

Programmation Par Objets et Java - TD 1

Premiers petits programmes

Les exercices 1 à 6 doivent se faire dans une classe unique, avec une ou plusieurs fonctions par exercice.

Chaque exercice doit être appelé dans un `main`, qui affiche le résultat escompté.

Code indenté, identificateurs parlants, pas de duplication de code. Dans ce qui suit, `[String]methode()` signifie que je vous conseille d'utiliser la fonction `methode` de la classe `String`.

Exercice 1

Écrire une fonction `ex1SaisieChaine` qui permet de saisir et de renvoyer une chaîne. Utiliser `Scanner`.

Écrire une seconde fonction qui appelle la première tant que la chaîne saisie n'est pas en majuscules.

Utiliser `[String]toUpperCase()` et `[String]equals(...)`.

Surcharger la fonction `ex1SaisieChaine` pour qu'on puisse y passer un paramètre pour spécifier le message à afficher ("Saisir une chaîne : ", "Quel est ton nom ?", etc.)

Exercice 2

Écrire une fonction `estAlphabetique(String s)` qui renvoie vrai si la chaîne est exclusivement composée de caractères alphabétiques, sans utiliser d'expressions régulières.

Utiliser `[String]length()`, `[String]charAt(c)`, `Character.isAlphabetic(c)`.

Utiliser cette fonction dans une fonction qui boucle jusqu'à ce que la chaîne saisie soit alphabétique.

Exercice 3

Écrire une fonction qui génère une chaîne uniquement composée de voyelles, minuscules ou majuscules. Cette fonction prend en paramètre la longueur de la chaîne voulue.

Exercice 4

Utiliser la fonction précédemment écrite pour lire une chaîne. Renvoyer la chaîne "cryptée" en utilisant les règles suivantes :

- A → 4, E → 3, G → 6, I → 1, O → 0, S → 5, Z → 2.

Les autres lettres sont inchangées. La conversion de chaque lettre doit se faire que la lettre soit minuscule ou majuscule.

Utiliser `[String] length()`, `substring(d, f)`, `toUpperCase()`, `indexOf(c)`, `charAt(c)`.

Essayer d'écrire un code optimisé, très facilement modifiable si l'on ajoute une règle de codage.

Remarque : En Java, une chaîne est immuable.

Exercice 5

Écrire une fonction qui génère une chaîne de 5 caractères dont chaque caractère est généré aléatoirement. On boucle tant que la chaîne n'est pas considérée comme étant en majuscules (la chaîne peut contenir des caractères non alphabétiques sur lesquels la casse n'a pas de sens). À chaque itération, afficher la chaîne générée. En fin de boucle, afficher le nombre d'itérations nécessaires pour obtenir une chaîne correcte. Utiliser `65 + Math.random() * (123-65)` permet d'obtenir un caractère aléatoire entre 'A' et 'z'. Un transtypage (`cast, (char) unEntier`) sera nécessaire pour bien obtenir des caractères et pas des entiers. La concaténation de chaîne peut se faire avec `[String]concat`, ou plus simplement, l'opérateur `+`.

Exemple de déroulement :

```
Tentative 1 : NnkCR
Tentative 2 : ^RUJe
Tentative 3 : VfOWc
Tentative 4 : mAz\A
Tentative 5 : XhXD[
Tentative 6 : \oJLu
Tentative 7 : V_B][
Nb itérations : 7
```

Dans un second temps, faites une version qui n'accepte pas les caractères non alphabétiques (réutiliser la fonction de l'exercice 2).

Exercice 6

Écrire une fonction qui affiche tous les entiers entre 1 et `n`, cette valeur étant saisie par une fonction que vous écrirez. L'affichage des entiers est perturbé de la façon suivante :

- Affichage de "Jazz" si l'entier est un multiple de 3.
- Affichage de "Bundle" si l'entier est un multiple de 5, sauf si cet entier est aussi un multiple de 3, dans ce cas affichage de "Jazz Bundle".

Faire une version avec autant de `System.out.print(\n)` que vous voulez, puis une version avec un seul `System.out.println`.

Exercices sur les tables

Créer une nouvelle classe pour cette partie.

Faire tout ce qui suit avec des tableaux de taille fixe.

- Écrire une fonction qui permet la saisie d'un tableau d'entiers dont la taille est passée en paramètres. Utiliser cette fonction pour les questions suivantes.
- Écrire une fonction qui additionne les entiers stockés dans un tableau.
- Écrire une fonction qui compte le nombre d'entiers pairs dans un tableau.
- Écrire une fonction qui prend en paramètres un certain nombre d'entiers et renvoie le plus grand pair (la fonction peut avoir pour paramètre un tableau ou une suite d'entiers : pour gérer cela, la fonction

doit utiliser des *varargs* `public static int plusGrandPair(int... valeurs)`.

- Écrire une fonction qui fusionne deux tableaux triés, et renvoie un tableau trié. La fonction doit tirer parti du fait que les tableaux en entrée sont triés.

Refaire ces exercices avec des instances d'`ArrayList<Integer>`. Essayez d'utiliser les fonctions toutes faites de la classe : `addAll`, `sort`.