

# Initiation à la programmation Java

## IP2 - Séance No 4

Yan Jurski

- Attention à ne pas vous laisser distancer, réalisez qu'on avance, et même si cela peut sembler simple à chaque étape, les combinaisons sont une vraie gymnastique !
- Lors de la séance de question/réponse de la semaine dernière j'ai pu constater qu'il y avait du sérieux (et c'est bien !) mais aussi quelques petites choses à consolider.
- Profitez de la semaine de pause du 22/02 pour vous mettre à jour, ou vous renforcer
- Constatez que ce cours suit un cycle facile-moyen-plus difficile, et enchaîne ensuite sur une autre notion pour laquelle ce qui précède est nécessaire.

**Bonne chance pour le contrôle Continu de Td de cette semaine**

Initiation à la programmation Java  
IP2 - Séance No 4  
Variations sur les liaisons d'objets  
(une introduction progressive aux listes)

Yan Jurski

# Variations sur les liaisons d'objets

- les attributs (qui sont des références) tissent des interconnexions, possiblement complexes entre les objets



- L'étude de ces structures va nous occuper tout le reste de l'année
  - listes chaînées (simples, doubles, circulaires, piles, files)
  - arbres binaires, avec ou sans liaison au père etc ...

# Exemple 1 - Une liaison simple déjà rencontrée

fichier : Cercle.java

```
public class Cercle{  
    private Point centre;  
    private int rayon;  
    Cercle (Point x, int d){  
        this.centre=x; // référence extérieure  
        this.rayon=d;  
    }  
}
```

- L'attribut `this.centre` établit une liaison d'un cercle vers un point
- Elle n'est pas réciproque (le point ne sait pas à qui il est lié)

## Exemple 2 - Liaison avec un leader



- Des individus se choisissent un leader commun
- Celui qui est considéré comme le plus sympathique
- A leur création les individus reçoivent un capital sympathie  $\in [0..100]$ .  
Le leader peut changer à cette occasion
- Si un individu change de capital, il challenge le leader
- Le capital du leader ne peut pas décroître (...disons ça...)

## Exemple 2 - Liaison avec un leader

### Fichier : Individu.java

```
public class Individu {  
    private static Individu leader=null; // relation partagée  
    private int capital;  
    private final String nom;  
    public Individu(String n){  
        this.nom=n;  
        Random r = new Random();  
        this.capital = r.nextInt(101);  
        challenge(); // à écrire  
    }  
}
```

## Exemple 2 - Liaison avec un leader

### Fichier : Individu.java

```
public class Individu {
    private static Individu leader=null; // relation partagée
    private int capital;
    private final String nom;
    public Individu(String n){
        this.nom=n;
        Random r = new Random();
        this.capital = r.nextInt(101);
        challenge();
    }
    public void change(int dx){ // dx : variation de popularité
        if ((this != Individu.leader) || (dx >0) ) {
            this.capital += dx; //rq : on ne se contraint pas à une limite à 100
            challenge();
        }
    }
}
```



## Fichier : Individu.java

```
public class Individu {  
    private static Individu leader=null; // relation partagée  
    private int capital;  
    private final String nom;  
    public Individu(String n){  
        this.nom=n;  
        Random r = new Random();  
        this.capital = r.nextInt(101);  
        if (Individu.leader==null) Individu.leader=this;  
        else challenge();  
    }  
    public void change(int dx){...}  
    private void challenge(){  
        if (Individu.leader==null) { Individu.leader=this; return; }  
        if (this.capital > Individu.leader.capital) Individu.leader = this;  
    }  
}
```

## Fichier : Individu.java - Avec tests

```
public class Individu {
    private static Individu leader=null; // relation partagée
    private int capital;
    private final String nom;
    public Individu(String n){...}
    public void change(int dx){...}
    private void challenge(){...}

    public static void main(String [] args){
        Individu [] population = new Individu[10]; // init. 10 objets nulls
        String nom="x";
        for (i=0; i<population.length; i++){
            population[i] = new Individu(nom+i);
        }
        System.out.println(leader.nom); // pourquoi les accès sont possibles ?
        Individu john=population[5]; // peu importe
        System.out.println(john.leader.nom); // ce n'est pas un attribut
            d'objet, mais on peut l'écrire ainsi par commodité
        System.out.println(leader.leader.leader.leader.nom); // vous suivez ?
    }
}
```

