Partie III

La programmation orientée objets



Informatique 1
Introduction à la programmation

Mission 11 : INTRODUCTION

Les listes chaînées

Kim Mens – Siegfried Nijssen – Charles Pecheur

Mission 11 : Matière à lire

Objects

- 1 - Classes and objects - Basics
- 2 - Classes and objects - Advanced
- 3 - Even more object-oriented programming
- 4 - Overloading and polymorphism
- 5 - Collections of objects
- 6 - Inheritance
- 7 - Linked lists
- Appendix - Source code of phone class
- Appendix - Worked out example: accounts
- Appendix - Source code of card game
- Appendix - Source code of linked lists

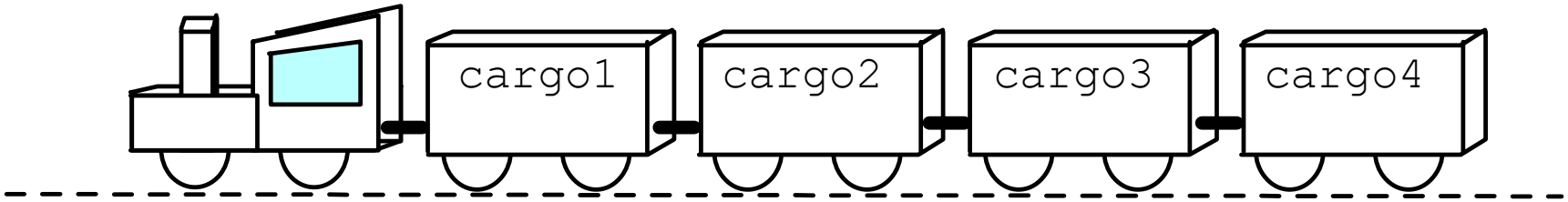
listes chaînées

principes

opérations

variantes

classes internes



Structure de données

Une **structure de données** est une manière de grouper et d'organiser des données afin de faciliter certaines opérations sur ces données.

> Par exemple, traverser tous les éléments d'une collection de données.

Une **liste chaînée** est une structure de données, groupant une collection d'éléments en une séquence de nœuds chainées et qui fournit des opérations comme:

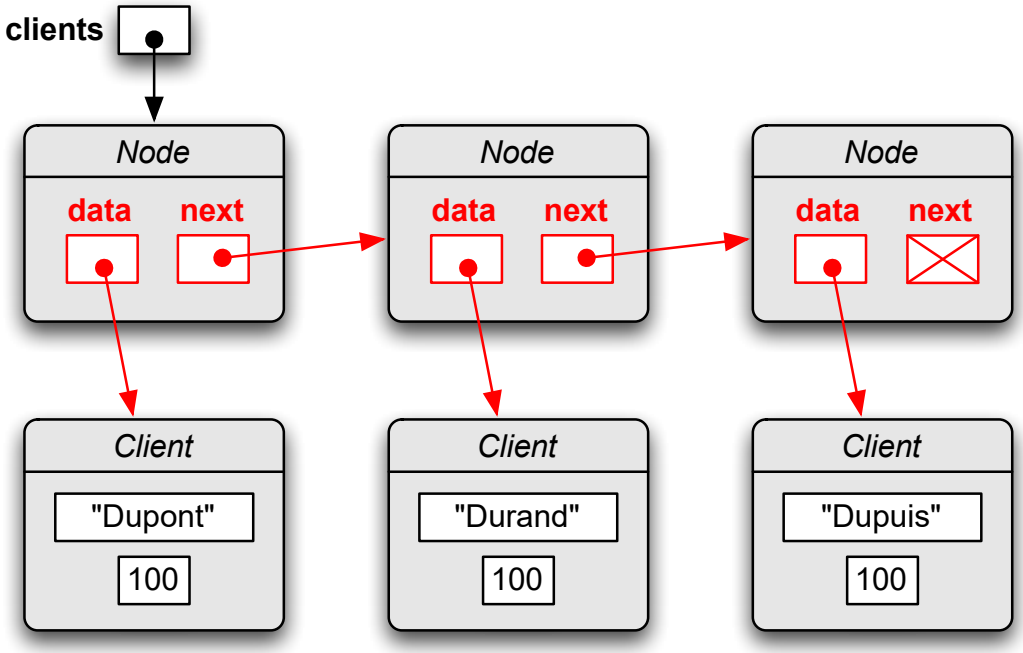
- parcourir les éléments séquentiellement
- imprimer tous les éléments
- ajouter un élément
- supprimer un élément
- insérer un élément

Chaîne de nœuds

Chaque nœud est un objet contenant:

- Un attribut `data` pour stocker une donnée (par ex. une instance de la classe `Client`)
- Un attribut `next` contenant une référence vers le nœud suivant

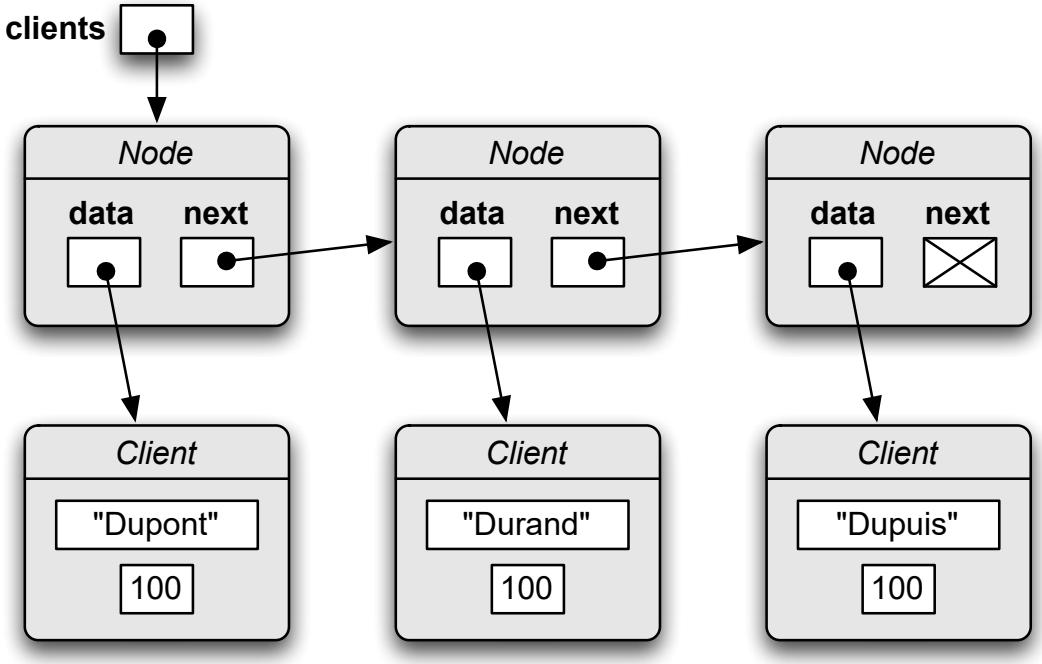
Les nœuds servent essentiellement à chaîner les données.




```
class Client:
  def __init__(self,n,c):
    self.name = n
    self.credit = c
```

```
class Node:
  def __init__(self,d,nx):
    self.data = d
    self.next = nx
```

