TD 11

UFR Math-Info
UE – L2/S4 – Programmation 4

L'interface *Iterator* de Java

Java propose une API appelée ***Iterator*** dont la structure est présentée ci-dessous :

```
public interface Iterator {
/**
* Returns < tt >true </ tt > if the iteration has more elements.
*
* @return < tt >true </ tt > if the iterator has more elements.
*/
boolean hasNext ();

/**
* Returns the next element in the interation .
*
* @returns the next element in the interation .
* @exception NoSuchElementException iteration has no more elements.
*/
Object next ();

/**  Removes from the underlying collection the last element returned by the iterator
(optional operation).
*/
void remove();
}
```

Dans l'API , il existe les classes Vector et LinkedList qui sont toutes les deux des listes mais dont les structures sont différentes : la première stocke ses éléments dans un tableau et la seconde dans une liste chaînées. Leur comportement étant le même, elles *implémentent* toutes les deux l'interface List et possèdent toutes les deux la méthode Iterator iterator() qui retourne un objet de type Iterator, c'est-à-dire un objet qui permet de parcourir la liste avec les méthodes de l'interface Iterator.

Question 1.
Ecrire le morceau de code qui permet de parcourir un vecteur à l'aide d'un objet de type Iterator.
Quel est d'après votre analyse, l'intérêt de cette approche ?

[**Solution**

```
public void parcours(Vector v){
            Iterator iterator = v.iterator();
            while(iterator.hasNext()){
                  Object valeur = iterator.next();
                  System.out.println("Valeur : " + valeur);
            }
      }
```

L'intérêt est donc de parcourir un ensemble de structures dynamiques en utilisant les mêmes méthodes
]

Question 2

Ecrire la classe *TableauIterator* qui implémente l'interface Iterator pour un tableau, on n'implémentera uniquement les méthodes hasNext() et next().

[**Solution**

```
import java.util.Iterator;
public class TableauIterator implements Iterator{
            private int indice ; // l'indice courant
            private Object [] tableau ; // le tableau parcouru

            //Constructeur
            public TableauIterator (Object [] tableau) {
                  this.tableau=tableau;
                  indice = 0;
            }

            //méthodes
            public boolean hasNext () {
                  return indice < tableau . length ;
            }

            public Object next (){
                  return tableau [indice++];
            }
            //Cette méthode n'est pas implémentée
            public void remove(){}

            //Test
            public static void main(String[] args) {
                  String[] tab =
{"Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"};
                  TableauIterator ti = new TableauIterator(tab);
                  while(ti.hasNext()){
                        System.out.println("Valeur : "+ ti.next());
                  }
            }

}
```

]