## Exemple 3 - Liaison d'attention



- Chacun focalise librement son attention vers un autre
- Le graphe résultant est différent de celui du leader

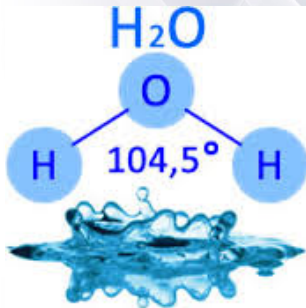
## Exemple 3 - Liaison d'attention

### Fichier : Individu.java

```
public class Individu {  
    private Individu focus=null; // propre à chacun  
    private final String nom;  
    public Individu(String n){ this.nom=n; }  
    public void changeFocus(Individu x){ this.focus=x; }  
    public static void main(String [] args){  
        Individu david=new Individu("david");  
        Individu syd=new Individu("syd");  
        Individu lenny=new Individu("lenny");  
        david.changeFocus(syd);  
        syd.changeFocus(david);  
        lenny.changeFocus(david);  
    }  
}
```

## Exemple 4 - Molécule d'eau

- chaque Hydrogène a une liaison possible
- les Oxygènes ont deux liaisons possibles



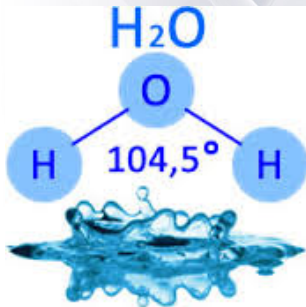
fichier : Hydrogène.java

```
public class Hydrogène {  
    private Oxygène x=null; // par défaut  
    Hydrogène (){} // construction par défaut  
}
```

fichier : Oxygène.java

```
public class Oxygène {  
    private Hydrogène h1=null, h2=null;  
    private static final double angle=104,5;  
    Oxygène (){}  
}
```

## Exemple 4 - Molécule d'eau



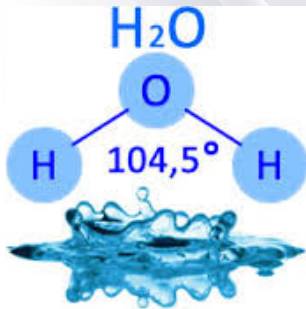
fichier : Hydrogène.java

```
public class Hydrogène {  
    private Oxygène x=null; // par défaut  
    Hydrogène (){} // construction par défaut  
    public boolean adoptLiaison(Oxygène o){  
        if (x==null) {x=o; return true;}  
        return false;  
    }  
}
```

fichier : Oxygène.java

```
public class Oxygène {  
    private Hydrogène h1=null, h2=null;  
    private static final double angle=104,5;  
    Oxygène (){}  
    public boolean adoptLiaison(Hydrogène h){  
        if (h1==null) {h1=h; return true;}  
        if (h2==null) {h2=h; return true;}  
        return false;  
    }  
}
```

Vers une généralisation



## fichier : Atome.java

```
public interface Atome {  
    public boolean adoptLiaison(Atome a);  
}
```

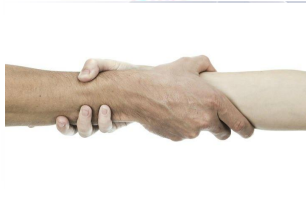
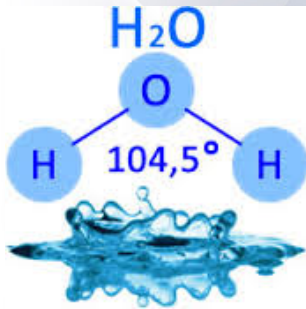
## fichier : Hydrogène.java

```
public class Hydrogène implements Atome {  
    private Atome x=null;  
    public boolean adoptLiaison(Atome a){  
        if (x==null) {x=a; return true;}  
        return false;  
    }  
}
```

## fichier : Oxygène.java

```
public class Oxygène implements Atome {  
    private Atome x1=null, x2=null;  
    public boolean adoptLiaison(Atome a){  
        if (x1==null) {x1=a; return true;}  
        if (x2==null) {x2=a; return true;}  
        return false;  
    }  
}
```

