LES STRUCTURES DE DONNEES EN JAVA

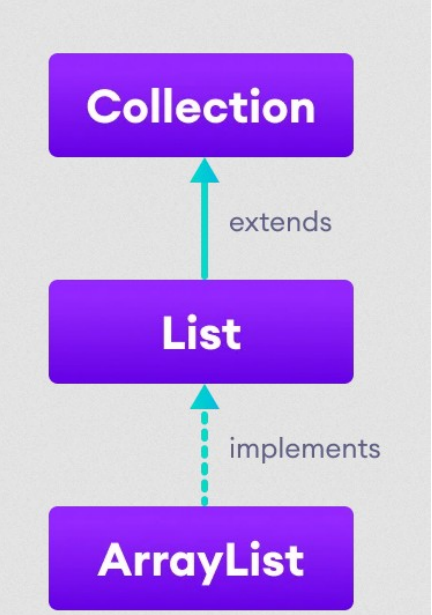
Réalisé par :

Rafed Safour

Table de Matière

I. ARRAYLIST

➔ ArrayList Définition :



ArrayList est une partie de collections, il est présenté dans le package de java.util. Il nous fournit des tableaux dynamiques en Java. Cependant, cela peut être plus lent que les tableaux standards, mais peut être utile dans les programmes où de nombreuses manipulations dans le tableau sont nécessaires. ArrayList permet une gestion plus simple des tableaux et son taille gérer automatiquement, par contre il utilise plus la mémoire.

➔ Utilisation

Pour utiliser la classe **ArrayList**, vous devez utiliser l'une des instructions d'importation suivantes :

IMPORT java.util.ArrayList ;

IMPORT java.util.* ;

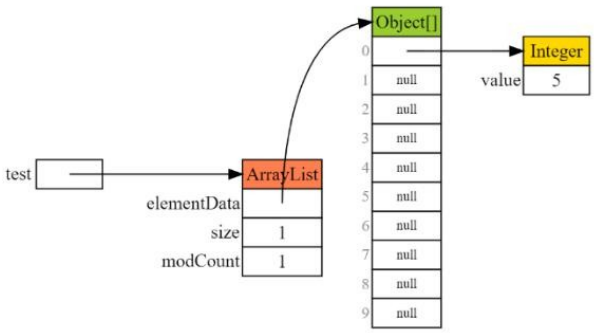
➢ Ensuite, pour déclarer un ArrayList, vous pouvez utiliser le constructeur par default :

ArrayList <Type d'éléments> test = new ArrayList <Type d'éléments>();

Exemple avec le schéma ARTOZ : (implémentation)

ArrayList <Integer> test = new ArrayList <Integer>();

test.add(5);



➔ *3 points essentiels à mémoriser :*

◆ Pour récupérer ou supprimer un élément, tu as besoin de connaitre sa position.

◆ ArrayList maintient l'ordre, le premier élément ajouté est dans la première case.

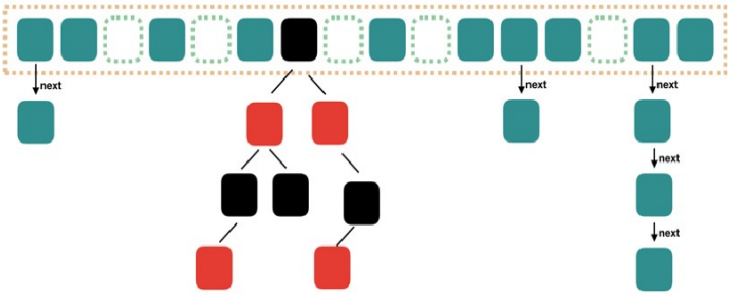
◆ Un élément peut changer sa position dans l'ArrayList à chaque fois, il n'est pas attaché à une position spécifique.

➔ Méthodes

Méthode	Description
public void add(Object)	Ajouter un élément à une ArrayList ; la version par défaut ajoute surchargée vous permet de spécifier une position à laquelle nous voulons ajouter l'élément
public void add(int, Object)	
public void remove(int)	Supprimer un élément d'une ArrayList à un emplacement spécifié
public void set(int, Object)	Modifier un élément à un emplacement spécifié dans une ArrayList
Object get(int)	Récupérer un élément d'un emplacement spécifié dans une ArrayList
public int size()	Renvoyer la taille actuelle de ArrayList
boolean contains(Object)	Renvoie vrai si la liste contient la valeur l'Object
int indexOf(Type élément)	Renvoie la position d'élément dans la liste, et -1 s'il n'y apparaît pas
clear()	Vider la liste

II. HASHMAP

➔ Définition



Cette classe est une implémentation pratique d'une structure où on accède à des éléments non plus avec un indice numérique comme dans un tableau ou un ArrayList, mais avec une clé qui peut être de n'importe quel type. Ce type doit être précisé lors de la déclaration d'une variable ou de l'appel à un constructeur pour créer un objet. HashMap utilise une table de hachage, elle accepte la valeur null, et la clé doit être unique pour bien la parcourir.

