

Chapitre V : Structures Séquentielles

- Les listes.

- Les piles.

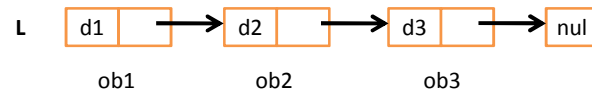
- Les files.

Les Listes chaînée

- Une **liste chaînée désigne une structure de données** représentant une collection ordonnée et de taille arbitraire d'objets.
- L'accès aux éléments d'une liste se fait de manière séquentielle.
- Chaque objet permet l'accès au suivant (contrairement au cas du tableau dans lequel l'accès se fait de manière absolue, par adressage direct de chaque cellule du tableau).
- Un objet contient un accès vers des données.
- Le principe de la liste chaînée est que chaque objet possède, en plus des données, des références vers les objets qui lui sont logiquement adjacents dans la liste.
- **premier(L)** : désigne le premier objet de la liste L.
- **nul** : désigne l'absence d'objet.
- Liste simplement chaînée :
 - **donnée(ob)** : désigne les données associée à l'objet ob.
 - **suivant(ob)** : désigne l'élément suivant ob.

Les Listes chaînée

- Représentation d'une liste L : \longrightarrow correspondant au suivant



- $\text{premier}(L) = \text{ob1}$.
- $\text{suivant}(\text{ob1}) = \text{ob2}$.
- $\text{suivant}(\text{ob2}) = \text{ob3}$.
- $\text{suivant}(\text{ob3}) = \text{nul}$.

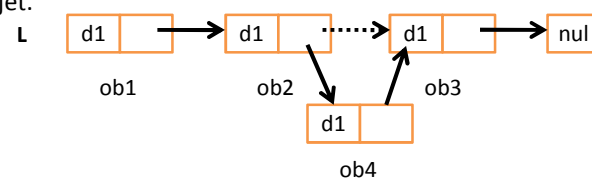
Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

3

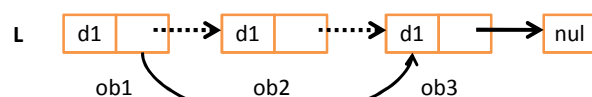
Les Listes chaînée

- **Opérations sur liste:** Trois opérations principales.

- Parcours de la liste.
- Ajout d'un objet.



- Suppression d'un objet.



- A partir de là d'autres opérations vont être obtenues : recherche d'une donnée, remplacement, concaténation de liste, fusion de listes, etc.

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

4

Les Listes chaînée

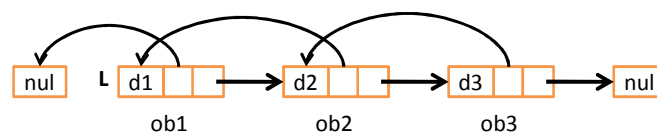
- **Tableau vs Liste** : Le principal **avantage** des listes sur les tableaux
 - L'ordre des objets de la liste peut être différent de leur ordre en mémoire.
 - Les listes chaînées vont permettre l'ajout ou la suppression d'un objet en n'importe quel endroit de la liste en temps constant.
- **En revanche**, certaines opérations peuvent devenir coûteuses comme la recherche d'un objet contenant une certaine donnée: on ne peut pas atteindre un objet sans parcourir la liste.

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

5

Les Listes doublement chaînée

- Représentation : \longrightarrow correspond à un lien.



- $\text{premier}(L) = \text{ob1}$.
- $\text{suivant}(\text{ob1}) = \text{ob2}$, $\text{précédent}(\text{ob1}) = \text{nul}$.
- $\text{suivant}(\text{ob2}) = \text{ob3}$, $\text{précédent}(\text{ob2}) = \text{ob1}$.
- $\text{suivant}(\text{ob3}) = \text{nul}$, $\text{précédent}(\text{ob3}) = \text{ob2}$.