## Exemple 4 - Molécule d'eau



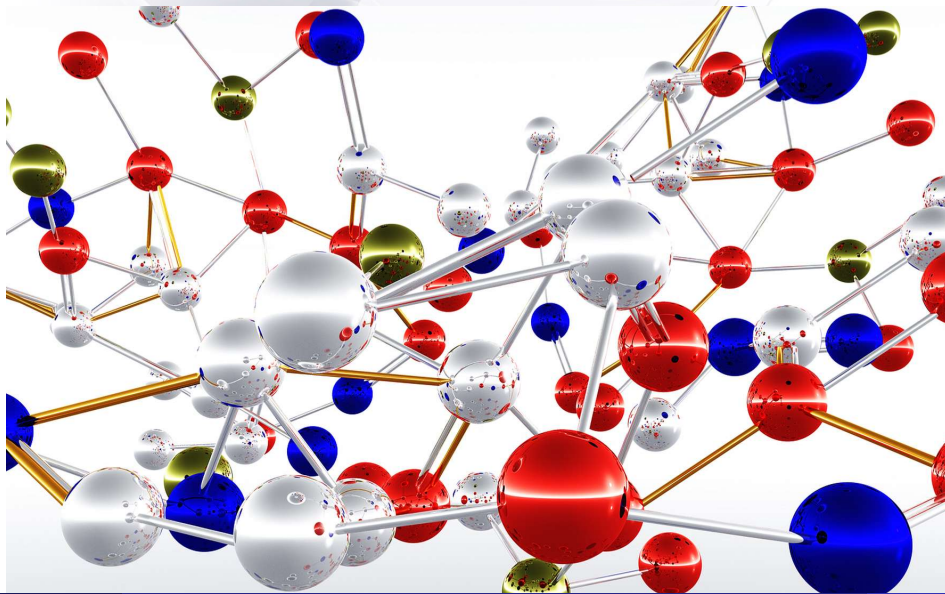
fichier : Test.java

```
public class Test {  
    public static void main(String [] args){  
        Atome a1=new Hydrogène();  
        Atome a2=new Hydrogène();  
        Atome a3=new Oxygène();  
        // établissement des liaisons  
        a1.adoptLiaison(a3);  
        a2.adoptLiaison(a3);  
        a3.adoptLiaison(a1);  
        a3.adoptLiaison(a2);  
    }  
}
```

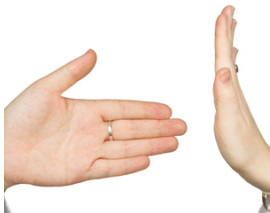


## Exemple 4 - Molécule d'eau

On peut étendre à toutes sortes d'atomes - Analyse du réseau difficile en général



## Exemple 4 - Molécule d'eau



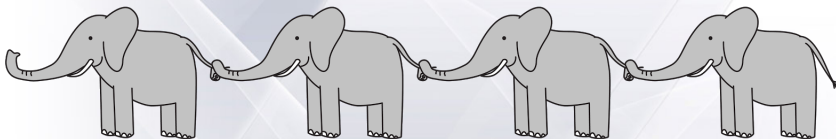
Il est très fréquent de faire des contresens !

```
public class Test {  
    public static void main(String [] args){  
        Atome a1=new Hydrogène();  
        Atome a2=new Hydrogène();  
        Atome a3=new Oxygène();  
        // établissement des liaisons  
        a1.adoptLiaison(a3);  
        a3.adoptLiaison(a1);  
        a2.adoptLiaison(a1); // oups !  
        a3.adoptLiaison(a3); // sic !  
    }  
}
```

