Exercices du module
Renforcement Technique JAVA

→ Important !

Vous devrez rendre vos exercices sur un dépôt GitHub que vous aurez préalablement créé. Il devra être public et s'appeler « **JavaExercises_[votre nom-votre prénom]** ». Par exemple : « JavaExercises_dupont-martin ».

A l'intérieur de votre dépôt, vous devrez ajouter un dossier nommé « file5 ». Chacun de vos exercices devra être dans un fichier distinct et déposé dans ce dossier. Vous nommerez les fichiers « **exercise1.java** », « **exercise2.java** », etc.

Rappel des commandes Git essentielles à utiliser sur votre terminal :

- git clone [nom de dépôt]
- git add [nom de fichier]
- git commit -m "[description de vos modifications]"
- git push origin main
- git status
- git log

⇨ Préambule

Dans ces exercices, nous apprendrons à manipuler des tableaux de données ainsi que la classe à travers plusieurs class. Pour ces exercices vous devrez définir un « **package** » à vos fichiers.

⇨ Génération de tableau

Créez donc un fichier nommé « **GenerateArray.java** » et insérez à l'intérieur une classe publique nommée « **GenerateArray** » dans laquelle se trouve :

- Une variable privée statique nommé « **list** » de type ArrayList, instancié avec aucun argument.
- Une méthode publique statique « **generate** » qui prends en paramètres 2 éléments qui sont la taille du tableau attendu et le nombre maximum que l'on souhaite pour un entier. Cette méthode vous permettra de générer un tableau d'entiers. Cette méthode ne retourne rien.
- Une méthode publique statique « **getList** » qui retourne la variable privée « **list** »

Dans la méthode « **generate** » vous devrez faire appel à la class « **Random** » pour générer X nombres aléatoires et les ajouter au tableau « **list** ».

Attention : Les nombres aléatoires doivent être compris entre 1 et le nombre max inclus. Vous devrez vérifier qu'il n'y a pas deux fois le même nombre dans la liste.

⇨ Exercice 1

Dans ce premier exercice vous allez devoir réaliser un tri par ordre croissant en utilisant la méthode du tri à bulle (ou Bubble sort) des nombres de la liste d'entiers. Dans un premier temps, vous allez devoir générer cette liste à l'aide de la class rédigé précédemment. Pour ce faire, vous allez devoir inclure à votre fichier deux éléments : Le package et l'importation de la class.

Dans la méthode « **main** », n'oubliez pas d'instancier la class « **GenerateArray** » avant de l'utiliser. Vous pourrez ensuite générer une liste de 10 entiers qui seront compris entre 1 et 100 inclut. Vous afficherez ce tableau non trier pour s'assurer qu'il comporte bien 10 valeurs. Pour récupérer la valeur de la liste, vous devrez faire appel à la méthode « **getList** » de « **GenerateArray** »

Il y a plusieurs façons de faire un tri à bulles. L'une de ces façons consiste à créer une première boucle for qui parcourra l'entièreté du tableau avec une variable « i ». A l'intérieur, vous aurez une seconde boucle qui parcourra le même tableau avec une variable « j ». Le principe de celle-ci sera de vérifier que « j » est inférieur à la position de « i » dans le tableau.

Vous rajouterez ensuite à l'intérieur de la seconde boucle une condition selon laquelle, si la valeur à la position « j » est supérieure à la suivante, alors vous échangez leurs valeurs. Pour y parvenir, pensez à utiliser une variable « temp » pour effectuer la permutation de données.

Exemple de résultat attendu :

```
Liste d'origine : [30, 74, 24, 23, 14, 20, 32, 94, 18, 68]
Liste triée : [14, 18, 20, 23, 24, 30, 32, 68, 74, 94]
```

⇨ Exercice 2

Maintenant que vous avez compris comment faire un tri à bulles, vous allez devoir l'effectuer par ordre décroissant en utilisant la même méthode que l'exercice 1, sur une liste d'entier toujours générer aléatoirement.

Exemple de résultat attendu :

```
Liste d'origine : [53, 95, 41, 78, 32, 40, 26, 82, 36, 63]
Liste triée : [95, 82, 78, 63, 53, 41, 40, 36, 32, 26]
```

⇨ Exercice 3

Votre programme affiche désormais un tableau trié de manière croissante et décroissante. Puisqu'il fonctionne très bien avec des nombres aléatoires, il est temps de faire intervenir un utilisateur en plus.

Dans votre fichier « **GenerateArray.java** » vous allez devoir ajouter une méthode publique statique « **addItem** » qui prends en paramètre un entier et qui l'ajoutera à la variable publique de la class.

De retour dans le fichier de l'exercice, vous allez devoir faire une boucle pour, en plus d'avoir une liste générer aléatoirement, d'avoir 5 entrées utilisateur. Vous devrez vous assurez que ces entrées soient bien des nombres et si c'est bien le cas, ils seront ajoutés à la « **list** » par le biais de « **addItem** ». N'oubliez pas de récupérer la liste mises à jour après.

Vous pourrez ensuite effectuer un tri par ordre décroissant de vos 15 nombres de la liste.

Exemple de résultat attendu :

```
Liste d'origine : [56, 28, 69, 22, 18, 16, 33, 9, 10, 84]
Entrez un nombre :
12
Entrez un nombre :
45
Entrez un nombre :
34
Entrez un nombre :
65
Entrez un nombre :
78
Liste d'origine avec inputs : [56, 28, 69, 22, 18, 16, 33, 9, 10, 84, 12, 45, 34, 65, 78]
Liste triée : [84, 78, 69, 65, 56, 45, 34, 33, 28, 22, 18, 16, 12, 10, 9]
```

⇨ Pour aller plus loin

Voici quelques idées de bonus pour aller plus loin :

- Pour une génération de 500 nombres, est-ce que votre programme fonctionne toujours pour triller ?
- Est-ce que votre algorithme est performant et rapide sur des gros tableaux ? est-ce qu'il n'y à pas un moyen d'améliorer la vitesse ?
    - Sous linux et macOS, utiliser la commande « time | java … »
    - Sous Windows utiliser la commande « Measure-Command { java … } »
- Dans un fichier « bonus.java », essayer d'autres solutions de tri pour augmenter la vitesse de celui-ci sur des grosses listes.