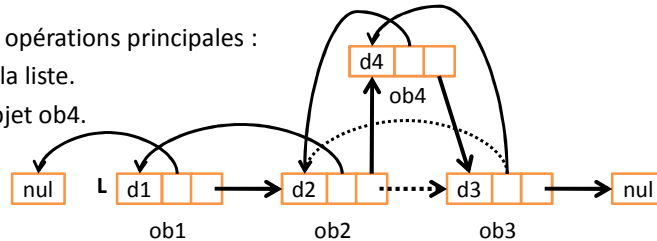
Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

6

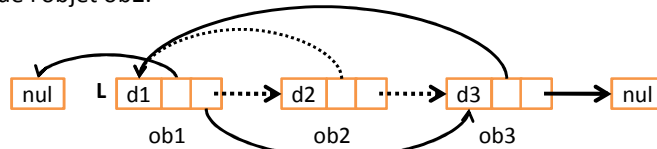
Les Listes doublement chaînée

• Opération : Trois opérations principales :

- Parcours de la liste.
- Ajout de l'objet ob4.



- Suppression de l'objet ob2.



• A partir de là d'autres opérations vont être obtenues : recherche de donnée, remplacement, concaténation de liste, fusion de listes, etc.

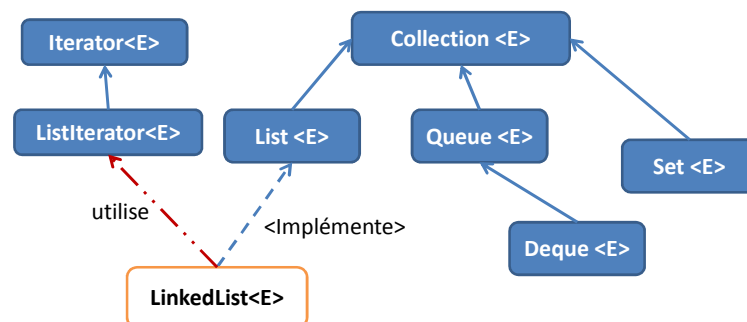
Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

7

Les Listes chaînée

• En java les listes sont manipulées par la classe générique **LinkedList<E>** qui implémente l'interface générique **List<E>** dérivé de l'interface générique **Collection<E>** qui sont définies dans le package java.util.* .

• Pour parcourir une liste , la classe **LinkedList<E>** utilise l'interface **ListIterator<E>** (itérateur bidirectionnel) qui est dérivée de l'interface **Iterator** (itérateur mono-bidirectionnel).



Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

8

Les Listes chaînée

• **L'interface générique List<E>** : c'est une interface qui détermine les différentes méthodes qui peuvent s'opérer sur une liste. Voici quelques méthodes :

Nom de la méthode	rôle
void add (int index, E elem)	Ajoute elem à la position index
E get (int index)	Fournit l'élément de rang index
ListIterator <E> listIterator ()	Fournit un itérateur bidirectionnel
ListIterator <E> listIterator (int index)	Fournit un itérateur bidirectionnel initialisé à index
E remove (int index)	Supprime l'objet de rang index et en fournit la valeur
E set (int index, E elem)	Remplace l'élément de rang index par elem
List<E> subList (int debut,int fin)	Fournit une vue d'une sous liste de la liste d'origine, constituée des éléments de rang debut à fin

Les Listes chaînée

• **L'interface générique Collection<E>** : c'est l'interface de base de tous types de listes. Voici quelques méthodes de cette interface :

Nom de la méthode	rôle
boolean add (E elem)	Ajoute elem à la collection
void clear ()	Supprime tous les éléments de la collection
boolean contains (Object elem)	True si elem appartient à la collection false sinon
Iterator <E> iterator ()	Fournit un itérateur monodirectionnel
boolean isEmpty ()	True si la collection est vide, false sinon
boolean remove (Object elem)	Supprime elem s'il existe et retourne true, sinon retourne false.
int size ()	Fournit le nombre d'éléments

Les Listes chaînée

- **L'interface générique ListIterator<E>** : c'est une interface qui fournit des méthodes permettant de parcourir une liste bidirectionnellement :

Nom de la méthode	Rôle
void add (E elem)	Ajoute elem à la position courante
boolean hasPrevious ()	True s'il existe un élément précédant la position courante
int nextIndex ()	Fournit le rang de l'élément courant
E previous ()	Fournit l'élément précédant la position courante et y déplace l'itérateur
int previousIndex ()	Fournit le rang de l'élément précédant l'élément courant
void set (E elem)	Remplace l'élément courant par elem.

