

GÉNIE LOGICIEL ORIENTÉ OBJET (GLO-2004)
ANALYSE ET CONCEPTION DES SYSTÈMES ORIENTÉS OBJETS (IFT-2007)

Automne 2016

Module 10 - Exemples de diagrammes de séquence

Martin.Savoie@ift.ulaval.ca

**B. ing, Chargé de cours, département
d'informatique et de génie logiciel**

Exercice - Tracez le diagramme de classes et le diagramme de séquence à partir du code (page suivante)

- En exercice
 - Tracez le diagramme de classes de conception correspondant à l'ensemble du code.
 - Tracez le diagramme de séquence pour un appel à l'opération *Manager.embauche*.

Tracez le diag. de classes et le diag. de séquence

```
public class Manager
{
    private Compagnie compagnie;
    public void embauche(int noEmployé)
    {
        if (noEmployé != 0)
            compagnie.embauche(noEmployé);
    }
}
```

```
public class Compagnie
{
    public ListeEmployés listeEmployés;
    public void embauche(int _noEmployé)
    {
        Employé nouveau = new Employé();
        nouveau.passeNoEmployé(_noEmployé);
        listeEmployés.ajoute(nouveau);
    }
}
```

```
public class Employé
{
    private noEmployé;
    public void passeNoEmployé(int _noEmployé)
    {
        noEmployé = _noEmployé;
    }
}
```



Solution – Diag. de classes de conception

```
public class Manager
{
    private Compagnie compagnie;
    public void embauche(int noEmployé)
    {
        if (noEmployé != 0)
            compagnie.embauche(noEmployé);
    }
}
```

```
public class Employé
{
    private noEmployé;
    public void passeNoEmployé(int _noEmployé)
    {
        noEmployé = _noEmployé;
    }
}
```

```
public class Compagnie
{
    public ListeEmployés listeEmployés;
    public void embauche(int _noEmployé)
    {
        Employé nouveau = new Employé();
        nouveau.passeNoEmployé(_noEmployé);
        listeEmployés.ajoute(nouveau);
    }
}
```

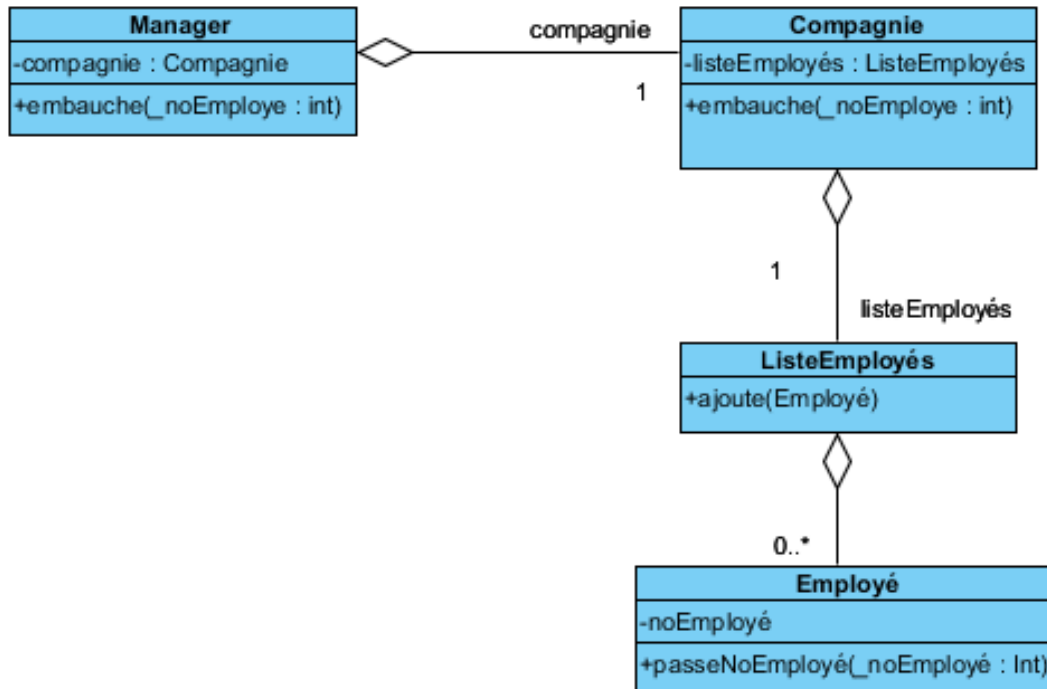
Solution – Diag. de classes de conception

```
public class Manager
{
    private Compagnie compagnie;
    public void embauche(int noEmployé)
    {
        if (noEmployé != 0)
            compagnie.embauche(noEmployé);
    }
}
```

```
public class Employé
{
    private noEmployé;
    public void passeNoEmployé(int _noEmployé)
    {
        noEmployé = _noEmployé;
    }
}
```

```
public class Compagnie
{
    public ListeEmployés listeEmployés;
    public void embauche(int _noEmployé)
    {
        Employé nouveau = new Employé();
        nouveau.passeNoEmployé(_noEmployé);
        listeEmployés.ajoute(nouveau);
    }
}
```

Visual Paradigm for UML Standard Edition (Université Laval)



Solution – Diag. de séquence de l'opération Manager.embauche

```
public class Manager
{   private Compagnie compagnie;
    public void embauche(int noEmployé)
    {
        if (noEmployé != 0)
            compagnie.embauche(noEmployé);
    }
}
```

```
public class Employé
{   private noEmployé;
    public void passeNoEmployé(int _noEmployé)
    {   noEmployé = _noEmployé;
    }
}
```

```
public class Compagnie
{   public ListeEmployés listeEmployés;
    public void embauche(int _noEmployé)
    {
        Employé nouveau = new Employé();
        nouveau.passeNoEmployé(_noEmployé);
        listeEmployés.ajoute(nouveau);
    }
}
```