Création d'une chaîne de nœuds

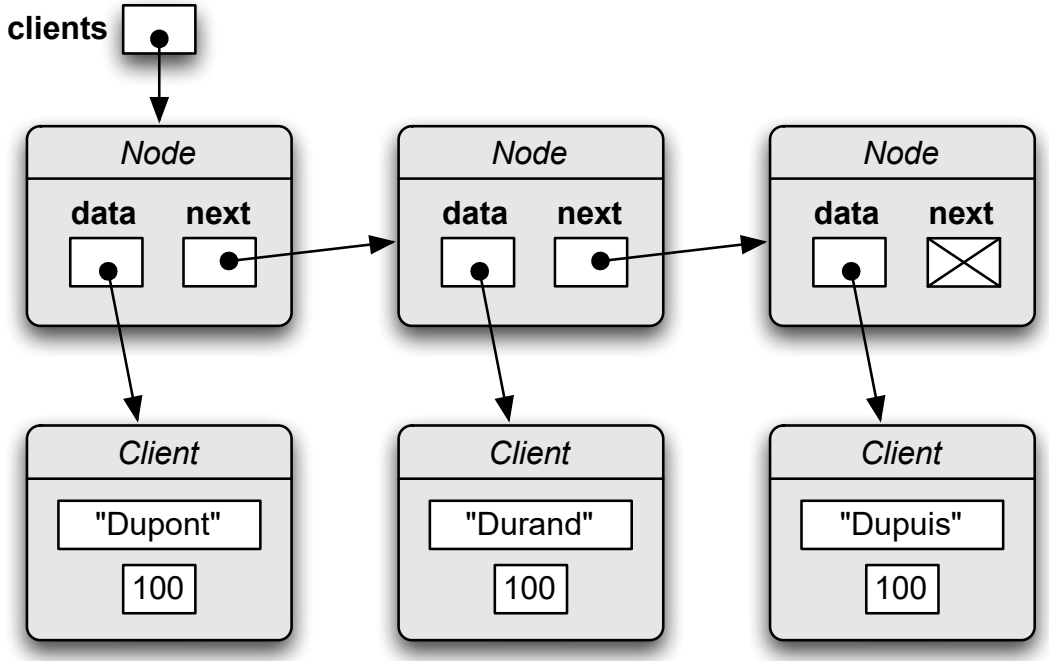



```
class Node:
  def __init__(self,d,nx):
    self.data = d
    self.next = nx
```

```
class Client:
  def __init__(self,n,c):
    self.name = n
