TP1 Java

Instructions pour l'envoi de votre code

1. Vous êtes libres de discuter entre vous mais le code est *individuel*.
2. Un fichier (une classe) pour chaque exercice.
3. Chaque méthode (fonction) doit être testée sur au mois trois exemples. Votre code doit contenir ces tests explicitement.
4. Merci d'inclure des commentaires pour aider à la lecture de votre code.
5. Faire une archive (`.zip` ou `.tar`) de vos fichier sources (`.java`) pour les exercices suivants (ne pas inclure les échauffements) et l'envoyer à `omar.fawzi@ens-lyon.fr` avant samedi 20 septembre 23h59. Merci d'inclure dans le sujet du courriel `[Proj1] [TP1]` et de nommer votre archive `NomPrenomTP1.zip`

Les bases du langage Java

Java a été développé dans les années 1990. Quelques propriétés (positives) de Java
– Langage très "Orientée Objet". Tout programme contient au moins une classe.
– Portable. On compile le code source `.java` qui crée un ficher `.class` (appelé byte code). Le même byte code tourne sur n'importe quelle plate-forme qui a une machine virtuelle Java (JVM).
– Beaucoup de librairies pour interface graphique, accès réseaux, etc...
Par rapport à Caml, une différence importante est que Java est fortement typé, c'est à dire que toute variable a un type fixé.
Ressources plus detaillées :
– http://introcs.cs.princeton.edu/java/11cheatsheet/
– http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-java

0.1 Création d'un programme Java

En ligne de commande.

1. Créer un fichier texte appelé `MyClass.java` avec votre éditeur texte préféré.
2. Commencer par taper la structure de base de chaque programme
```
public class MyClass {
  public static void main(String[] args) {
    System.out.println("Hello");
    //Write code here
  }
}
```
3. Sauvegarder et ouvrir un terminal sur le répertoire ou le fichier est sauvegardé.
4. Pour compiler, taper `javac MyClass.java`. Cette étape compile le code et crée un fichier appelé `MyClass.class`.
5. Pour exécuter, taper `java MyClass`.

Utilisant un IDE. Vous pouvez faire tout ça plus facilement en utilisant un "integrated development environment" tel que Eclipse.

0.2 Bases de la syntaxe

- Toute ligne de code se termine pas ;
- Un bloc est delimité par des accolades `{ ... }`
- Commentaires `// comments` ou
  ```
  /* multiple line
  comments */
  ```
- Déclaration d'une variable
  ```
  int x;
  double d = 1.543;
  char c = 'a';
  String s = "java";
  ```
  Les types : http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html
  Les opérateurs http://docs.oracle.com/javase/tutorial/java/nutsandbolts/opsummary.html
- Pour afficher
  ```
  System.out.println("Hello"); // Write to the standard output stream
  System.out.println("s = " + s + " and d = " + d); // + for concatenation
  ```
- Pour pouvoir lire une entrée, il faut d'abord importer la classe `java.util.Scanner` en rajoutant au début du fichier
  ```
  import java.util.Scanner
  ```
  Ensuite, dans votre programme, créer une variable de type `Scanner` (on parle d'objet de type `Scanner`) en utilisant le code suivant
  ```
  Scanner sc = new Scanner(System.in); // Scan the standard input stream
  String s = sc.nextLine();
  int x = sc.nextInt();
  ```
- Tableaux http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
  - Declaration `int[] tableInt;`
    En général `type[] tableType;`
  - Initialization `tableInt = new int[10];`
    ```
    tableString = new String[20];
    tableChar = {'a', 'b', 'c', 'd'};
    System.out.println("First element in tableInt is " + tableInt[0]);
    System.out.println("Length of tableInt is " + tableInt.length);
    ```
  Point technique sur le passage de tableaux à une fonction : une variable de type simple est passée par valeur mais un tableau est passé par adresse. En d'autres termes c'est possible de changer les valeurs d'un tableau par un sous-programme.

0.3 Fonctions et constructions de bases

- Fonctions :
  ```
  static type functionName(type1 x1, type2 x2) {
  // code of the function
  return z; // z have the type "type"
  }
  ```

Dans le prochain TP où nous parlerons plus de programmation objet, nous verrons la signification de `public` et `static`.

– Conditions

```
if(a == 0) {
  System.out.println("The variable is 0"); } else {
  System.out.println("The variable is nonzero");
}
```

– Boucles

```
for(int i=0;i<10;i++) {
  System.out.println(i);
}
while(i<20) {
  i = i*2;
  if(i > 15) break;
}
```

Plus de détails : http://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html

0.4 Pour s'échauffer

1. Ecrire un programme qui prend en entrée deux entiers n et m, et affiche le produit $n \cdot m$.
2. Ecrire un programme qui prend en entrée un entier n et affiche $n!$.
3. Ecrire un programme qui prend en entrée une chaîne de caractères et affiche sa longueur ainsi que le nombre de `'a'` dans la chaîne.
4. Ecrire un programme qui prend en entrée un entier n et affiche l'entier n en toutes lettres si $0 \leq n \leq 10$ et affiche `nombre trop grand` sinon.

1 Factorisation

1. Ecrire une fonction `isPrime(long n)` qui prend en entrée un entier n et qui renvoie un booléen avec la valeur `true` si et seulement si l'entier n est un nombre premier.
2. Ecrire une fonction `factor(long n)` qui renvoie dans l'ordre croissant liste des facteurs premiers de n avec multiplicité.
3. Ecrire une fonction `printIntList(long[] list)` qui imprime la liste des entiers `list` separés par le caractère `'*'`.
4. Ecrire une fonction `printFactorization(long n)` qui affiche la factorisation en nombre premier de n en affichant explicitement la multiplicité de chaque facteur premier. Par exemple, sur l'entrée `45`, votre programme doit afficher `45 = 3 ^ 2 * 5`.
5. Essayer votre programme sur les entiers `27, 25008, 1900988, 87178291199`.

2 Chaînes de caractères

1. Ecrire une fonction `numOccurrences(String s, String p)` qui prend en entrée deux chaînes de caractères s et p et renvoie le nombre de fois que la chaîne p apparait dans la chaîne s en tant que sous-chaîne.
2. Ecrire une fonction `numOccurrencesWithWildcard(String s, String p)` qui prend en entrée deux chaînes de caractères $s \in \{A, C, T, G\}^*$ et $p \in \{A, C, T, G, ?\}^*$ et renvoie

le nombre de fois que la chaîne p apparait dans la chaîne s en tant que sous-chaîne, le symbole `?` peut prendre la place de n'importe quel caractère.

3. Ecrire une fonction `maxPattern(String s, int k)` qui prend en entrée une chaîne de caractères s et un entier k et renvoie le chaîne de caractère de taille k qui apparait le plus souvent en tant que sous-chaîne de s.

3 Produit de matrices

1. Ecrire une fonction `multiply(int[][] m1, int[][] m2)` qui prend en entrée deux matrices $m1$ et $m2$ et renvoie le produit des deux matrices.
2. Ecrire une fonction `matrixPower(int[][] m, int p, int z)` qui prend en entrée une matrice m et un entier $p \geq 0$ et renvoie la matrice m^p où les calculs sont effectués modulo z. Votre fonction devrait prendre un temps raisonnable pour tout $p \leq 2^{32} - 1$.
3. Ecrire une fonction `printMatrix(int[][] m)` qui affiche la matrice m.