R2.01 TP N°10

Semaine 19

Objectifs du TP

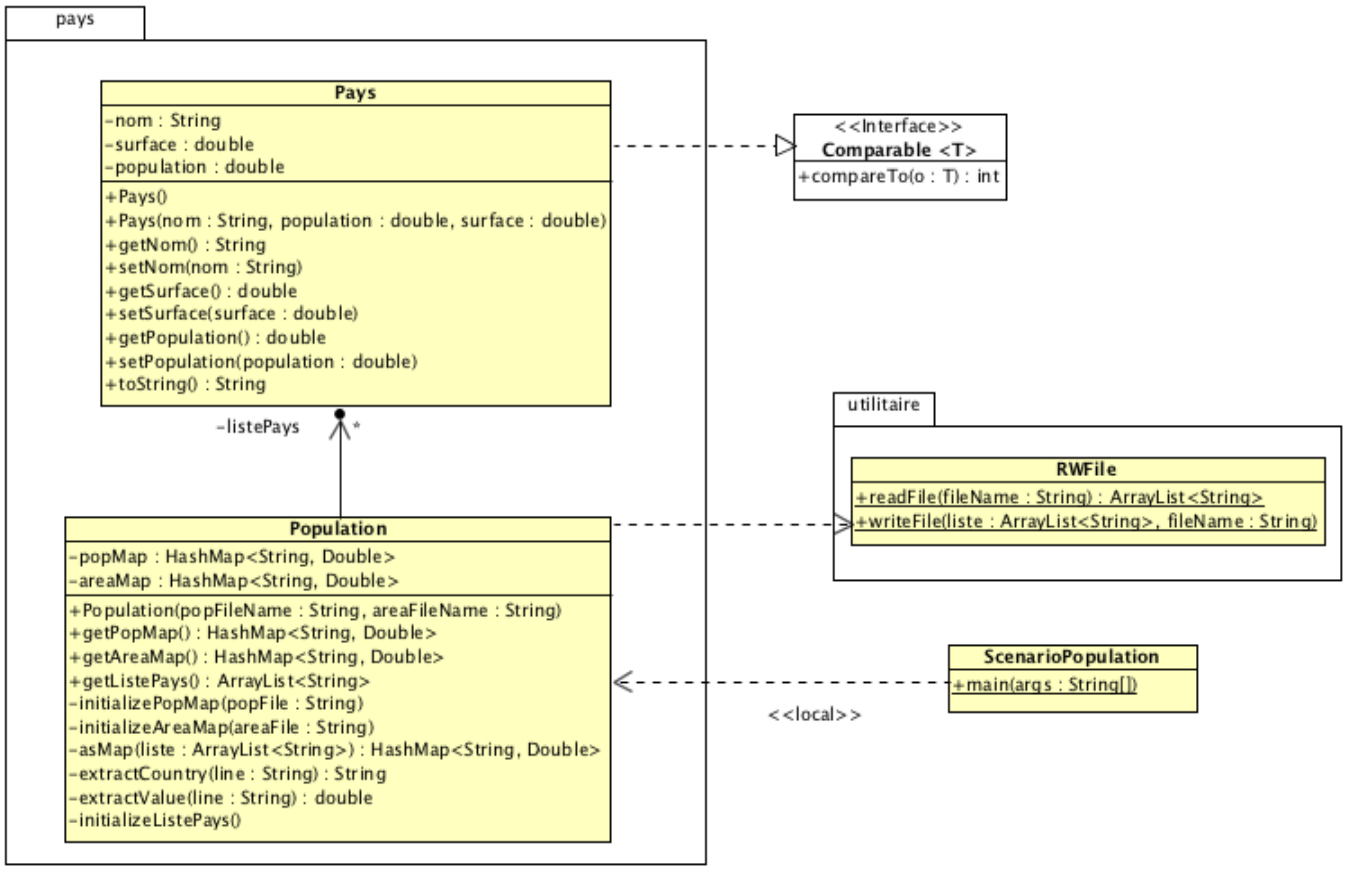
Manipuler des collections (ArrayList et HashMap)
Apprendre à lire et écrire des fichiers de texte

Préambule : Relecture des cours

Le premier travail est de relire les cours n°8-9-10 (HashMap, Exceptions, Fichiers)

Version finale de l'application pays/population

Voici le diagramme finale de l'application :



Nous allons commencer par créer l'utilitaire de lecture/écriture de fichiers.

Ensuite nous modifierons notre classe Population pour la connecter aux fichiers, puis nous ferons une nouvelle classe `SenarioPopulation.`

1- Utilitaire de lecture/écriture de données dans un fichier

Nous allons en premier définir une classe qui nous permettra de lire des données dans un fichier et

d'écrire des données dans un fichier. Nous vous donnons deux fichiers de données pour cela.

1.1 Utilisation de fichiers de données

Nous allons utiliser deux fichiers de données : un fichier donnant les surfaces des pays du monde et l'autre le nombre habitants des pays du monde, afin créer des instances de la classe `Pays` en nombre suffisant pour qu'un tri de données soit intéressant à observer dans le TP suivant.

