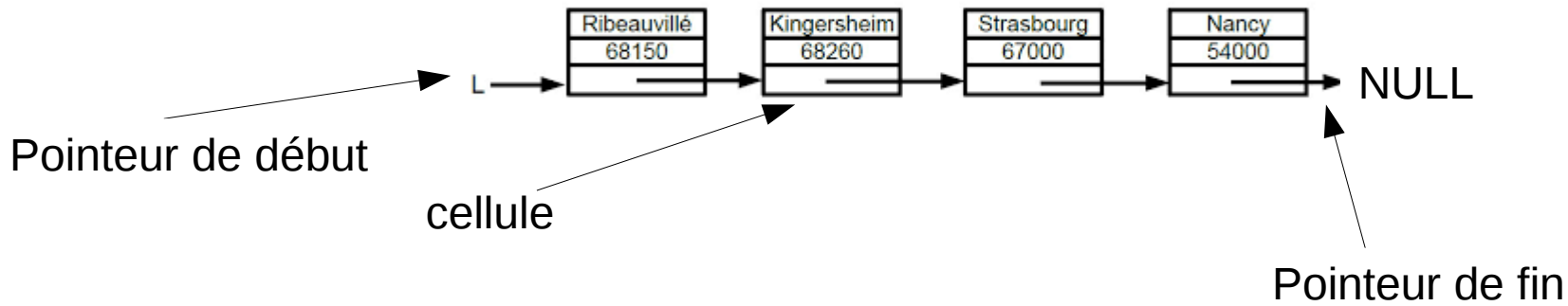


Première structures de données :
les listes simplement chaînées

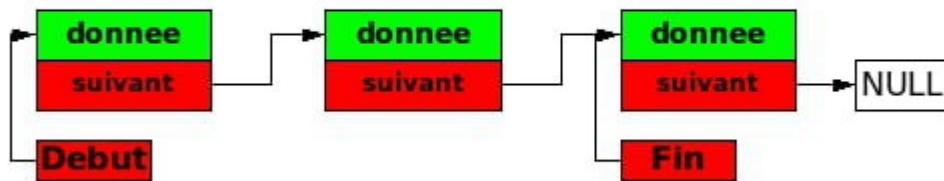
Réursion

- Nous avons vu la semaine dernière qu'il est possible de définir des méthodes récursives, c'est-à-dire des méthodes qui s'appellent elles-mêmes dans leurs corps.
- Il est également possible de définir une classe en l'utilisant elle-même :
 - soit en tant que type pour les méthodes : on l'a déjà vu dans les méthodes de la classe `Rect`,
 - soit en tant que type d'un ou plusieurs de ses champs.

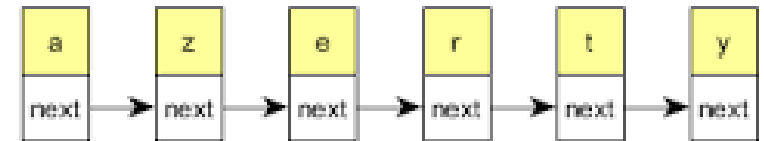
Une liste simplement chaînée



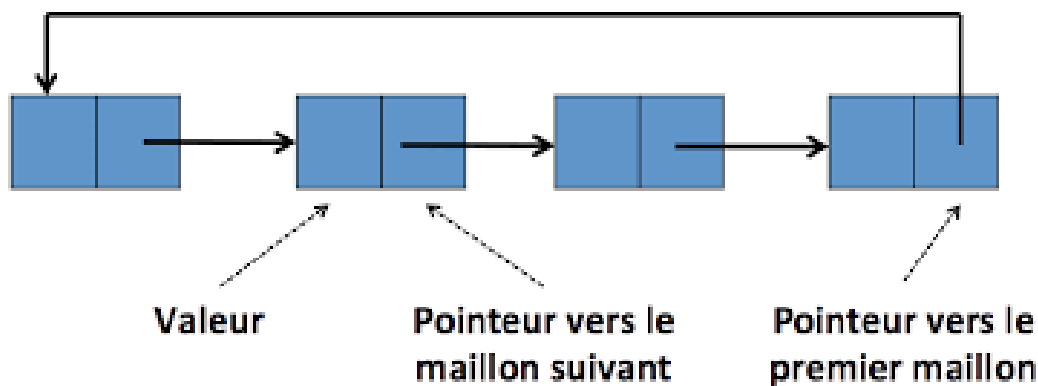
Avec un pointeur de queue... pas pareil



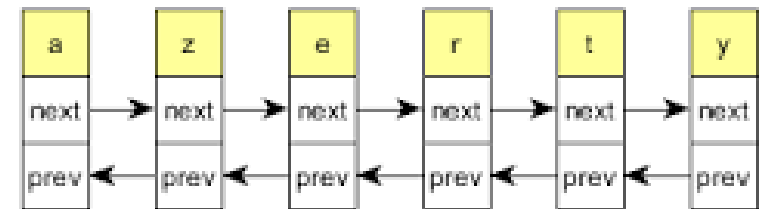
Liste simplement chaînée



Liste circulaire (pas pareil)



