TP1 JAVA

1. Présentation de l'IDE Netbeans

1.2 Compiler et exécuter : `javac` et `java`

Pour faire un programme en java il faut taper son code via un éditeur de texte, dans notre cas il s'agit d'Emacs, et d'enregistrer le code dans un fichier : *nom_fich*.java .

Pour compiler le programme il faut taper la commande :

```
> javac nom_fich.java
```

Cette commande génère un fichier compilé de nom : *nom_fich*.class. Et pour lancer l'exécution de ce programme il faut taper :

```
> java nom_fich
```

2 Les bases d'un programme java

2.1 Le code minimal

En Java, tout programme commence par la définition d'une classe publique contenant la fonction main() qui est exécutée au lancement du programme. Comme vous le verrez plus tard, en règle générale, chaque classe est stockée dans un fichier différent. Le nom du fichier doit être identique au nom de la classe qu'il contient avec en plus l'extension '.java', en respectant les minuscules et les majuscules.

Voici le code minimal pour un programme Java :

```
public class nom_classe {
                    //point d'entrée
public static void main(String[] args ){
     //votre code ...
        }
}
```

Rmq: - le symbole // permet d'écrire une ligne de commentaire (ignorée du compilateur).

- les noms indiqués ci-dessus en italique sont des noms de classe ou de variables qui peuvent être choisis à votre guise.

Ce code doit donc être enregistré dans un fichier portant le nom : *nom_classe*.java

2.2 Exemple de programme java

L'exemple suivant est le code complet d'un programme java qui se contente d'afficher la chaîne de caractère : "Hello world !!!"

```
public class Hello {

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

Enregistrer ce code dans le fichier : `Hello.java`. Compiler ensuite ce programme en tapant :

```
> javac Hello.java
```

Et exécuter-le en tapant : `> java Hello`

Notez au passage une fonction très importante en Java :

System.out.println("*votre texte*");

Cette fonction vous permet d'afficher une ligne de texte. Vous pouvez l'utiliser aussi pour afficher la valeur d'une variable :

```
int n=5;
     System.out.println(n);
```

Ou encore une chaîne structurée avec un texte et la valeur d'une variable :

```
int n=5;
     System.out.println("La valeur de n est : "+n);
```

Exercice 1 : Coder la fonction factorielle itérative avec affichage du résultat.

Rappel : cette fonction calcule le factoriel de n, où n est une variable de type `int` dont la valeur est fixée dans le code au moment de sa déclaration (`int n=5`;). Fact(n) = n*n-1*...*2*1

exemple, si n = 5 :

```
> java Fact

120
```

3 Les paramètres d'un programme

Un programme peut recevoir des paramètres que l'on spécifie dans la ligne de commande au moment du lancement de l'exécution du programme.

Exemple : `> java Prog p1 p2 p3`

Cette ligne de commande lance donc l'exécution du programme Prog avec les paramètres : p1 p2 et p3. Les paramètres sont des chaînes de caractères, séparés les uns des autres par des espaces.

3.1 Récupérer les paramètres transmis

Dans le programme les paramètres sont placés dans l'argument args de la fonction main(), comme vous l'avez remarquez args est un tableau de String, c'est a dire un tableau de chaînes de caractères. Dans l'exemple précédent on aura alors : la chaîne p1 dans args[0] , la chaîne p2 dans args[1] et la chaîne p3 dans args[2].

Rappel : le nombre d'éléments que contient un tableau est donné par la valeur `tab.length` . Ainsi toujours dans notre exemple : `args.length` vaut 3.

Exercice 2 : Ecrire un programme (Affiche) qui affiche tous les paramètres qui lui sont transmis en ligne de commande. Exemple:

```
> java Affiche bob 5 a
bob
5
a
```

3.2 Convertir un paramètre

Comme vous venez de le voir, un paramètre est toujours du type String. Si on veut qu'un paramètre soit une valeur entière il faut faire une conversion du type String vers le type int.

Pour cela on peut utiliser la fonction suivante :

```
Integer.parseInt(chaîne)
```

qui retourne la valeur entière écrite sous forme de chaîne de caractère dans la variable *chaîne*.

Ainsi si on veut mettre la valeur entière du premier paramètre du programme dans une variable i de type int, il suffit d'écrire :

```
int i = Integer.parseInt(args[0]);
```

Exercice 3 : Reprendre la fonction factorielle de l'exercice 1, et la modifier pour que la valeur de n soit donnée en paramètre au moment du lancement du programme. Exemple :

```
> java Fact  5

120
```

3.3 Le compte en Banque

Refaire le programme de compte en banque vu en cours. On rappelle que l'on crée deux classes: Compte et Banque. La classe Compte comprend deux attributs 'solde' et 'numero' et quatre méthodes 'deposer(s)', 'retirer(s)', 'avoirSolde' et 'avoirNumero'.

a) Ecrire la classe Compte et la compiler

b) Ecrire une classe Banque qui comprend une methode 'public static void main(...)' dans laquelle on instancie plusieurs fois la classe Compte. Afficher le solde et le numéro de chaque Compte.

Note: Pour compiler la Banque, vous verrez que le compilateur vous enverra des erreurs de type "unresolved symbol Compte". En fait, le compilateur ne trouve pas la classe Compte que vous avez compilé. Pour compiler la classe Banque, en lui demandant d'aller chercher les autres classes (telles que Compte) dans le dossier (directory) courant faire:

```
javac -classpath . Banque.java
```

3.4 Analyser les paramètres (avec Emacs ou KWrite)

On peut également extraire des caractères d'une chaîne de caractère à l'aide de la fonction suivante :

chaîne`.charAt( p )`

Cette fonction retourne le caractère situé à la position p dans la chaîne chaîne (de type String). Ainsi si on veut récupérer le premier caractère du premier paramètre du programme, il suffit d'écrire :

```
char c = args[0].charAt(0);
```

Exercice 4 : Ecrire un programme (Calcul) qui retourne le résultat d'une opération donnée en paramètre, il doit gérer les opérateurs : + - * et / .

Exemple:

```
> java Calcul  5 + 6

11
```

Attention : l'exécution dans une console Unix de la commande : `> java Calcul 3 * 4` provoque une erreur car le caractère * est interprété par le shell et est remplacé par l'ensemble des noms de fichier contenu dans le répertoire courant (s'en convaincre en exécutant la commande : `> java Affiche *`) . Il faut donc taper à la place :

> java Calcul 3 "*" 4

Ceci n'a aucune conséquence pour le programme, il recevra toujours les paramètres 3, * et 4.

Exercice 5 : Ecrire un programme convertisseur hexadécimal -> décimal (HexaToInt), qui affiche la valeur décimal **d'un** caractère hexadécimal fournit en paramètre.

```
ex:    > java HexaToInt  A
          10
```

Exercice 6 : Améliorer le programme précédent pour qu'il puisse convertir une valeur hexadécimale de plusieurs caractères. Exemple:

```
> java HexaToInt  B5
181
```

Pour connaître le nombre de caractères que contient un String on peut utiliser la fonction suivante :

chaîne`.length()`

Cette fonction retourne le nombre de caractères que contient la chaîne chaîne (de type String). Ainsi si on veut récupérer le nombre de caractères contenues dans le premier paramètre du programme, il suffit d'écrire :

```
int i = args[0].length();
```

(On peut également convertir une chaîne de caractères en tableau de caractères de la façon suivante : `char c[ ] = chaine.toCharArray();`)