    self.credit = c
```

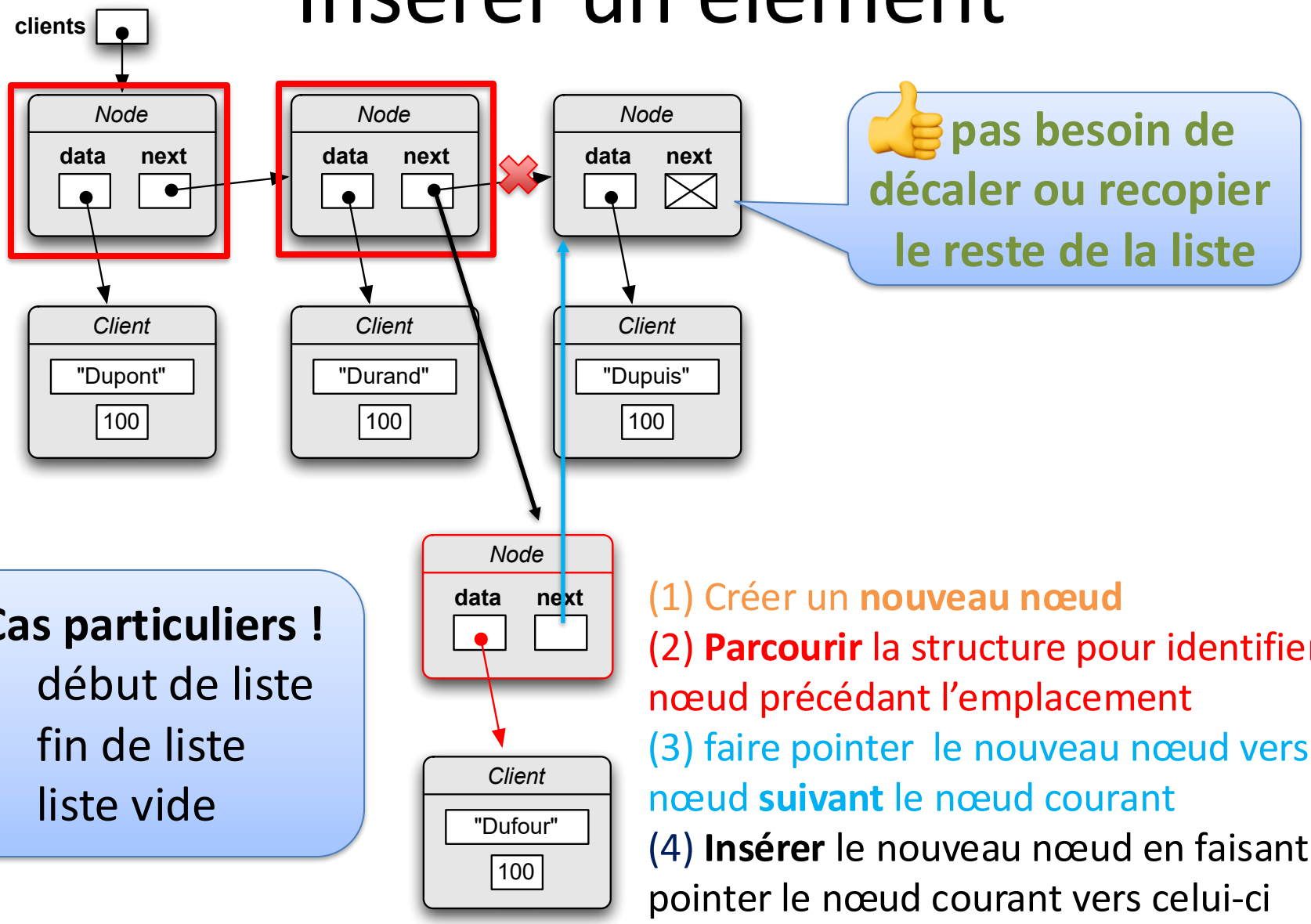
```
clients = Node(Client("Dupont",100),
               Node(Client("Durand",100),
                    Node(Client("Dupuis",100),None)))
```

