TP 12 Programmes Java

D'abord, démarrer un terminal en utilisant une des méthodes indiquées sur le site ci-dessous, comme au TP précédent. Le mieux est peut-être d'utiliser la machine virtuelle Antix.
`cedric.cnam.fr/~porumbed/vari1/term/`

Taper les commandes suivantes pour commencer. Cela pourrait vous aider à vous mieux organiser votre travail ; vous pourriez écrire tous vos fichiers `Java` dans un dossier `tp12`.

```
cd                #se placer dans le dossier personnel
mkdir tp12        #créer un nouveau dossier
cd tp12           #se placer dans le nouveau dossier tp12
touch Exo1.java   #créer nouveau fichier
geany Exo1.java& #editer le fichier, n'oublier pas le '&' sinon le terminal reste bloqué
```

Exercice 1 Modifier le fichier `Exo1.java` pour écrire un programme `Java` qui affiche *Ciao Bella*. Pour des exemples, n'hésitez pas à regarder la première diapo du Cours 12 sur le `Java`, disponible sur le site web

`cedric.cnam.fr/~porumbed/vari1`

Pour compiler et exécuter, il faut utiliser `javac` et `java`, voir les indications fournies dans les diapos du cours 12 ou le TP précédent.

Exercice 2 Modifier le programme précédent et écrire une classe `Exo2` dans un fichier `Exo2.java`. Ajouter une fonction statique `puissance4(float x)` qui renvoie $x^4 = x \times x \times x \times x$. Appeler cette fonction dans le programme principal (c. à. d. dans la fonction `main(...)`) pour afficher 2^4, 3^4, 4^4 et 5^4.

Exercice 3 Écrire une classe `Exo3` dans un fichier `Exo3.java`. Définir une fonction statique

`static double impots(double revenuFiscalRef)`

qui renvoie le montant des impôts pour un revenu fiscal référence `revenuFiscalRef`. Faire fonctionner le code ci-après qui permet à l'utilisateur de saisir son revenu fiscal. Il s'agit d'un impôt progressif, par tranches, basé sur les règles (simplifiées) suivantes définies par la loi :

- Pas d'impôt pour la tranche $0 - 9710$ euros.
- 14% pour la tranche de 9710 à 26818 euros
- 30% pour tout revenu supérieur à 26818 euros.

Attention : l'impôt est progressif en France. Tester votre programme sur les revenus suivants :
- `impots(10000)`$= 0.14 \times (10000 - 9710) = 40.6$;
- `impots(30000)`$= 0.14 \times (26818 - 9710)$
$+0.3 \times (30000 - 26818) = 3349.72$.

```
class Exo3{
    static double impots(double ...){
            ...
    }
    public static void main(String[] args){
        java.util.Scanner scan = new java.util.Scanner(System.in);
        System.out.println("Entrer votre revenu fiscal de référence:");
        double revenuNet = scan.nextDouble();                    //lecture clavier
        double mesImpots = impots(revenuNet);
        System.out.println("Vos impôts="+mesImpots);
    }
}
```

Exercice 4 Écrire une classe `Exo4` dans un fichier `Exo4.java`. Ce programme `Exo4` doit demander à l'utilisateur de saisir son salaire annuel *comme un nombre entier*. Si le salaire est supérieur ou égal à 1000000, il faut afficher « vous avez besoin d'un parachute doré ». Si le salaire est inférieur à ou égal à 10000 euros, il faut afficher « vous ne payez pas d'impôt ; sympa». Pour tout salaire entre 10001 euros et 999999 euros, il faut afficher « je pense que vous allez payer des impots ». **Il faut écrire que la fonction main(...)**

Exercice 5 Écrire une fonction

```
void trouveMinMax(double[] tabd)
```

qui enregistre la valeur minimale et la valeur maximale du tableau dans les variables globales `min` et `max`. Afficher `min` et `max` dans le programme principal pour `double[] t={1.5, 9,0.3,8, 12.4,13}`
Indication : voir les programmes faits en classe au cours 12, voir `cedric.cnam.fr/~porumbed/vari1/`

Exercice 6 Écrire une classe `Exo6` dans un fichier `Exo6.java` pour tester si un entier saisi par l'utilisateur appartient à un tableau. Il faut écrire **que la fonction main(...)** ; le tableau est initialisé au début du main :

```
int[] tab = {12, 15, 13, 10, 8, 9, 13, 14};
```

**Indication : ** Utiliser une variable booléene `valSaisieExiste` qui sera au début `false`. Mais si on trouve la valeur saisie, on met `valSaisieExiste=true`. Ainsi, à la dernière ligne du programme il suffit de vérifier si `valSaisieExiste` vaut `true` of `false`. Cette logique est illustré par le pseudo-code suivant :

```
valSaisieExiste = false
for i = 0 to n-1
    if valSaisie==tab[i]
            valSaisieExiste =true
if valSaisieExiste
    print "trouvé"
else
    print "pas trouvé"
```

Exercice 7 Écrire une fonction d'en-tête

```
static boolean premier(int n){
    .....
}
```

qui renvoie `true` si `n` est premier ou `false` sinon.
Rappel : `n` est premier si on ne trouve **aucun** nombre i dans la liste $2, 3, \ldots n-1$ qui divise `n`, c. à. d. $n\%i \neq 0\ \forall i \in \{2, 3, \ldots n-1\}$. Tester la fonction `premier(...)` sur les nombres $5, 19, 51$ dans le `main(..)`.

Exercice 8 Écrire un programme qui affiche un carré composé de caractères '*', dont la taille est entrée par l'utilisateur. Par exemple, pour une taille 4, ce sera :

```
****
****
****
****
```

Exercice 9 **(bonus)** Dans la continuité de l'exercice précédent, tracer un triangle avec la pointe tournée vers le haut. Le nombre de lignes est fixé à $n = 4$, mais vous pouvez tester d'autres valeurs. Pour chaque ligne, il faudra afficher un certain nombre d'espaces sans saut de ligne (via `System.out.print(" ")`), puis un certain nombre d'étoiles, puis un saut de ligne (via `System.out.println("")`). Il faut déterminer le nombre d'espaces et le nombre d'étoiles pour chaque numéro de ligne $i \in \{0, 1, \ldots n-1\}$ (ex, $n-i$ espaces ?)

```
   *                #//3 espaces, 1 étoile
  ***               #//2 espaces, 3 étoiles
 *****              #//1 espace , 5 étoiles
*******             #//0 espaces, 7 étoiles
```