BPOO - Sujet TD 5

Dut/Info-S2/M2103

Table des matières

PreReq	TD1 : implantations mémoires, TD3 classes objets-encapsulation, TP2 : Utilisation d'objets.
ObjTD	**Stockage de données : listes chaînées**.
Durée	**2 séances** de 1,5h

1. Principes

Une structure autoréferrentielle (parfois appelée structure récursive) correspond à une structure dont au moins un des champs contient une référence vers une structure de même type.

De cette façon on crée des éléments **cellule** (appelés parfois noeuds ou liens) contenant des données, mais, contrairement à un tableau, celles-ci peuvent être éparpillées en mémoire et reliées entre elles par des liens logiques (des références), c'est-à-dire un ou plusieurs champs dans chaque structure contenant la référence d'une ou plusieurs structures de même type.

Plusieurs cas sont possibles :

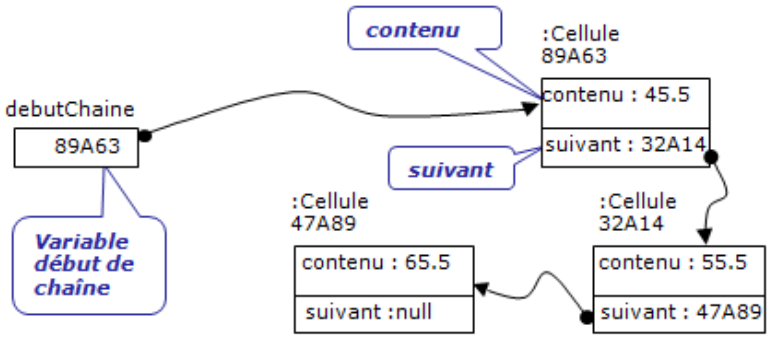
- Lorsque la structure contient des données et une référence vers la structure suivante : on parle de structure/liste chaînée.
- Lorsque la structure contient des données, une référence vers la structure suivante, et une référence vers la structure précédente : on parle de structure/liste chaînée double.
- Lorsque la structure contient des données, une référence vers une première structure suivante, et une référence vers une seconde, on parle d'arbre binaire.

2. Listes chaînées

2.1. Définition

On va considérer uniquement ici les structures chaînées de type liste simplement chaînée.

1. Pour gérer de telles listes, il faut :
   - des éléments **cellule** contenant une valeur (contenu) et une référence vers la **cellule suivante**,
   - repérer a minima la première **cellule** de la chaîne.
2. Exemple de liste chaînée contenant des réels :



   On a donc des **cellules** reliées entre elles comme des maillons d'un chaîne.
3. Attention : importance de la valeur **null** de la dernière Cellule → repère la fin de la chaîne (dernier maillon).
4. Attention : `debutChaine` est ici une variable "externe" à la structure chaînée. Il faut nécessairement repérer **au moins** le début de la structure chaînée (une variable, un attribut d'une classe plus tard, …).

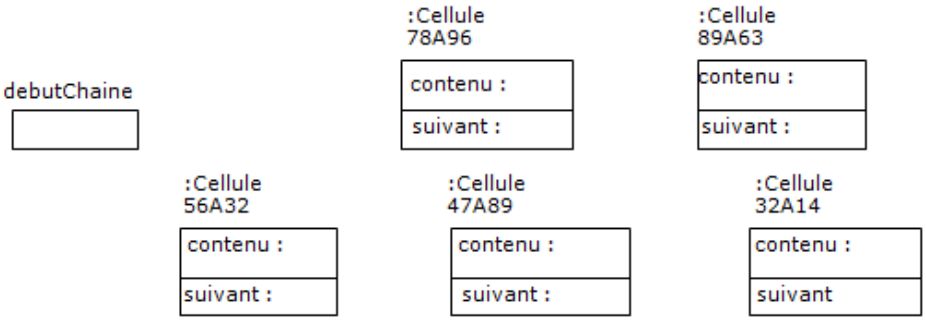
A noter :

