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

▷ **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'API[1] `http://docs.oracle.com/javase/7/docs/api/`. Seulement, ensuite, vous pourrez compléter la classe.

Réponse

```
//
//  Classe effectuant le parcours d'une collection en utilisant un iterator
//
package tp5;

import java.util.ArrayList ;
import java.util.Iterator ;

public class TestIterator {

    public static void main (String[] args) {
        if (args.length == 0) {
            System.out.println("Usage: donner un entier (la taille du tableau)");
            System.exit(1);
        }

        int taille = Integer.parseInt(args[0]) ;

        // Creation et instanciation d'un ArrayList d'Integer
        ArrayList l  = new ArrayList();
        for (int i=0; i< taille; i++)
            l.add(new Integer(i));

       // Completer avec les instruction suivantes :
       // 1) Creer un objet Iterator avec la methode iterator()

       /* BEGINKILL */
       Iterator iter = l.iterator();
       /* ENDKILL */

       // 2) Parcourir la  structure en utilisant les deux methodes
       //    hasNext() et next() de la classe Iterator
       //    On fera une boucle for ou une boucle while

       /* BEGINKILL */
        while (iter.hasNext()) {
            Integer v = (Integer) iter.next();
            System.out.println(v);
        } // while
       /* ENDKILL */

    } // main

} //class

```

Fin réponse

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

▷ **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`.

Réponse

```
1  package tp5;
2
3  public interface Iterateur {
4
5      /* BEGINKILL */
6
7      // retourn true si il reste des elements a parcourir
8      public boolean hasNext();
9
10     // retourne le prochain element du parcours
11     public int next();
12
13     /* ENDKILL */
14 }
```

Fin réponse

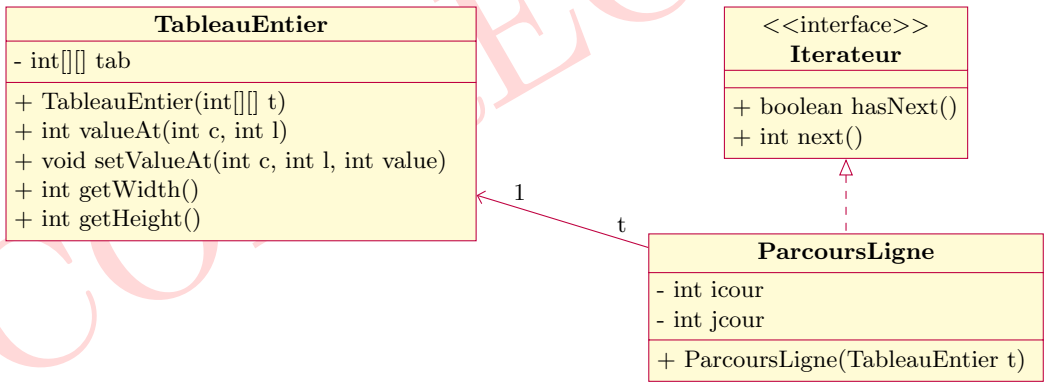
Dans la suite, nous allons écrire plusieurs algorithmes permettant de parcourir de différentes manières un tableau de valeurs entières.

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

`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.** 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`.



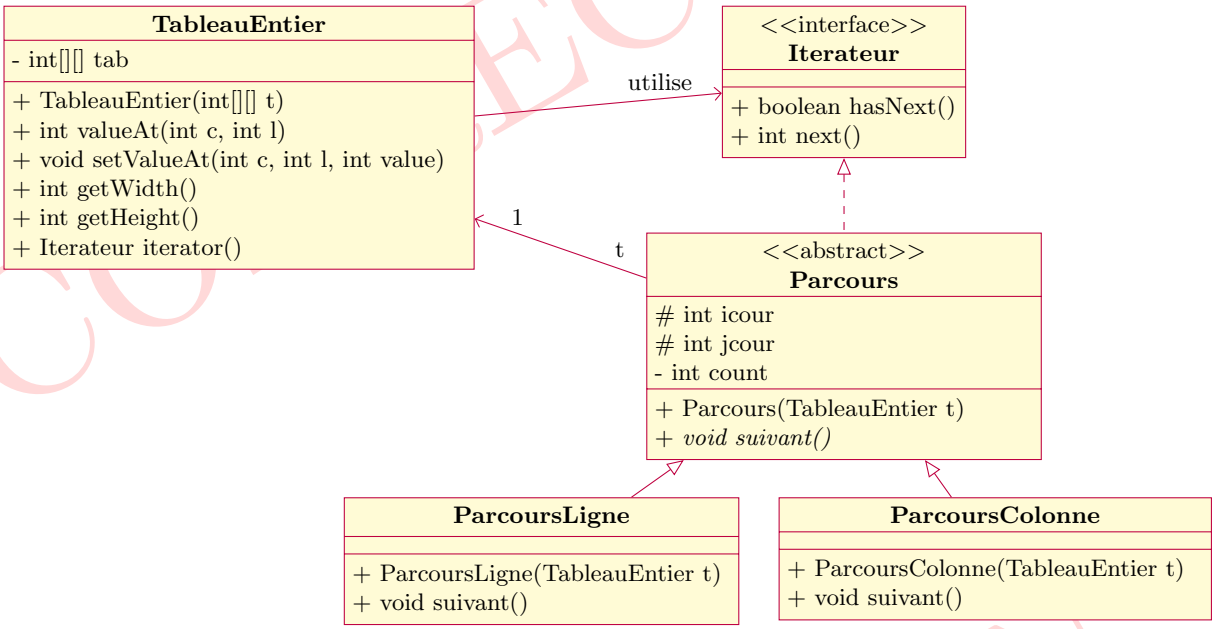
(b) Écrire une classe `tp5.TestParcoursLigne` afin de tester le parcours en ligne de divers tableaux. L'automatisation de cette classe de test (le test compare le résultat obtenu au résultat attendu) serait un plus.

