TD/TP N°8– R2.01

Objectifs :

Savoir utiliser des interfaces
Savoir définir des interfaces

Rappels de cours : Les interfaces (cours n°8)

- Une interface Java est une spécification d'un type, sous la forme d'un nom de type et d'un ensemble de signatures de méthodes
- Une interface spécifie des comportements qu'une classe implémentera.

L'interface Comparable

De nombreuses classes des bibliothèques java utilisent l'interface Comparable qui se trouve dans java.lang. Il n'y a donc pas d'import à faire pour l'utiliser.
L'interface est paramétrée :

```
public interface Comparable <T>
où T est le type des objets avec lesquels cet objet peut être comparé
```

Cette interface impose un ordre total des objets de chaque classe qui l'implémente. Cet ordre est l'ordre naturel de la classe.
Comparable ne possède qu'une méthode :

```
public int compareTo(T o)
```

compare cet objet avec l'objet spécifié en paramètre.
Retourne:

- un entier négatif si cet objet est plus petit que l'objet spécifié
- zéro si cet objet est égal à l'objet spécifié
- un entier positif si cet objet est plus grand que l'objet spécifié

lance : `NullPointerException` si l'objet spécifié est null

Attention le type paramétré doit être compatible avec le type de l'objet qui implémente l'interface.

1 - La classe Pays

Nous allons utiliser une classe qui va nous servir pour plusieurs TP : la classe **`Pays`** qui et dans un package **`pays`**.

Cette classe définit des données pour un pays, en particulier sa surface et sa population.

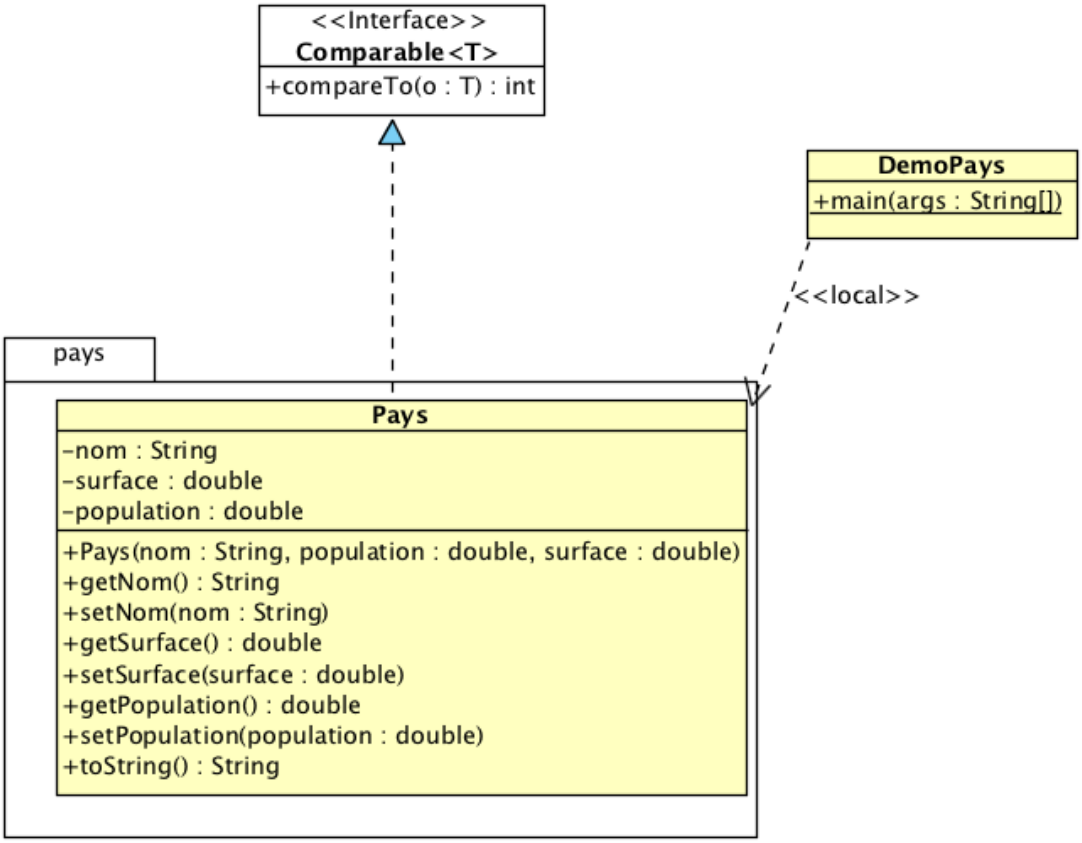
Note premier objectif est de pouvoir comparer deux pays entre eux. Pour cela nous allons utiliser l'interface Java ***Comparable***.

Le diagramme UML ci-dessous spécifie que la classe Pays va implémenter l'interface.

On remarque que pour indiquer qu'il s'agit d'une interface en UML on utilise le stéréotype <<Interface>>.

De plus, en UML les méthodes spécifiées dans l'interface n'apparaissent pas dans la boîte de la classe qui implémente l'interface. C'est pourquoi la méthode compareTo n'apparaît pas dans la classe Pays. Le lien d'implémentation est très fort.

