Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

TP7: Lamba expressions et introductions aux streams

L'objectif des exercices ci-dessous est de mettre en œuvre l'écriture de code en java en manipulant les lambda expressions et une approche simple des streams.

Exercice 1

Récupérer et organisez le code source fourni selon l'arborescence ci-dessous. Toutes les méthodes fournies sont implémentées en utilisant une écriture procédurale : itération sur la liste complète ; traitement des éléments sélectionnés de la liste sur la base d'une condition (if) ; et suppression d'éléments. **Attention, il manque des implémentations pour certaines méthodes et les niveaux d'accès sont probablement trop restrictifs. !**

```
├── tp7
        ├── data
        │   └── sightings.csv
        │
        └── src
            └── animalmonitoring
                    └── procedural
                        ├── AnimalMonitor.java
                        ├── Main.java
                        ├── Sighting.java
                        ├── SightingReader.java
```

La classe **`Sighting`** enregistre les détails de chaque rapport d'observation, d'un seul observateur pour un animal particulier, une fois que les détails de l'observation ont été traités. Nous ne nous préoccuperons pas directement du format des données envoyées par l'observateur. La classe d'observation est très simple.

La classe **`SightingReader`** ne doit pas être modifiée. Elle est appelée pour extraire des données depuis un fichier texte au format csv.

Le fichier **`sightings.csv`** contient un échantillon d'enregistrements d'observations au format CSV (valeurs séparées par des virgules).

La classe **`AnimalMonitor`** est utilisée pour agréger les observations individuelles dans une liste. À ce stade, toutes les observations de tous les différents observateurs et zones sont réunies dans une seule collection. Cette classe contient des méthodes pour imprimer la liste, rechercher des animaux par critère, compter le nombre total d'observations d'un animal donné, répertorier toutes les observations d'un observateur particulier, supprimer les enregistrements ne contenant aucune observation, etc.

POLYTECH NICE-SOPHIA — Université Nice Sophia Antipolis — Membre de UNIVERSITÉ CÔTE D'AZUR

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

Question 1

Implémentez les méthodes manquantes en **utilisant l'écriture procédurale** et en modifiant les accès de sorte à rester le plus restrictif possible.

Obtenir le bon nom de fichier est en fait plus délicat qu'il n'y paraît. Si nous utilisons simplement `addSightings("sightings.csv")`, l'application ne pourra pas trouver le fichier ; le fichier se trouve dans data/sightings.csv par rapport à votre projet, mais la JVM ne le sait pas.
TIPS : utilisez `System.getProperty("user.dir") + "/data/sightings.csv")`.

- Passez en paramètre le nom du fichier à la méthode addSightings de l'objet AnimalMonitor,
- appelez `printList` pour afficher les animaux.

```
* ------------------------------------------------------- *
* TP java                                                 *
*                                                         *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD7 - ex1 - question 1                         *
* ------------------------------------------------------- *
Pikachu, count = 3, area = 1, spotter = 0, period = 0
Greninja, count = 10, area = 1, spotter = 0, period = 0
Arceus, count = 0, area = 1, spotter = 0, period = 0
Pikachu, count = 1, area = 2, spotter = 1, period = 0
Pikachu, count = 3, area = 3, spotter = 2, period = 0
Pikachu, count = 0, area = 2, spotter = 3, period = 0
Greninja, count = 2, area = 1, spotter = 3, period = 0
Mew, count = 25, area = 1, spotter = 3, period = 0
Pikachu, count = 4, area = 1, spotter = 0, period = 1
Greninja, count = 16, area = 1, spotter = 0, period = 1
Mew, count = 20, area = 1, spotter = 1, period = 1
Greninja, count = 0, area = 2, spotter = 3, period = 1
Mew, count = 30, area = 2, spotter = 3, period = 1
Pikachu, count = 1, area = 1, spotter = 0, period = 2
Pikachu, count = 2, area = 2, spotter = 1, period = 2
Pikachu, count = 0, area = 3, spotter = 2, period = 2
Mew, count = 30, area = 2, spotter = 3, period = 2
Arceus, count = 24, area = 2, spotter = 3, period = 2
```