Les Listes chaînée

- **L'interface générique Iterator<E>** : c'est l'interface de base de **ListIterator** qui fournit des méthodes permettant de parcourir une liste mono-directionnellement :

Nom de la méthode	Rôle
boolean hasNext ()	True si la position courante désigne un élément
E next ()	Fournit l'élément courant et avance l'itérateur
void remove ()	Supprime l'élément courant

Les Listes chaînée

- **La classe générique LinkedList <E>**: c'est une classe permettant de créer des listes, cette classe implémente les interface List<E>, Queue<E> (Java5), Deque<E>(Java 6), voici quelques méthodes :

Nom de la méthode	Rôle
void addFirst (E elem)	Ajoute elem en debut de liste
void addLast (E elem)	Ajoute elem en fin de liste
E getFirst ()	Fournit le premier élément de la liste
E getLast ()	Fournit le dernier élément de la liste
LinkedList ()	Construit une liste
E removeFirst ()	Supprime le premier élément de la liste et en fournit la valeur
E removeLast ()	Supprime le dernier élément de la liste et en fournit la valeur

Les Listes chaînée

- **Exemple 1 : La création d'une liste**

```
import java.util.*;
public class Listel {
    public static void main(String[] args) {
        LinkedList<Integer> l = new LinkedList<>();
        l.add(4);
        l.add(2);
        l.add(3);
        l.add(6);

        for(int i=0;i<4;i++)
            System.out.print(l.get(i)+" ");
    }
}
```

- Le programme affiche :

4 2 3 6

Les Listes chaînée

• Exemple 1 : L'ajout d'un élément

```
import java.util.*;
public class Listel {
    public static void main(String[] args) {
        LinkedList<Integer> l = new LinkedList<>();

        l.add(4); l.add(2); l.add(3); l.add(6);

        for(int i=0;i<4;i++)
            System.out.print(l.get(i)+" ");

        ListIterator <Integer> it = l.listIterator();
        it.next(); it.next();
        it.add(15);
        System.out.println();
        it = l.listIterator();
        while(it.hasNext()) {
            System.out.print(it.next()+" "); }
    }}

```

• Le programme affiche :

```
4 2 3 6
4 2 15 3 6
```

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

15

Les Listes chaînée

• Exemple 1 : La modification d'un élément

```
import java.util.*;
public class Listel {
    public static void main(String[] args) {
        LinkedList<Integer> l = new LinkedList<>();

        l.add(4); l.add(2); l.add(3); l.add(6);

        for(int i=0;i<4;i++)
            System.out.print(l.get(i)+" ");

        ListIterator <Integer> it = l.listIterator();
        it.next(); it.next();
        it.set(35);
        System.out.println();
        it = l.listIterator();
        while(it.hasNext()) {
            System.out.print(it.next()+" "); }
    }}

```

• Le programme affiche :

```
4 2 3 6
4 35 3 6
```

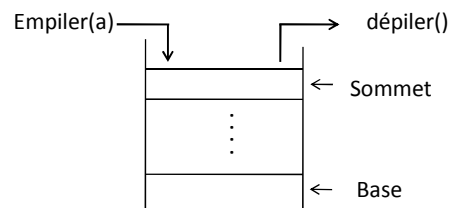
Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

16

Les Piles

- Une pile (stack en anglais) est une liste chaînée dans laquelle :
 - Un élément ne peut être ajouté qu'au sommet de la pile,
 - Un élément ne peut être retiré que du sommet de la pile.

