TP 13 Programmes Java

D'abord, démarrer un terminal
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 tp13       #créer un nouveau dossier
cd tp13          #se placer dans le nouveau dossier tp12
kate  Exo1.java& #editer le fichier, n'oublier pas le '&' sinon le terminal reste bloqué
```

Exercice 1 Écrire dans le fichier `Exo1.java` un programme `Java` qui affiche le texte ci-dessous (3×3 étoiles) :

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

Il suffit d'ajouter trois lignes au code ci-dessous.

```
class Exo1{
    public static void main(String[] args){
        ....
    }
}
```

Note (bonus) : On peut réaliser cela en une seule ligne grâce au caractère « `\n` » qui représente un saut de ligne. Exemple : `System.out.println("un \n deux");` affiche deux lignes.

Exercice 2 Écrire une classe `Exo2` dans un fichier `Exo2.java`. Ajouter une fonction statique `somme(float x, double y)` qui renvoie la somme $x + y$ comme un double. Appeler cette fonction dans le programme principal et vérifier si le calcul est correct.

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 de référence. Il s'agit d'un impôt calculé grâce à deux règles simplifiées :

- Pas d'impôt pour la tranche $0 - 9710$ euros.
- 14% pour tout revenu qui dépasse 9710, ex, pour un revenu de 10000, il faut payer $0.14 \cdot (10000 - 9710)$.

```
1  class Exo3{
2      static double impots(double ...){
3              ...
4      }
5      public static void main(String[] args){
6          java.util.Scanner scan = new java.util.Scanner(System.in);
7          System.out.println("Entrer votre revenu fiscal de référence:");
8          double revenuNet = scan.nextDouble();                    //lecture clavier
9          double mesImpots = impots(revenuNet);
10         System.out.println("Vos impôts="+mesImpots);
11     }
12 }
```

Vous pourriez avoir besoin de faire `Kate` afficher les numéros de lignes pour corriger les erreurs : taper `F11`, ou `Configuration→Configurer Kate→Apparence→Onglet Bordures→Afficher les numéros de lignes`.

Exercice 4 Écrire une classe `Exo4` dans un fichier `Exo4.java` pour tester si un entier saisi par l'utilisateur appartient au 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 5 Écrire une classe `Exo5` dans un fichier `Exo5.java`. Ajouter une fonction `sommeTab(int[] tab)` qui calcule la somme des éléments du tableau `tab`. Il faut utiliser une boucle `for`. Tester la nouvelle fonction dans le main.

Exercice 6 Écrire une classe `Exo6` dans un fichier `Exo6.java`. Ajouter une fonction

`sommeTabVal(int val, int[] tab)`

qui vérifie si la somme des éléments du tableau `tab` est égale à `val`. Cette fonction doit envoyer un `boolean`. Dans cette fonction `sommeTabVal` il faut faire appel à la fonction `sommeTab` écrite à l'exercice précédent. Tester la nouvelle fonction dans le main.

```
class Exo6{
  static boolean sommeTabVal(...){
          ...
  }
  public static void main(String[] a){
      int[] t = {10,20,30};
      if(sommeTabVal(60,t))
          System.out.println("Somme égale à la valeur 60");
  }
}
```

Exercice 7 Écrire une classe `Exo7` dans un fichier `Exo7.java`. Ajouter une méthode (fonction qui renvoie `void`) nommé `toutesLesSommes(int n)` qui calcule toutes les sommes ci-dessous :

```
1
1+2
1+2+3
1+2+3+4
...
1+2+3+4...+ n
```

Il faut afficher que le résultat de chaque somme. Par exemple, pour $n = 4$, il faut afficher

```
1
3
6
10
```

Exercice 8 Écrire une classe `Exo8` dans un fichier `Exo8.java`. Ajouter une méthode (fonction qui renvoie `void`) nommé `lesProduits(int n)` qui calcule tous les produits :

```
1*2/2
2*3/2
3*4/2
4*5/2
...
n*(n+1)/2
```

Il faut afficher que le résultat de chaque produit. Par exemple, pour $n = 4$, il faut afficher

```
1
3
6
10
```

En fait, cette fonction devrait finir par afficher les même résultats que la fonction de l'exercice précédent.

Exercice 9 Écrire une classe `Exo9` dans un fichier `Exo9.java`. Ajouter une fonction `cubeParfait(int n)` qui indique si n est un cube parfait ou pas. Il faut renvoyer un `boolean`. Un cube parfait est une valeur comme $8 = 2{\cdot}2{\cdot}2$ ou $125 = 5{\cdot}5{\cdot}5$. **Astuce :** utiliser une boucle `for` pour parcourir toutes les valeurs `i` de 1 à `n-1`. S'il y a une seule valeur `i` tel que `n` = `i`^{3}, on renvoie `true`. Si le programme finit cette boucle sans renvoyer `true`, il faut renvoyer `false` à la fin de la fonction. Tester la nouvelle fonction dans le main.