TP NOTÉ VARI1 (NFP135)

Écrire une classe `ControleContinu` dans un fichier `ControleContinu.java`. Dans cette classe vous allez pouvoir écrire 5 fonctions et méthodes, plus la méthode `main(String[] args)` qui vous permettra de tester vos fonctions.
Ce TP noté comporte 6 exercices mais je vous propose de **résoudre uniquement 5 exercices** à votre choix parmi les 6. Je vais donner 4 points pour chaque exercice choisi, pour un total de maximum 20.

Exercice 1 Écrire une fonction Java `somme(...)` qui prend un paramètre `double` et un `int` et qui renvoie la somme des deux paramètres. Naturellement, le résultat sera un `double`. On dit que l'en-tête (ou la signature) de la fonction est :

double somme(**double** x, **int** y)

Exercice 2 Écrire une fonction d'en-tête

double noteFinale(**double** exam, **double** projet)

qui calcule et renvoie la note finale d'une UE en fonction d'une note d'examen et d'une note de projet.

- Une note d'examen inférieure à 6 est éliminatoire et `noteFinale(...)` renvoie 0 dans ce cas.
- Une note de projet inférieure à 10 est éliminatoire et `noteFinale(...)` renvoie 0 dans ce cas.
- La note finale est la moyenne entre la note d'examen et la note de projet.

Exemple 1 : Pour exam=12 et projet=10, la fonction renvoie la note finale $\frac{12+10}{2} = 11$.
Exemple 2 : Pour exam=12 et projet=9, la fonction renvoie 0, car la note de projet est trop petite.

Exercice 3 Écrire une méthode `lireFichier()` qui permet de lire 7 entiers naturels à partir d'un fichier texte `in.txt` (utiliser la classe `BufferedReader`). On suppose que ce fichier comporte un entier par ligne, donc un total de 7 lignes. Stocker tous les entiers négatifs dans un tableau `int[] tabNegatifs`, déclaré dans la fonction `lireFichier()`. Afficher ces entiers en ordre inverse, séparés par des virgules (attention à ne pas laisser une virgule suivie par rien à la fin).

Exemple : pour le fichier à droite, le programme devrait afficher :

```
-9, -11
```

in.txt

```
3
7
−11
14
12
8
−9
```

Exercice 4 Écrire une fonction `nombrePremier(int n)` qui prend un paramètre n et renvoie `true` si n est premier ou `false` sinon. Pour tester si un nombre est premier, il faut vérifier s'il est divisible par 2 ou 3 ou 4 ou ... ou $n-1$. Si n est divisible par au moins un de ces nombres, alors n n'est pas premier. Le résultat sera un boolean.
Note 1 : Un cas particulier/exceptionnel à gérer : 1 est premier.
Note 2 : L'expression `n%i` calcule le reste de la division n/i ; si ce reste est 0, alors n est divisible par i.

Exercice 5 Écrire une méthode `diviseursPremiers(int n)` qui prend un argument n et qui affiche les diviseurs premiers de n. Il est (très) souhaitable d'appeler la fonction `nombrePremier(...)` dans le corps de la fonction `diviseursPremiers(int n)`.

Exercice 6 Une fraction $\frac{a}{b}$ où a et b sont des entiers ($b \neq 0$) représente un nombre rationnel. Un inconvénient de cette représentation est qu'un même nombre rationnel peut-être représenté par une infinité de fractions, ex, $3/2 = 6/4 = 9/6 = 300/200$, etc. Écrire une méthode `simplifier(int a, int b)` qui permet de simplifier la fraction $\frac{a}{b}$ et d'écrire le résultat dans deux variables globales `num` et `den` (déclarées avec le mot clé `static` dans la classe `ControleContinu` au début du code). Par exemple, si l'utilisateur appelle (dans le `main`) `simplifier(300,200)`, alors la fonction doit écrire `num`= 3 et `den`=2.