

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

▷ **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).
```

Compléter 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.util.ArrayList` de l'API¹ <http://docs.oracle.com/javase/7/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`.

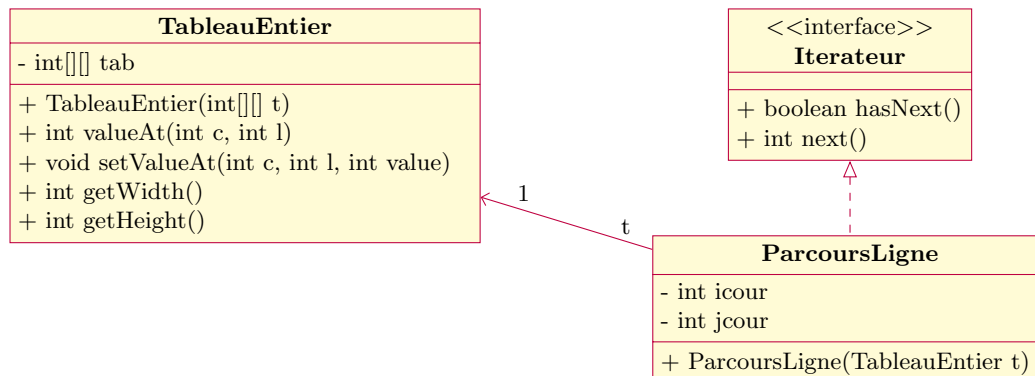
Dans la suite, nous allons écrire plusieurs algorithmes permettant de parcourir de différentes manières un tableau de valeurs entières.

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

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

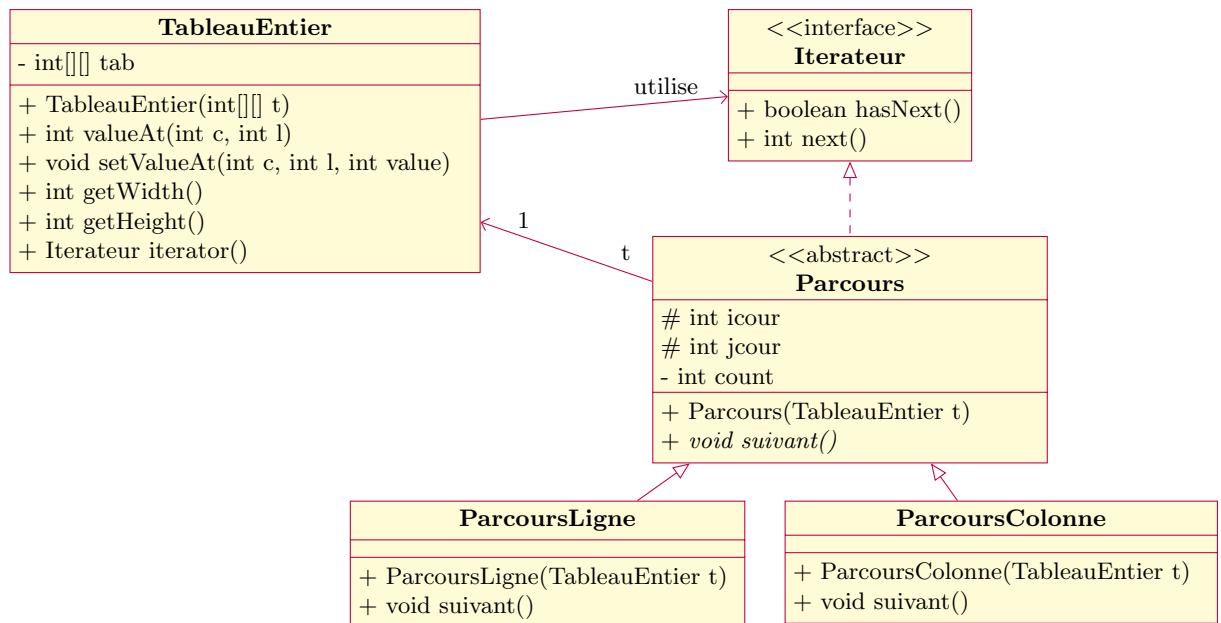
▷ **Question 4.** Nous souhaitons implémenter différents parcours de tableaux à deux dimensions : parcours par ligne, par colonne, en diagonale, en zigzag, en diagonale avec zigzag. Nous souhaitons abstraire l'utilisation de ces parcours en utilisant des interfaces et l'héritage.

(a) Écrire une classe `tp5.ParcoursLigne` permettant de faire un parcours en ligne sur un objet de type `TableauEntier`. Cette classe réalisera l'interface `tp5.Iterateur`.



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

- (b) Écrire une classe `tp5.TestParcoursLigne` afin de tester le parcours en ligne de divers tableaux. L'automatisation de cette classe de test (le test compare le résultat obtenu au résultat attendu) serait un plus.
- (c) Ajouter à la classe `tp5.TableauEntier` une méthode `iterator()` qui retournera un itérateur (pour l'instant un objet de type `tp5.ParcoursLigne`) sur le tableau.
- (d) Implémenter le parcours en colonne de façon analogue (ajout d'une classe `tp5.ParcoursColonne`, ajout d'une classe de tests `tp5.TestParcoursColonne`).
- (e) Écrire une classe abstraite `tp5.Parcours` réalisant l'interface `tp5.Itérateur`. Cette classe contiendra comme attributs les indices nécessaires au parcours. En reposant sur une méthode abstraite `suiivant()`, elle réalisera les méthodes `next()` et `hasNext()` de l'itérateur.
- (f) Modifier vos classes `tp5.ParcoursLigne` et `tp5.ParcoursColonne` afin que celles-ci héritent de la classe abstraite `tp5.Parcours`. Supprimez les valeurs et traitements qui sont factorisés dans la classe `tp5.Parcours`.



- (g) 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()`.
- (h) Implémenter les autres parcours (en zigzag, en diagonale, en diagonale zigzag, etc.) de façon analogue.
- (i) Modifiez la classe `tp5.TableauEntier` afin que celle-ci soit maintenant une classe générique modélisant un tableau à deux dimensions. Cette classe sera paramétrée par le type des objets stockés dans le tableau.
- (j) Modifier les classes de parcours et l'interface `Itérateur` selon le même principe.

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

```