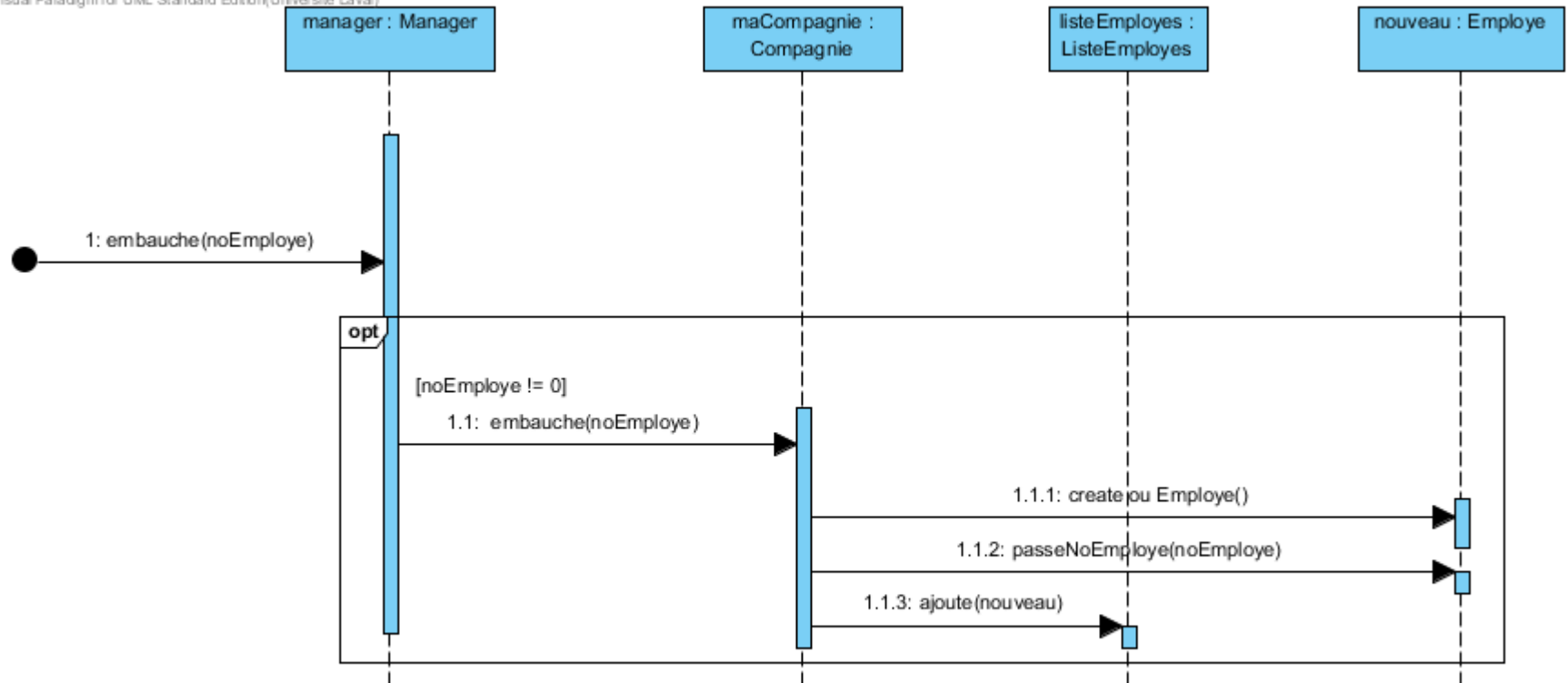
Solution – Diag. de séquence de l'opération Manager.embauche

```
public class Manager
{
    private Compagnie compagnie;
    public void embauche(int noEmployé)
    {
        if (noEmployé != 0)
            compagnie.embauche(noEmployé);
    }
}
```

```
public class Employé
{
    private noEmployé;
    public void passeNoEmployé(int _noEmployé)
    {
        noEmployé = _noEmployé;
    }
}
```

```
public class Compagnie
{
    public ListeEmployés listeEmployés;
    public void embauche(int _noEmployé)
    {
        Employé nouveau = new Employé();
        nouveau.passeNoEmployé(_noEmployé);
        listeEmployés.ajoute(nouveau);
    }
}
```

Visual Paradigm for UML Standard Edition (Université Laval)



Robo sapiens



* A.ROB *

Sortie 1



Sortie 2



☐ Lier les sorties

☒ Mémoriser

Interrupteur 1:



Interrupteur 2:



Exécuter une
procédure



Ouvrir



Nouveau



Enregistrer



Quitter



Aide

```
; La procédure « Longe_un_mur » fait ceci
; ■ le robot avance jusqu'à rencontrer un
; obstacle (interrupteur 1)
; ■ il se tasse d'une largeur et recommence
; jusqu'à ce qu'il se retrouve a l'extrémité
; de l'obsacle

; Ce fichier contient d'autres procédure
; intéressantes (utilisées par Longe_un_mur)
; Avance_Un_Peu, Recule_Un_Peu
; FonceDansObstacle, Avance_D'une_Longueur
; Tasse_D'une_Largeur
```

```
[Longe_Un_mur]
FonceDansObstacle
Tant que il actif Fais Tasse_d'une_largeur
```

```
[TasseD'uneLargeur]
ReculeUnPeu
Droite90
attends 0,5
AvanceD'uneLongueur
Gauche90
AvanceUnPeu
```

```
[ReculeUnPeu]
recule
attends 0,3 sec
arreteetout
```

```
[AvanceUnPeu]
Avance
attends 0,4 sec
ArreteTout
```

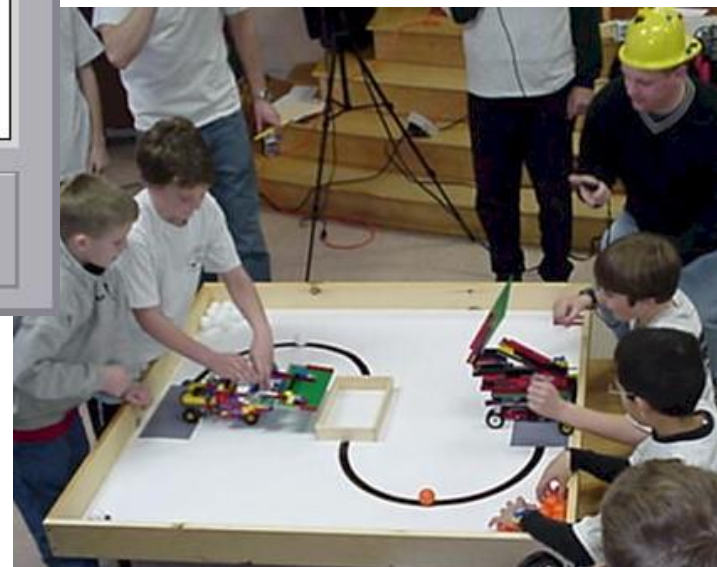
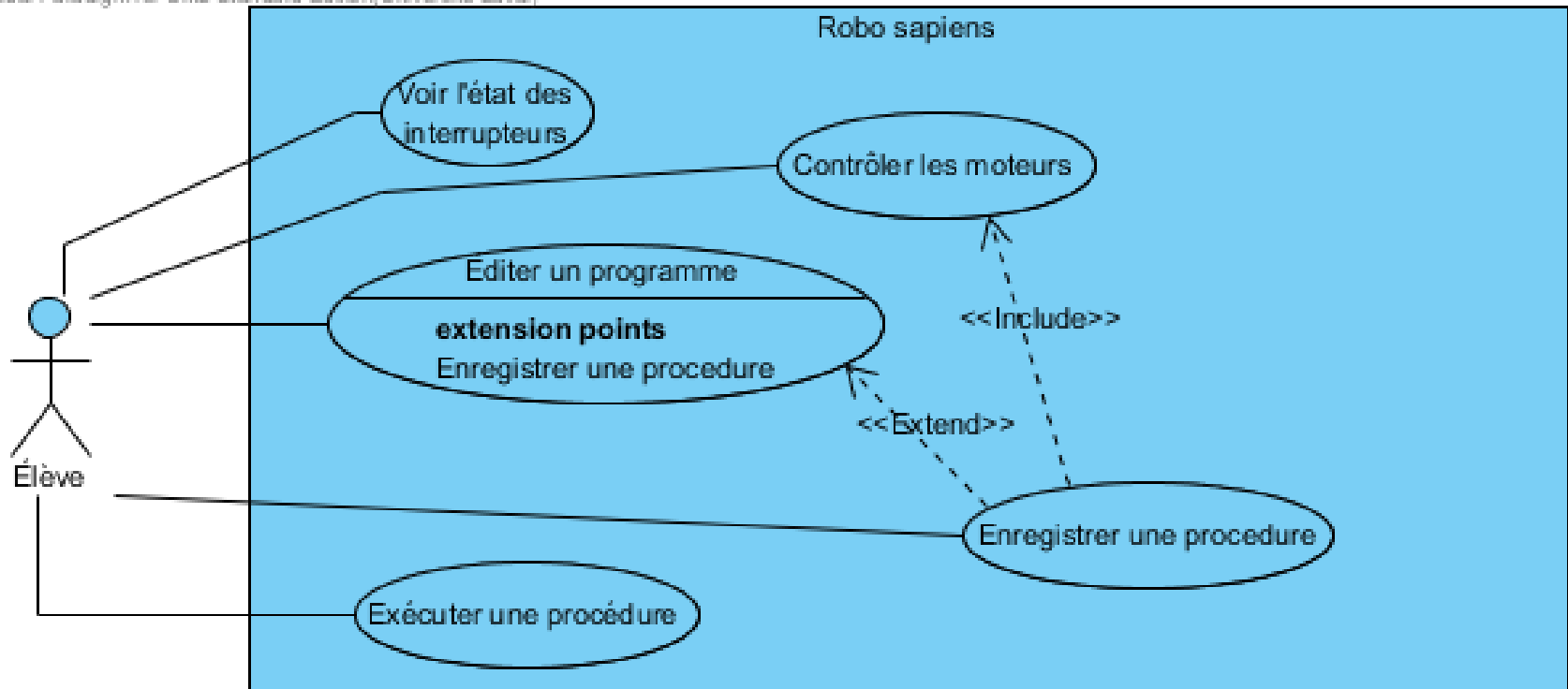


