Comprendre l'API des collections

Le site `https://www.prix-carburants.gouv.fr/` permet de connaître le prix des carburants dans tous les points de vente en France. Chaque point de vente à l'obligation de renseigner sur ce site tout changement de prix sur les carburants qu'il vend. Dans le cadre de l'Open Data, les données peuvent être récupérées. Nous allons les exploiter dans les exercices suivants.

Les fichiers de prix des carburants peuvent être récupérés dans [1] :

```
/mnt/n7fs/ens/tp_cregut/prix-carburants
```

Exercice 1 : Informations de Prix-carburants

Il est possible de récupérer par année (ou par période de 30 jours), la liste de tous les points de vente (PDV) de carburant.

Pour chaque point de vente, on peut obtenir :

— son identifiant (unique par définition),
— sa latitude et sa longitude (position du point de vente),
— son code postal,
— sa ville,
— son adresse,
— les services proposés (chaque service est décrit par un texte, une chaîne de caractères),
— pour chaque carburant vendu, la liste des changements de prix : la date du changement et le nouveau prix exprimé en millième d'euro,
— les périodes de rupture par carburant (non traitées),
— la fermeture temporaire ou définitive du point de vente (non traitée).

Des cas d'usages classiques consistent à :

— obtenir les services fournis par un point de vente
— obtenir, pour un carburant donné, le prix de ce carburant à une date donnée.

1.1. Proposer un diagramme de classe pour modéliser un point de vente et ses caractéristiques en veillant à ce que l'on puisse obtenir efficacement, pour un carburant donné, son prix à une date donnée, par exemple, le point de vente d'identifiant 31075001, pour le gazole, affiche un prix de vente de 1235 millième d'euro le 25 janvier 2017.

1.2. Indiquer les collections de l'API Java à utiliser pour implanter ce diagramme de classe.

Exercice 2 : Comprendre et utiliser le code fourni

Les classes à regarder sont `Carburant` et `PointDeVente`.

2.1. Lister les principaux choix faits et les comparer aux réponses des questions précédentes.

2.2. Écrire le code de la méthode `getPrix` de la classe `PointDeVente` et la tester en utilisant la classe de test `PointDeVenteTest`.

1. Si le système de gestion de fichiers de l'N7 est inaccessible depuis votre machine, vous pouvez faire, en étant sur le VPN : `scp monLogin@c201-01.enseeiht.fr:/mnt/n7fs/ens/tp_cregut/prix-carburants/\*.zip .`
Les fichiers sont aussi disponibles à l'URL : `http://cregut.perso.enseeiht.fr/pc/`

2.3. La classe `Main` correspond au programme principal. Sa méthode principale prend en paramètre le nom du fichier qui contient les prix. Elle utilise la méthode `fromXML` de la classe `PointsDeVente` qui retourne un objet de type `Iterable<PointDeVente>`.

Compléter la méthode `repondreQuestions` de la classe `Main` pour construire un tableau associatif qui recense tous les points de vente avec un accès par l'identifiant du point de vente. Il faudra exécuter la classe en ajoutant comme arguments l'un des fichiers de prix des carburants. On peut directement fournir le fichier d'archive .zip, inutile d'en extraire le contenu.

Exercice 3 : Exploitation des points de vente

De nombreuses informations peuvent être calculées sur ces points de vente. Nous allons nous limiter à quelques unes que nous implanterons directement dans la méthode `repondreQuestions` même s'il serait préférable de les définir sous forme de sous-programmes. Concentrons nous sur l'utilisation de l'API des collections !

1. Afficher le nombre de points de vente.
2. Afficher le nombre de services existants.
3. Afficher tous les services existants.
4. Afficher le prix du gazole du point de vente d'identifiant 31075001 [2] le 25 janvier 2017 à 10h.
5. Afficher le nombre de villes offrant au moins un point de vente.
6. Afficher le nombre moyen de points de vente par ville [3].

Exercice 4 : Exploitation de Java8 et des streams

Java8 apporte deux améliorations importantes : les lambdas [4] et les streams qui permettent d'écrire de manière plus concise et efficace des traitements sur des collections et plus généralement des sources de données.

4.1. Lire et comprendre le code suivant :

```
1 // le nombre de villes offrants un certain nombre de carburants
2 for (int nbCarburants = 1; nbCarburants < 7; nbCarburants++) {
3    final int nb = nbCarburants;
4    long nbPdvNbCarburantsEtPlus = pdvs.values().stream()
5          .filter(p -> p.getPrix().keySet().size() >= nb)
6          .count();
7    System.out.printf("PDV␣proposant␣au␣moins␣%d␣carburant(s)␣:␣%d%n",
8          nbCarburants, nbPdvNbCarburantsEtPlus);
```

2. Attention en Java à noter 31075001L (L pour indiquer qu'il s'agit d'un entier long). En effet, en raison des conversions implicites vers les classes enveloppes, si on note `get(31075001)`, le compilateur va le transformer en `get(Integer.valueOf(31075001))` et comme toutes les clés sont des entiers longs (`Long` et non `Integer`), le point de vente associé ne sera pas trouvé.

3. On s'appuiera sur l'information "ville" même si cette information n'est pas bien codifiée, n'utilise pas un référentiel et n'est donc pas en qualité. Pour s'en convaincre, on exécutera la commande suivante : `zgrep ville PrixCarburants_annuel_2017.zip | grep -i paris`

4. Les lambdas en Java constituent une facilité syntaxique qui évite l'écriture de classes anonymes. Dans le même ordre d'idée, Java8 permet d'utiliser une méthode (opérateur : :) là où une interface fonctionnelle (interface avec une seule méthode abstraite) est attendue.

```
9  }
10
11 // Afficher le nombre et les points de vente dont le code postal est 31200
12 List<PointDeVente> pdvs31200 = pdvs.values().stream()
13       .filter(p -> p.getCodePostal().equals("31200"))
14       .collect(Collectors.toList());
15 System.out.println("Nombre de PDV en 31200 : " + pdvs31200.size());
16 pdvs31200.stream().forEach(System.out::println);
```

4.2. Afficher le nombre de points de vente de la ville de Toulouse qui proposent Gazole et GPLc.
4.3. Afficher le nom et le nombre de points de vente des villes avec au moins 20 points de vente.

Remarque : Pour en apprendre un peu plus sur les lambdas et les streams, on peut lire :

- `https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html`
- `http://www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646.html`
- `https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html`