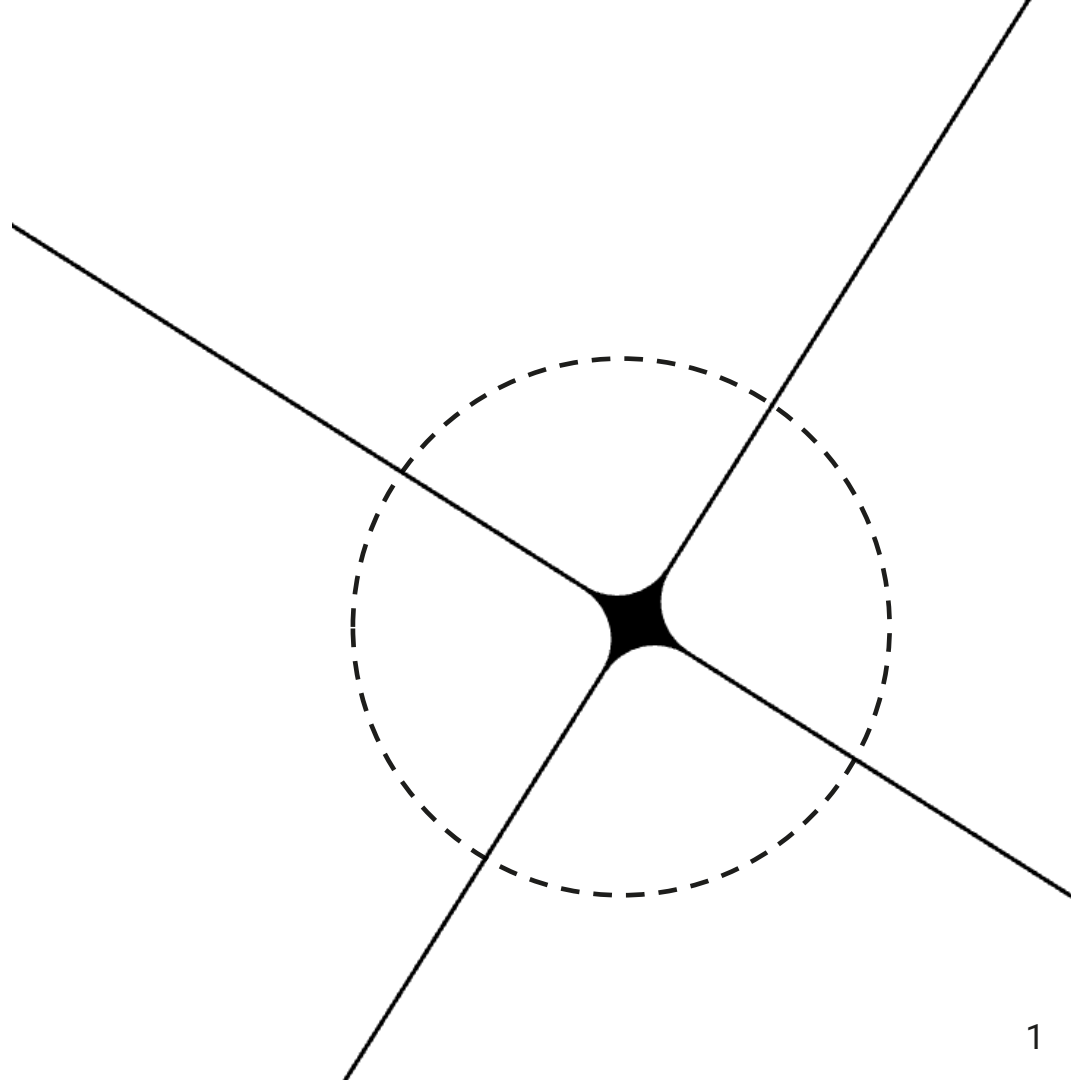
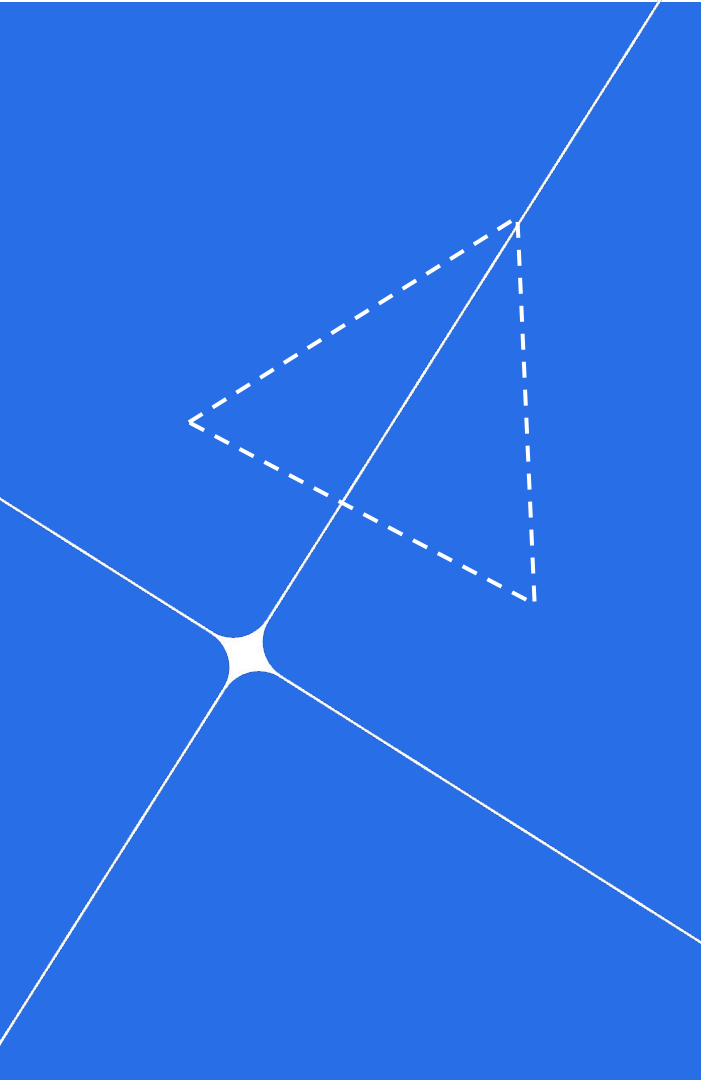