- Liste chaînée fait penser à liste et ArrayList ⇒ on attendrait des méthodes
- On présente plutôt ici des **structures** chaînées, sans opération spécifique définie (ajout, retrait, …).
- Par abus de langage, on parlera de "listes" chaînées

2.2. Exemple

Question : Proposer un schéma de cette liste (barrer les cellules inutiles) avec :

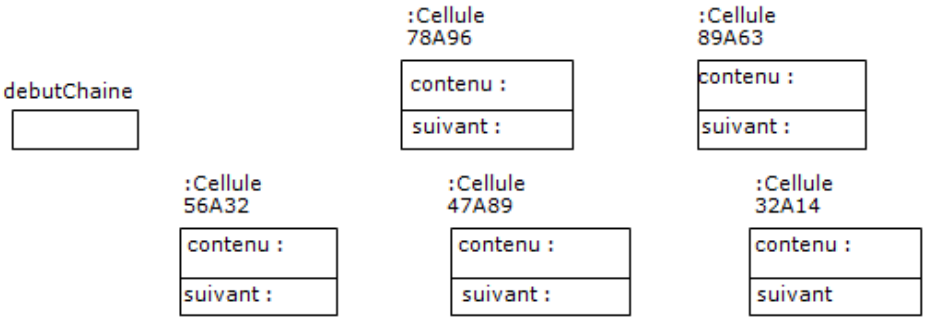
1. Un élément 35.5 de plus au début.
2. Un élément 75.5 de plus en fin.



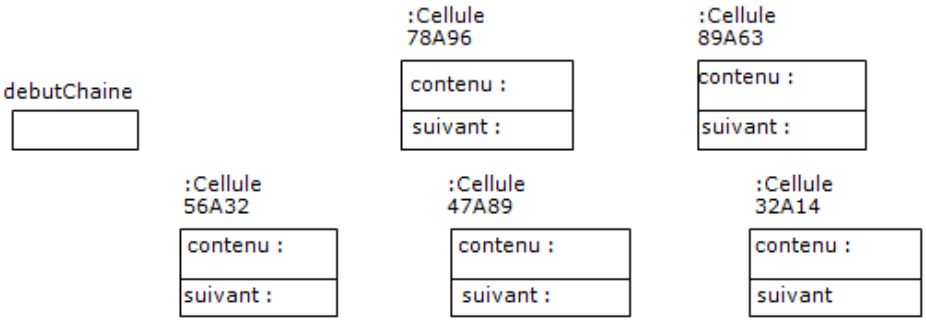
3. L'élément 35.5 en moins.

4. L'élément 55.5 en moins.

5. L'élément 75.5 en moins.



6. Comment représenter une liste chaînée ne comportant aucune valeur ?

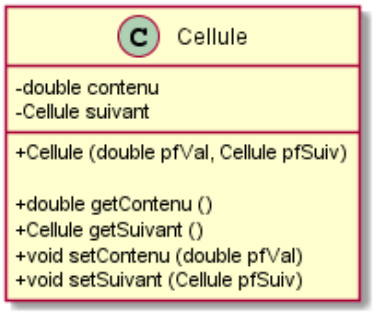


3. Utiliser "manuellement" une liste chaînée

3.1. La classe Cellule

Afin de pouvoir gérer les cellules d'une liste chaînée, on propose la classe ci-dessous :