Parcourir les éléments




```
n = clients
while n is not None:
    print(n.data)
    n = n.next
```

Insérer un élément



Cas particuliers !

- début de liste
- fin de liste
- liste vide

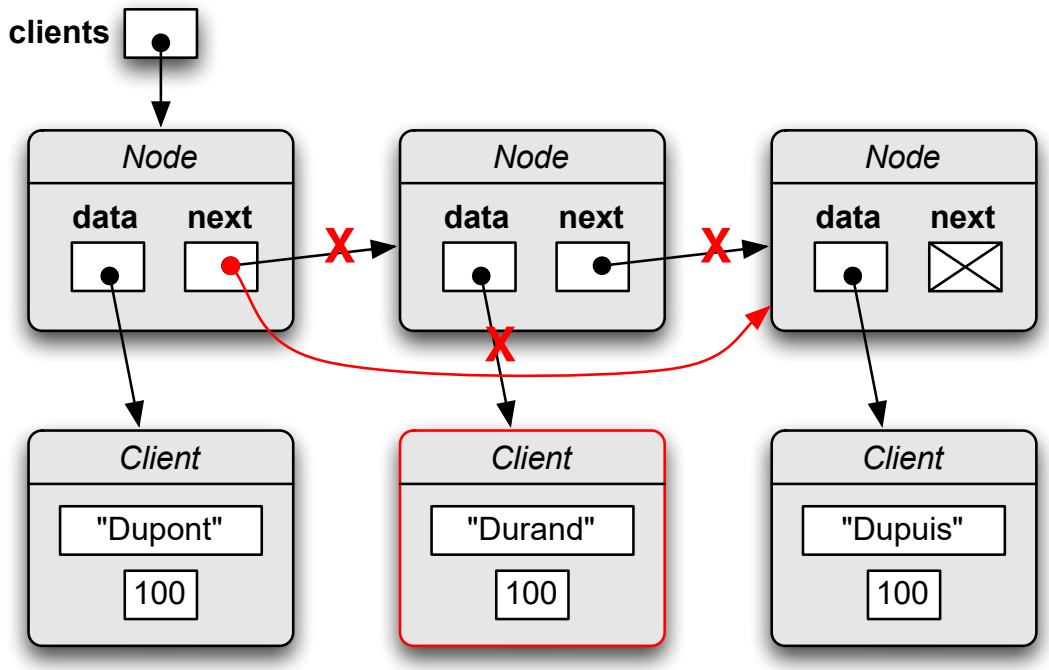
(1) Créer un **nouveau nœud**

(2) **Parcourir** la structure pour identifier le nœud précédant l'emplacement

(3) faire pointer le nouveau nœud vers le nœud **suivant** le nœud courant

(4) **Insérer** le nouveau nœud en faisant pointer le nœud courant vers celui-ci

Retirer un élément

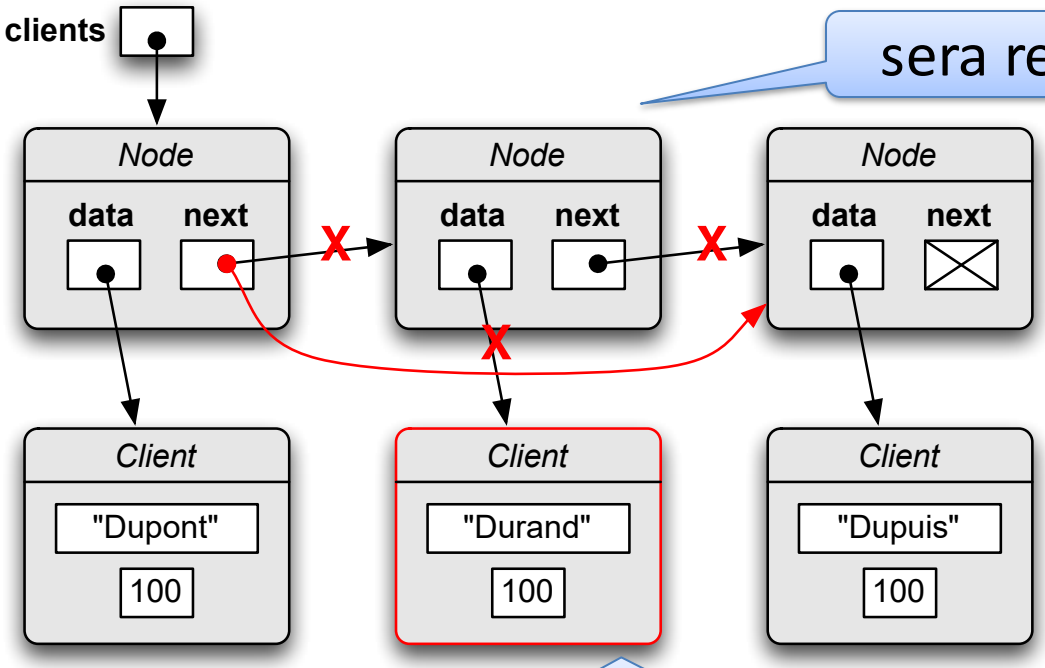


(1) Vérifier la **valeur** de l'élément à retirer

(2) **Parcourir** la structure pour identifier le nœud précédant l'emplacement du nœud à supprimer

(3) **Retirer** le nœud correspondant

Retirer un élément



sera recyclé (ramasse-miettes)