Introduction TP 2

Julien Romero





Le tableau extensible

Le tableau extensible

Inconvénient des tableaux: On ne peut pas ajouter plus d'éléments que la taille du tableau.

Solution: Quand un tableau est plein, on va augmenter sa taille pour permettre d'ajouter de nouveaux éléments. C'est le **tableau extensible**.

Notre tableau extensible implémente deux opérations:

- L'ajout d'élément dans le tableau
- L'extension du tableau

Le tableau extensible - Ajout d'un élément

Contrairement à un tableau classique, le tableau extensible n'est rempli que jusqu'à un indice donné, que l'on doit conserver.

Pour ajouter un élément (en considérant que l'on a la place), il faut:

1. Ajouter l'élément à l'indice d'insertion actuel
2. Incrémenter l'indice d'insertion

Au début, notre tableau est vide, l'indice d'insertion est 0.

Indice d'insertion



Le tableau extensible - Ajout d'un élément

Contrairement à un tableau classique, le tableau extensible n'est rempli que jusqu'à un indice donné, que l'on doit conserver.

Pour ajouter un élément (en considérant que l'on a la place), il faut:

1. **Ajouter l'élément à l'indice d'insertion actuel**
2. Incrémenter l'indice d'insertion

Au début, notre tableau est vide, l'indice d'insertion est 0.

Indice d'insertion



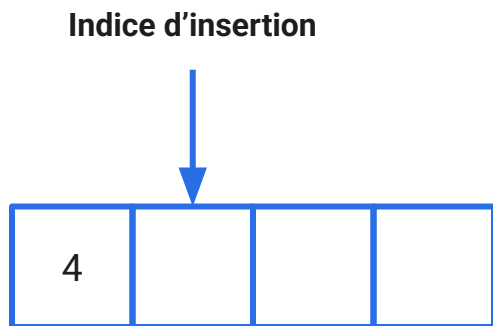
Le tableau extensible - Ajout d'un élément

Contrairement à un tableau classique, le tableau extensible n'est rempli que jusqu'à un indice donné, que l'on doit conserver.

Pour ajouter un élément (en considérant que l'on a la place), il faut:

1. Ajouter l'élément à l'indice d'insertion actuel
2. **Incrémenter l'indice d'insertion**

Au début, notre tableau est vide, l'indice d'insertion est 0.



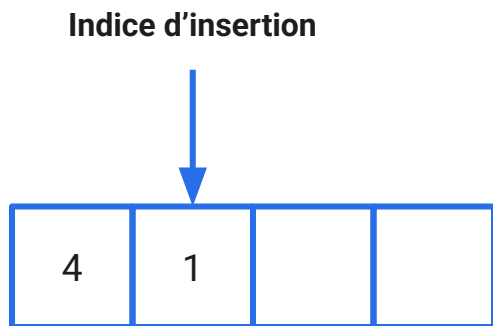
Le tableau extensible - Ajout d'un élément

Contrairement à un tableau classique, le tableau extensible n'est rempli que jusqu'à un indice donné, que l'on doit conserver.

Pour ajouter un élément (en considérant que l'on a la place), il faut:

1. **Ajouter l'élément à l'indice d'insertion actuel**
2. Incrémenter l'indice d'insertion

Au début, notre tableau est vide, l'indice d'insertion est 0.



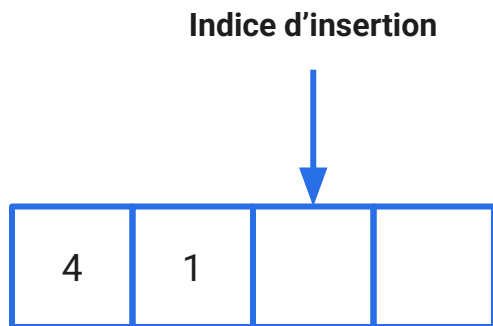
Le tableau extensible - Ajout d'un élément

Contrairement à un tableau classique, le tableau extensible n'est rempli que jusqu'à un indice donné, que l'on doit conserver.

Pour ajouter un élément (en considérant que l'on a la place), il faut:

1. Ajouter l'élément à l'indice d'insertion actuel
2. **Incrémenter l'indice d'insertion**

Au début, notre tableau est vide, l'indice d'insertion est 0.



