Premiers pas en Java, chaîne de caractères, tableau, boucle

Exercice 1 - Hello Groland

On rappelle qu'il est préférable en Java que chaque classe soit définie dans un fichier qui lui est propre. Le nom du fichier doit être le nom de la classe qu'il contient, auquel on ajoute le suffixe **.java**. Les noms des classes doivent être constitués de mots accolés dont la première lettre est une majuscule.
Dans un premier temps nous allons écrire des petits programmes permettant de se familiariser avec le compilateur, la machine virtuelle et les méthodes.

1 Écrire le programme suivant :

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

dans votre éditeur de texte préféré et sauvegarder celui-ci sous le nom `HelloGroland.java`

2 Compiler le programme en utilisant le commande `javac` puis vérifier que le fichier `.class` correspondant existe bien.

```
javac HelloGroland.java
```

3 Exécuter le programme avec la commande `java`

```
java HelloGroland
```

On ne met pas ".class" parce que la machine virtuelle le rajoute toute seule.

Exercice 2 - Afficher les arguments de la ligne de commande

Écrire une classe `PrintArgs` qui affiche les arguments de la ligne de commande.

```
$ java Voici des arguments
Voici
des
arguments
```

Les arguments de la ligne de commande sont stockés dans le tableau de chaînes de caractères passé en argument à la méthode `public static main(String[] args)`.

Dans un premier temps, afficher le premier argument de la ligne de commande (dans notre exemple Voici).

Que ce passe-t'il si l'on ne passe pas d'argument lors de l'exécution du programme ?

Écrire une boucle affichant le contenu du tableau en sachant qu'en Java les tableaux possèdent un champ (un attribut) `length` qui renvoie la taille du tableau.

Changer votre programme pour utiliser la syntaxe dite 'foreach' `for(Type value:array)`

Exercice 3 - Calculette simple

Écrire un programme prenant un nombre sur l'entrée standard et affichant celui-ci
Pour cela, on utilisera un objet `Scanner` et particulièrement sa méthode `nextInt()`.

```
import java.util.Scanner;

public class Calc {
  public static void main(String[] args) {
    Scanner scanner;
    scanner=new Scanner(System.in);
    int value;
    value=scanner.nextInt();
    // compléter ici
  }
}
```

Pour comprendre le programme, il est utile de regarder la documentation disponible ici, et même de mettre les liens en bookmark (signet, favoris, etc.)

- la javadoc (`http://java.sun.com/javase/6/docs/api/index.html`) .
- les guides (`http://java.sun.com/javase/6/docs/`) .
- des tutoriaux (des fois un peu datés) (`http://java.sun.com/docs/books/tutorial/`) .

1. Indiquer dans le programme où sont les variables et quel est leur type associé.
2. Recopier le programme précédent et le compléter pour qu'il affiche le nombre saisi par l'utilisateur.
3. Modifier le programme pour déclarer et initialiser les variables en une seule ligne
4. Expliquer la ligne :

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

5. Modifier le programme pour qu'il demande deux entier et affiche la somme de ceux-ci.
6. Afficher en plus de la somme, la différence, le produit, le quotien et le reste.

Exercice 4 - Bien le bonjour

Écrire un programme demandant à un utilisateur quel est son nom et affichant "bonjour" suivi du nom rentré précédemment.
Faite en sorte d'afficher le texte "bonjour" et le nom sur une seule et même ligne

- Soit un utilisant `System.out.print()`
- Soit en utilisant le concaténation des chaînes de caractères (le +).

Exercice 5 - De C vers Java

Cet exemple a pour but de montrer les différences de performance entre un programme en C et le même en Java.

```
#include <stdio.h>
#include <stdlib.h>

int pascal (int nBut, int pBut){
  int * tab;
  unsigned int n, i;
```

```
  tab = (int *)malloc ((nBut+1)*sizeof(int));
  if(tab==NULL){
    fprintf(stderr,"Pas assez de place\n");
    exit(0);
  }

  tab[0] = 1;

  for(n=1; n<=nBut; n++){
    tab[n] = 1;

    for(i=n-1; i>0; i--)
      tab[i] = tab[i-1] + tab[i];
  }

  int result=tab[pBut];
  free(tab);
  return result;
}

int main(int argc, char * argv[]) {
  printf(" Cn, p = %d\n", pascal (30000, 250));
  return 0;
}
```

1 Compiler (`gcc pascal.c`) et exécuter le programme `a.out` en demandant au système le temps d'exécution du programme. (`time a.out`).

2 Écrire le programme (`Pascal.java`) équivalent en Java. Pour une fois, servez-vous du copier/coller. Compiler le programme. Exécuter le en mesurant le temps (toujours avec `time`).

Comment peut-on expliquer la différence de vitesse ?