Les deux fichiers sont structurés de façon identique :

- l'un contient pour chaque pays le nombre d'habitants (**worldpop.txt)**
- l'autre contient pour chaque pays la superficie en km2 (**worldarea.txt)**

Chaque ligne des fichiers contient deux données séparées par un ou plusieurs espaces. Par exemple quelques lignes du fichier worldarea.txt :

```
Afghanistan    647500
Akrotiri       123
Albania        28748
Algeria        2381740
American Samoa      199
Andorra        468
Angola         1246700
```

A faire : Récupérer les deux fichiers de données et les mettre dans un répertoire nommé **`data`** au même niveau que votre répertoire **`src`** (pas dedans).

1.2 Lecture et écriture des données dans des fichiers

La classe qui permettra de lire ou écrire des données dans les fichiers est nommée **`RWFile`**. Il s'agit d'un utilitaire. Elle est à développer et à mettre dans un répertoire nommé « **utilitaire** ».

Remarquez qu'une partie du code vous est donné dans le cours 10.

La classe RWFile possède deux méthodes de classe statiques et publiques:

- `public static ArrayList<String> readFile (String fileName)` : cette méthode lit les données d'un fichier texte , dont le nom est passé en paramètre, et les met dans un ArrayList qui est retourné en résultat. Pour cela elle lit le fichier ligne par ligne.
- `public static void writeFile (ArrayList<String> liste, String fileName)` : cette méthode écrit le contenu d'un ArrayList dans un fichier texte. L'ArrayList et le nom du fichier sont passés en paramètre. L'écriture se fait ligne par ligne.

L'utilisation de la classe pourra être par exemple :

```
ArrayList<String> popArray ;
popArray = RWFile.readFile("data/worldpop.txt") ;
```

A faire

1. Après avoir récupéré les fichiers de données et les avoir mis dans le répertoire **`data`**, créer le répertoire **`utilitaire`** dans votre dossier source dans le lequel vous mettrez la classe **`RWFile.java`**

Vous devez respecter l'organisation des fichiers proposée. On doit retrouver dans votre dossier src/ la même organisation que dans le dossier class/

2. Ecrite et compiler le code Java de la classe **`RWFile.java`**
3. Ecrire une classe de scénario pour vérifier le fonctionnement de vos méthodes de lecture et écriture. Vous définirez un `ArrayList` qui contiendra le résultat lu dans un fichier et ensuite vous pourrez écrire cet `ArrayList` dans un nouveau fichier (Attention de ne pas écraser les deux fichiers de départ). Cette classe doit être à l'extérieur du package utilitaire.

Pour vérifiez le contenu de l'ArrayList après la lecture : soit avec une boucle d'impression ou utiliser `java.util.Arrays` après avoir converti l'ArrayList en un tableau

```
Object[] tab=popArray.toArray();
System.out.println(Arrays.toString(tab));
```

2- Finalisation de la classe Population

Maintenant il faut faire les modifications et ajouts dans la classe pour obtenir la version finale.

Définition des deux méthodes d'initialisation des HashMap.

- **`initialisePopMap(String popFile)`** :
  - envoie le message de lecture de fichier à `RWFile` (le nom du fichier à lire est passé en paramètre du constructeur) et récupère le contenu dans un dans un `ArrayList` ;
  - ensuite utilise la méthode **`asMap`** pour transformer l'`ArrayList` en un `HashMap`.
  - Initialise **`popMap`** avec ce **`HashMap`**.

- **`initialiseAreaMap(String areaFile)`** : réalise la même chose que la méthode précédente et initialise **`areaMap`**.

Modification du constructeur

Le constructeur reçoit en paramètres le noms de fichier.

Il crée l'attribut `listePays`.

Il envoie ensuite les 2 messages d'initialisation des HashMap et leur passe le nom de fichier en paramètre.

Ajouter l'envoi du message initialisePays() .

Initialisation de listePays

- **`initialiseListePays()`** : cette méthode initialise l'attribut **`listePays`** avec le contenu des deux `HashMap` ; Elle vérifie que les 2 `HashMap` ont la même taille sinon imprimez un message d'erreur .

Une fois les deux `HashMap` remplis avec les données, on crée des instances de la classe `Pays` en

récupérant le nom du pays et les données population et surface, qui sont dans les `HashMap` popMap et popArea On ajoute les instances au fur et à mesure dans l'`ArrayList listePays`. Le remplissage se fait en ajoutant au fur et à mesure les instances de `Pays` créées et initialisées avec le parcours des deux `HashMap`.

- Reprenez votre classe **`ScenarioPopulation`** et complétez-la pour vérifier le contenu de **`listePays`** .

Astuces :

Vous pouvez récupérer l'ensemble de tous les noms de pays en demandant toutes les clés d'un HashMap, par exemple :

```
Set<String> nomsPays= popMap.keySet();
```

Ensuite l'ensemble nomPays peut être parcouru avec un for each.

Vérification de la robustesse

La robustesse est la qualité de votre classe Population de fonctionner dans des cas anormaux.

Ici on va la vérifier dans le cas où les fichiers donnés en entrée sont vides.

Pour cela vous créez un fichier vide que vous passez en paramètre du constructeur de Population dans la classe Scenario (pour popFile et areaFile).

S'il y a des erreurs à l'exécution, vous corrigez la ou les méthodes qui sont en cause.

Travail à rendre :

Déposez votre archive contenant les répertoires pays et utilitaire contenant les sources des classes **`Pays`**, **`Population`** et **`RWFile`**, plus le source de la classe **`ScenarioPopulation`** ainsi que la javadoc générée globalement.