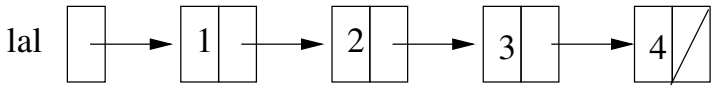
Corrigé des exercices sur les listes (suite)

Exercice 1.2.1 *travail sur une liste*

Cet exercice reprend la classe `Liste` vue en cours.
Nous allons travailler sur la liste obtenue au moyen du code java suivant :

```
ListeIter lal = new ListeIter();
lal.ajouterAuDebut(4);
lal.ajouterAuDebut(3);
lal.ajouterAuDebut(2);
lal.ajouterAuDebut(1);
```

1. dessinez la liste `lal`.



2. définissez une liste contenant les mêmes valeurs dans le même ordre en utilisant une seule fois le constructeur et en insérant les éléments au moyen de `ajouerALaFin`.

```
ListeIter lal = new ListeIter();
lal.ajouterALaFin(1);
lal.ajouterALaFin(2);
lal.ajouterALaFin(3);
lal.ajouterALaFin(4);
```

3. par quel code Java utilisant la liste `lal` peut-on désigner l'élément 1 de la liste ?
   `lal.getPremier().getValeur()`
4. par quel code Java utilisant la liste `lal` peut-on désigner l'élément 2 de la liste ?
   `lal.getPremier().getSuivant().getValeur()`
5. par quel code Java utilisant la liste `lal` peut-on désigner l'élément 3 de la liste ?
   `lal.getPremier().getSuivant().getSuivant().getValeur()`
6. par quel code Java utilisant la liste `lal` désigner la queue de liste qui ne contient que deux éléments ?
   `lal.getPremier().getSuivant().getSuivant()`
7. par quel code java peut-on remplacer l'entier 3 par 5, dans la liste ?
   Pour avoir une liste avec les mêmes valeurs sauf que le 3 est remplacé par 5, il y a deux façons de faire :
   - **garder les mêmes maillons et changer la valeur de la variable `element` du troisième maillon**

– créer un nouveau maillon avec l'élément 5 et l'insérer dans la liste entre le 2 et le 4.
Les deux possibilités peuvent être envisagées pour des applications réelles.

Solution 1 :

```
lal.getPremier().getSuivant().getSuivant().getSuivant().setValeur(5);
```

Solution 2 :

```
lal.getPremier().getSuivant().setSuivant(new ElementListe(5,
    lal.getPremier().getSuivant().getSuivant().getSuivant()))
```

Voici un programme qui permet de tester ce code.

```
class Exo22_1{
    public static void main(String[] args){
        ListeIter lal = new ListeIter();
        lal.ajouterALaFin(1);
        lal.ajouterALaFin(2);
        lal.ajouterALaFin(3);
        lal.ajouterALaFin(4);
        ElementListe p = lal.getPremier();
        while (p != null){
            Terminal.ecrireString("␣␣"+p.getValeur());
            p = p.getSuivant();
        }
        Terminal.sautDeLigne();
        Terminal.ecrireIntln(lal.getPremier().getValeur());
        Terminal.ecrireIntln(lal.getPremier().getSuivant().getValeur());
        Terminal.ecrireIntln(lal.getPremier().getSuivant().getSuivant().getValeur());
        lal.getPremier().getSuivant().setSuivant(new ElementListe(5,
                lal.getPremier().getSuivant().getSuivant().getSuivant()));
        p = lal.getPremier();
        while (p != null){
            Terminal.ecrireString("␣␣"+p.getValeur());
            p = p.getSuivant();
        }
        Terminal.sautDeLigne();
    }
}
```

Exercice 1.2.2 *insertion d'élément*

Question 1

Notre but est d'écrire une fonction qui ajoute un élément à un rang donné d'une liste. Nous avons déjà vu l'insertion en début et en fin de liste. Avec l'insertion à un rang donné, nous complétons la panoplie de méthodes permettant d'ajouter un élément à une liste. Procédons par étape.

L'algorithme consiste à avancer dans la liste jusqu'au bon endroit. Là, il faut insérer la nouvelle cellule. Il faut que la cellule précédente pointe sur elle et qu'elle pointe sur l'élément suivant.

Voyons d'abord un exemple.

Supposons que la liste contient 1, 2, 3, 4 et qu'on veut insérer 5 en position 3. Dessinez la représentation de l'état initial. Faut-il une variable auxiliaire ? Si oui, comment doit-elle être initialisée ? Si oui toujours, ajoutez-la à votre représentation.

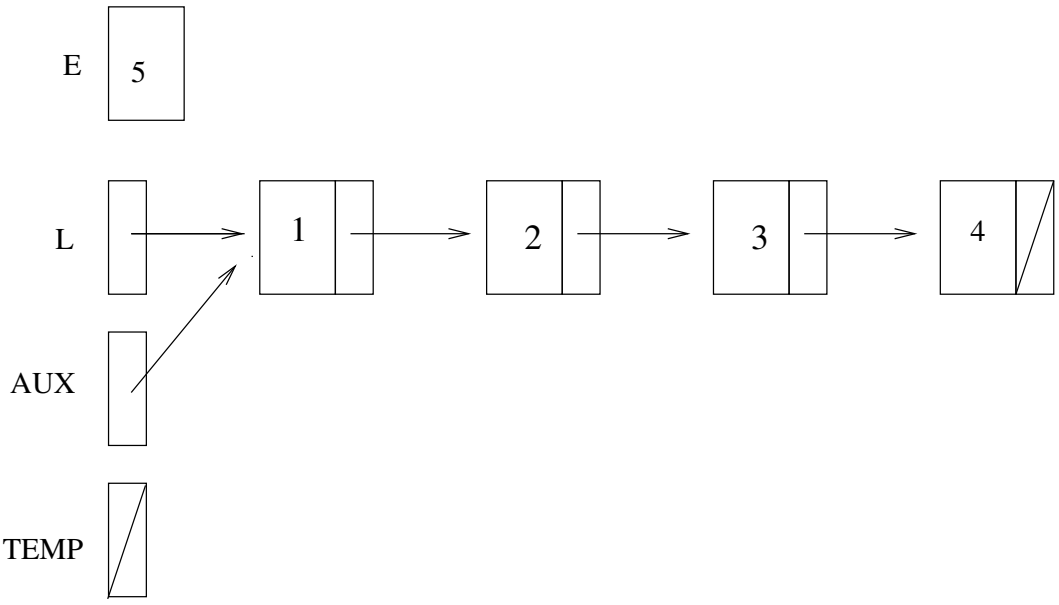
Dessinez ensuite tous les états intermédiaires par lesquels il faut passer pour réaliser l'opération. Faites bien un dessin séparé pour chaque état.

Essayez de décrire le changement d'état par une instruction Ada (ou une portion d'instruction comme nous l'avons fait dans les rappels sur les listes). Cette instruction doit être une affectation ou un appel de méthode du genre :

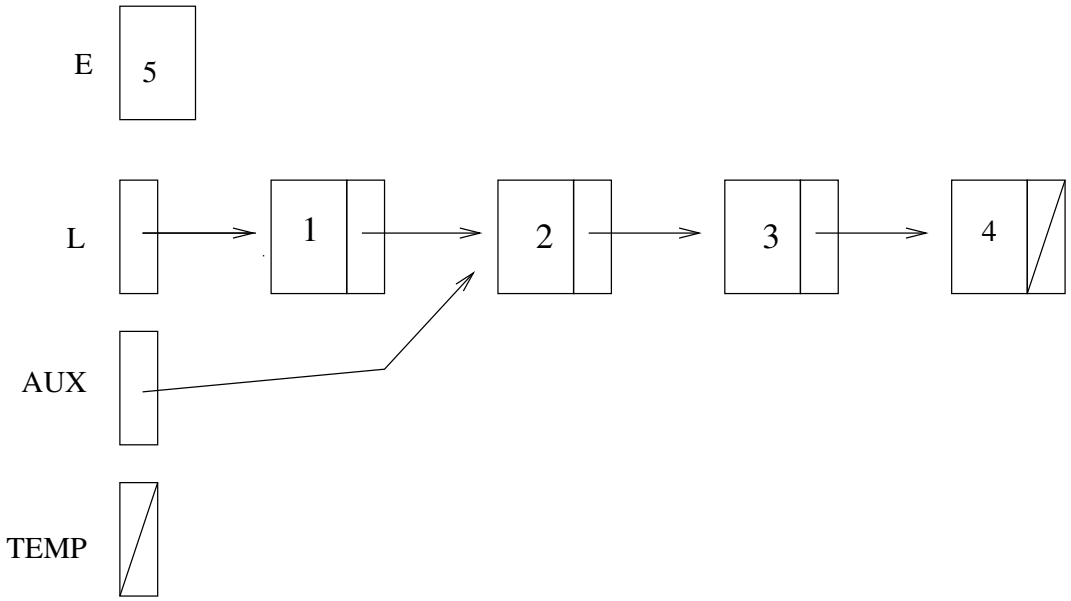
- `l1 = l2;`
- `l=l.getSuivant();`
- `l1= new ElementListe(x,l2);`
- `l1.setSuivant(l2);`