Sans être des erreurs de syntaxe

# Abstraction et outils - Listes chaînées

Difficulté progressive, intéressantes, assez simple



# Abstraction et outils

Nous implémenterons nous même un sous ensemble d'une bibliothèque Java

## Fichier ListIP2.java – écrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(E x); // Appends the element to the end
    void add(int index, E x); // Inserts the element at the position
    void clear(); // Removes all of the elements from this list
    boolean contains(E x); // Returns true if the list contains the element
    E get(int index); // Returns the element at the specified position
    int indexOf(E x); // Returns the index of the first occurrence of the
        element, or -1 if not there
    boolean isEmpty(); // Returns true if this list contains no elements
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    E set(int index, E x); // Replaces the element at a position
    int size(); // Returns the number of elements in this list
}
```

# Distinction du tout, du liant, et du contenu

## Exemples :

- le troupeau, la trompe, l'éléphant
- le groupe, les mains, l'individu
- le train, l'attelage, le wagon
- le système de canalisation, les pièces de raccords, le tuyau

## Distinctions sémantiques :

- Le tout
  - peut être vide
  - il détermine un point d'entrée
- Le liant peut être
  - mono directionnel
  - multi directionnel
- Le contenu est secondaire du point de vue de la structure générale

# Distinction du tout, du liant, et du contenu

## Exemples :

- le troupeau, la trompe, l'éléphant
- le groupe, les mains, l'individu
- le train, l'attelage, le wagon
- le système de canalisation, les pièces de raccords, le tuyau

## Distinctions sémantiques :

- Le tout  $\Rightarrow$  une classe **MaListe implements ListIP2**
  - peut être vide
  - il détermine un point d'entrée  $\Rightarrow$  dans le cas des éléphants : à droite
- Le liant peut être  $\Rightarrow$  une classe **Cellule**
  - mono directionnel
  - multi directionnel
- Le contenu est secondaire du point de vue de la structure générale  $\Rightarrow$  une quelconque classe **Cellule**

# Modèle à 3 classes

## Fichier E.java - le contenu

```
public class E{  
    // peu importe  
}
```

## Fichier Cellule.java - rôle auxiliaire, précise la nature des liaisons

```
public class Cellule{  
    private E content;  
    private Cellule next;  
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{  
    private Cellule first;  
}
```

## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(E x); // Appends the element to the end
    void add(int index, E x); // Inserts the element at the position
    void clear(); // Removes all of the elements from this list
    boolean contains(E x); // Returns true if the list contains the element
    E get(int index); // Returns the element at the specified position
    int indexOf(E x); // Returns the index of the first occurrence of the
        element, or -1 if not there
    boolean isEmpty(); // Returns true if this list contains no elements
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    E set(int index, E x); // Replaces the element at a position
    int size(); // Returns the number of elements in this list
}
```



## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public MaList(){
        this.first=null;
    }
    public boolean isEmpty() {
        return ( this.first==null );
    }
}
```

## Fichier Cellule.java - rôle auxiliaire, précise la nature des liaisons

```
public class Cellule{
    private E content;
    private Cellule next;
}
```

## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(E x); // Appends the element to the end
    void add(int index, E x); // Inserts the element at the position
    void clear(); // Removes all of the elements from this list
    boolean contains(E x); // Returns true if the list contains the element
    E get(int index); // Returns the element at the specified position
    int indexOf(E x); // Returns the index of the first occurrence of the
        element, or -1 if not there
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    E set(int index, E x); // Replaces the element at a position
    int size(); // Returns the number of elements in this list
    ...
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public ListIP2(){
        this.first=null;
    }
    public boolean isEmpty() {
        return ( this.first==null );
    }
    public void clear() {
        this.first=null;
    }
}
```

## Fichier Cellule.java - rôle auxiliaire, précise la nature des liaisons

```
public class Cellule{
    private E content;
    private Cellule next;
}
```

## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(E x); // Appends the element to the end
    void add(int index, E x); // Inserts the element at the position
    boolean contains(E x); // Returns true if the list contains the element
    E get(int index); // Returns the element at the specified position
    int indexOf(E x); // Returns the index of the first occurrence of the
        element, or -1 if not there
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    E set(int index, E x); // Replaces the element at a position
    int size(); // Returns the number of elements in this list
    ...
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    ...
    public void add(E x){ // ajoute en fin
        if (this.isEmpty()) first=new Cellule(x);
        else first.add(x); // Déléguer = responsabiliser
    } // par convenance on a utilisé le même nom add dans MaList et Cellule
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule(E x){
        this.content=x;
        next=null;
    }
    public void add(E x){
        Cellule tmp=this;
        while(tmp.next != null) tmp=tmp.next;
        tmp.next=new Cellule(x);
    }
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    ...
    public void add(E x){ // ajoute en fin
        if (this.isEmpty()) first=new Cellule(x);
        else first.add(x); // Déléguer = responsabiliser
    } // par convenance on a utilisé le même nom add dans MaList et Cellule
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule(E x){
        this.content=x;
        next=null;
    }
    public void add(E x){
        Cellule tmp=this; // attention à bien vous représenter ce que font
        while(tmp.next != null) tmp=tmp.next; // ces quelques lignes
        tmp.next=new Cellule(x);
    }
}
```

## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(int index, E x); // Inserts the element at the position
    boolean contains(E x); // Returns true if the list contains the element
    E get(int index); // Returns the element at the specified position
    int indexOf(E x); // Returns the index of the first occurrence of the
        element, or -1 if not there
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    E set(int index, E x); // Replaces the element at a position
    int size(); // Returns the number of elements in this list
    ...
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public boolean contains(E x){
        if (this.isEmpty()) return false;
        else return first.contains(x); // on responsabilise !
    }
    ...
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    ...
    public boolean contains(E x){
        Cellule tmp = this;
        while ( (tmp != null) && (tmp.content != x ) ) { // ordre des tests !
            tmp=tmp.next;
        }
        if (tmp==null) return false; // ordre des tests !
        return true; // par déduction
    }
}
```