Liste doublement chaînée



```
class CelluleEntier{

    private int valeur;
    private CelluleEntier celluleSuivante;

    /** Creation d'une cellule contenant la valeur n. */
    CelluleEntier(int n){
        this.valeur = n;
        this.celluleSuivante = null;
    }
    /** Creation d'une cellule contenant la valeur n et ayant comme
        cellule suivante la cellule c. */
    CelluleEntier(int n, CelluleEntier c){
        this.valeur = n;
        this.celluleSuivante = c;
    }
}
```

```
/** Renvoie la valeur contenue dans la cellule. */  
public int valeur(){  
    return this.valeur;  
}
```

```
/** Renvoie la cellule suivante. */  
public CelluleEntier celluleSuivante(){  
    return this.celluleSuivante;  
}
```

```
}
```

Utilisation de la classe CelluleEntier

- On peut utiliser cette classe pour faire une suite de cellules, qu'on appelle *liste chaînée*
- Par exemple :

```
CelluleEntier c = new CelluleEntier(1, new CelluleEntier  
                                     (2, new CelluleEntier(3)))
```

- Mais on ne peut pas avoir de liste vide de cellules sur laquelle on pourrait tout de même appeler les méthodes qu'on a enfin d'écrire (comme la déterminer la longueur d'une liste, une méthode toString, etc.).

La classe ListeDEntiers

```
public class ListeDEntiers{
    CelluleEntier cellule;

    /** Liste vide. */
    ListeDEntiers(){
        this.cellule = null;
    }
    /** Liste d'un élément. */
    ListeDEntiers(int n){
        this.cellule = new CelluleEntier(n);
    }
    /** Construction d'une liste par
     * ajout d'un élément en tête d'une
     * liste.
     */
    ListeDEntiers(int n, ListeDEntiers l){
        this.cellule = new CelluleEntier(n,l.cellule);
    }
}
```

La classe ListeDEntiers

```
/** Renvoie la valeur de la tête de la liste. */  
public int tete(){  
    assert(this.cellule!=null);  
    return this.cellule.valeur();  
}  
/** Supprime la premier valeur de la liste */  
public void queue(){  
    assert(this.cellule!=null);  
    this.cellule = this.cellule.celluleSuivante();  
}  
/** Ajoute un entier en tête de liste */  
public void ajoute(int n){  
    this.cellule = new CelluleEntier(n, this.cellule);  
}  
/** Indique si la liste est vide */  
public boolean estVide(){  
    return cellule==null;  
}
```


Version récursive du calcul de la longueur

```
/** Renvoie la longueur de la liste version récursive. */  
private int longueur_aux(CelluleEntier c){  
    if(c==null)  
        return 0;  
    else  
        return 1 + longueur_aux(c.celluleSuivante());  
}  
public int longueur_rec(){  
    return longueur_aux(this.cellule);  
}
```

Autres méthodes

- version récursive toString;
- méthodes pour calculer la somme des valeurs de la liste :
 - méthode itérative,
 - méthode récursive;
- méthodes pour calculer la moyenne.

Exercices

- Recopier et compiler
- Ajouter une méthode main pour faire vos tests...
- Ajouter une méthode « toString ». Exemple [1;2;3]
- Ajouter une méthode calculant la moyenne des valeurs des cellules
- Ajouter une méthode permettant d'inverser une liste. [1;2;3]=>[3;2;1]. `ListEntier inverse(ListEntier l)`
- Ajouter une méthode permettant de concatener 2 listes. `Concat([1;2;3],[4;5;6])=> [1;2;3;4;5;6]`
- **DIFFICILE** : Faire une classe de listes doublement chaînées...

Pour plus tard dans l'année...

La classe ListeDEntiers

```
/** Renvoie la longueur de la liste */  
public int longueur(){  
    int longueur = 0;  
    CelluleEntier c = this.cellule;  
    while (c!=null) {  
        longueur = longueur + 1;  
        c = c.celluleSuivante();  
    }  
    return longueur;  
}
```

La classe ListeDEntiers

```
/** Renvoie une chaîne de caractères  
    représentant la liste. */  
public String toString(){  
    String s = "[";  
    CelluleEntier c = this.cellule;  
    while (c!=null) {  
        s = s + c.valeur();  
        c = c.celluleSuivante();  
        if (c!=null) s = s + ";";  
    }  
    s = s + "];"  
    return s;  
}  
}
```