Programmation avec des objets : Cours 7

Menu du jour

1. Retour sur la classe Liste
2. Précisions sur l'interface
3. Difficultés dans le cas d'erreurs
4. Soulever des exceptions
5. Utilisation des Listes

Opérations souhaitées sur les listes

- Constructeurs: liste vide, liste contenant un élément
- `valeur(), reste(), estVide()`
- Ajout au début, à la fin
- Supprimer
- Calculer la longueur, vérifier si un entier appartient à la liste,

Principe de la conception/utilisation

- On donne les spécifications précises de chaque méthode ou fonction publique
- L'utilisateur de la classe liste ne doit pas connaître les détails de l'implantation
- Le concepteur de la classe doit satisfaire les spécifications
- Problème avec les mauvaises utilisations de la classe

Constructeurs de listes

- Construire une liste vide
- Construire une liste qui ne contient qu'un entier

Tester si une liste est vide

- Une méthode (non statique) appelée par `x.extVide()`
- Une fonction statique appelée par `estVide(x)`

Valeur et reste

- La valeur d'une liste est le premier élément.
- La méthode `reste()` donne comme résultat une liste obtenue en supprimant le premier élément, elle ne doit pas recopier car doit fonctionner en peu d'opérations (indépendant de la taille)
- Les deux méthodes ne doivent pas altérer la liste
- `valeur(), reste()` sont des méthodes non statiques
- La liste vide n'a ni valeur ni reste, un utilisateur doit veiller à ne pas appeler ces méthodes sur la liste vide

Ajouter, supprimer un entier

- On peut ajouter soit au début soit à la fin d'une liste
- La suppression d'un élément ne s'effectue que si l'élément est présent dans la liste, si tel n'est pas le cas la liste est inchangée .
- Les trois méthodes: `ajoutDebut(int a)` , `ajoutFin(int a)`, `supprimer(int a)` donnent un résultat de type `void` sont non statiques et opèrent par effet de bord

Longueur, appartient

- La longueur d'une liste est son nombre d'éléments, donnée par une méthode non statique `int longueur()` qui peut s'appliquer à la liste vide.
- Vérifier si un entier appartient à une liste s'effectue à l'aide d'une méthode non statique `boolean appartient(int a)`.
- Les deux méthodes: `longueur()` , `appartient(int a)` ne modifient pas les listes qui les appellent

Afficher une liste

- Une liste s'affiche en la transformant en chaîne de caratères
- On utilise pour cela la méthode

  ```
  String toString()
  ```

  qui permet d'utiliser par la suite la fonction usuelle d'affichage:

  ```
  System.out.println()
  ```

Les listes vues par l'utilisateur: une interface

```
public class Liste{

    Liste () // construit la liste vide
    Liste (int a)
           // construit la liste ne contenant que a
    public boolean estVide()
          // verifie si la liste est vide
    public static boolean estVide(Liste x)
          // la meme en statique
    public int valeur()
          // donne le  premier element,
          //  ne pas appliquer a la liste vide
    public Liste reste()
         //  donne la liste obtenue en supprimant le
         // premier element
         // ne pas appliquer a la liste vide
```

```
    public void  ajoutDebut(int a)
         //  ajoute au debut
         // sans modifier la reference au premier
    public void ajoutFin (int a)
        // ajoute a la fin
        // sans modifier la reference au premier
    public void supprimer (int a)
        //   supprime si appartient
        // sans modifier la reference au premier
    public int longueur()
        //     donne le nombre d'elements
    public boolean appartient (int a)
        //  vrai si a appartient a la liste d'appel
    public String toString()
      // transforme en chaine de caracteres affichable
}
```

Les liste vues par l'implanteur

- Faire que l'on puisse appliquer des méthodes à la liste vide
- Bien gérer les effets de bord
- Attention à la valeur et le reste sur une liste vide

Les liste vues par l'implanteur: champs, constructeurs

```
public class Liste{
    private int val;
    private Liste suiv;
    Liste (){
        val = 0; suiv = null;}
    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;}
    static boolean estVide(Liste x){
        return (x.suiv == null);}
```

Les liste vues par l'implanteur: exceptions sur listes vides

```
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;
    }
```

Les liste vues par l'implanteur: la suite

```
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;}}
```

Fonctions sur les listes

```
 static Liste reverse (Liste x){
      if (x.estVide()) return new Liste();
      else {
           Liste u = reverse(x.reste());
           int a = x.valeur();
           u.ajoutFin(a);
           return u;}}
 static Liste reverseAux(Liste u, Liste v){
        if (u.estVide()) return v;
        else {v.ajoutDebut(u.valeur());
              return reverseAux(u.reste(), v);}
}
 static Liste reverse1(Liste u) {
        Liste v = new Liste();
        return reverseAux(u, v);}
    }
```

Remarques sur les listes

- Attention aux effets de bords
- Liste d'appel modifiée ou conservée?
- Fonction sur la liste vide déclanche une exception
- Destructif ou pas
- Pas besoin de libérer de la place le glaneur de cellules (garbage collector) le fera
- Attention à ne jamais effectuer: `null.valeur` ou `null.reste()` .....

Classes abstraites

- Une classe abstraite est une classe qui contient une ou plusieurs méthodes abstraites
- Une méthode est abstraite si on ne définit que sa signature
- Une classe réalise une classe abstraite si c'est une sous-classe qui contient le corps de toutes les méthodes abstraites
- Obligation pour le programmeur de remplir toutes les spécifications demandées

Programmation avec des classes abstraites

- Une classe abstraite contient des fonctions avec seulement une déclaration
- Il faut les préciser dans les classes qui en héritent
- Lors de l'exécution le type de l'objet détermine la méthode utilisée
- Pour les listes : Une classe abstraite `Liste` etdes classes qui en héritent: