Programmation Objet en Java - Feuille de TP 2

Attention, vous devez faire tous les exercices de ce TP en utilisant uniquement les informations qui se trouvent dans les pages de documentation officielle !

Exercice 1. Chercher la documentation de la classe `ArrayList` ici : <https://docs.oracle.com/en/java/javase/17/docs/api/>.

- a) Dans une classe `ArrayListTest`, créer une liste modifiable `integers` et permettant de contenir des entiers. Quel est le type de cette liste?
- b) Ajouter tous les entiers de 5 à 20 à cette liste, puis afficher la liste. Afficher également la taille de la liste. Jusqu'à quelle taille pouvez-vous aller?
- c) Afficher le dixième élément de `integers`, puis son vingtième élément. Que se passe-t-il?
- d) Retirer l'entier 10 de la liste, puis retirer son 10^e élément. Vérifier en affichant la liste. Pourquoi est-ce que les deux méthodes de suppression n'ont pas le même type de retour?
- e) Tester que la liste contient bien l'entier 13.
- f) Écrire une méthode `toStrings` qui prend en paramètre une `ArrayList` d'entiers et qui renvoie une `ArrayList` de chaînes des caractères représentant ces entiers. Quelle est la particularité de cette méthode ?
- g) Tester la méthode `toStrings` sur la liste `integers` et tester que la liste obtenue contient bien la chaîne de caractères "13".

Exercice 2. Chaînes de caractères

- a) En java, il est possible de récupérer des informations depuis l'entrée standard (ici, le clavier) grâce aux objets de la classe `Scanner`. Chercher dans la doc comment récupérer une chaîne de caractères entrée par l'utilisateur (et la stocker dans une variable `user`) : <https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Scanner.html>. Essayer dans une classe `StringTest`.
- b) Faire afficher la chaîne de caractères entrée par l'utilisateur en majuscules. Vous trouverez ce dont vous avez besoin ici : <https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html>.
- c) Est-il possible de modifier la chaîne `user` ? Pourquoi ?
- d) S'il existe, afficher le septième caractère de la chaîne `user`, et sinon afficher "pas de septième caractère".
- e) Tester et afficher si la chaîne `user` contient la chaîne "cha", puis si la chaîne `user` est exactement la chaîne "cha".
- f) Écrire un méthode `characters` qui, étant donnée une chaîne, renvoie une liste de ses caractères qui ne sont pas des "blancs" (espace, tabulation, retour à la ligne, etc...). Quelle est la particularité de cette méthode ?
Remarque : on vous conseille de regarder la documentation de la classe `Character`.
- g) Écrire un méthode `userStrings` qui prend un `Scanner` en paramètre et attend que l'utilisateur entre des chaînes de caractères (sans espaces) au clavier jusqu'à ce qu'il rentre exactement "stop". La méthode renvoie alors la liste des chaînes de caractères de l'utilisateur.

Exercice 3. Des livres

- Définir un record `Book` pour représenter des livres. Chaque livre possède un titre (`title`) et un auteur (`author`) qui sont des chaînes de caractères.
- Écrire une classe `BookTest` avec une méthode `main` pour tester le record `Book`. Créer et afficher 3 livres.
- Dans `Book`, écrire une méthode d'instance `authorBeginsWith` qui prend une chaîne de caractères `begin` en paramètre et renvoie `true` si le nom de l'auteur commence par `begin` et `false` sinon.
- Que se passe-t-il si on utilise cette méthode sur le livre suivant ?

```
Book b = new Book("Contes chinois", null);
```

- Corriger le problème en empêchant la construction d'un `Book` avec un nom d'auteur `null`. Quel autre problème du même type pourrait se produire ? Corriger le code.
- Écrire un autre constructeur qui prend juste un `title` et pas d'`author`. On initialisera le champ `author` avec "no author" dans ce cas.
- Ajouter une méthode `firstInAlphabeticalOrder` qui prend un `Book` en paramètre et renvoie celui qui est le premier dans l'ordre alphabétique du titre, entre ce paramètre et le `Book` auquel on applique la méthode.

Exercice 4. Si ce n'est pas déjà fait, terminer l'exercice du TD2 sur les **listes de poissons**.

Exercice 5. (Optionnel) Chemins

On se place dans le plan à coordonnées entières. On appelle *pas* un vecteur horizontal ou vertical de longueur 1. On considère qu'un chemin entre deux points p_1 et p_n est une liste de points $[p_1, p_2, p_3, \dots, p_n]$ telle que les couples $(p_1, p_2), (p_2, p_3), \dots, (p_{n-1}, p_n)$ forment des pas vers la droite ou vers le bas et dont les extrémités sont p_1 et p_n .

Par exemple, la liste $[(0, 2), (1, 2), (1, 1), (2, 1), (2, 0), (3, 0)]$ forme bien un tel chemin, mais pas la liste $[(0, 2), (2, 1), (2, 0), (3, 0)]$ car il manque des points, ni la liste $[(0, 1), (1, 1), (1, 2), (2, 2), (2, 1), (2, 0)]$ car les points $(1, 1)$ et $(1, 2)$ forment un pas vers le haut.

- Dans le record `Point` de la semaine dernière, écrire une méthode `isPathInRectangle` qui prend en paramètre une liste de `Points` et qui renvoie `true` si la suite des points forme un chemin entre le premier et le dernier point de la liste, et `false` sinon.
- Si ce n'est pas déjà fait, écrire une version récursive de la méthode précédente.
- Ajouter une méthode `pathsInRectangle` qui prend un point en paramètre et renvoie un liste de tous les chemins (des listes de points) allant du point auquel on applique la méthode jusqu'à celui passé en paramètre.