➔ Utilisation

Pour utiliser la classe **HashMap**, vous devez utiliser l'une des instructions d'importation suivantes :

IMPORT java.util.HashMap;

IMPORT java.util.* ;

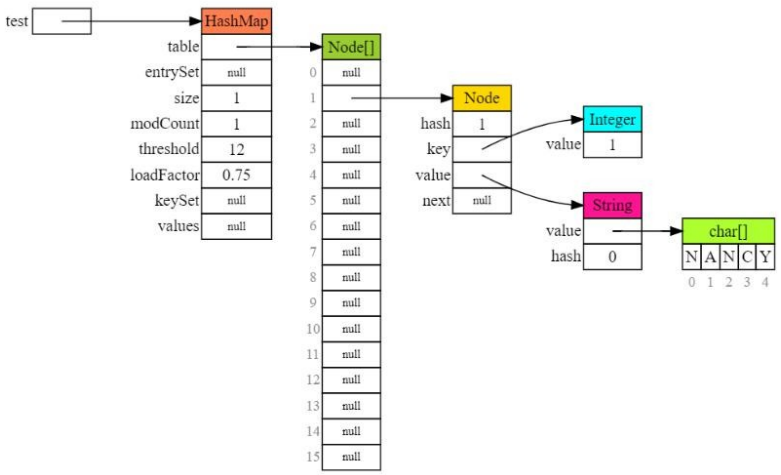
➢ Ensuite, pour déclarer un HashMap, vous pouvez utiliser le constructeur par default :

HashMap <Type d'éléments_clé , Type d'éléments_valeur > test = new HashMap <Type_d'éléments_clé , Type_d'éléments_valeur > () ;

Exemple avec le schéma ARTOZ : (implémentation)

HashMap <Integer,String> test = new HashMap <Integer, String>();

test.put(1, "NANCY");



➔ ***3 points essentiels à mémoriser :***

- Pour récupérer un élément, tu as besoin de connaitre sa clé.
- Le stockage d'objet se fait en deux objets : une clé et une valeur.
- La relation entre une valeur et sa clé n'est pas arbitraire.

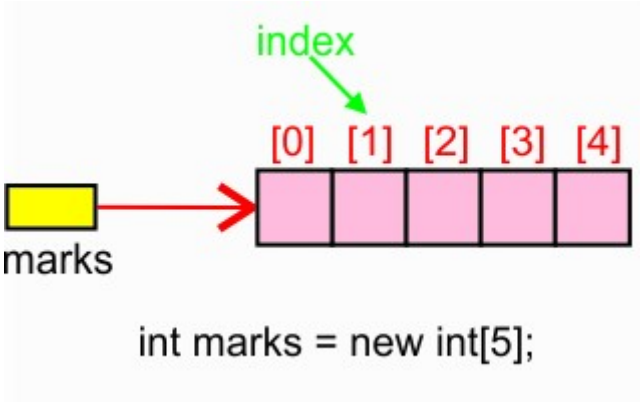
➔ Méthodes

Méthode	Description
clear()	Efface tous les éléments de la structure
boolean containsKey(K key)	Teste si une clé appartient à la structure.
boolean containsValue(V value)	Teste si un élément est présent dans la structure.
boolean isEmpty()	Teste si la structure est vide.
V get(K key)	Obtenir des valeurs de HashMap
V put(K key, V value)	Rajoute un nouveau couple (clé, élément) dans la structure.
V remove(K key)	Élémine un élément de la structure. La clé est donnée en paramètre.
int size()	Renvoie la taille (nombre d'éléments).

III. TABLEAU

➔ Définition

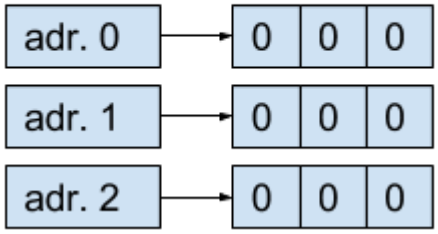
Un tableau est une structure de données contenant un groupe d'éléments tous du même type. Le type des éléments peut être un type primitif ou une classe.



- La classe Table est utilisée pour afficher les données sous forme de tableau. Il est composé de lignes et de colonnes
- La taille est fixée lors de l'instanciation et ne peut plus changer
- Les éléments rangés de façon contigu en mémoire
- Nous pouvons faire un tableau de 2 dimensions ou plus : **int[ ][ ] tab**
- Nous pouvons créer plus d'un type de tableau

int | String | char | double | . . . . | un type de classe

t = new int[3][3];



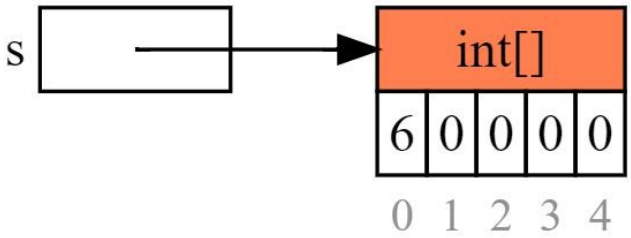
➔ Utilisation

- Pour déclarer un tableau :