Diagramme des cas d'utilisation

Visual Paradigm for UML Standard Edition (Université Laval)



Modèles du domaine / Diagramme de classes conceptuel

Visual Paradigm for UML Standard Edition (Université Laval)

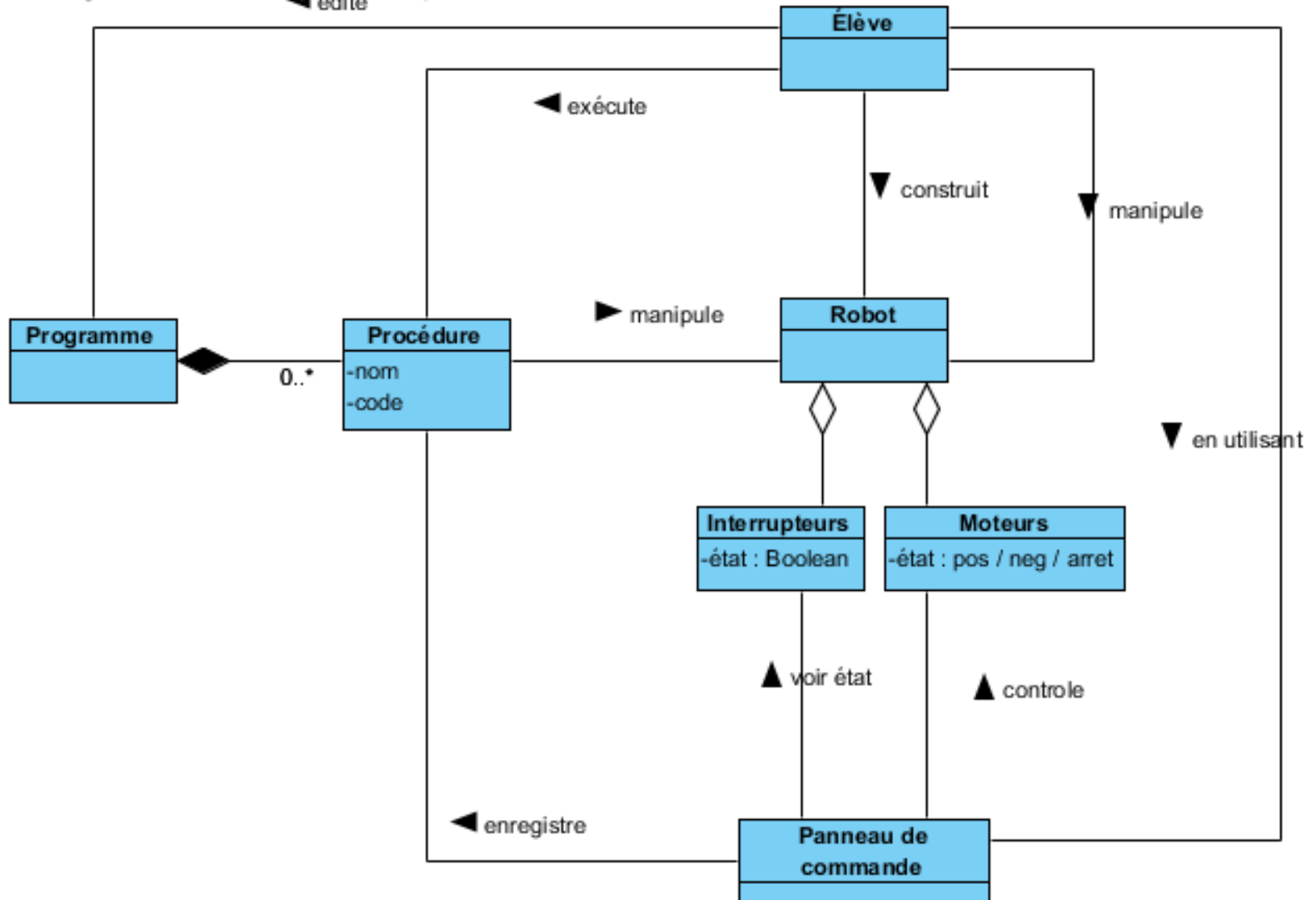
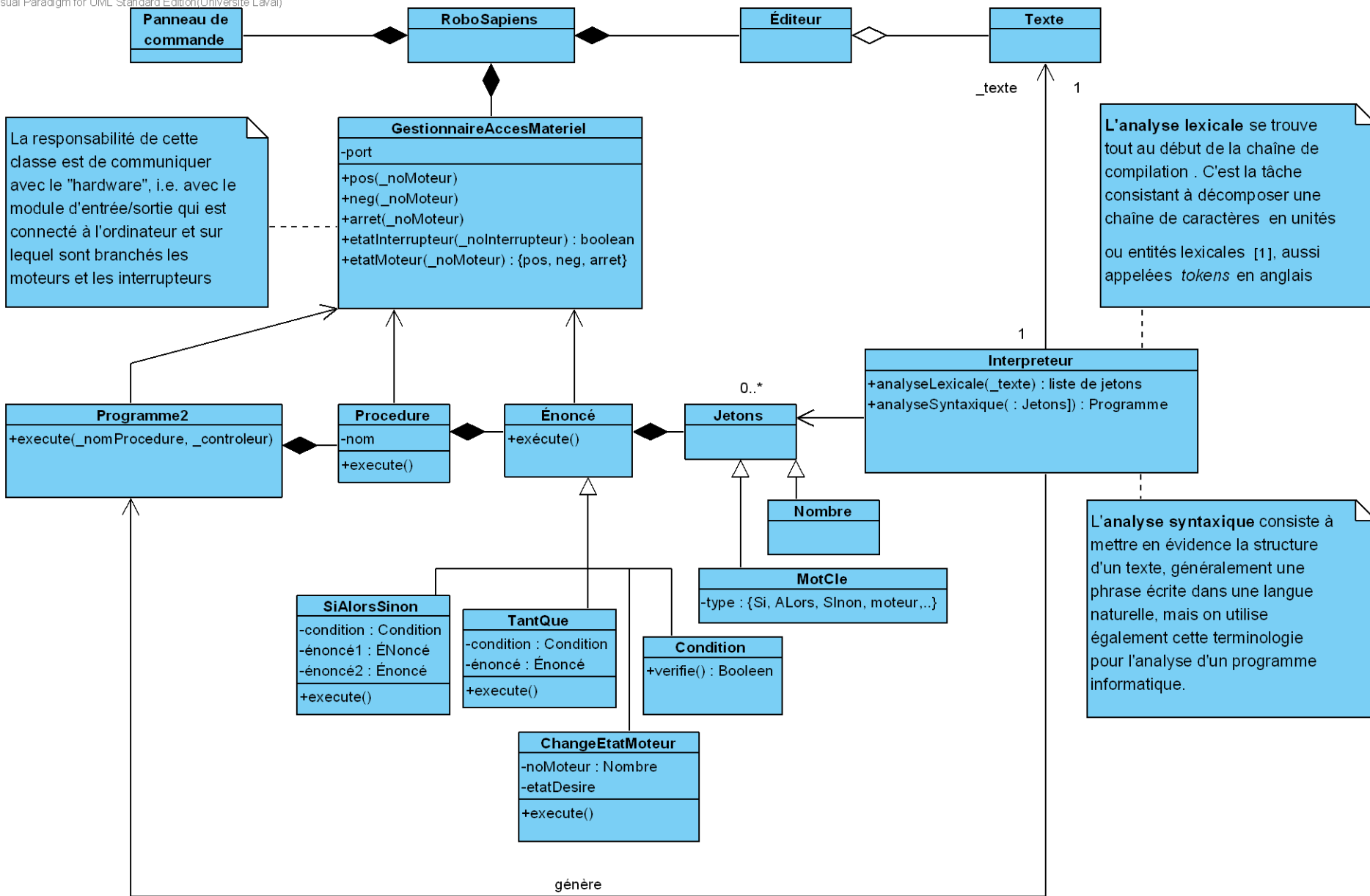