Il s'agit donc d'une structure de type LIFO (Last In First Out: dernier entré, premier sortie). On ne travaille que sur le sommet de la pile. Les piles sont comparables à des piles d'assiettes.



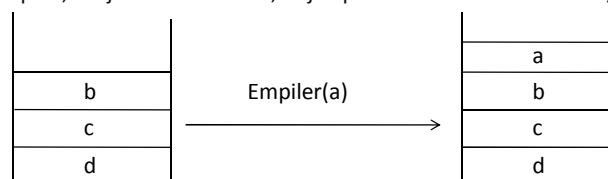
- On peut associer à une pile deux termes :
 - empiler(a) : pour ajouter l'élément 'a' au sommet de la pile → push(a)
 - dépiler() : pour supprimer l'élément se trouve au sommet de la pile → pop()

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

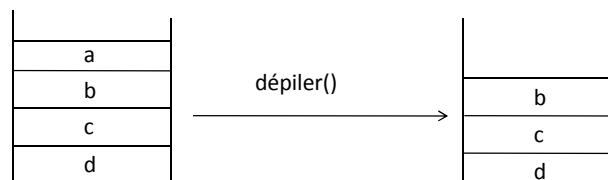
17

Les Piles

- Les opérations autorisées avec une pile sont :
 - empiler, toujours au sommet, et jusqu'à la limite de la mémoire,



- dépiler, toujours au sommet, si la pile n'est pas vide,



- vérifier si la pile est vide ou non.

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

18

Les Piles

- Les piles servent à revenir à l'état précédent et sont utilisées pour :
 - implanter les appels de procédures (pour revenir à l'état d'avant l'appel),
 - annuler une commande,
 - Dans un navigateur web, une pile sert à mémoriser les pages Web visitées. L'adresse de chaque nouvelle page visitée est empilée et l'utilisateur dépile l'adresse de la page précédente en cliquant le bouton 'Afficher la page précédente'
 - évaluer des expressions arithmétiques,
 - ...

Exemple : évaluation de l'expression $5*(3+4)$ selon la notation polonaise utilisé dans les machine HP ($5*(3+4)$ devient $4\ 3\ +\ 5\ *$) : chaque nombre est ajouté à la pile et chaque opération prend des éléments dans la pile et retourne le résultat sur la pile :

- push(4)
- push(3)
- push(pop() + pop())
- push(5)
- push(pop() * pop())

A la fin il reste donc juste 35 sur la pile.

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

19

Les Piles

- **Les piles en utilisant la classe générique Stack<E>** : c'est une classe permettant la création d'une pile, et offre différentes méthodes pour la manipulation de celle-ci, parmi lesquelles:

Méthode	Rôle
Stack ()	construire une pile VIDE
boolean empty ()	la pile est-elle vide
E peek ()	consulter l'objet au sommet sans le retirer de la pile
E pop ()	retirer l'objet au sommet et le retourne
void push (E elem)	empiler un objet au sommet de la pile

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

20

Les Piles

- Exemple : une pile qui contient 3 éléments

```
import java.util.*;
public class LesPiles {
    public static void main(String[] args) {
        Stack <String> pile = new Stack<>();

        pile.push("a1");
        pile.push("a2");
        pile.push("a3");

        System.out.println(pile.pop());
        System.out.println(pile.pop());
        System.out.println(pile.pop());
    }
}
```

- Le programme affiche :

```
a3
a2
a1
```

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

21

Les Piles

- Les piles en utilisant la classe générique **LinkedList<E>** : Nous pouvons utiliser aussi la classe **LinkedList<E>** pour créer une pile en utilisant les méthodes :

Nom de la méthode	Rôle
void addFirst (E elem)	Empiler elem au sommet de la pile
E getFirst ()	Fournit l'élément au sommet de la pile sans le supprimer
LinkedList ()	Construit une pile
E removeFirst ()	Dépiler un élément du sommet de la pile