Il faut une variable auxiliaire (`aux`) pour parcourir la liste sans perdre le début. De plus, on va utiliser une deuxième variable auxiliaire (`temp`) pour pointer provisoirement sur le nouveau maillon créé. Ceci n'est pas nécessaire, mais rend le cheminement plus simple à comprendre.

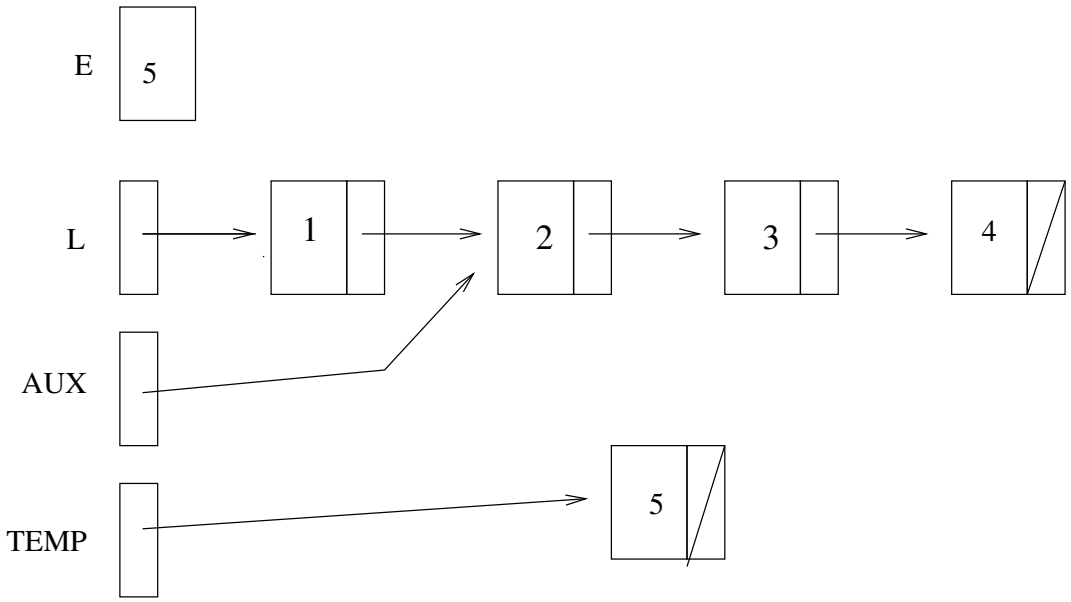
```
aux = L.getPremier();
```



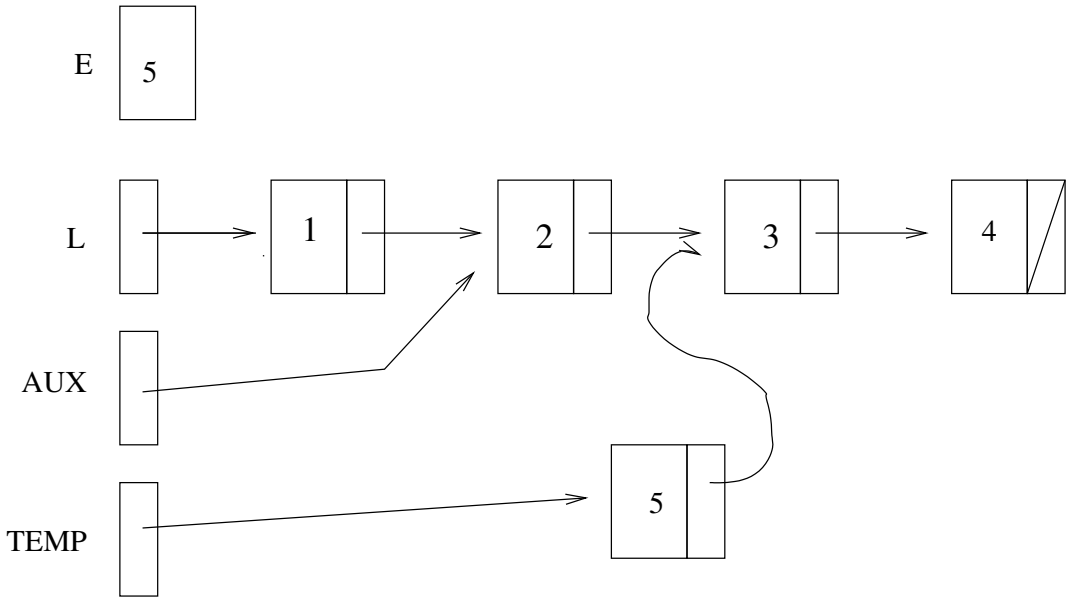

```
aux = aux.getSuivant();