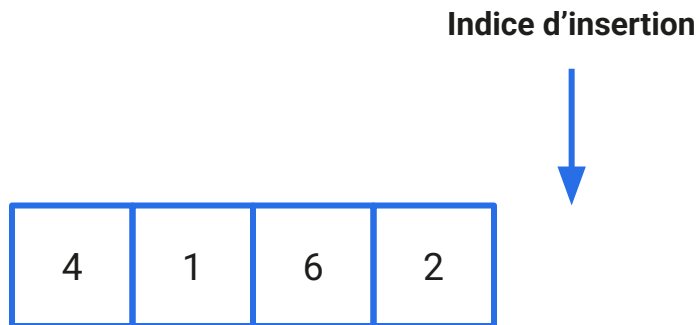
Le tableau extensible - Extension du tableau

À force d'ajouter des éléments, nous allons **remplir notre tableau** (notre indice d'insertion dépasse la taille du tableau). Il va falloir d'étendre.

Pour **l'étendre**, nous devons:

1. Créer un nouveau tableau deux fois plus grand
2. Recopier les anciens éléments dans le nouveau tableau

On peut alors faire l'insertion.



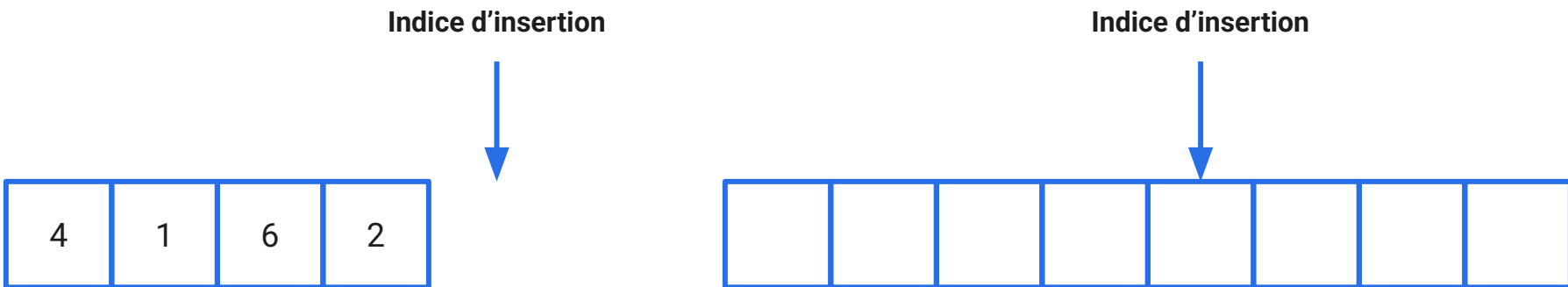
Le tableau extensible - Extension du tableau

À force d'ajouter des éléments, nous allons **remplir notre tableau** (notre indice d'insertion dépasse la taille du tableau). Il va falloir d'étendre.

Pour **l'étendre**, nous devons:

1. **Créer un nouveau tableau deux fois plus grand**
2. Recopier les anciens éléments dans le nouveau tableau

On peut alors faire l'insertion (on a conservé le même indice d'insertion).



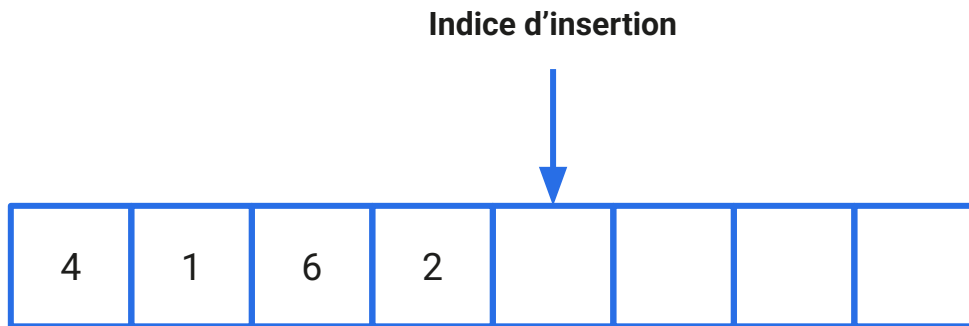
Le tableau extensible - Extension du tableau

À force d'ajouter des éléments, nous allons **remplir notre tableau** (notre indice d'insertion dépasse la taille du tableau). Il va falloir d'étendre.

Pour **l'étendre**, nous devons:

1. Créer un nouveau tableau deux fois plus grand
2. **Recopier les anciens éléments dans le nouveau tableau**

On peut alors faire l'insertion (on a conservé le même indice d'insertion).