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

22

Les Piles

- Exemple : création d'une pile contenant 3 élément

```
import java.util.*;
public class JavaApplication48 {
    public static void main(String[] args) {
        LinkedList <String> pile = new LinkedList<>();

        pile.addFirst("a1");
        pile.addFirst("a2");
        pile.addFirst("a3");

        System.out.println(pile.removeFirst());
        System.out.println(pile.removeFirst());
        System.out.println(pile.removeFirst());
    }
}
```

- On obtient :

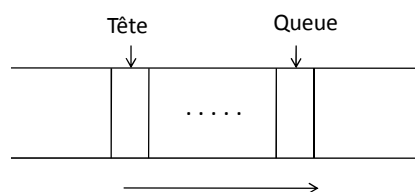
a3
a2
a1

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

23

Les Files

- Une file (queue en anglais), ou file d'attente, est une liste chaînée dans laquelle :
 - Un élément ne peut être ajouté qu'à la queue de la file,
 - Un élément ne peut être retiré qu'à la tête de la file.



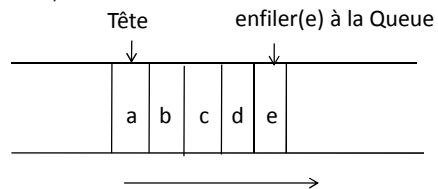
- Il s'agit donc d'une structure de type FIFO (First In First Out : premier entré, premier sorti). Les données sont retirées dans l'ordre où elles ont été ajoutées. Une file est comparable à une queue de clients à la caisse d'un magasin.

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

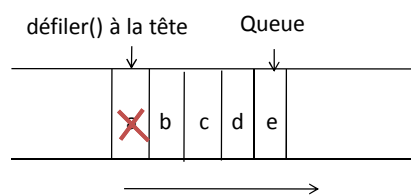
24

Les Files

- Les opérations autorisées avec une file sont :
 - enfiler(e) : enfiler l'élément e à la queue. Cette opération est réalisable jusqu'à la limite de la mémoire,



- défiler() : toujours à la tête si la file n'est pas vide,



- vérifier si la file est vide ou non.

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

25

Les Files

- Les files servent à traiter les données dans l'ordre où on les a reçues et permettent de :
 - gérer des processus en attente d'une ressource système (par exemple la liste des travaux à éditer sur une imprimante)
 - construire des systèmes de réservation
 - le routage de paquets réseau
 - etc.

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

26

Les Files

- Java (JDK 5.0) manipule les files via l'interface **Queue<E>** qui dérivée de l'interface **Collection<E>**, il s'agit de structures dans lesquelles on peut :
 - ☐ introduire un nouvel élément, si la file n'est pas pleine,
 - ☐ prélever le premier élément de la file.
- Les méthodes de cette interface sont données par :

Méthode	Rôle
E element()	Fournit, sans le supprimer, le premier élément de la file.
boolean offer (E elem)	Place elem dans la file, si possible et renvoie true; renvoie false sinon.
E peek()	Fournit le premier élément de la file sans le supprimer (null si file vide).
E poll()	Fournit le premier élément de la file en le supprimant (null si file vide).
E remove()	Fournit le premier élément de la file, en le supprimant (exception si file vide)

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

27

Les Files

- Pour la création de notre file nous utilisons la classe **LinkedList<E>** qui implémente l'interface **Queue<E>**.

- Exemple : Gestion de file d'attente**

```
import java.util.*;
public class LesFiles {
    public static void main(String[] args) {
        // TODO code application logic here
        LinkedList <String> file = new LinkedList<>();

        file.offer("Mohamed");
        file.offer("Khaled");
        file.offer("Amina");
        file.offer("Ridha");

        System.out.print(file.poll() + " ");
        System.out.print(file.poll() + " ");
        System.out.print(file.poll() + " ");
        System.out.print(file.poll() + " ");
        System.out.print(file.poll());
    }
}
```

Le programme affiche : Mohamed Khaled Amina Ridha null

Cours A.S.D M.LAKEHAL Dep.
Informatique Univ. M'sila

28