Licence Informatique **UE INF203**
Université Bordeaux 1 **Programmation par objets**
2006-2007 **Travaux Dirigés**
feuille 4

Télécharger les fichiers qui accompagnent le td à l'aide de la commande
`wget http://dept-info.labri.fr/ENSEIGNEMENT/poo/src/fichiers-4.tar.gz` puis décompacter cette archive au moyen de la commande `tar -xvzf fichiers-4.tar.gz`. Se placer dans le répertoire td04.

Liste chaînée

Le but de ce TD est d'utiliser et de compléter la classe Liste (liste d'entiers chaînée avec sentinelle) vue en cours.

1. Ecrire un programme de test de la classe `Liste`. Toutes les méthodes disponibles doivent être testées. Lors de l'affichage d'une liste, afficher également la valeur de la sentinelle. Cette valeur peut-elle varier au cours du temps ?
2. Ecrire un nouveau constructeur de la classe `Liste`. Ce constructeur prendra en paramètre un tableau d'entiers donnant les éléments de la liste.
3. Ecrire trois méthodes donnant respectivement le nombre d'occurences d'un entier dans la liste, l'indice de la première occurence d'un entier et l'indice de la dernière occurence d'un entier.
4. Ecrire la méthode `public boolean equals(Object o)` permettant de tester l'égalité de deux listes.
5. Ecrire une méthode donnant l'élément de la liste se trouvant à un indice donné.
6. Ecrire une méthode supprimant l'élément de la liste se trouvant à un indice donné.
7. Ecrire une méthode ajoutant un élément dans la liste à un indice donné.
8. Ecrire une méthode construisant une sous-liste de la liste étant donnés les indices de début et de fin.
9. Ecrire une méthode de concaténation qui ajoute à la liste courante la liste passée en paramètre.
10. Ecrire une fonction de concaténation qui construit la liste résultat de la concaténation des deux listes passées en paramètre.

Toutes les méthodes doivent être testées **au fur et à mesure**.

```
public class Liste{
    private int val;
    private Liste suiv;

    public Liste(){ // liste vide avec sentinelle
        val = 0;
        suiv = null;
    }

    public Liste(int a){
        Liste x = new Liste(a, null);
        val = 0;
        suiv = x;
    }

    private Liste (int a, Liste x){
        val = a;
        suiv = x;
    }

    public boolean estVide(){
        return suiv == null;
    }
```

```
    public int valeur() {
        if (estVide())
            throw new RuntimeException("Valeur pour une  liste vide");
        return suiv.val;
    }

    public Liste reste(){
        if (estVide())
           throw new RuntimeException("Reste pour une  liste vide");
        return suiv;
    }

    public void ajoutDebut(int a){
        Liste x = new Liste(a, suiv);
        this.suiv = x;
    }

    public void ajoutFin (int a){
        if (estVide())
            ajoutDebut(a);
        else
            reste().ajoutFin(a);
    }

    public void supprimer (int a){
        Liste x = this;
        boolean trouve = false;
        while (!trouve && x.suiv != null ){
            if (x.suiv.val == a) {
                trouve = true;
                x.suiv = x.suiv.suiv;
            }
            x = x.suiv;
        }
    }

    public int longueur(){
        return estVide() ? 0: 1 + reste().longueur();
    }

    public boolean appartient (int a){
        if (estVide())
            return false;
        else if (valeur() == a)
            return true;
        else
            return reste().appartient(a);
    }

    public String toString(){
        if (estVide())
            return "-> ";
        else
            return "-> " + valeur() + reste().toString();
    }
}
```