## Fichier ListIP2.java – Ecrire pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(int index, E x); // Inserts the element at the position
    E get(int index); // Returns the element at the specified position
    int indexOf(E x); // Returns the index of the first occurrence of the
        element, or -1 if not there
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    E set(int index, E x); // Replaces the element at a position
    int size(); // Returns the number of elements in this list
    ...
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    ...
    public E get(int index){
        if ( this.isEmpty() || index < 0 ) return null;
        else return first.get(index); // on responsabilise
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    ...
    public E get(int index){ // les index commencent à 0
        Cellule tmp = this;
        while ( (index!=0) && (tmp!=null)) {
            tmp=tmp.next;
            index--;
        }
        if (tmp==null) return null;
        return tmp.content;
    }
}
```

## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(int index, E x); // Inserts the element at the position
    int indexOf(E x); // Returns the index of the first occurrence of the
        element, or -1 if not there
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    E set(int index, E x); // Replaces the element at a position
    int size(); // Returns the number of elements in this list
    ...
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public int indexOf(E x){
        if (this.isEmpty()) return -1;
        else return first.indexOf(x); // on responsabilise
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public int indexOf(E x){
        int rep=0;
        Cellule tmp = this;
        while ( (tmp != null) && (tmp.content != x) ){
            tmp=tmp.next;
            rep++;
        }
        if (tmp==null) return -1;
        else return rep;
    }
}
```

## Fichier ListIP2.java – Ecrire pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(int index, E x); // Inserts the element at the position
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    E set(int index, E x); // Replaces the element at a position
    int size(); // Returns the number of elements in this list
    ...
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public int size(){
        if (this.isEmpty()) return 0;
        else return first.size(); // on responsabilise
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public int size(){
        int rep=0;
        Cellule tmp=this;
        while ( tmp != null ){
            tmp=tmp.next;
            rep++;
        }
        return rep;
    }
}
```

## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(int index, E x); // Inserts the element at the position
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    E set(int index, E x); // Replaces the element at a position. Returns
        the old value if done, or null otherwise
    ...
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public E set(int index, E x){
        if (this.isEmpty() || index <0 ) return null; // aucun remplacement
        else return first.set(index,x); // on responsabilise
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public E set(int index, E x){
        Cellule tmp=this;
        while ( (tmp != null) && (index!=0) ){
            tmp=tmp.next;
            index--;
        }
        if (tmp==null) return null;
        E old=tmp.content;
        tmp.content=x;
        return old;
    }
}
```



## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(int index, E x); // Inserts the element at the position
    int lastIndexOf(E x); // see indexOf
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    ...
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public int lastIndexOf(E x){
        if (this.isEmpty()) return -1;
        return first.lastIndexOf(E x);
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public int lastIndexOf(E x){
        int rep=-1;
        int i=0;
        Cellule tmp=this;
        while (tmp !=null){
            if (tmp.content==x) rep=i;
            tmp=tmp.next;
            i++;
        }
        return rep;
    }
}
```

## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    void add(int index, E x); // Inserts the element at the position
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    ...
}
```

