TD N°9– M2103

Semaine 14

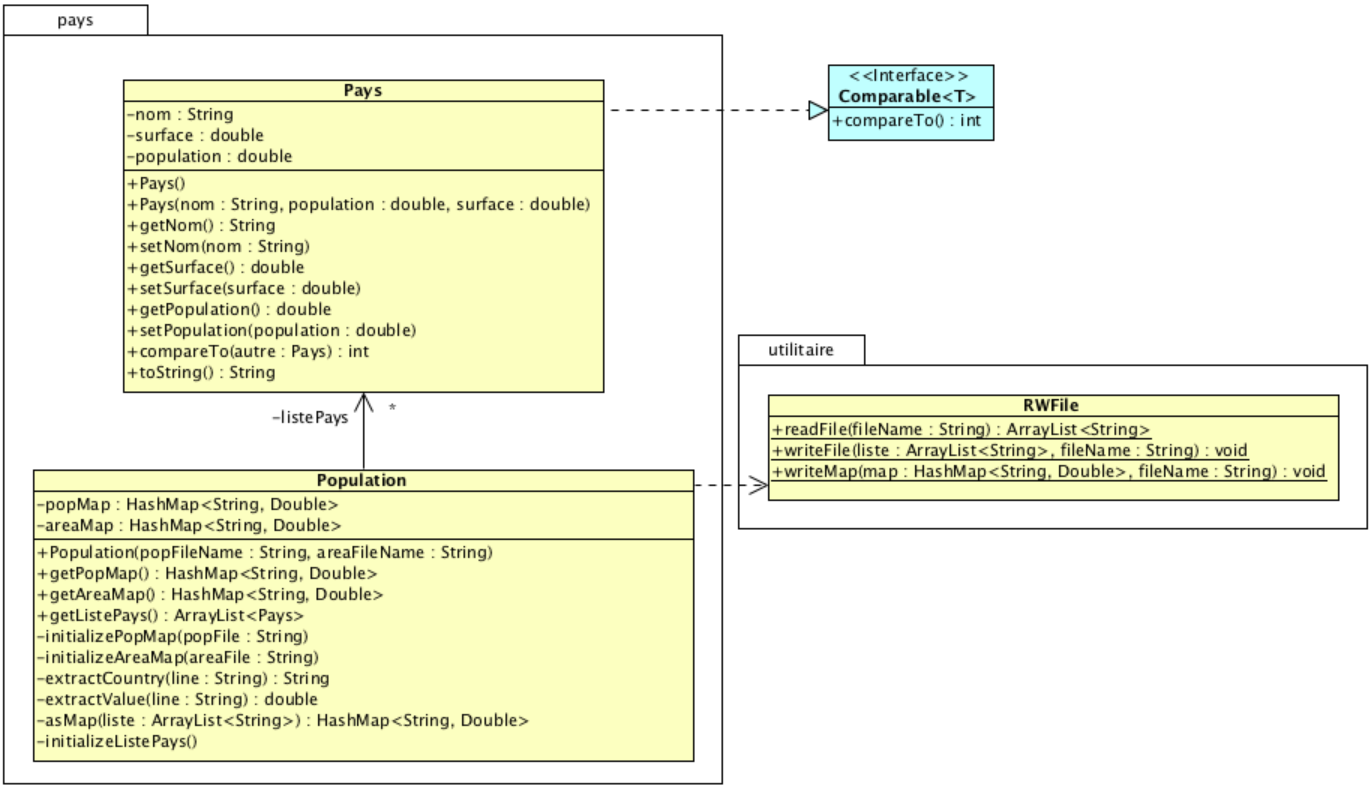
Objectifs du TD-TP

Manipuler des collections (ArrayList et HashMap)
Manipuler la lecture et l'écriture de fichiers de texte

Rappel de cours : Cours n°8
Les structures de données de type HashMap

1- La classe Population

Au TD/TP précédent nous avons défini et utilisé la classe Pays. Elle va être utilisée ici pour représenter des pays dont les données sont lues dans des fichiers de texte qui vous sont fournis.



La classe Population est une classe qui va réaliser des manipulations sur des données lues dans des fichiers de texte et permettre de créer une liste d'objets de la classe Pays, représentée par un ArrayList. Elle possède 3 attributs :

- **listePays** : une collection d'objets de type Pays représentée par un ArrayList<Pays>
- **popMap** : une collection de couple de données (nom_pays, nb_habitants) représentée par un HashMap<String, Double> dont les clés sont les noms des pays et les objets associés les nombres d'habitants.
- **areaMap** : une collection de couple de données (nom_pays, surface) représentée par un

HashMap<String, Double> dont les clés sont les noms des pays et les objets associés les surfaces en km2.

2 – Organisation complète de l'application

On remarque qu'il y a deux packages :

- le package **`pays`** qui contient les classes **`Pays`** et **`Population`**
- le package **`utilitaire`** qui contient la classe **`RWFile`** chargée de la lecture des fichiers.

Dans votre espace de travail il faudra créér les répertoires sources correspondants aux packages :

- **pays/**
- **utilitaire/**
- **data/** qui contiendra les fichiers **`worldarea.txt`** et **`worldpop.txt`** qui doivent être placés dans un répertoire nommé **`data`** sinon la classe **`RWFile`** ne pourra pas les lire.

3 – Utilisation de fichiers de données

Nous allons utiliser deux fichiers de données : un 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.

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)

```
pays1 info1
pays2 info2
....
paysn infon
```

`infoi` étant respectivement le nombre d'habitant et la superficie.

Chaque ligne des fichiers contient donc 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
Anguilla 102
Antigua and Barbuda 443
Argentina 2766890
Armenia 29800

3.1 Extraction des données dans une chaîne de caractères

Les fichiers de données vont être lus ligne par ligne et notre objectif est d'obtenir un ArrayList dont les éléments sont des chaînes de caractères du type :

"Afghanistan 647500"
"Akrotiri 123"
"Albania 28748"
"Algeria 2381740 "

Exercice 1

On a besoin de pouvoir séparer le nom du pays de la donnée associée.

Ecrire une méthode privée (on l'associera à la classe Population plus tard) :

`private String extractCountry(String line)` : qui prend une chaîne de caractères en paramètre et retourne le nom du pays extrait de la chaîne.

Pour cela :

- Il faut analyser la chaîne caractère par caractère pour trouver l'indice dans la chaîne où commence les caractères qui sont des nombres.
- Ensuite, on extrait la sous-chaîne qui correspond au nom du pays
- enfin il faut supprimer les espaces superflus.

Les méthodes nécessaires de la classe **`String`** :

• **`public char charAt(int index)`** : Returns the char value at the specified index

• **`public String trim()`** : Returns a copy of the string, with leading and trailing whitespace omitted.

• **`public String substring(int beginIndex)`** : Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.es par au moins un espace.

Pour tester si un caractère est un nombre, on va utiliser une méthode de la classe wrapper **`Character :`**

• **`public static boolean isDigit(char ch)`** : Determines if the specified character is a digit.

Exercice 2

2) Ecrire une méthode privée (on l'associera à la classe Population plus tard) :

`private double extractValue(String line)` : qui prend une chaîne de caractères en paramètre et qui retourne le nombre contenu dans la deuxième partie de la chaîne.

• Il faudra donc repérer comme pour la méthode précédente l'indice du premier caractère numérique.

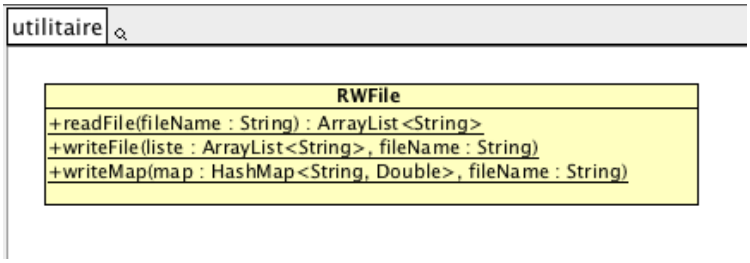
• Puis, extraire la chaîne de cet indice jusqu'à la fin de la chaîne.

• Supprimer les espaces pouvant subsister

• Enfin transformer le résultat en double avec la méthode classe **`Double.parseDouble(String s)`**

3.2 Lecture des données dans les fichiers

La classe qui permettra de lire les données dans les fichiers sera à développer. Elle sera à mettre dans un répertoire nommé « utilitaire » et à compiler.



Dans l'application que nous réaliserons en TP, nous aurons besoin de lire les fichiers de données en utilisant cette classe RWFile.

Par exemple pour initialiser l'Arraylist popArray avec le contenu du fichier :

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

Donc après popArray contient des éléments du type :

"Yemen 527970 "

"Zambia 752614 "

"Zimbabwe 390580"

3.3 Utilisation des structures de données type HashMap (voir cours 8)

Nous avons deux fichiers contenant des données différentes pour une même liste de pays.

Le plus simple est d'utiliser une structure de données permettant d'associer une clé à une valeur, c'est ce que permet de faire la collection **`java.util.HashMap`** de Java.

Un hashMap se déclare avec 2 types HashMap<K,V> : K le type des clés et V le type des valeurs associées.

Exemple :

```
private HashMap<String, Double> popMap; // Clé de type String et valeur de type Double
```

(attention pas de primitif, seuls les objets peuvent être mis dans des collections)

```
popMap = new HashMap<String, Double>();
popMap.put("France", new Double(55000000)) ;
popMap.put("Belgium", new Double("45000000") ) ;
```

3.4 Transformation d'un ArrayList en HashMap

Exercice 3

Dans le paragraphe 3.2 on voit comment remplir l'ArrayList **`popArray`** à partir d'un fichier de données.

Maintenant ont doit initialiser les HashMap à partir de l'ArrayList contenant les données. Pour cela on besoin de convertir l'ArrayList créé par la lecture de fichier en un HashMap :

```
private HashMap<String,Double> asMap(ArrayList<String> liste)
```

Prend en paramètre un ArrayList contenant des objets String de la forme "pays, donnée " et retourne en résultat un HashMap dont les éléments auront comme clé le nom du Pays et la donnée en valeur.

– Cette méthode sera utilisée dans les méthodes d'initialisation de **`popMap`** et **`popArea`**.

TP N°9– M2103

Semaine 14

1- Configuration des répertoires

Les fichiers que vous avez récupérés sur Moodle doivent être placés dans des répertoires :

- Les fichiers **`wordlarea.txt`** et **`worldpop.txt`** doivent être placés dans un répertoire nommé **`data`**
- La classe **`RWFile.java`** se trouvera dans le package **`utilitaire`**
- Les autres classes du TP : **`Pays`**, **`Population`**, **`TestPopulation`** se trouveront dans le package **`pays`**

2- La classe Population

(voir le diagramme de classes dans le sujet de TD)

La classe Population est une classe qui va réaliser des manipulations sur des données lues dans des fichiers de texte et permettre de créer un ArrayList de Pays. Elle possède 3 attributs (voir explication dans TD).

Le constructeur de la classe Population prend en paramètre les noms des fichiers de données et appelle successivement les méthodes qui font les initialisations.

La classe possède un certain nombre de méthodes privées qui réalisent les initialisations des différentes collections et qui sont appelées dans le constructeur :

- **`initialisePopMap(String popFile)`** : appelle la méthode de RWFile qui permet de lire un fichier (le fichier à lire ici est worldpop.txt) et le récupère dans un ArrayList ; ensuite utilise la méthode **`asMap`** pour transformer l'ArrayList en un HashMap.
- **`initialiseAreaMap(String areaFile)`** : réalise la même chose que la méthode précédente avec le fichier worldarea.txt
- **`initialiseListePays()`** : 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, et 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.

La classe possède également des méthodes privées pour réaliser l'extraction des données (méthodes **`extractCountry`** et **`extractValue`** vues en TD) qui sont utilisées pour la méthode de conversion des ArrayList en HashMap :

- **`private HashMap<String,Double> asMap(ArrayList<String> liste)`** : parcoure la liste passée en paramètre dont les éléments sont de la forme "pays info" et retourne un HashMap dont la clé est le nom du pays, extrait avec **`extractCountry,`** et l'objet qui est un Double contenant info qui est extrait avec **`extractValue`**.
-

Travail à Faire :

1. Récupérer les classes du précédent TP et ajouter les packages (penser à recompiler) :
   - Mettre la classe Pays dans le package « pays ».
   - Mettre dans un package **`tri`** l'interface **`ITri`** et la classe **`TriParSelection`** que vous avez déjà définies.
2. Ecrire le code de la classe RWFile : on écrire les deux méthodes statiques de noms readFile et write File.
3. Ecrire le code Java de la classe **`Population.`**
4. Réaliser une classe scénario TestPopulation
   - qui permettra de vérifier que la classe Population fonctionne et
   - qui ensuite récupèrera la collection listePays avec l'accesseur public prévu et utilisera le tri par sélection pour trier la liste.

Si l'on souhaite mesurer le temps pris par le tri on peut encadrer l'appel du tri par des appels à **`System.currentTimeMillis()`** ou par **`System.nanoTime()`** comme indiquée ci-dessous.

On peut aussi utiliser j**`ava.util.Arrays`** pour afficher simplement un tableau à l'écran.

```
TriParSelection triS = new TriParSelection(tabPays);

long startTime = System.currentTimeMillis();
triS.trier();
long estimatedTime = System.currentTimeMillis()- startTime;
System.out.println("temps du tri ="+ estimatedTime);
System.out.println(Arrays.toString(tabPays));
```