Diagramme des classes de (conception)

Visual Paradigm for UML Standard Edition (Université Laval)



Le cas d'utilisation le plus complexe: « Exécuter une procédure »

[Ma_procédure]

Si interrupteur 1 actif, alors moteur 1 positif,

Sinon, moteur 1 négatif

Attends 5 secondes

Moteur 1 arrêté



Il faut bâtir un
interpréteur !

1. Transformation du texte en éléments appelés« jetons »

[Ma_procédure]

Si interrupteur 1 actif, alors moteur 1 positif,

Sinon, moteur 1 négatif

Attends 5 secondes

Moteur 1 arrêté

2. Mettre en évidence la structure du programme rédigé par l'utilisateur

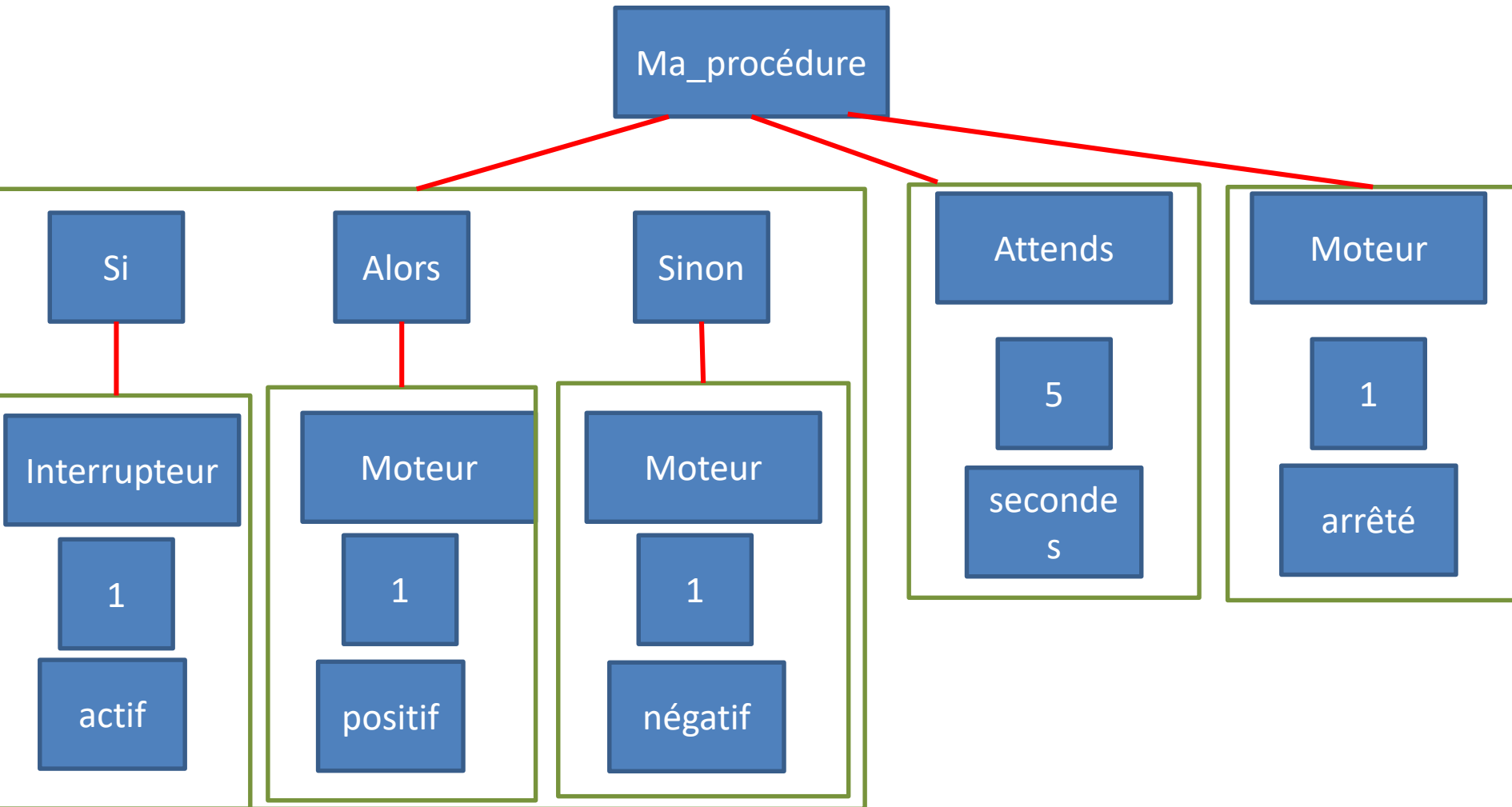
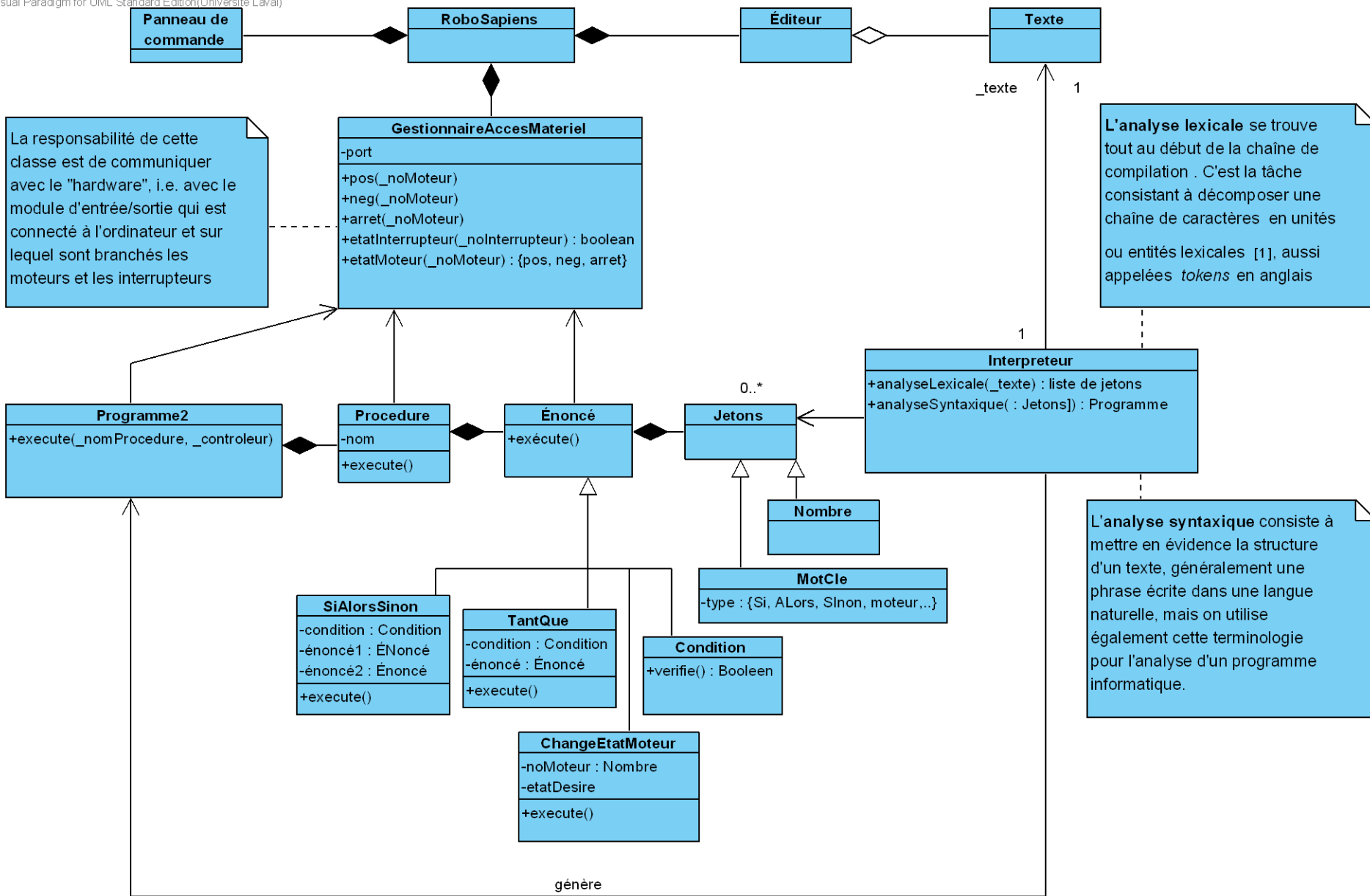


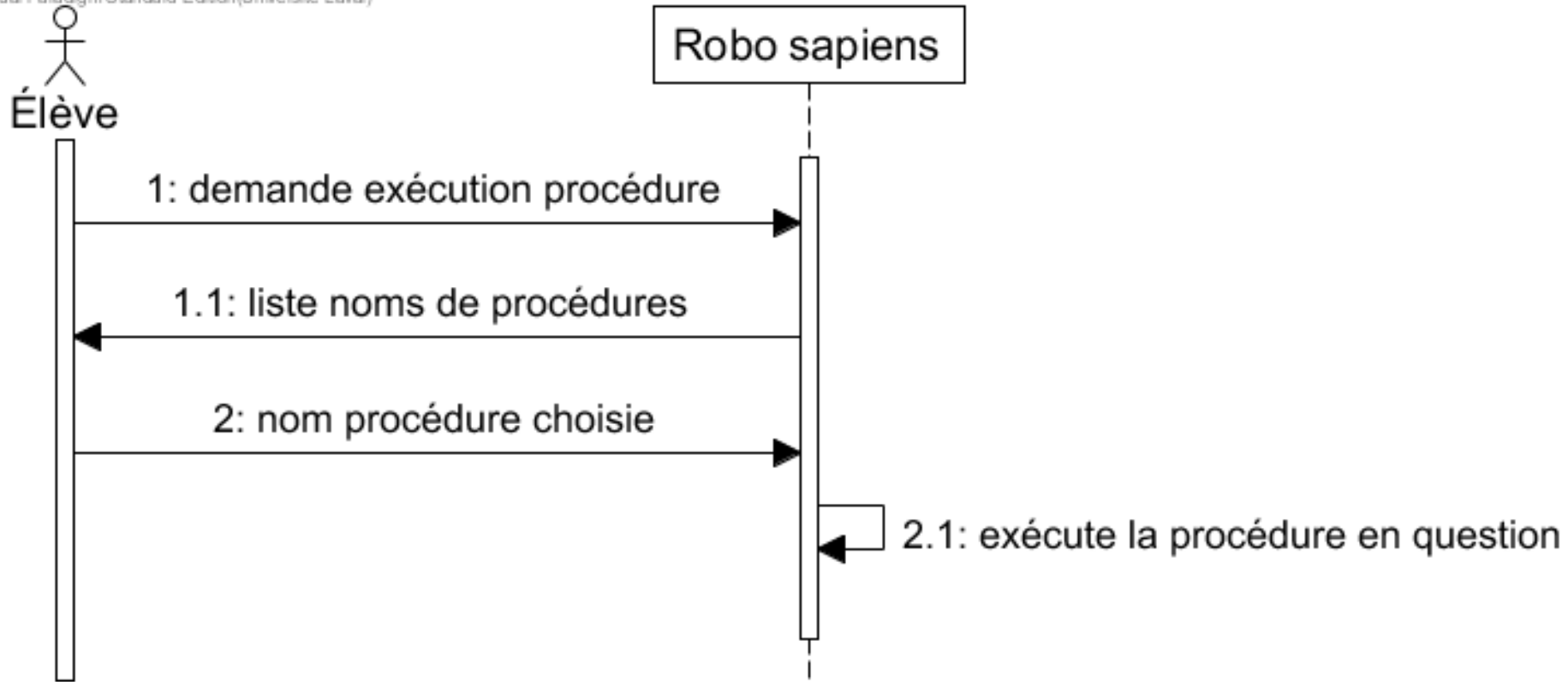
Diagramme des classes de (conception)

