Les itérateurs en Java

Marc Champesme
mailto:Marc.Champesme@lipn.univ-paris13.fr

13 janvier 2021

À quoi servent les itérateurs ?

En Java :

```
// for amélioré (foreach)
List<String> uneListe = ...
for (String s : uneListe) {
    System.out.println(s);
}
```

est équivalent à :

```
List<String> uneListe = ...
Iterator<String> iter = uneListe.iterator();
while (iter.hasNext()) {
    String s = iter.next();
    System.out.println(s);
}
```

Pas de foreach sans itérateur !

À quoi servent les itérateurs ?

Quelle différence avec :

```
List<String> uneListe = ...
for (int i = 0; i < uneListe.size(); i++) {
    System.out.println(uneListe.get(i));
}
```

???

- Selon que `uneListe` contient une instance de `LinkedList` ou de `ArrayList`, ça fait une grosse différence ! !
- Et si à la place de `List<String> uneListe = ...` on a `Set<String> unSet = ...` ???
- Indice : pas de `get(i)` dans l'interface `Collection` (ni dans `Set`)

Qu'est-ce qu'un itérateur ?

- Une abstraction pour itérer sur des objets
- Qui permet de masquer le moyen d'accéder à l'élement suivant
- En Java : instance d'une classe implémentant l'interface `Iterator<E>`

```
public interface Iterator<E> {
    boolean hasNext(); // Reste t'il un élément?
    E next(); // Renvoie l'élément suivant et avance le curseur
    void remove(); // Supprime le dernier élément
                   // renvoyé par next()
                   // (Opération optionelle)
}
```

- Une instance d'itérateur ne peut servir qu'un seule fois ! (pour une seule itération sur l'ensemble des éléments)

Et l'interface Iterable<E> ?

Pour toute classe implémentant Iterable<E> :

```
public interface Iterable<E> {
     Iterator<E> iterator();
}
```

on peut utiliser le for amélioré !
Exemple (si Piece implements Iterable<ObjetZork>) :

```
Piece p = ...
for (ObjetZork oz : p) {
    System.out.println(oz);
}
```

Comment définir la méthode iterator ?

Cas simple : Piece utilise une ArrayList (ou autre Iterable) pour mémoriser les ObjetZork...

```
public class Piece implements Iterable<ObjetZork> {
    private List<ObjetZork> mesObjets;
    ...
    public Iterator<ObjetZork> iterator() {
        return mesObjets.iterator(); // Facile!
     }
}
```

Plus compliqué si les objets ne sont pas mémorisés grâce à une classe Iterable...

Comment définir la méthode iterator(2) ?

- Si Piece utilise un tableau pour mémoriser les ObjetZork ?
- Les tableaux n'implémentent pas Iterable...
- ...mais on peut utiliser le for amélioré quand même

```
ObjetZork[] tab = ...
// Iterator<ObjetZork> iter = tab.iterator(); impossible!
for (ObjetZork oz : tab) {
    System.out.println(oz); // Ça marche!
}
```

est équivalent à :

```
for (int i = 0; i < tab.length; i++) {
    System.out.println(tab[i]);
}
```

Mais ça ne nous aide pas à définir iterator() pour la classe Piece

Comment définir la méthode iterator(3) ?

Alternative : définir une classe implémentant Iterator, par exemple TableauIterator

```
public class Piece implements Iterable<ObjetZork> {
    private ObjetZork[] mesObjets;
    private int nbObjets;
    ...
    public Iterator<ObjetZork> iterator() {
        return new TableauIterator(mesObjets, nbObjets);
     }
}
```

Ne reste plus qu'à définir la classe TableauIterator...

Implémenter l'interface Iterator

```
public class TableauIterator implements Iterator<ObjetZork> {
    private ObjetZork[] tab;
    private int nbElts;
    private int index;

    public TableauIterator(ObjetZork[] tab, int nbElts) {
        this.tab = tab;
        this.nbElts = nbElts;
    }

    public boolean hasNext() {
         return index < nbElts;
    }

    public ObjetZork next() {
        return tab[index++];
    }
}
```

Implémenter l'interface Iterator pour PureListString

```
public class PureListStringIterator
                          implements Iterator<String> {
    private PureListString current;

    public PureListStringIterator(PureListString liste) {
        current = liste;
    }

    public boolean hasNext() {
         return !current.isEmpty();
    }

    public String next() {
        String result = current.getFirst();
        current = current.removeFirst();
        return result;
    }
}
```

Un petit exercice pour pour terminer. . .

- Définir TableauIterator<E> implements Iterator<E>
- Définir PureListIterator<E> implements Iterator<E>
- Bon courage !