Question 2

Pour chaque animal on indique s'il est ou non en danger et le nombre d'individus restants. Dans ce qui précède, le seuil de danger a été fixé à 25.

```
* ------------------------------------------------------- *
* TP java                                                 *
*                                                         *
* @author frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD7 - ex1 - question 2                         *
* ------------------------------------------------------- *
Pikachu is endangered.
Arceus is endangered.
```

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

Question 3

Mettez en œuvre les autres méthodes et assurez-vous qu'elles fonctionnent... !

```
Pikachu is endangered: 14 left
Arceus is endangered: 24 left
Greninja is not endangered: 28 left
Mew is not endangered: 105 left
```

Question 4

Codez les autres méthodes et assurez-vous qu'elles fonctionnent... ! Attention pour les méthodes `removeZeroCounts()` et `removeSpotter(int spotterID)`, vous ne devez pas créer de collection temporaire !

Exercice 2

Vous devez convertir le traitement des collections écrit en style procédural en traitement de collection fonctionnel. Dans votre projet, copiez tout le code du package `animalmonitoring.procedural` dans un nouveau package `animalmonitoring.functional`. Assurez-vous qu'il compile et s'exécute (laissez le fichier sightings.csv où il se trouve). **A partir de maintenant, nous allons travailler dans le package fonctionnel.**

Réécrivez la méthode `printList` dans votre version de la classe `AnimalMonitor` pour utiliser une fonction lambda comme `sightings.forEach`

Exercice 3

Modifiez la méthode `printSightingsOf` pour remplacer la boucle for par un pipeline d'opérations : stream, filter et forEach pour imprimer toutes les observations d'un animal donné. Comme d'habitude, testez que cela fonctionne,

Ecrivez une méthode `AnimalMonitor#printSightingsOn(int period)` pour afficher les détails de toutes les observations enregistrées au cours d'une période particulière, qui est passé en paramètre à la méthode.

Écrivez une méthode `AnimalMonitor#printSightingsOf(String animal, int period)` qui utilise deux appels de filtre pour afficher les détails de toutes les observations d'un animal particulier faites au cours d'une période particulière - la méthode prend le nom de l'animal et la période comme paramètres.

L'ordre des deux appels de filtre est-il important ? Justifiez votre réponse dans un commentaire de votre code.

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

Exercice 4

Modifiez les méthodes `printSightingsBy` et `printEndangered` qui contiennent une boucle for par un **flux (stream)** d'opérations : stream, filter, map et forEach pour imprimer toutes les observations d'un observateur donné.

Réécrivez votre méthode `getCount` en utilisant des flux, avec **reduce** comme opération de terminal.

Ajoutez une méthode `AnimalMonitor#getCount(String animal, int spotter, int period)` qui prend trois paramètres : animal, spotter ID et period, et renvoie le nombre d'observations de l'animal donné effectuées par l'observateur au cours d'une période donnée .

Ajoutez une méthode `AnimalMonitor#getAnimals(int spotter, int period)` qui prend deux paramètres - l'identifiant de l'observateur et la période - et renvoie une chaîne contenant les noms des animaux vus par l'observateur au cours d'une période particulière.

> → Vous devez inclure uniquement les animaux dont le nombre d'observations est supérieur à zéro, mais ne vous inquiétez pas d'exclure les noms en double si plusieurs enregistrements d'observations non nuls ont été faits pour un animal particulier.

TIPS : les principes d'utilisation de reduce avec des éléments String et un résultat String sont très similaires à ceux de l'utilisation d'entiers. Décidez de l'identité correcte et formulez un lambda à deux paramètres qui combine la « somme » en cours avec l'élément suivant du flux.

Exercice 5

Réécrivez la méthode `removeZeroCounts` à l'aide de la méthode `removeIf`.

Écrivez une méthode `removeSpotter` qui supprime tous les enregistrements signalés par un observateur donné.