Visual Paradigm for UML Standard Edition (Université Laval)



Exécuter une procédure – Rappel - DSS

Visual Paradigm Standard Edition (Université Laval)

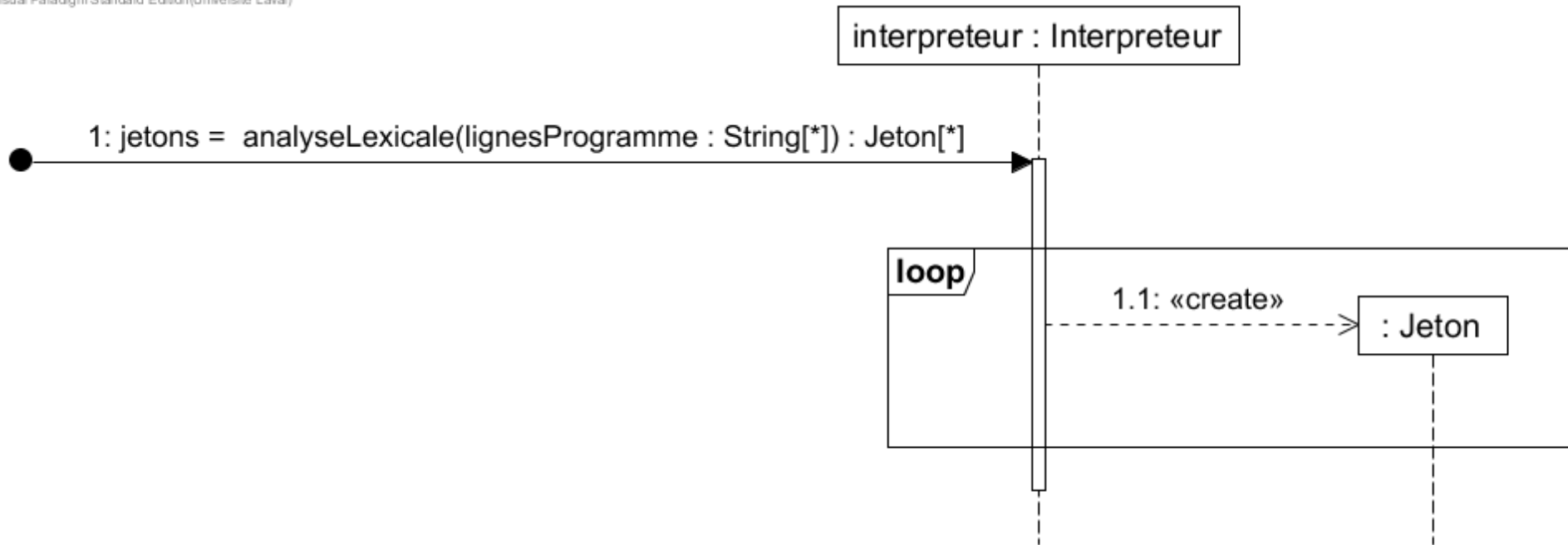


Exercice - Robo sapiens

- Exécuter une procédure
 - Plusieurs petits **diagrammes de séquence de conception** pour détailler comment nos classes réaliseront le scénario décrit dans le **DSS**

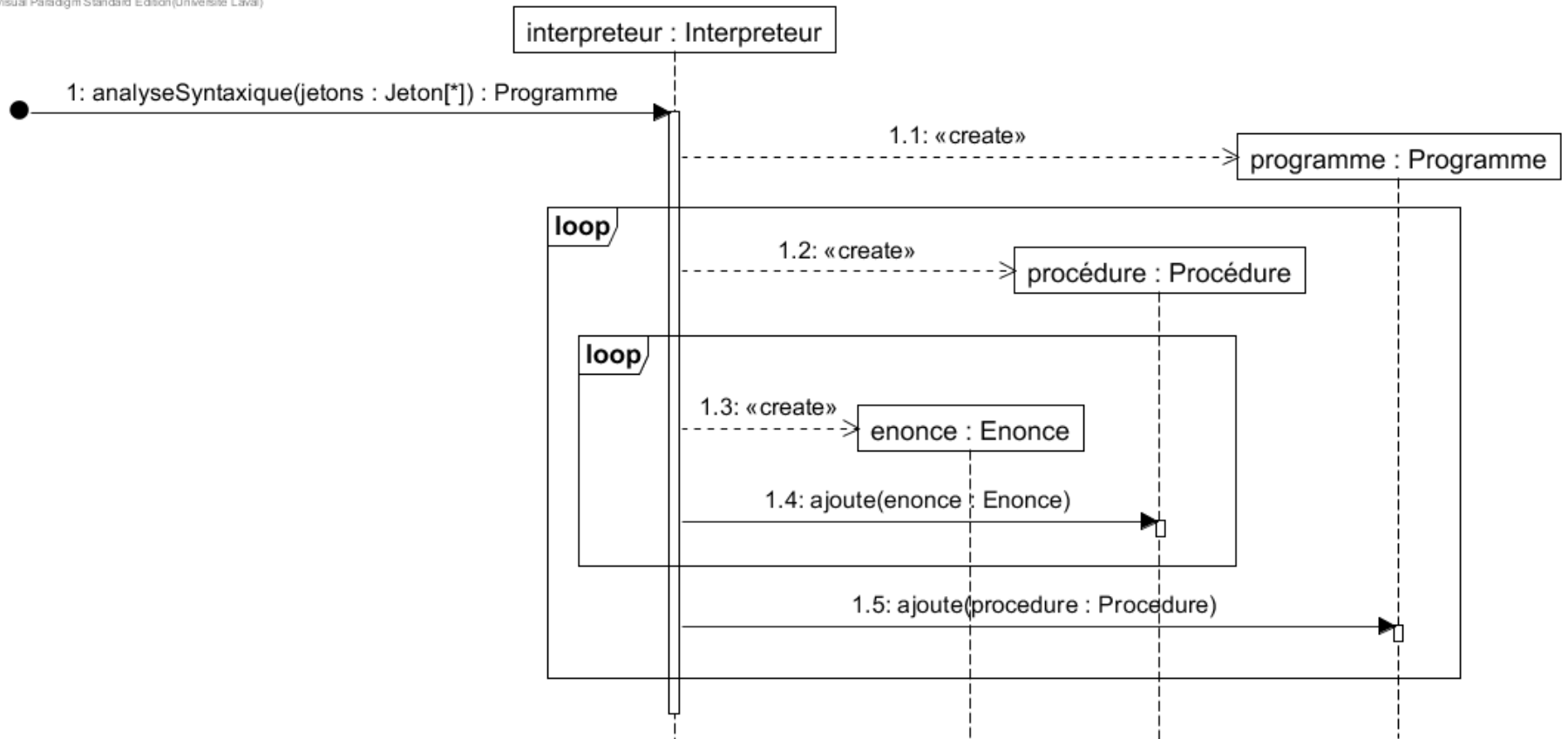
Analyse lexicale (incomplet)