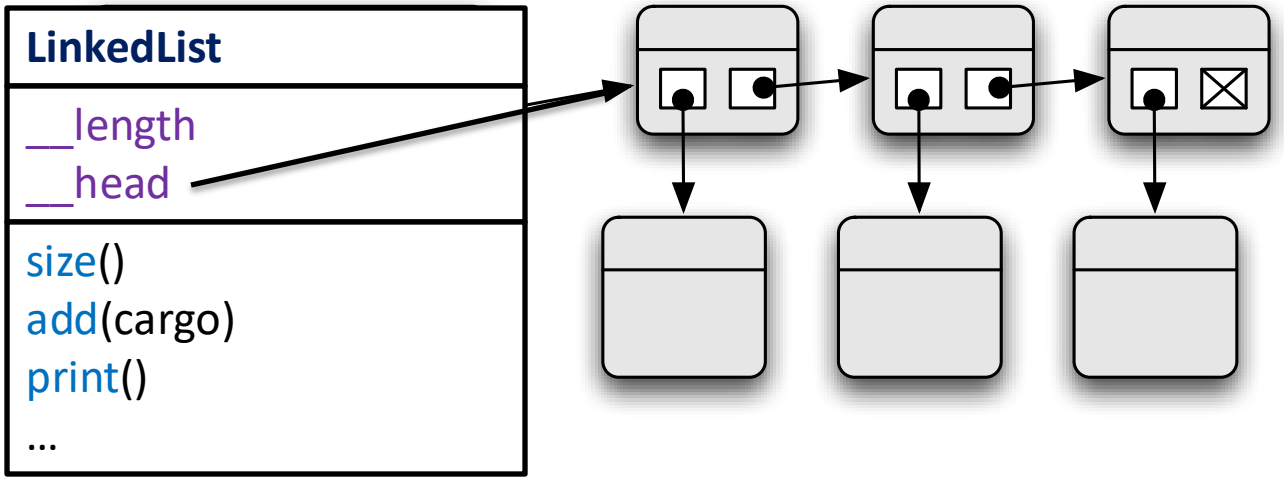
pourra être recyclé
(si pas d'autres références)

+ cas particuliers !

Classe **LinkedList**

Implémentation d'une liste (simplement) chaînée comme classe Python :

- Utilise une **chaîne de nœuds** comme composant, en gardant un référence `__head` vers le premier nœud
- Ainsi qu'un attribut `__length` représentant la taille
- Gère automatiquement le **chaînage des nœuds**
- Offre des méthodes spécifiques supplémentaires :
  - ajout, impression, insertion et retrait d'éléments, …



Classe **LinkedList**

```
class LinkedList :

  def __init__(self):
    self.__length = 0
    self.__head = None

  def size(self):
    return self.__length