```



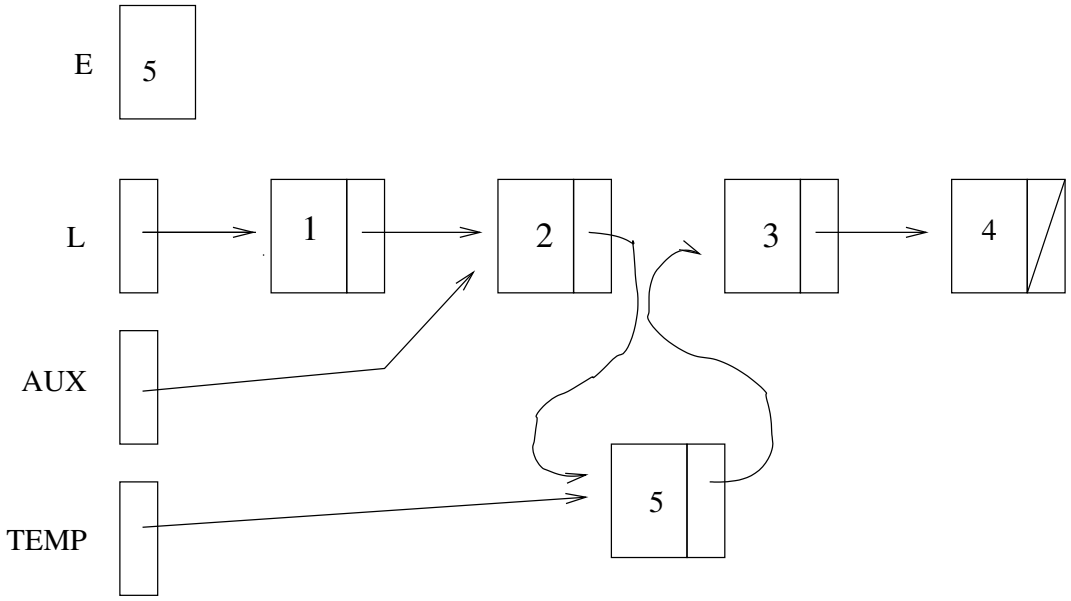

```
temp := new ElementListe(E);
```




```
temp.setSuivant(aux.getSuivant());
```




```
aux.setSuivant(temp);
```



Ceci est bien le résultat demandé.

Question 2

Ecrivez une procédure qui fonctionne dans tous les cas simples (comme l'exemple étudié). Dans cette première version, on traitera pas les cas particuliers ni les problèmes qui peuvent arriver. Testez votre procédure.

