is no `break` clause at case 1.

> **Proposed exercises**:
>
> **2.2.1.1.** Create a program called *MarkCheck* that asks the user to enter 3 marks. The program must print one of these messages, depending on the mark values:
>
> - *All marks are greater or equal than 4*
> - *Some marks are not greater or equal than 4*
> - *No mark is greater or equal than 4*
>
> **2.2.1.2.** Create a program called *GramOunceConverter* that will be an improved version of exercise 2.1.6.2. of previous unit. In this case, the user will type a weight (float), and a unit (`g` for grams, `o` for ounces). Then, depending on the unit chosen, the program will convert the weight to the opposite unit. For instance, if the user types a

> weight of 33 and chooses `o` as unit, then the program must convert 33 ounces to grams. You must solve this program using a `switch` structure. If the unit is other than `g` or `o`, then the program must output an error message: "Unexpected unit", with no final result.

2.2.2. Loops

Java has the most common iterative structures that (almost) every programming language has: `while`, `do..while` and `for`. The way you use them is similar to other languages such as C or C#.

2.2.2.1. while

We will use this loop when we don't know how many iterations are expected, and we don't even know if there will be one iteration. This example counts from 1 to 10:

```
int n = 1;
while (n <= 10)
{
    System.out.println(n);
    n++;
}
```

2.2.2.2. do..while

We will use this loop when we don't know how many iterations are expected, but we know that there will be (at least) one iteration. It is very usual when we ask the user to type something and then check the input and ask the user again. If we do the same loop than in previous example with a `do..while` structure, it would look like this.

```
int n = 1;
do
{
    System.out.println(n);
    n++;
} while (n <= 10);
```

2.2.2.3. for

We will use this loop when we know how many iterations are expected. The counter from 1 to 10 should be done with this structure preferably, and it would look like this:

```
for (int n = 1; n <= 10; n++)
{
    System.out.println(n);
}
```

Note that we can declare variables in `for` loops (and in the middle of other code, as in other languages such as C#).

2.2.2.4 Another "for"

There is another way of using the `for` clause, applied to collections or arrays. It consists in using a variable with the same type, this way:

```
for (int number: numbers)
    System.out.println("" + number);
```

where `numbers` is expected to be a collection or array of integers. This structure is equivalent to the `foreach` structure of other languages such as C#, and is expected to be used in a read-only way (only to check the values, but not to modify them).

> **Proposed exercises**:
>
> **2.2.2.1.** Create a program called *GroupPeople* that asks the user to enter how many people is going to attend to a conference. The program must create groups of (preferably) 50 people. Whenever this is not possible, then it will attempt to create groups of 10 people, and finally it will create groups of 1 person. The program must output how many groups of each category are necessary. For instance, if 78 people are going to attend to the conference, then we need 1 group of 50 people, 2 groups of 10 people and 8 groups of 1 people.
>
> **2.2.2.2.** Create a program called *SumDigits* that asks the user to enter numbers (integer values) until he enters 0. The program must sum up all the numbers entered by the user and then show the final result, and how many digits it has. For instance, if the user types 12, 20, 60, 33, 99 and 0, then the program must output: "*The result is 224, and it has 3 digits*".

2.2.3. Additional exercises: Java challenges

2.2.3.1. Introduction to Java challenges

As we learn new concepts of Java, we can apply them to solve some exciting challenges from Acepta el reto web site, so first of all, you must register in this site with the login and

password that you want.

There, you can find many programming challenges, grouped by category. In this unit and the following ones, you will be encouraged to solve some of them.

2.2.3.2. Sample challenge: Hello world

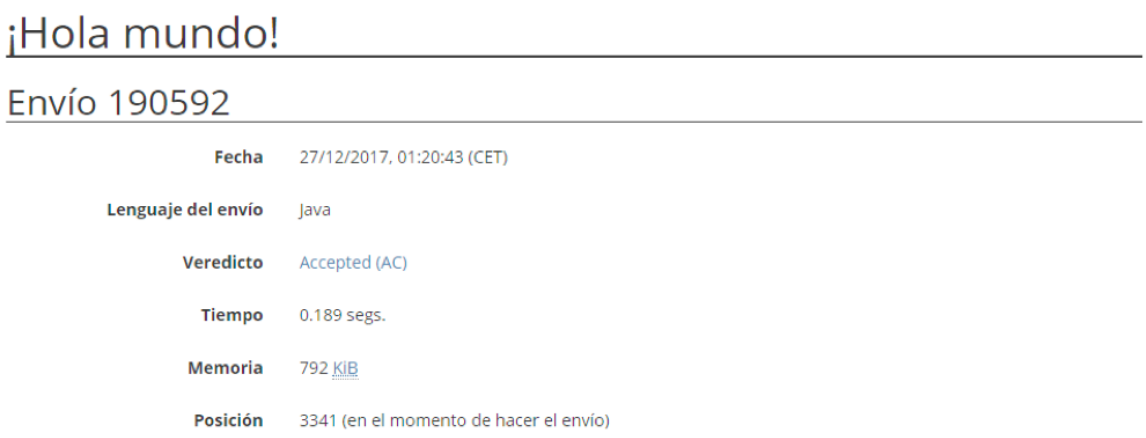
Let's have a look at this challenge from *Acepta el reto*. It asks you to read a number N and print N times the string "Hola mundo.". To solve this challenge, we could implement something like this in Java:

```
import java.util.Scanner;

public class Challenge116_Hola_mundo
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int times = sc.nextInt();

        for (int i = 0; i < times; i++)
            System.out.println("Hola mundo.");
    }
}
```

Try to upload this code to *Acepta el reto* and see how it is accepted.



Proposed exercises:

2.2.3.1. Try to solve these challenges from *Acepta el reto* web site:

- Boredom at tabletalks
- Last factorial digit