Type_elements nom_var = new type_elements [Capacité] ;

Exemple avec le schéma ARTOZ : (implémentation)

```
int [] s = new int [5] ;
s[0] = 6 ;
```



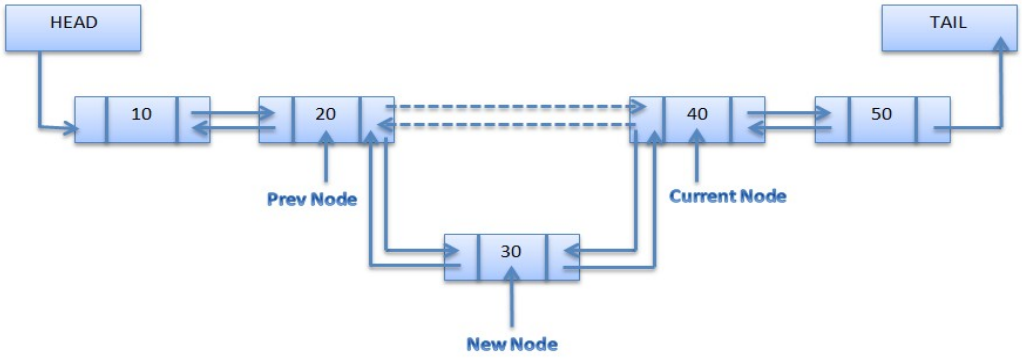
Les inconvénients	***Les avantages***
Il ne peut pas être redimensionner.	*On peut utiliser l'opérateur = pour modifier une case dans le tableau.*
	Rapide au niveau de l'exécution.
	Facile à implémenter

IV. LINKEDLIST

➔ Définition

LinkedList implémente le concept de la liste doublement chainé.

Dans LinkedList, les éléments peuvent être stockés à n'importe quel emplacement mémoire disponible car l'adresse du nœud est stockée dans le nœud précédent.



✓ Utilisation

Pour utiliser la classe **LinkedList**, vous devez utiliser l'une des instructions d'importation suivantes :

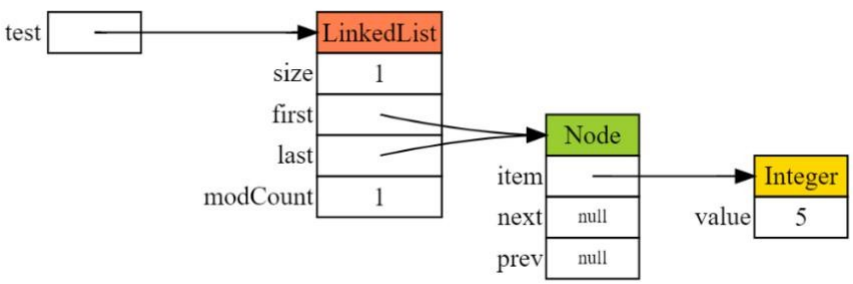
```
IMPORT java.util.LinkedList;
IMPORT java.util.* ;
```

- Ensuite, pour déclarer une LinkedList, vous pouvez utiliser le constructeur par default :

LinkedList <Type d'éléments> test = new LinkedList <Type_d'éléments > () ;

Exemple avec le schéma ARTOZ : (implémentation)

```
LinkedList <Integer> test = new LinkedList <Integer>();
test.add(5);
```



✔ Méthodes

Méthode	Description
public void addFirst()	Ajoute un élément au début de la liste
public void addLast()	Ajouter un élément à la fin de la liste
public void removeFirst()	Supprimez un élément du début de la liste
public void removeLast()	Supprimer un élément de la fin de la liste
public void getFirst()	Récupérez l'élément au début de la liste
public void getLast()	Récupérez l'article à la fin de la liste

Les avantages	***Les inconvénients***
Vous pouvez redimensionner LinkedList au moment de l'exécution.	*LikedList nécessite plus de mémoire pour stocker les éléments qu'un tableau,*
Vous pouvez effectuer les opérations d'insertion et de suppression d'une façon rapide.	*On ne peut pas accéder à un index spécifique.*

CONCLUSION

Après avoir voir les différents types de collections, on va attaquer les cas d'utilisation conseiller pour chacune.

- Si vous avez une taille fixe de la collection, et vous n'avez pas besoin de la redimensionner, c'est préférable d'utiliser un tableau.
- Ou cas d'une taille variable :
  - Vous voulez stocker et accéder aux données, c'est préférable d'utiliser une ArrayList.
  - Vous voulez manipuler les données, c'est préférable d'utiliser une LinkedList.
  - Vous voulez accéder à un élément en utilisant une clé spécifique, vous pouvez utiliser HashMap.

Remarque :

On ne peut pas les type primitives (int, double ...) dans les collections, on utilise les classes enveloppe.