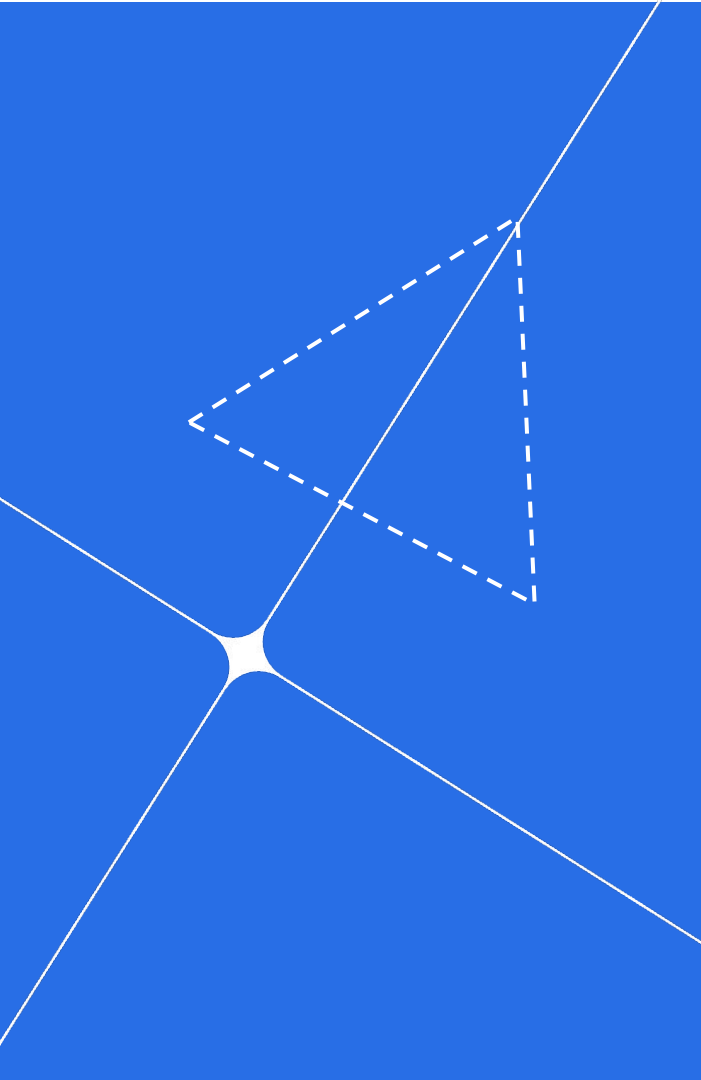
Le tableau extensible - Complexité

- L'extension est **linéaire** dans tous les cas (création du tableau et recopie)
- L'ajout d'un élément est **linéaire** dans le pire des cas (cas où il n'y a plus la place et il faut faire une extension)
 - Constant dans le meilleur des cas (pas d'extension)
 - Constant en moyenne (c.f. TP)
- L'accès à l'élément à l'indice i est **constant** (car c'est un tableau)

Le tableau extensible est utilisé en pratique grâce à la classe `ArrayList<E>` de Java (c.f. futur cours).

Même si l'ajout est de complexité constante en moyenne, elle peut créer de mauvaises surprises quand une extension à lieu.

D'où l'introduction des **listes chaînées**.



La liste chaînée

La liste chaînée

- **Idée:** Au lieu de créer plusieurs cases continues dans la mémoire pour les tableaux, nous allons séparer chaque case. Pour ce faire, chaque élément de notre liste chaînée connaîtra sa valeur ET la référence vers la case suivante.

Une liste chaînée est composée de nœuds. Chaque nœud contient:

- Une valeur
- Une référence vers le nœud suivant (null si dernier nœud)

Nous avons ici une structure récursive (chaque nœud référence un autre nœud). Les listes sont naturellement compatibles avec les fonctions récursives.

En général, on ne se rappelle que du premier élément de la liste.

Nous allons voir comment:

- Ajouter un élément à la liste
- Supprimer un élément
- Accéder à l'élément i de la liste

La liste chaînée - Ajout

Considérons que nous avons une variable `first` qui réfère le premier nœud. Au début, elle vaut `null`.

Pour ajouter un élément à la liste, nous devons:

1. Créer un nouveau nœud avec la bonne valeur (un monstre dans le TP) et référent le nœud suivant (`next`) comme étant `first`
2. Redéfinir `first` comme référent vers ce nouveau nœud

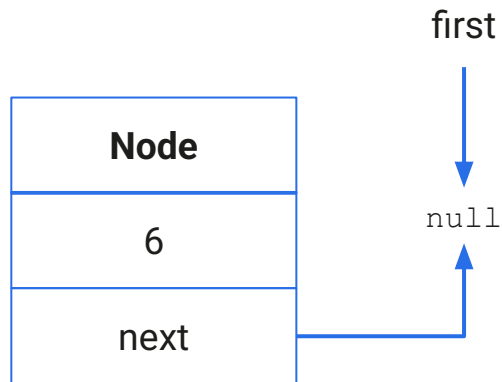


La liste chaînée - Ajout

Considérons que nous avons une variable `first` qui réfère le premier nœud. Au début, elle vaut `null`.

Pour ajouter un élément à la liste, nous devons:

1. **Créer un nouveau nœud avec la bonne valeur (un monstre dans le TP) et référent le nœud suivant (`next`) comme étant `first`**
2. Redéfinir `first` comme référent vers ce nouveau nœud

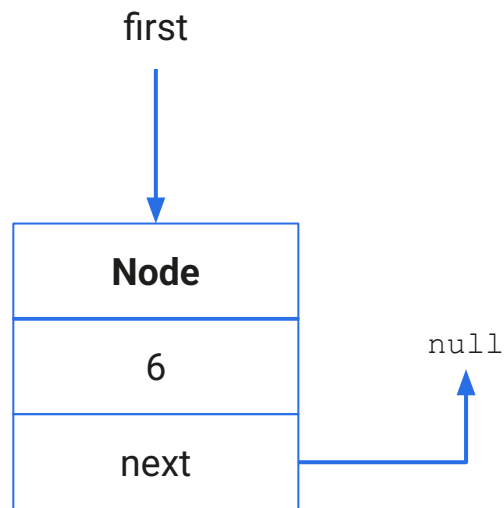


La liste chaînée - Ajout

Considérons que nous avons une variable `first` qui réfère le premier nœud. Au début, elle vaut `null`.

Pour ajouter un élément à la liste, nous devons:

1. Créer un nouveau nœud avec la bonne valeur (un monstre dans le TP) et réfèrent le nœud suivant (`next`) comme étant `first`
2. Redéfinir `first` comme référent vers ce nouveau nœud

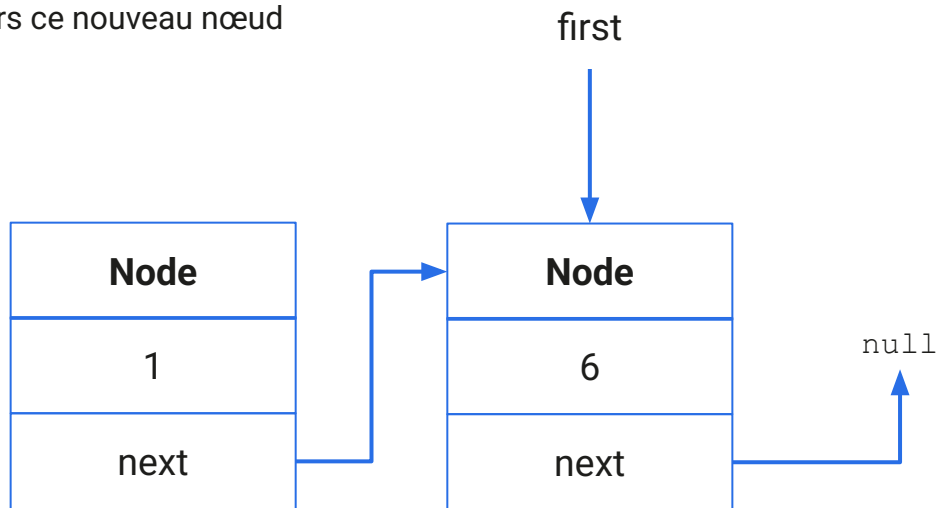


La liste chaînée - Ajout

Considérons que nous avons une variable `first` qui réfère le premier nœud. Au début, elle vaut `null`.

Pour ajouter un élément à la liste, nous devons:

1. **Créer un nouveau nœud avec la bonne valeur (un monstre dans le TP) et référer le nœud suivant (`next`) comme étant `first`**
2. Redéfinir `first` comme référent vers ce nouveau nœud

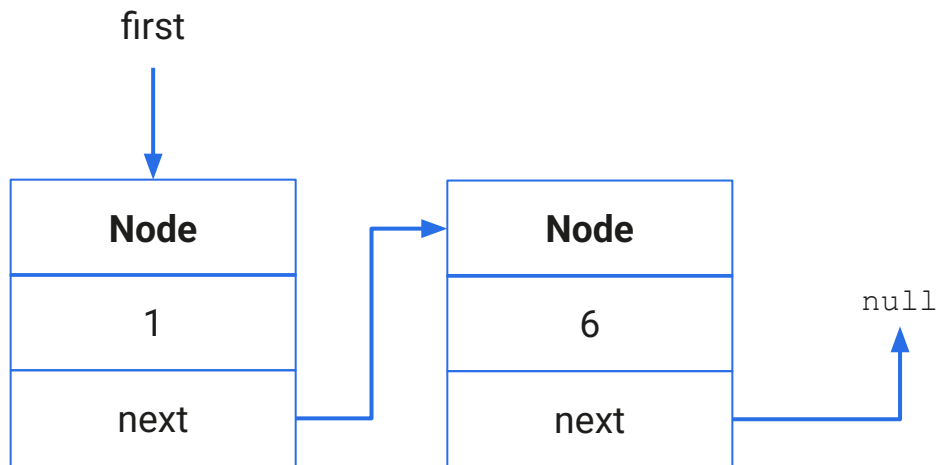


La liste chaînée - Ajout

Considérons que nous avons une variable `first` qui réfère le premier nœud. Au début, elle vaut `null`.

Pour ajouter un élément à la liste, nous devons:

1. Créer un nouveau nœud avec la bonne valeur (un monstre dans le TP) et référer le nœud suivant (`next`) comme étant `first`
2. Redéfinir `first` comme référent vers ce nouveau nœud

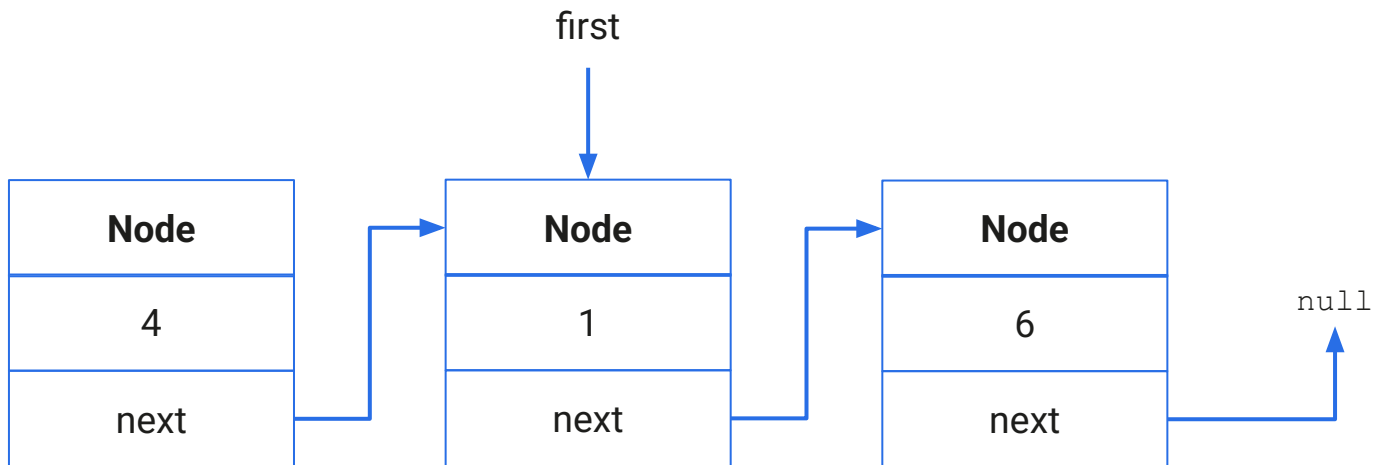


La liste chaînée - Ajout

Considérons que nous avons une variable `first` qui réfère le premier nœud. Au début, elle vaut `null`.

Pour ajouter un élément à la liste, nous devons:

1. **Créer un nouveau nœud avec la bonne valeur (un monstre dans le TP) et référer le nœud suivant (`next`) comme étant `first`**
2. Redéfinir `first` comme référent vers ce nouveau nœud

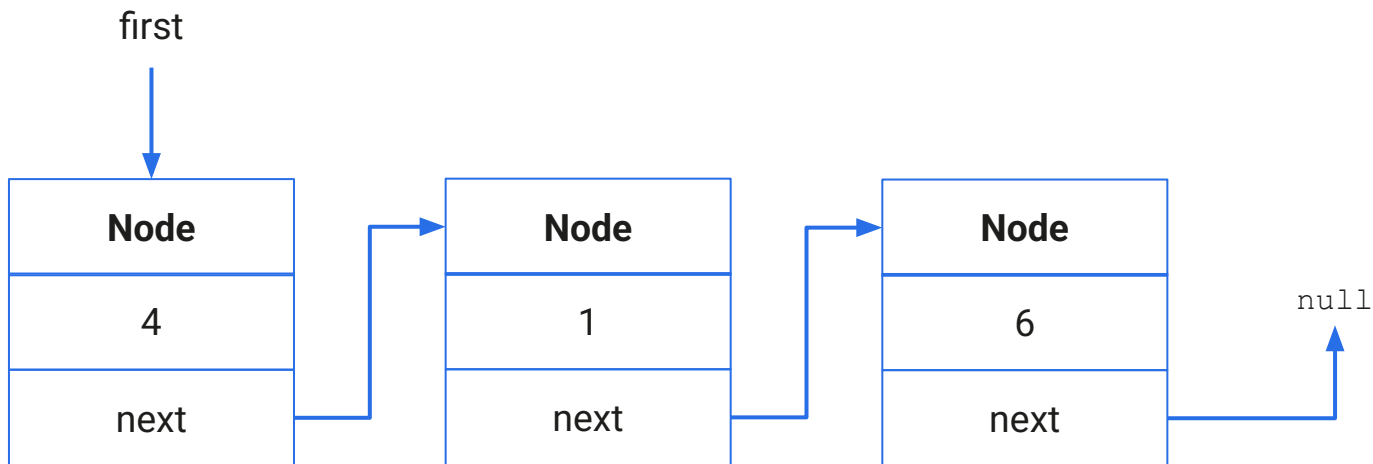


La liste chaînée - Ajout

Considérons que nous avons une variable `first` qui réfère le premier nœud. Au début, elle vaut `null`.

Pour ajouter un élément à la liste, nous devons:

1. Créer un nouveau nœud avec la bonne valeur (un monstre dans le TP) et référer le nœud suivant (`next`) comme étant `first`
2. Redéfinir `first` comme réfèrent vers ce nouveau nœud

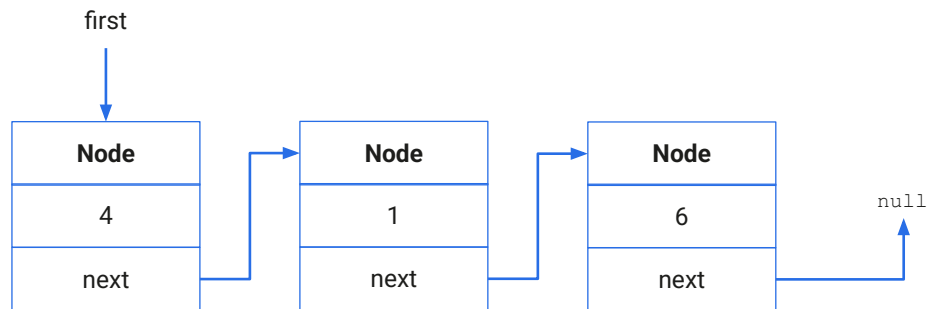


La liste chaînée - Accéder à un élément

Accéder à un élément dans une liste n'est pas aussi facile que pour un tableau. Nous devons sauter de nœud en nœud jusqu'à rencontrer l'élément voulu.

Par exemple, nous accédons à l'élément à la position N:

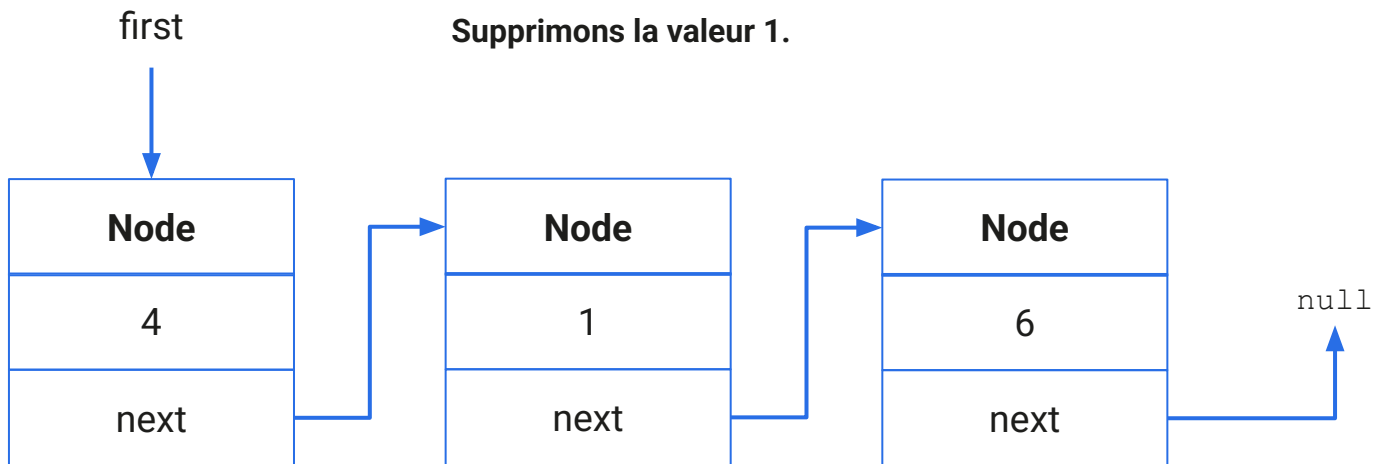
```
current_node = first
for(int i = 0; i < N; i++){
    current_node = current_node.next;
}
```



La liste chaînée - Retrait d'un élément

Pour retirer un élément de la liste, nous devons:

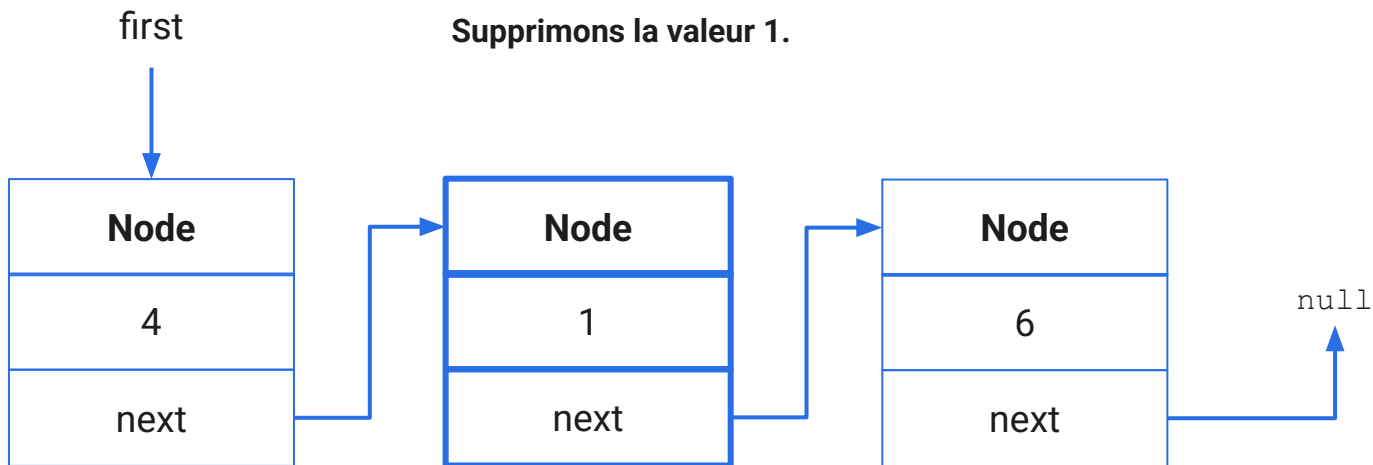
1. Le localiser dans la liste
2. Le supprimer
 - a. Si c'est le premier élément, on redéfinit `first` comme étant le deuxième élément
 - b. Sinon, on définit le `next` du nœud précédent comme étant le `next` du nœud à supprimer



La liste chaînée - Retrait d'un élément

Pour retirer un élément de la liste, nous devons:

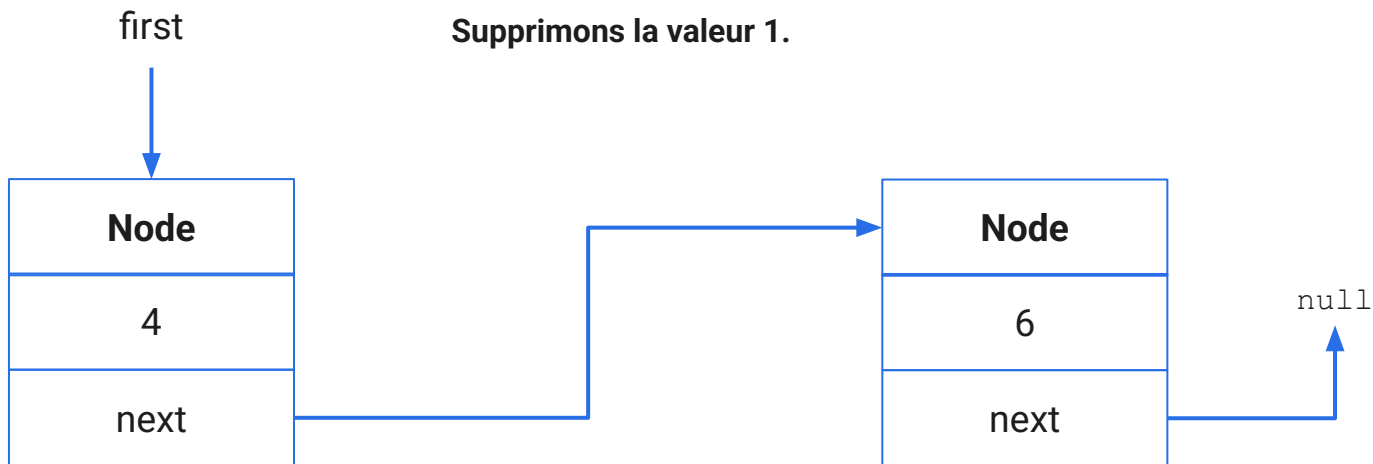
1. **Le localiser dans la liste**
2. **Le supprimer**
 - a. Si c'est le premier élément, on redéfinit `first` comme étant le deuxième élément
 - b. Sinon, on définit le `next` du nœud précédent comme étant le `next` du nœud à supprimer



La liste chaînée - Retrait d'un élément

Pour retirer un élément de la liste, nous devons:

1. Le localiser dans la liste
2. **Le supprimer**
 - a. Si c'est le premier élément, on redéfinit `first` comme étant le deuxième élément
 - b. Sinon, on définit le `next` du nœud précédent comme étant le `next` du nœud à supprimer

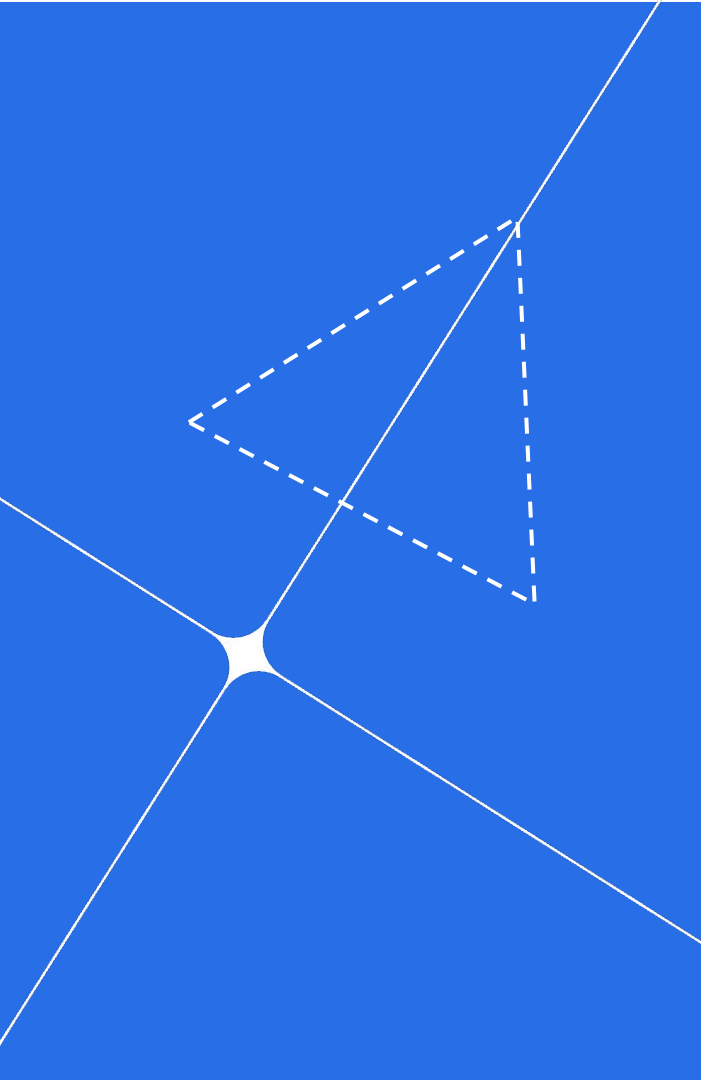


La liste chaînée - Complexité

- La complexité d'un ajout est **constante**
- Trouver un élément est **linéaire** dans le pire des cas (l'élément est à la fin, et il faut parcourir tous les nœuds)
- Supprimer un élément est **linéaire** dans le pire des cas (car il faut localiser l'élément)

Nous avons bien un coût d'ajout constant dans tous les cas. Par contre, l'accès à un élément est plus coûteux.

En Java, nous avons accès à la classe prédéfinie `LinkedList<E>` qui implémente les listes chaînées (c.f. futur cours).

A blue background with white geometric lines. A dashed white triangle is positioned in the upper left quadrant. A solid white line runs diagonally from the top left towards the bottom right, passing through the triangle. Another solid white line runs diagonally from the bottom left towards the top right, also passing through the triangle. These two lines intersect at a point within the triangle. A third solid white line runs diagonally from the bottom left towards the top right, passing through the intersection point of the other two lines.

En route vers le TP !