IS2A4 – PPO – T.P. 7b

Résumé

Ce TP aborde les lambda expressions en java.
N'hésitez pas à consulter la documentation en ligne sur le site d'oracle : `https://docs.oracle.com/javase/8/docs/api/` (regardez la version correspondant la version de java que vous utilisez).

I Préparation du TP (5 min)

I.1. Créer un dossier spécifique pour les TP de PPO qui vous appellerez `TP_PPO` par exemple.

I.2. Dans ce dossier, créer un dossier `TP7b`. C'est dans ce dossier que vous travaillerez aujourd'hui. Durant les prochaines séances de TP, essayez de garder cette pratique.

I.3. Récupérez le fichier `TP7b_source.zip` sur moodle et le décompresser dans `TP7b`.

L'objectif de ce TP est de parser un fichier csv où chaque ligne est une information sur une gare de france et de récupérer des informations en utilisant les lambda expressions pour le faire.

II Parseur ligne par ligne (10 min)

II.1. Créer une classe `VoirGare.java` qui va prendre le nom d'un fichier en paramètre et va vous permettre d'afficher le contenu du fichier ligne par ligne (vous pourrez créer un `BufferedReader` en passant par un `FileReader`).

II.2. Tester votre code avec le fichier `liste_gare_short.csv`.

III transformer chaque ligne en une gare (20 min)

III.1. Décomposer chaque ligne en une liste de strings séparés par le caractère `\t` (vous pourrez utiliser la méthode `split(...)` des instances de `String` et la méthode `static Arrays.asList(...)`).

III.2. Identifier les différentes colonnes du fichier.

III.3. Utiliser la classe `Gare.java` pour transformer chaque ligne en une instance de la classe `Gare`.

III.4. Tester votre code avec le fichier `liste_gare_short.csv`.

IV Une nouvelle classe ListeGare (25 min)

IV.1. Créer une classe `ListGare` qui prendra comme constructeur une liste de gares.

IV.2. Créer une méthode `getSetDepartement()` dans `ListGare` qui donne l'ensemble des départements des gares.

IV.3. Créer une méthode `getListGaresWhere(String m)` dans `ListGare` qui donne l'ensemble des gares qui ont `m` comme sous-chaîne.

IV.4. Créer une méthode `getSortedLibelle()` dans `ListGare` qui donne l'ensemble des libelles des gares dans l'ordre lexicographique.

IV.5. Créer une méthode `getCountCommunes(String m)` dans `ListGare` qui donne le nombre de commune des gares qui sont dans le département `m`.

Indices :

- `https://howtodoinjava.com/java8/java-stream-distinct-examples/`

V Plus proche gares (20 min)

V.1. Créer une méthode `plusProcheGares(Coordonnees c)` dans `ListGare` qui renvoie la liste des gares des plus proches aux plus lointaines de la coordonnée `c`. Vous limiterez le nombre de gares dans la liste à 20 maximum.

Indices :

- `https://howtodoinjava.com/java8/java-stream-limit-method-example/`

V.2. Créer une méthode `cheminExtremite(Coordonnees debut, Coordonnees fin)` dans `ListGare` qui renvoie la gare où commencer le périple et la gare où s'arreter.

VI Autres questions (Bonus)

VI.1. Créer un méthode `getGareIsole()` dans `ListGare` qui donne la gare la plus isolée, c'est à dire celle qui est la plus loin de toutes les autres.

VI.2. Créer un méthode `getGareLinked(double a)` dans `ListGare` qui donne pour une distance `a`, les gares qui ont le plus d'autres gares dans un rayon de `a`.

VI.3. Créer une méthode `getBetterCommune(String dep)` dans `ListGare` qui pour un département `dep`, donne l'ensemble des communes ordonnées de manière décroissante par le nombre de gares dans chaque commune. (N'utilisez qu'un seul `stream`)

Indices :

- `https://docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html`

VI.4. Créer une méthode `getNombreMoyenGareParCommune()` dans `ListGare` qui donne le nombre moyen de gare par commune.

Indices :

- `https://docs.oracle.com/javase/8/docs/api/java/util/IntSummaryStatistics.html`

`Temps total du sujet : 1 heures et 20 minutes`