#1.en - First steps in Java, strings, arrays and loops

Exercice 1 - Hello Groland

In Java, each public class must be defined in its own file. The name of the file must be the name of the class it contains, to which we add the suffix **.java** . Class names must consist of adjoining words whose first letter is a capital letter (*CamelCase* style).
First we will write small programs to familiarize yourself with the compiler, the virtual machine and the methods.

1. Write the following program:

   ```
   public class HelloGroland {
     public static void main(String[] args) {
       System.out.println("Hello Groland");
     }
   }
   ```

   in your favorite text editor and save it under the name `HelloGroland.java`
2. Compile the program using the command `javac` then check that the corresponding `.class` file exists.

   ```
   javac HelloGroland.java
   ```

3. Run the program with the command `java`

   ```
   java HelloGroland
   ```

   We don't write ".class" because the virtual machine adds it on its own.

Exercice 2 - Print the arguments on the command line

Write a class `PrintArgs` that prints all the arguments, one by line.

```
$ java PrintArgs these are args
these
are
args
```

Command line arguments are stored in the array of strings passed as an argument to the method `public static void main(String[] args)`.

- First, display the first argument from the command line (in our example `these`).
  What happens it you don't pass any argument during the execution of the program?
- Write a loop that prints the content of an array, knowing that in Java, arrays have an attribut (`lenght`) for that purpose.
- Change your program in order to user the 'foreach' syntax `for(Type value: array)`

Exercice 3 - Simple Calculator

Write a program that read a number on the standard input and display it on the console.
For that purpose, use `Scanner` and its method `nextInt()`.

To understand this program, it is useful to look at the documentation **avalaible**, and to register the links in your bookmark.

1. Copy the program above and add some code in order to print a number typed by the user.
2. Indicate in the program where the variables are and what is their type.
   Modify the program to declare and initiate the variables in one line.
3. Why `nextInt()` is not a function?
   So what is `nextInt()`?
4. Explain this line:

   ```
           import java.util.Scanner;
   ```

5. Modify the program in order to ask for two integer and to display their sum.
6. Display also the difference, the product, the quotient and the rest.

Exercice 4 - Conversion from String to integer

We want to write a program that displays the sum of integers read from the command line.
Here is an example:

```
       $ java Sum 15 5 231
       integers: 15 5 231
       sum: 251
```

This program is divided in some functions :

1. Write a method that takes in argument an array of String and returns an array of integer taken from these strings.
   The static method `parseInt(String s)` of the class `java.lang.Integer` allows to get the value of an integer stored in a string
2. What means static for a method?
3. What happens if the word taken from the command line is not a number
4. Write a method that takes as argument an array of int and returns their sum.
5. Write the `main` method that uses the two methods above to display the array of int and its sum.
   There is a little trick to display the array. You should take a look at `java.util.Arrays`.

Exercice 5 - From C to Java

The purpose of this example is to show the diferences in perfomance between a program in C and the same written in Java.

1. Compile (`gcc pascal.c`) and run the file `a.out` while asking the system to measure the processing time `(time a.out)`.
2. Write the same program in Java (`Pascal.java`). You can use copy/paste. Compile this program, and run it while measuring time (use `time` again).

How can you explain the difference in speed?