Licence Mention Informatique, Programmation avec des Objets

Cours de Robert Cori

Devoir surveillé du 24 mars 2007

Documents autorisés : sujets, corrigés de travaux dirigés et copies des transparents.
Les livres et photocopies de livres sont interdits. Le barême est indicatif.

Exercice 1 : Chaînes de caractères et tableaux *(8 points)*

On souhaite gérer un ensemble de chaînes de caractères par un tableau. On peut pour guider l'intuition prendre comme exemple l'ensemble de clients d'un commerçant.

Pour cela on défini une classe dont une variable globale `u` est un tableau de chaînes de caractères (qui contiendra les noms des clients du commerçant), dont la longueur est donnée par un entier `tailleMax` :

```
public class Exo1{
    static final int tailleMax = 1000;
    static String[] u = new String[tailleMax];
}
```

Question 1. Ecrire une fonction :

`static void initialise(String[] tab)`

qui fait que les premiers éléments de `u` soient des références sur les chaînes contenues dans le tableau `tab` (correspond à l'initialisation de l'ensemble des clients lors de l'installation du commerçant).

Question 2. Ecrire une fonction :

`static void ajoute(String f)`

qui ajoute la chaîne f au tableau `u` en la première position `u[i]` libre (c'est à dire contenant la valeur `null`). Si aucune place n'est libre alors on affichera un message d'erreur (correspond à l'arrivée d'un nouveau client).

Question 3. Ecrire une fonction :

`static boolean appartient(String f)`

qui indique si la chaîne `f` apparaît dans le tableau `u`. Pour cela vous utiliserez la méthode `x.equals(y)` qui donne pour résultat `true` si les chaînes `x` et `y` sont égales.

Question 4. Un programmeur a écrit la fonction `supprime` suivante pour supprimer un élément du tableau `u`, (correspond au départ d'un client).

```
public static void supprime (String x){
       int i = 0;
       while(i < tailleMax && !u[i].equals(x)) i++;
       if (u[i].equals(x)) u[i] = null;
       else System.out.println(x + "n'existe pas");
   }
```

Cette fonction donne lieu, dans beaucoup de cas, à un message `java.lang.NullPointerException`, pourquoi ? Comment modifier cette fonction pour éviter ce problème.

Exercice 2 : Construction d'une classe, listes *(12 points)*

On se propose de construire une classe `Message` qui permettra de simuler les envois de messages par Internet. Pour cela on considère une version très simplifiée dans laquelle un message contient l'adresse IP de l'émetteur (on supposera qu'il s'agit d'un entier), la date (un entier long correspondant au nombre de secondes écoulées depuis le 1 janvier 1970), le contenu du message (une chaîne de caractères) et les adresses IP de tous les destinataires du message (sous forme d'un tableau d'entiers).

I : La classe Message *(8pts)*

On souhaite que cette classe contienne un constructeur d'un message à partir de ses 4 composants et les méthodes suivantes :

- Méthode `int longContenu()` qui donne la longueur du contenu du message.
- Méthode `String toString()` qui construit pour un message donné, une chaîne de caractères contenant les adresses d'emission et d'envoi, le moment d'emission (tout ceci sous forme d'entiers), suivis du contenu du message.
- Méthode `boolean estDestineA(int x)` telle que `m1.estdestineA(x)` a pour résultat `true` si le nombre `x` est l'adresse IP d'un des destinataires du message.

Question 1. Ecrire la définition de cette classe, du constructeur et des trois méthodes.

Question 2. Ecrire une fonction statique

```
static Message fusion (Message m1, Message m2)
```

qui a partir de deux messages émis par un même utilisateur construise un nouveau message défini par :

- Le moment d'emission est le plus récent des deux moments d'emission de `m1` et `m2`,
- le contenu est la concaténation des deux contenus et les destinataires
- les destinataires sont tous ceux de l'ensemble formé par l'union des destinataires de `m1` et de `m2`.

Question 3. Ecrire une méthode :

```
Message[] décomposer()
```

qui à partir d'un message destiné à k utilisateurs construit un tableau de k messages destiné chacun à l'un des utilisateurs. Les contenus et moments d'emissions seront identiques pour tous les messages construits.

II : Listes de messages *(4 points)*

Question 4. On se propose de définir des listes de messages. Pour cela on reprend une partie de la classe des listes d'entiers étudiée en cours et qui est donnée sur la feuille suivante.

Comment doit-on modifier cette classe pour obtenir une classe `ListeMessage` permettant de gérer des listes de messages ? Vous ferez apparaître les modifications sur la feuille et insérerez cette feuille dans votre copie.

Question 5. Ecrire une méthode :

```
boolean doublons()
```

qui pour une liste de messages , dont on sait à priori que chacun ne s'adresse qu'à un seul destinataire détermine s'il existe deux messages, non n'ecessairement identiques, destinés au même utilisateur.

Nom : **Prénom :** **Groupe :**

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

    // par convention la liste vide est null
    // attention a ne pas lui appliquer de methode

    public Liste (int a) {
        val = a; suiv = null;
    }

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

    public static boolean estVide(Liste x){
        return (x == null);
    }

    public int valeur(){
        return val;
    }

    public Liste reste(){
        return suiv;
    }

    public int longueur(){
        if (suiv == null) return 1;
        else return 1 + reste().longueur();
    }

    public void  ajoutDebut(int a){
        // La nouvelle liste est a la meme adresse que la liste d'appel
        // la Liste initiale est modifiee
        // On ne peut pas utiliser cette methode pour la liste vide
        Liste x = new Liste(this.val, suiv);
        this.val = a;
        this.suiv = x;
    }

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