Enfin pour utiliser `Comparable` vous devrez préciser le type T des éléments à comparer.
Comme la comparaison concerne un pays avec un autre le type T sera `Pays`.
L'entête de la classe Pays sera donc : **`public class Pays implements Comparable<Pays>`**



Question 1.1 :
Dans un premier temps on réalisera la comparaison de deux pays par rapport à leur surface.
Donner le code Java de la classe Pays ainsi que la javadoc générée. Ne pas oublier le package.

Question 1.2 :
Ecrire la classe scénario `DemoPays` qui permet de créer deux Pays et de les comparer.

2 - Définition d'une interface simple

2.1 Définition d'une interface de tri

Comme il existe beaucoup d'algorithme de tri, nous avons décidé de créer une nouvelle interface **`ITri`** qui spécifie la méthode **`trier()`** que l'on veut retrouver dans tous les algorithmes de tri.

L'interface **`Itri` est dans le package `tri`.**

- Écrire et compiler le code Java de l'interface **`ITri`** représentée dans le diagramme de classes ci-dessous.

2.2 Implémentation de tris simples en Java

L'objectif maintenant est de pouvoir trier un ensemble de Pays avec un tri simple, c'est-à-dire un tri par sélection ou un tri par insertion.

Rappel du principe du tri par sélection :

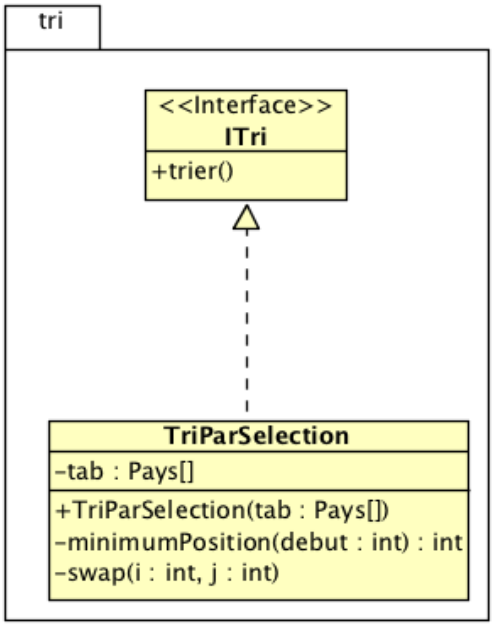
Sur un tableau de n éléments (numérotés de 1 à n), le principe du tri par sélection est le suivant :

- rechercher le plus petit élément du tableau, et l'échanger avec l'élément d'indice 1 ;
- le second plus petit élément du tableau, et l'échanger avec l'élément d'indice 2 ;
- continuer de cette façon jusqu'à ce que le tableau soit entièrement trié.

Bien sûr si on veut utiliser un tableau Java, les indices iront de 0 à n-1 pour trier n éléments.

Le tri par sélection en Java

Nous avons également décidé d'appliquer l'algorithme du tri par sélection pour trier un tableau de pays. Le diagramme présentant la spécification de notre classe **`TriParSelection`** est le suivant :



Le constructeur reçoit le tableau à trier et le met dans l'attribut **`tab`** de la classe **`TriParSelection`**. L'algorithme de tri est réparti dans plusieurs méthodes pour décomposer le problème à résoudre :

- La méthode privée **`swap(int i, int j)`** échange deux entrées du tableau d'indice i et j.
- La méthode privée **`int minimumPosition(int debut)`** retourne la position du plus petit élément dans tab[debut] jusqu'à tab[tab.length-1].
- La méthode **`trier()`** implémente la boucle principale de l'algorithme.

La classe **`TriPar Selection`** sont dans le package **`tri`**.

Travail à réaliser :

- Ecrire et compiler la classe **TriParSelection** qui implémente l'interface **ITri** .
- Tester le tri simple des pays dans une classe **`DemoTri`**en dehors du package en utilisant le jeu de données donné ci-dessous.

Pour créer un tableau de pays, vous pouvez copier/coller ce qui suit :

```
Pays[] tabPays = new Pays[10];
    tabPays [0] = new Pays ("Cuba" , 11423952, 110860);
    tabPays [1] = new Pays ("Chile" , 16454143, 756950);
    tabPays [2] = new Pays ("Russia" , 140702094, 17075200);
    tabPays [3] = new Pays ("Norway" , 4644457, 323802);
    tabPays [4] = new Pays ("Nigeria" , 138283240, 923768);
    tabPays [5] = new Pays ("Paraguay" , 6831306, 406750);
    tabPays [6] = new Pays ("Oman" , 3311640, 212460);
    tabPays [7] = new Pays ("Yemen" , 23013376, 406750);
    tabPays [8] = new Pays ("Togo" , 5858673, 56785);
    tabPays [9] = new Pays ("France" , 64057790 , 643427);
```

Travail à rendre :

Déposez sur moodle le code source complet et **organisé** (respectez les répertoires de packages pour le code source) de toutes les classes définies et de l'interface `ITri`, ainsi que la javadoc générée.

Astuce :

Il existe une classe `java.util.Arrays` qui permet d'imprimer un tableau passé en paramètre (regarder l'API Java) .

Il faut penser à importer cette classe et utiliser son message toString().
Exemple :

```
System.out.println (Arrays.toString(tabPays)) ;
```