université PARIS DIDEROT PARIS 7

INITIATION À LA PROGRAMMATION (IP2)

TD6 : Un peu plus loin avec les listes

Dans ce TD, nous allons voir l'interaction des listes chaînées dans le contexte d'autres objets. Les méthodes récursives doivent être écrites de façon la plus naturelle possible, en faisant possiblement des méthodes auxiliaires dans les autres classes.

1 Resumé de liste

Supposons que nous avons deux classes `Cellule` et `Liste` avec un constructeur vide qui modélisent les listes d'entiers, ainsi qu'une méthode `ajouter` qui prend en argument un entier et l'ajoute au début de la liste.
La classe `ResumeListe` possède 3 attributs : un entier `min`, un entier `max` et une liste `liste`. Nous demandons que `min` et `max` contiennent respectivement la valeur minimale et maximale de la liste tant que la liste est non vide. Cette propriété doit être garantie par les méthodes dans la classe.

1. Écrire deux méthodes `int minimal()` et `int maximal()` **itératives** qui renvoient l'entier minimal et maximal dans la liste. Si la liste est vide, `int minimal()` et `int maximal()` doivent renvoyer 0 et -1 respectivement.
2. Écrire une méthode `boolean contient(int x)` dans `Liste` qui permet de savoir si un entier x est dans `this`.
3. Écrire la classe `ResumeListe`, avec son constructeur qui prend un paramètre du type `Liste`, les accesseurs, et une méthode `String toString()` qui renvoie un `String` des éléments dans `liste` séparés par des espaces.

2 Tableaux de Young

Un *tableau de Young* (ou simplement *tableau*) est un objet mathématique qui est composé par des lignes de boîtes de longueur décroissante de bas en haut, avec un entier dans chacune des boîtes. Donc la première ligne, située le plus bas, est la plus longue. On constate aussi que, si on retire la première ligne d'un tableau, ce qui reste est encore un tableau.
Nous pouvons modéliser un tableau avec une liste de listes d'entiers (une liste de `Liste`) ordonnées par longueur décroissante, avec la première ligne comme premier élément. Le tableau vide doit être représenté par une seule ligne vide, donc nous pouvons supposer qu'il existe toujours au moins un élément dans cette liste de `Liste`.

6	12			
4	8	13		
3	7	10		
1	2	5	9	11

FIGURE 1 – Un tableau de Young

1. Écrire une classe `Tableau` qui permet de modéliser un tableau (donc une liste de `Liste`). Elle doit porter deux attributs, `Liste ligne` pour la plus base ligne, et `Tableau sousTab` pour le tableau obtenu en supprimant la ligne la plus basse. Écrire un constructeur vide de `Tableau`, et un autre constructeur qui prend une ligne et un tableau en argument et qui renvoie le tableau obtenu en ajoutant la ligne en bas du tableau.
2. Écrire une méthode `boolean estValide()` qui dit si les lignes dans le tableau respectent l'ordre décroissant. On peut commencer par une méthode `int longueur()` qui renvoie la longueur d'une `Liste`.
3. Écrire une méthode **récursive** `void affiche()` pour afficher le tableau. Attention au sens d'affichage. Nous pouvons utiliser la méthode `String toString()` dans la classe `Liste`.
4. (*facultatif*) Écrire une méthode `Liste concat()` qui renvoie une `Liste` composée en concaténant toutes les lignes de bas en haut d'un `Tableau` **sans le détruire**. Nous pouvons commencer par une méthode `void concatEnTete(Liste queue)` dans la classe `Liste` qui ajoute les éléments de `this` un par un dans le bon ordre au début de `queue`.

3 Modélisation d'entrepôt

Les pièces d'un entrepôt sont référencées par leurs numéros de série. Elles sont rangées dans des compartiments. Sur chaque compartiment se trouve collée une fiche qui indique un intervalle dans lequel les numéros des pièces présentes se situent. Puis au sein de ce compartiment les choses sont plus où moins en vrac, les pièces y ont été déposées les unes derrière les autres. Selon cette présentation, un entrepôt peut donc être modélisé par un tableau de `ResumeListe`, avec les entiers dans la liste `liste` représentant les pièces, et `[min, max]` l'intervalle correspondant.

1. Écrire une classe `Entrepot` qui modélise les entrepôts, avec un constructeur correspondant.
2. Proposer un exemple d'une instance de `Entrepot` et faire une schématisation.
3. On dit que le classement est mauvais si les intervalles des résumés ne sont pas disjoints. Ecrivez une méthode `boolean estMauvais()` qui teste si le classement est mauvais. Nous pouvons commencer par penser au critère de la disjonction de deux intervalles `[min1, max1]` et `[min2, max2]`.
4. On suppose que le classement n'est pas mauvais. Écrire le plus efficacement possible une méthode `ResumeListe cherche(int ref)` qui cherche une pièce avec une certaine référence et renvoie le compartiment contenant cette pièce si on la trouve, et `null` sinon.