Visual Paradigm Standard Edition (Université Laval)



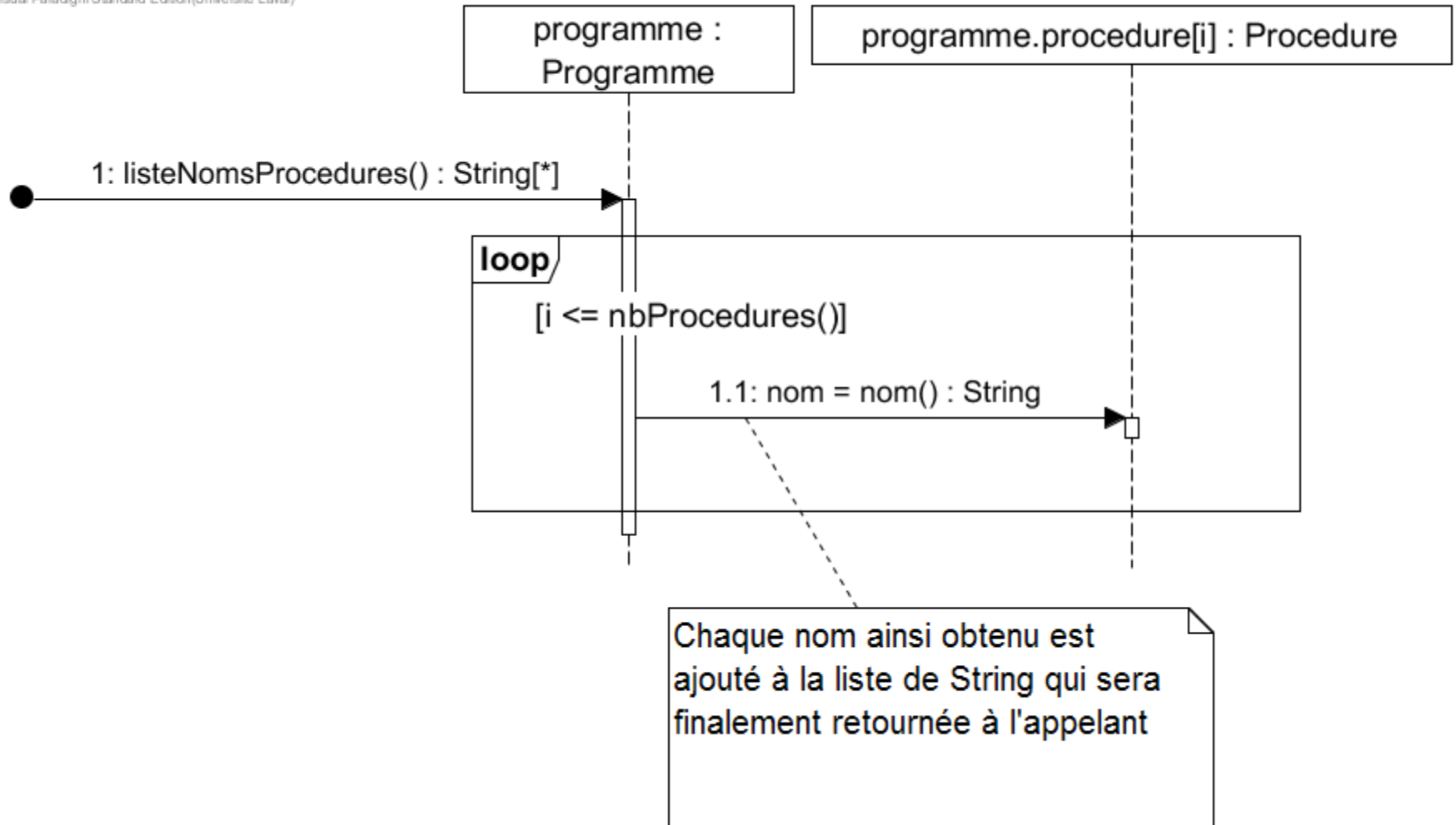
Analyse syntaxique (incomplet)

Visual Paradigm Standard Edition (Université Laval)