(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.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.

(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 `Iterateur` 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

Réponse

```
1 package tp5;
2
```

```java
public class TableauEntier {

    /* BEGINKILL */
    private int[][] tab;

    private int iteratorType = TableauEntier.ITERATEUR_LIGNE;

    public static final int ITERATEUR_LIGNE = 0;
    public static final int ITERATEUR_COLONNE = 1;
    public static final int ITERATEUR_ZIGZAG = 2;

    public TableauEntier(int n, int m) {
        this.tab = new int[n][m];
    }

    public TableauEntier(int[][] t) {
        this.tab = t;
    }

    public int valueAt(int c, int l) {
        return this.tab[c][l];
    }

    public void setValueAt(int c, int l, int value) {
        this.tab[c][l] = value;
    }

    public int getWidth() {
        return this.tab.length;
    }

    public int getHeight() {
        return this.tab[0].length;
    }

    public void configureIterator(int type) {
        this.iteratorType = type;
    }


    public Iterateur iterator() {
        Iterateur it = null;

        switch (this.iteratorType) {
        case ITERATEUR_COLONNE:
            it = new ParcoursColonne(this.tab);
            break;
        case ITERATEUR_ZIGZAG:
            it = new ParcoursZigZag(this.tab);
            break;
        case ITERATEUR_LIGNE:
        default:
            it = new ParcoursLigne(this.tab);
            break;
        }

        return it;
    }

    /* ENDKILL */
}
```

```java
package tp5;

// Classe abstraite implantant l'interface Iterateur

public abstract class Parcours implements Iterateur {
    /* BEGINKILL */
    protected TableauEntier tab; // tableau des valeurs
    protected int icour, jcour; // indices courants de ligne et de colonne
    private int compteur = 0; // nb de valeurs parcourues

    public Parcours(int[][] t) {
        tab = new TableauEntier(t);
        icour = 0;
        jcour = 0;
    }

    public boolean hasNext() {
        return this.compteur < this.tab.getWidth()*this.tab.getHeight();
    }

    public int next() {
        int v = this.tab.valueAt(icour,jcour);
        suivant();
```

```
        compteur++;
        return v;
    }

    protected abstract void suivant() ;
    /* ENDKILL */
}
```

```
package tp5;

// Classe implantant le parcours en ligne

public class ParcoursLigne extends Parcours {
    /* BEGINKILL */
    public ParcoursLigne(int[][] t) {
        super(t);
    }

    protected void suivant() {
        if (icour < tab.getWidth()-1) {
            icour++;
        } else {
            icour = 0;
            if (jcour < tab.getHeight()-1) {
                jcour++;
            } else {
                jcour = 0;
            }
        }
    }
    /* ENDKILL */
}
```

```
package tp5;

// Classe implantant le parcours en colonne

public class ParcoursColonne extends Parcours {
    /* BEGINKILL */
    public ParcoursColonne(int[][] t) {
        super(t);
    }

    protected void suivant() {
        if (jcour < tab.getHeight()-1) {
            jcour++;
        } else {
            jcour = 0;
            if (icour < tab.getWidth()-1) {
                icour++;
            } else {
                icour = 0;
            }
        }
    }
    /* ENDKILL */
}
```

```
package tp5;

// Classe implantant le parcours en zigzag

public class ParcoursZigZag extends Parcours {
    /* BEGINKILL */
    public ParcoursZigZag(int[][] t) {
        super(t);
    }

    protected void suivant() {
        if (jcour % 2 == 0) {
            if (icour < tab.getWidth()-1) {
                icour++;
            } else if (jcour < tab.getHeight()-1) {
                jcour++;
            }
        } else {
            if (0 < icour) {
                icour--;
            } else if (jcour < tab.getHeight()-1) {
                jcour++;
            }
        }
```

```
    }
    /* ENDKILL */
}
```

```
package tp5;

// Classe de test des differents implantations de parcours

public class TestParcours {

    public static void main(String[] args) {
        /* Cree un nouveau tableau 2D */
        int nl=4;
        int nc=5;
        TableauEntier tab = new TableauEntier(nc,nl);

        for (int i = 0; i<nc;i++){
           for (int j = 0; j <nl; j++)
               tab.setValueAt(i,j, (nc*j+i));
        }

        /* configure un parcours en ligne et appel testParcours() qui cree un iterateur */
        /* BEGINKILL */
        tab.configureIterator(TableauEntier.ITERATEUR_LIGNE);
        testParcours("Parcours en ligne", tab);
        /* ENDKILL */

        /* configure un parcours en colonne et appel testParcours() qui cree un iterateur */
        /* BEGINKILL */
        tab.configureIterator(TableauEntier.ITERATEUR_COLONNE);
        testParcours("Parcours en colonne", tab);
        /* ENDKILL */

        /* configure un parcours en zigzag et appel testParcours() qui cree un iterateur */
        /* BEGINKILL */
        tab.configureIterator(TableauEntier.ITERATEUR_ZIGZAG);
        testParcours("Parcours en zigzag", tab);
        /* ENDKILL */
    }

    public static void testParcours(String titre, TableauEntier tab) {
        Iterateur it = tab.iterator();
        System.out.print(titre+":\t");
        while (it.hasNext()) {
            int v = it.next();
            System.out.print(v+" ");
        }
        System.out.println();
    }
}
```

Fin réponse