TP5 : Itération, interface, héritage.

POO : Programmation Orientée Objet

Première année

L'objectif de ce TP est de manipuler des *itérateurs*. Vous utiliserez dans un premier temps l'interface `Iterator` de Java, puis vous réaliserez votre itérateur et ses différentes implémentations pour effectuer différents parcours d'un tableau d'entiers à deux dimensions.

Travail à réaliser

Éléments fournis Vous trouverez dans le répertoire **/home/depot/1A/POO/TP5/** les classes à compléter. Vous devez donc au préalable les recopier dans votre répertoire de travail.

▷ **Question 1:** L'interface `java.util.Iterator` est de la forme suivante :

`boolean`	`hasNext()`	retourne `true` si il reste des éléments à parcourir.
`Object`	`next()`	retourne le prochain élément du parcours.
`void`	`remove()`	supprimer de la collection le dernier élément parcouru (*méthode optionnelle*).

Complétez la classe `tp5.TestIterator` qui est fournie. L'objectif de cette classe est d'effectuer un parcours d'une collection Java de type `ArrayList`.

Vous commencerez par étudier la classe fournie, puis consulterez la documentation de l'interface `java.util.Iterator` et `java.lang.ArrayList` de l'API [1] `http://java.sun.com/javase/6/docs/api/`. Seulement, ensuite, vous pourrez compléter la classe.

▷ **Question 2:** Pour des raisons de simplicité dans la suite du TP nous ne souhaitons pas réaliser la méthode `remove()` et ne souhaitons pas non plus gérer les exceptions. De plus, nous nous restreignons à parcourir uniquement des tableaux de valeurs entières (de type primitif `int`). Nous allons donc écrire notre propre interface `Iterateur` comportant les deux méthodes suivantes :

`boolean hasNext()`	retourne `true` si il reste des éléments à parcourir.
`int next()`	retourne le prochain élément du parcours.

Écrivez l'interface `tp5.Iterateur`.

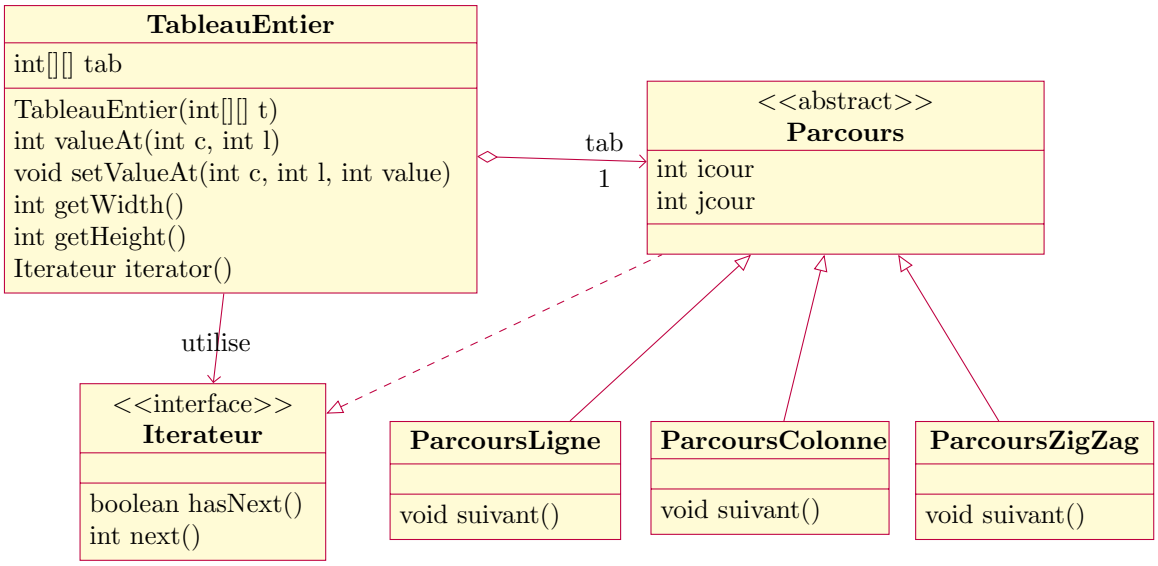
Dans la suite, nous allons écrire plusieurs algorithmes permettant de parcourir de différentes manière un tableau de valeurs entières.

▷ **Question 3:** Écrivez une classe `tp5.TableauEntier` encapsulant un tableau à deux dimensions de valeurs entières dont l'interface est la suite :

`TableauEntier(int n, int m)`	crée un nouveau tableau de dimension $n \times m$.
`TableauEntier(int[][] t)`	stocke le tableau passé en paramètre.
`int valueAt(int c, int l)`	retourne un élément du tableau.
`void setValueAt(int c, int l, int value)`	modifie la valeur d'une élément du tableau.
`int getWidth()`	retourne la largeur du tableau.
`int getHeight()`	retourne la hauteur du tableau.

▷ **Question 4:** Pour ceux qui ont suivi le module de "Préparation informatique", vous avez implementé différents parcours de tableaux à deux dimensions : parcours par ligne, par colonne, en diagonale, en zigzag, en diagonale avec zigzag. Nous souhaitons maintenant implanter ces parcours en utilisant les interfaces et l'héritage.

1. L'interface de programmation (*Application Programming Interface*) fournie la documentation sur les différences classes de la librairie standard Java.



(a) Ajouter à la classe `tp5.TableauEntier` une méthode `iterator()` qui retournera un itérateur sur le tableau.

(b) Écrivez une classe abstraite `tp5.Parcours` implantant votre interface `tp5.Iterateur`. Cette classe contiendra comme attributs les indices nécessaires au parcours. En reposant sur une méthode abstraite `suivant()`, elle réalisera les méthodes `next()` et `hasNext()` de l'itérateur.

(c) Écrivez une classe `tp5.ParcoursLigne` héritant de la classe `tp5.Parcours` et permettant de faire un parcours en ligne.

(d) Écrivez une classe `tp5.TestParcoursLigne` afin de tester le parcours en ligne d'un tableau. L'automatisation de cette classe de test (le test compare le résultat obtenu au résultat attendu) serait un plus.

(e) Implementez les autres parcours de façon analogue.

(f) Ajouter à la classe `tp5.TableauEntier` une méthode `configureIterator(int type)` permettant de configurer le type de l'itérateur qui sera retourné lors de l'appel à la méthode `iterator()`.

Rappels sur les différents parcours de tableau à deux dimensions

Parcours en ligne

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Parcours en colonne

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Parcours en zigzag

1	2	3	4
8	7	6	5
9	10	11	12
16	15	14	13

Parcours en diagonale

7	4	2	1
11	8	5	3
14	12	9	6
16	15	13	10

Parcours en diagonale zigzag

10	4	3	1
11	9	5	2
15	12	8	6
16	14	13	7