Diagramme UML de la classe Cellule



Source java de la classe Cellule

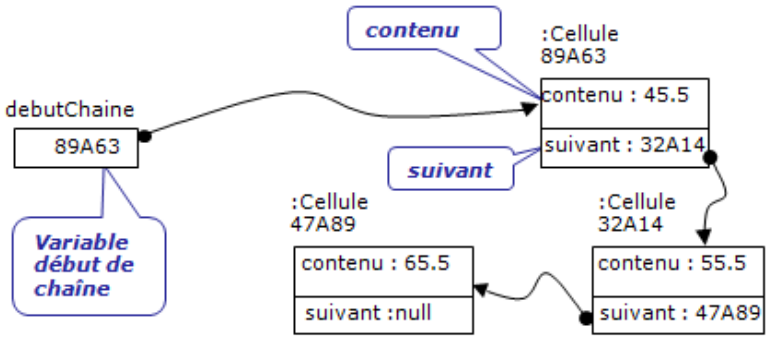
```
public class Cellule {
        private double contenu;
        private Cellule suivant;

        public Cellule (double pfVal, Cellule pfSuiv) {
                this.contenu = pfVal;
                this.suivant = pfSuiv;
        }
        public double getContenu () {
                return this.contenu;
        }
        public Cellule getSuivant () {
                return this.suivant;
        }
        public void setContenu (double pfVal) {
                this.contenu = pfVal;
        }
        public void setSuivant (Cellule pfSuiv) {
                this.suivant = pfSuiv;
        }
}
```

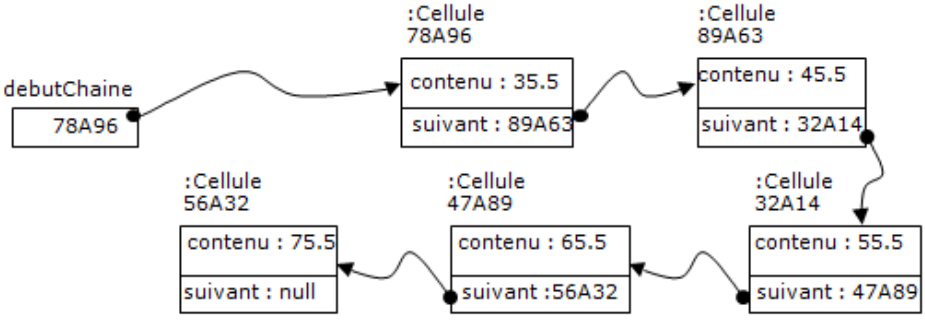
3.2. Utiliser la classe Cellule

Écrire le programme simple proposé en annexe.

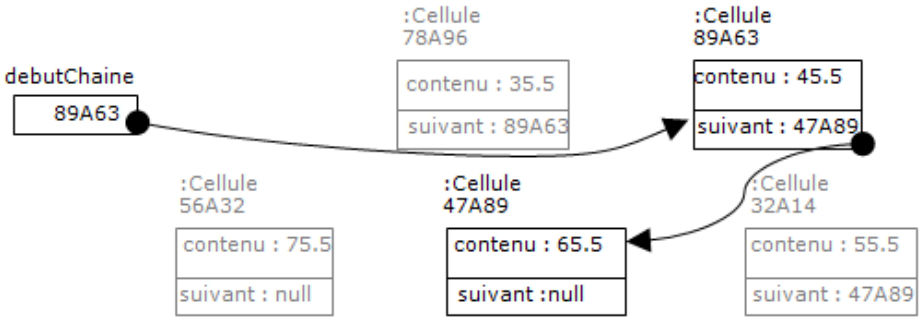
En utilisant trois variables, la partie 1 doit permettre de construire :



En ne conservant que l'information `debutChaine`, la partie 2 doit permettre de construire :



En ne conservant que l'information `debutChaine`, la partie 3 doit permettre de construire :



Enfin, chercher un algorithme permettant la suppression d'un élément (ou valeur, appelons-la `valASup`) n'importe où dans la structure chaînée et éventuellement absent de la structure. Il faudra réfléchir :

- au cas où l'élément n'est pas dans la structure chaînée,
- au cas où l'élément est présent et est le premier de la structure chaînée,
- au cas où l'élément est présent et est "au milieu" de la structure chaînée,
- au cas où l'élément est présent et est le dernier de la structure chaînée (mais est ce bien un cas particulier ?).

---

4. Différence tableau "dynamique" <> liste chaînée simple

Donner les différences entre tableaux "dynamiques" et listes chaînées simples dans les situations suivantes :

1. Ajout d'un élément en fin.
2. Ajout au début.
3. Suppression d'un élément (début, milieu, fin).
4. Accès élément i.

Dernière mise à jour 2017-03-05 16:56:56 CET