Les difficultés commencent

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public void add(int index, E x){
        if (index == 0) first=new Cellule(x,first); // nouveau constructeur
        else if (this.isEmpty()) // ??? faut il remettre en cause le void ?
            // on va choisir d'être robuste : d'interpréter les cas impossibles
        else
            // à compléter
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule(E e){ content=e; next=null; }
    public Cellule(E e, Cellule c){
        this.content=e;
        this.next=c;
    }
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public void add(int index, E x){
        if (index == 0) first=new Cellule(x,first); // nouveau constructeur
        else if (this.isEmpty()) first=new Cellule(x);
        // on va choisir d'être robuste : d'interpréter les cas impossibles
        else
            // à compléter
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule(E e){ content=e; next=null; }
    public Cellule(E e, Cellule c){
        this.content=e;
        this.next=c;
    }
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public void add(int index, E x){
        if (index == 0) first=new Cellule(x,first); // nouveau constructeur
        else if (isEmpty()) first=new Cellule(x,null); // c'est équivalent
        // on va choisir d'être robuste : d'interpréter les cas impossibles
        else
            // à compléter
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule(E e){ content=e; next=null; }
    public Cellule(E e, Cellule c){
        this.content=e;
        this.next=c;
    }
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public void add(int index, E x){
        if (index == 0) first=new Cellule(x,first); // nouveau constructeur
        else if (isEmpty()) first=new Cellule(x,first); // c'est équivalent
        // on va choisir d'être robuste : d'interpréter les cas impossibles
        else
            // à compléter
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule(E e){ content=e; next=null; }
    public Cellule(E e, Cellule c){
        this.content=e;
        this.next=c;
    }
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public void add(int index, E x){
        if ( (index == 0) || (first == null) ) first=new Cellule(x,first);
        // robustesse : cas des index négatifs ?
        else
            // à compléter
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule(E e, Cellule c){
        this.content=e;
        this.next=c;
    }
}
```



## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public void add(int index, E x){
        if ( (index <= 0) || (first == null) ) first=new Cellule(x,first);
        else
            // à compléter
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule(E e, Cellule c){
        this.content=e;
        this.next=c;
    }
}
```

## Fichier MaList.java

```
public class MaList implements ListIP2{
    public void add(int index, E x){
        if ( (index <= 0) || (first == null) ) first=new Cellule(x,first);
        else first.add(index,x); // avec index >=1
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule(E e, Cellule c){
        this.content=e;
        this.next=c;
    }
    public void add(int index, E x){ // précondition : index est >=1
        Cellule tmp=this;
        while ( (index !=1 ) && (tmp.next!=null) ){
            tmp=tmp.next; index--;
        } // tmp n'est jamais null !
        tmp.next=new Cellule(x,tmp.next); // capture bien tous les cas
    }
}
```

## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    E remove(int index); // Removes the element at the specified position
    boolean remove(E x); // Removes the first occurrence of the element
    ...
}
```

- La cellule précédente est concernée
- Et first peut changer

## Fichier MaList.java

```
public class MaList implements ListIP2{
    private Cellule first;
    public E remove(int index){
        if (this.isEmpty() || index<0) return null;
        if (index==0) {
            Cellule old_first = first;
            first=first.getNext(); // accesseur nécessaire
            return old_first.getContent(); // accesseur nécessaire
        }
        else return first.remove(index);
        // l'état de la mémoire est alors interessant, liaison perdue ?
    }
}
```

## Fichier Cellule.java

```
public class Cellule{
    private E content;
    private Cellule next;
    public Cellule getNext(){ return this.next; }
    public E getContent(){ return this.content; }
    ...
}
```

## Fichier Cellule.java

```
public class Cellule{
    public Cellule getNext(){ return this.next; }
    public E getContent(){ return this.content; }
    public Cellule remove(int index) { // ici index >=1
        Cellule tmp=this;
        while (index != 1 && tmp.next!=null){ // on cherche celui d'avant
            tmp=tmp.next;
            index--;
        }
        if (tmp.next==null) return null;
        else {
            Cellule old=tmp.next;
            tmp.next=old.next;
            return old.content;
            // état de la mémoire, que devient old ? (mecanisme nettoyage java)
        }
    }
}
```

## Fichier ListIP2.java – Ecrite pour des éléments de type E

```
public interface ListIP2 { // position starts at 0
    boolean remove(E x); // Removes the first occurrence of the element
    ...
}
```

- La cellule précédente est concernée
- Pensez que first peut changer
- Laisse en exercice ...

Notez que :

- Semaine du 22 février : vacances pour le L1 Info
- Semaine de la rentrée (1er mars) récursion ... (un gros morceau)
- Samedi 20 mars : partiel d'IP2 et de MI2