En gros, on doit retrouver les différentes affectations de l'exemple, organisées avec des structures de contrôle (boucles, if). Eventuellement, il faudra de nouvelles variables pour décrire les conditions de ces structures de contrôle.

```
class Exo22_2{
    static void insereNieme(ListeIter l, int element, int rang){
        ElementListe aux, temp;
        aux = l.getPremier();
        for (int i=1; i<rang-1; i++){
            aux = aux.getSuivant();
        }
        temp = new ElementListe(element);
        temp.setSuivant(aux.getSuivant());
        aux.setSuivant(temp);
    }
    static void ecrireListe(ListeIter l){
        ElementListe p = l.getPremier();
        while (p != null){
            Terminal.ecrireString("␣␣"+p.getValeur());
            p = p.getSuivant();
        }
        Terminal.sautDeLigne();
    }
    public static void main(String[] args){
        ListeIter lal = new ListeIter();
        lal.ajouterAuDebut(4);
        lal.ajouterAuDebut(3);
        lal.ajouterAuDebut(2);
        lal.ajouterAuDebut(1);
        Terminal.ecrireString("lal:␣");
        ecrireListe(lal);
        insereNieme(lal,5,3);
        Terminal.ecrireString("lal:␣");
        ecrireListe(lal);
    }
}
```

Question 3

Nous allons maintenant parachever le travail en traitant les erreurs et les cas particuliers éventuels.

Cas particuliers :

- la procédure marche-t-elle si on veut ajouter l'élément en première position ? **non, la procédure ne fonctionne pas dans ce cas puisqu'il faut changer le contenu de la variable** `premier` **de la liste, ce que ne fait pas la procédure. Par ailleurs, il est vain de chercher à faire pointer la variable auxiliaire** `aux` **sur l'élément précédent, puisqu'il n'existe pas.**
- la procédure marche-t-elle si on veut ajouter l'élément en dernière position ? **Oui, cela ne pose pas de problème particulier. La ligne de code** `temp.setSuivant(aux.getSuivant());` **recopie la valeur** `null`**.**

Modifiez la procédure pour qu'elle marche dans tous ces cas.

Le problème qui peut arriver est que la liste ne comporte pas assez d'éléments pour que le rang demandé existe. Par exemple, il n'est pas possible d'insérer au dixième rang d'une liste de trois entiers.

A quel endroit de la procédure peut-on déceler le problème, et comment ?

On peut déceler le problème quand on avance dans la liste avec la boucle qui avance `aux` d'un cran.

```
class Exo22_2Bis{
    static void insereNieme(ListeIter l, int element, int rang)
        throws MauvaisRang{
        if (rang < 1){
            throw new MauvaisRang();
        }
        if (rang == 1){
            l.ajouterAuDebut(element);
        }else{
            ElementListe aux = l.getPremier();
            for (int i=1; i<rang-1; i++){
                if (aux.getSuivant() == null){
                    throw new MauvaisRang();
                }
                aux = aux.getSuivant();
            }
            ElementListe temp = new ElementListe(element);
            temp.setSuivant(aux.getSuivant());
            aux.setSuivant(temp);
        }
    }
    static void ecrireListe(ListeIter l){
        ElementListe p = l.getPremier();
        while (p != null){
            Terminal.ecrireString("␣␣"+p.getValeur());
            p = p.getSuivant();
        }
        Terminal.sautDeLigne();
    }
    public static void main(String[] args) throws MauvaisRang{
        ListeIter lal = new ListeIter();
        lal.ajouterAuDebut(4);
        lal.ajouterAuDebut(3);
        lal.ajouterAuDebut(2);
        lal.ajouterAuDebut(1);
        Terminal.ecrireString("lal:␣");
        ecrireListe(lal);
        insereNieme(lal,5,3);
        Terminal.ecrireString("lal:␣");
        ecrireListe(lal);
        insereNieme(lal,6,1);
        Terminal.ecrireString("lal:␣");
        ecrireListe(lal);
        insereNieme(lal,7,7);
        Terminal.ecrireString("lal:␣");
        ecrireListe(lal);
        insereNieme(lal,8,17); // erreur de rang
    }
}
class MauvaisRang extends Exception{}
```