  def add(self, cargo):
    node = Node(cargo,self.__head)
    self.__head = node
    self.__length += 1

  ...
```


Objects
- 1 - Classes and objects - Basics
- 2 - Classes and objects - Advanced
- 3 - Even more object-oriented programming
- 4 - Overloading and polymorphism
- 5 - Collections of objects
- 6 - Inheritance
- 7 - Linked lists
- Appendix - Source code of phone class
- Appendix - Worked out example: accounts
- Appendix - Source code of card game
- Appendix - Source code of linked lists


```
clients = LinkedList()

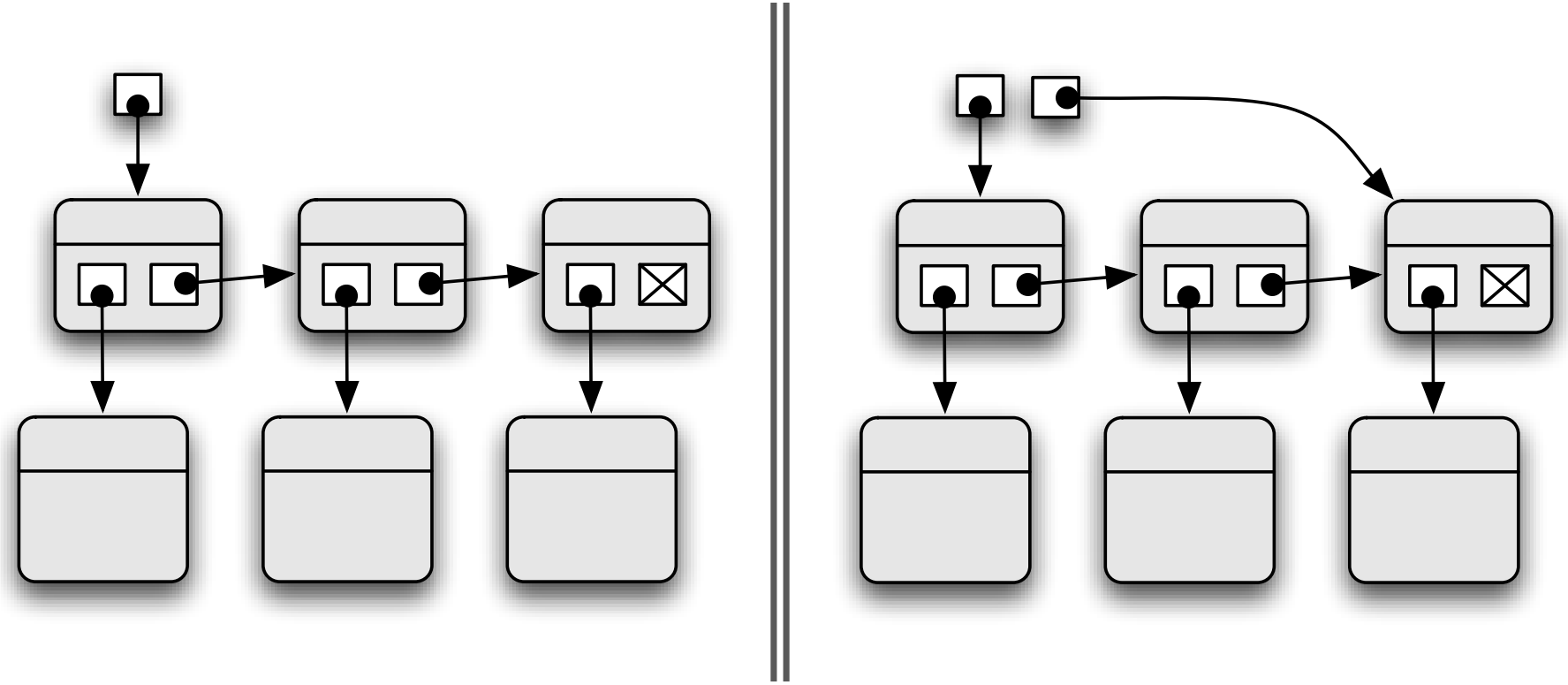
clients.add(Client("Dupuis",100))

clients.add(Client("Durand",100))

clients.add(Client("Dupont ",100))
```