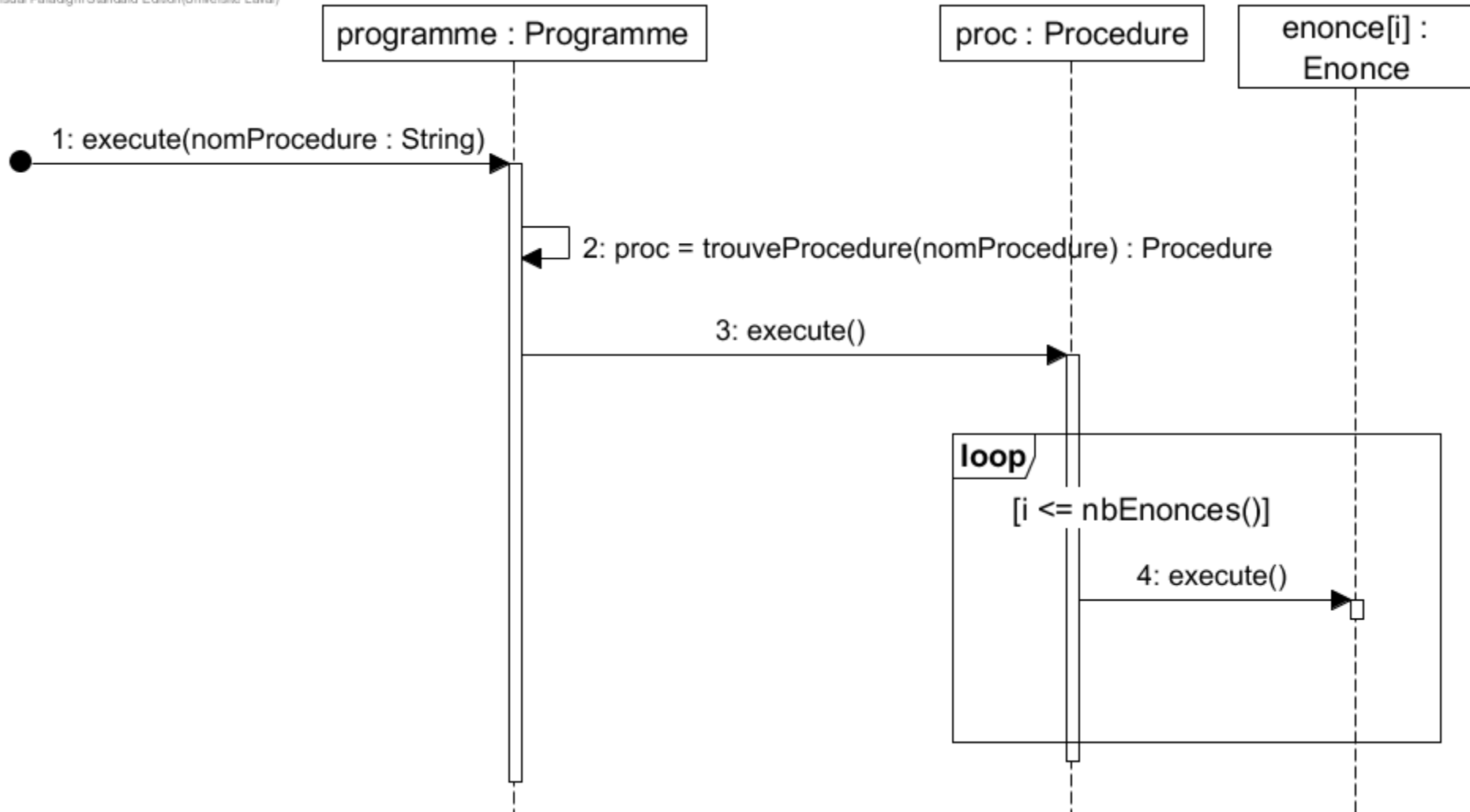
Obtenir la liste des noms de procédures

Visual Paradigm Standard Edition (Université Laval)



Exécute procédure (générique)

Visual Paradigm Standard Edition (Université Laval)

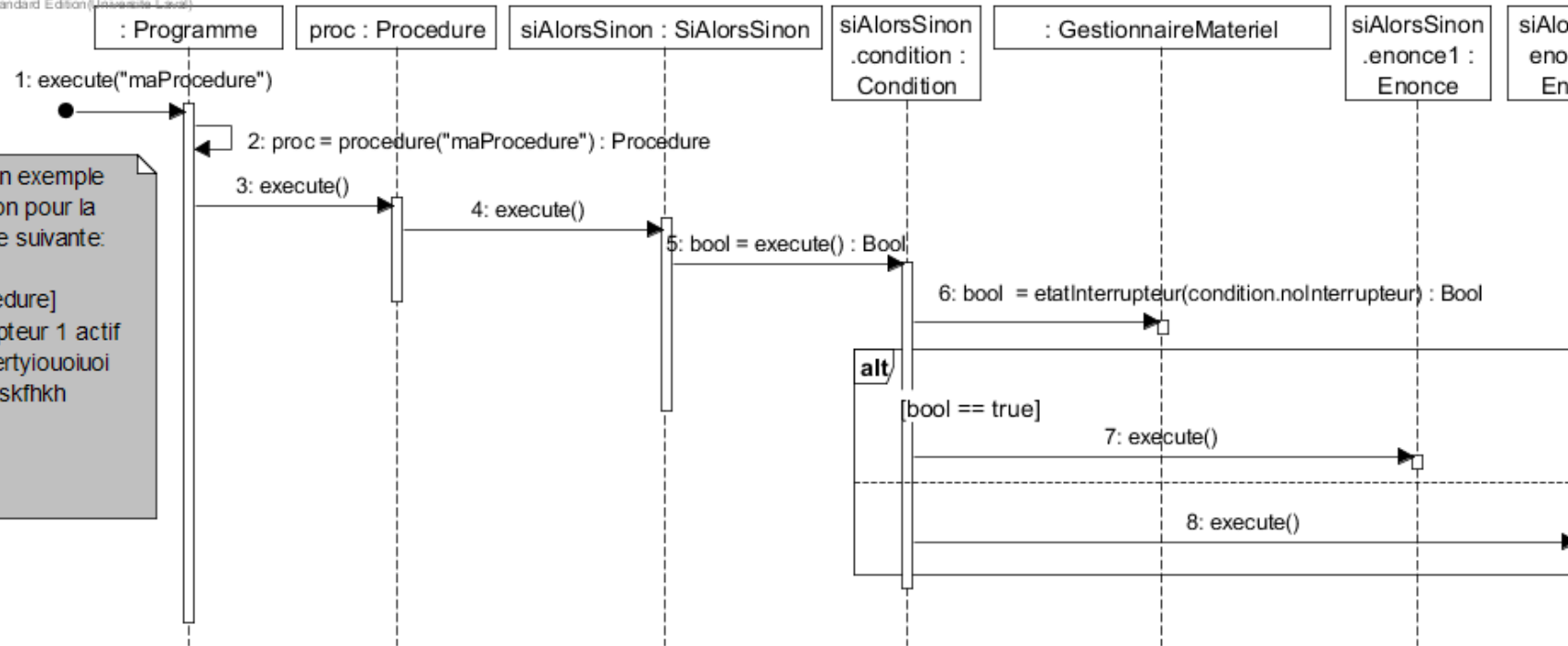


Exécute procédure (exemple)

Visual Paradigm Standard Edition (UML 2.5.1)

Voyons un exemple d'exécution pour la procédure suivante:

[maProcédure]
Si interrupteur 1 actif
alors qwertyiouioi
sinon kjfhskfhkh



Attention!!

- Nous avons une mauvaise correspondance entre les noms dans le diagramme des classes et dans les diagrammes de séquences:
 - Nouvelles méthodes à **ajouter** dans le diagramme des classes (**à corriger**)
 - Noms différents dans l'un et l'autre
- Il est impératif de retourner dans le diagramme des classes et faire les corrections
 - Notez bien: Visual Paradigm fait une bonne partie du travail de mise à jour pour vous

Les diagrammes de **séquence** peuvent être utilisés sous deux **formes**:

- **forme générique**: elle décrit **tous** les déroulements possibles d'un scénario et peut contenir des **branchements**, des **conditions**, et des **boucles**.
- **forme d'instanciation**: elle décrit le comportement du système pour un **aspect spécifique** d'un scénario et ne peut donc contenir **aucun branchement** et aucune **condition** ou **boucle**.

À faire cette semaine

- Lecture des chapitres
 - Version anglaise: 14,15 (pour les retardataires: 12 à 16)
 - Version française: 13,14 (pour les retardataires: 12 à 15)
- Comprendre la relation entre les concepts suivants:
 - Diagrammes d'interaction, diagramme de séquence, diagramme de communication
 - Diagramme de séquence système et Diagramme de séquence
 - Diagramme de séquence système, Cas d'utilisation, Diagramme de cas d'utilisation, Scénario
- Transformer du code en diagramme d'interaction, et vice-versa
- Projet de session

À faire cette semaine

- Étudier
- