TP1 - Manuel d’utilisation

Giraud Antoine

Faou Matthieu

La liste chainée et la suite chainée sont deux fonctionnalités relativement indépendantes et c’est pourquoi ce manuel s’attachera à expliquer l’utilisation de chacune d’entre elle.

# Liste Chainée

Ce module permet de créer une liste chainée c’est-à-dire une liste donc les éléments sont attachés les uns après les autres. Ainsi, à l’inverse d’un tableau, il est nécessaire de parcourir la liste pour atteindre un élément.

La liste chainée se déclare de la façon suivant :  
 ListeChainee myList = new ListeChainee();

Ensuite, de nombreuses méthodes permettent de travailler sur la liste chainée.

* **add(int element)** : Permet d’ajouter un élément à la liste chainée.
* **add(String chaine)** : Permet d’ajouter plusieurs éléments sous la forme *val1, val2, val2, …* à la liste chainée.
* **reset()**: vide la liste.
* **isEmpty()**: Retourne vrai si la liste est vide, faux sinon.
* **contains(int element)**: Retourne vrai si la liste contient l’élément element, faux sinon.
* **getAt(int position)**: Retourne la valeur contenue à l’index position.
* **setAt(int position, int element)**: Remplace la valeur contenue à l’index position par element.
* **removeAt(int position)**: Supprime le maillon qui est à la position position.
* **removeItem(int element)**: Supprime tous les éléments égaux à celui passé en paramètre.
* **getSize()**: Retourne la taille de la liste.
* **show() ou showByOrderedAsc()**: Affiche le contenu de la liste dans l’ordre dans lequel les éléments ont été ajoutés.
* **showByKeyOrderedDesc()**: Affiche le contenu de la liste dans l’ordre dans lequel les éléments sont stockés.
* **toString() :** Affiche les éléments de la liste dans leur ordre d’ajout, séparés par des virgules.

*Remarque :* l’index de notre liste chainée fonctionne sur le même principe que celui d’un tableau : l’élément 0 est le premier élément ajouté dans la liste.

*Remarque :* prenons une liste chainée équivalente à un tableau : [0 => ‘elem1’, 1 => ‘elem2’]. Si on retire le chainon 0 ou le chainon qui comporte « elem 1 » ; notre tableau deviendra [0 => ‘elem2’]

# Suite

Ce module permet de créer une suite à partir de deux valeurs et d’une opération (addition, soustraction, multiplication ou division). Cette suite est ensuite enregistrée dans un fichier qui permet de reprendre ultérieurement la génération de celle-ci.

Pour créer une suite, on utilisera l’instruction suivante :

Suite mySuite = Suite(path, operator, val1, val2, length, emptyState)

Où :

* **path** : nom du fichier de sauvegarde.
* **operator**: type d’opération (add, sub, mult ou div).
* **val1 et val2** : les deux premières valeurs permettant de générer la suite.
* **length**: longueur de la suite à générer (maximum 10).
* **emptyState**: True pour recalculer entièrement la suite, false pour reprendre la génération à partir des valeurs contenues dans le fichier de sauvegarde.

La fonction **show()** permet d’afficher les valeurs de la suite.

On notera que val1 et val2 seront les deux premières valeurs de la suite.

On notera que si les valeurs passées lors de l’initialisation de la suite diffèrent de celles contenues dans le fichier de sauvegarde, alors les valeurs de ce-dernier sont écrasées.

On notera aussi que si val1 et val2 ne correspondent pas aux deux dernières valeurs calculées et enregistrées dans le fichier de sauvegarde alors les calculs reprendront avec val1 et val2 mais sans ajouter ces deux valeurs dans la suite.

On notera qu’il est possible de continuer des calculs déjà commencés avec un opérateur différent.