Master 1 TIIL

Algorithmique et Programmation Objet

Alain Plantec

TP 2

Vous utilisez Eclipse ou NetBeans IDE pour développer.
N'oubliez pas les tests ! http://gfx.developpez.com/tutoriel/java/junit/

Exercice

En cours, nous avons vu les tableaux à taille dynamique. Ce type de collection autorise l'ajout et le retrait d'éléments. De plus, la notion de collection vide est mise en œuvre : lorsqu'une telle collection est créée, elle ne contient aucun élément.
La mise en œuvre présentée pendant le cours peut être très inefficace si beaucoup d'éléments doivent être ajoutés. En effet, à chaque ajout, le tableau interne doit être ré-alloué et les références des éléments déjà contenus doivent être recopiées dans le nouveau tableau interne.
On propose d'étudier une mise en œuvre plus efficace de collection dynamique en prenant en compte la notion de capacité.

La capacité d'une collection dynamique est la taille du tableau interne. Cette capacité est différente de la taille de la collection. Par exemple, une collection peut être vide et avoir une capacité de 100. Cela signifie qu'on peut ajouter 100 éléments dans la collection sans qu'il soit nécessaire de ré-allouer le tableau interne. Le tableau interne est ainsi ré-alloué par bloc et non à chaque ajout.

En utilisant cette méthode, on demande de développer et de tester la classe TabListe qui met en œuvre l'interface Liste donnée ci-après. Vous pouvez vous reporter à la documentation d'Oracle pour comprendre précisément le sens des méthodes de l'interface Liste.
Voir http://docs.oracle.com/javase/6/docs/api/java/util/List.html.

```
Interface Liste<E> {
   int size();
   boolean isEmpty();
   boolean contains(Object e);
   Iterator<E> iterator();
   boolean add(E e);
   boolean remove(Object e);
   boolean containsAll(Collection<?> c);
   boolean addAll(Collection<? extends E> c);
   boolean addAll(int idx, Collection<? extends E> c);
   boolean removeAll(Collection<?> c);
   boolean retainAll(Collection<?> c);
   void clear();
   boolean equals(Object arg0);
   int hashCode();
   E get(int idx);
   E set(int idx, E e);
   void add(int idx, E e);
   E remove(int idx);
   int indexOf(Object e);
   int lastIndexOf(Object e);
   List<E> subList(int idx1, int idx2);
}
```