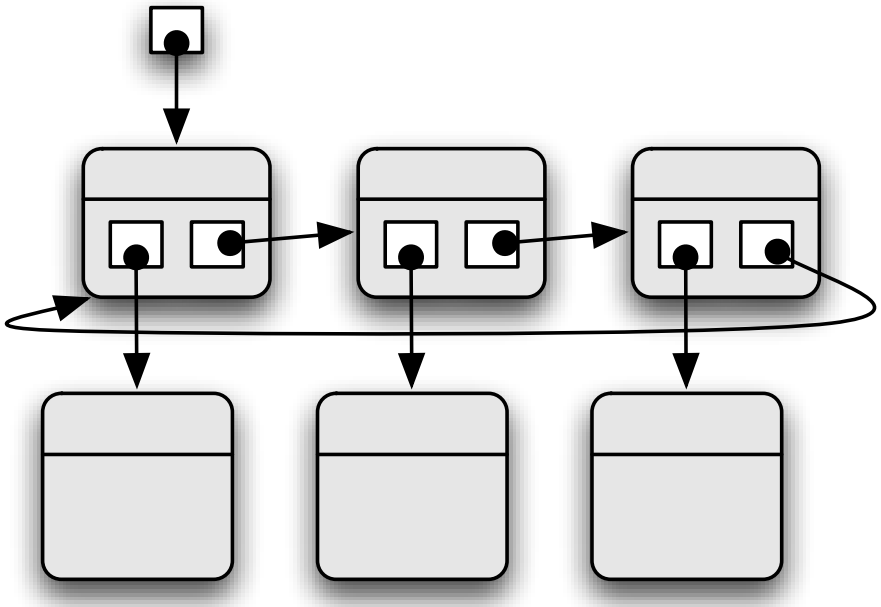
Variantes de listes chaînées

liste simplement chaînée

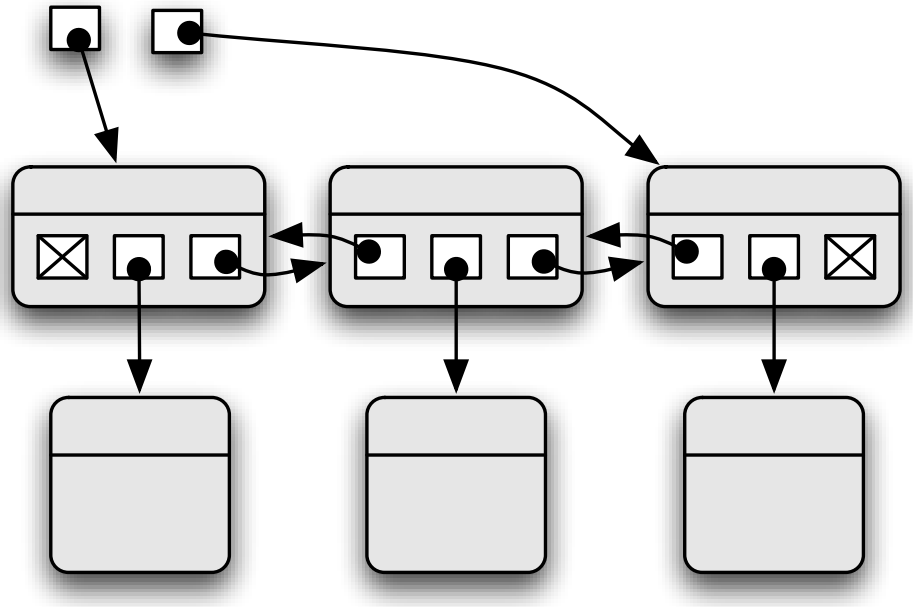


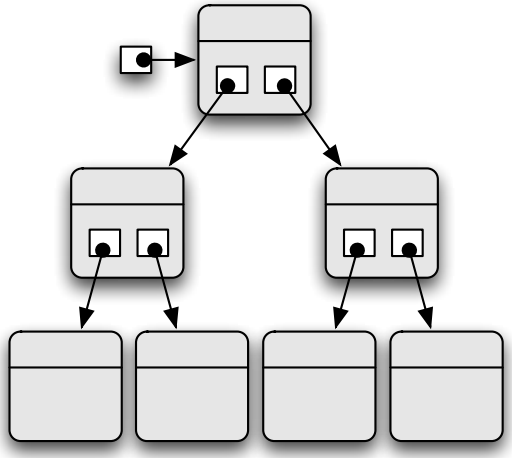
Variantes de listes chaînées

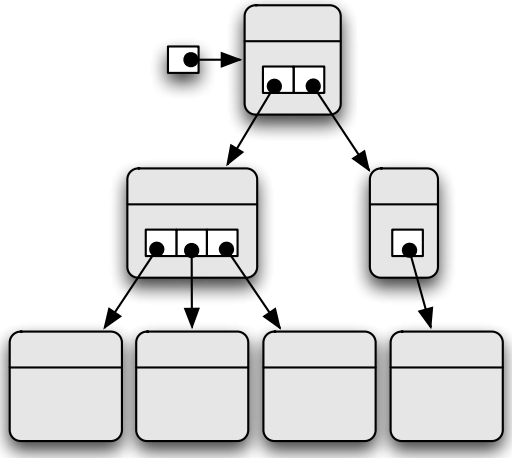
liste circulaire

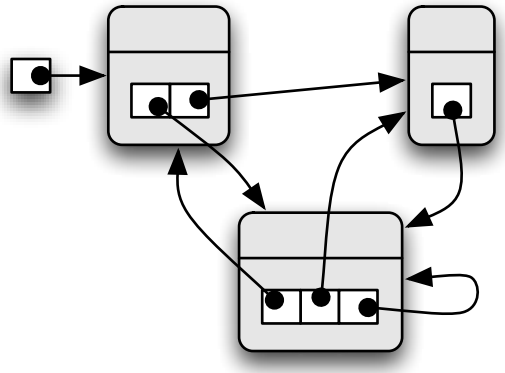


liste doublement chaînée









arbre binaire

arbre quelconque

graphe

Autres structures chaînées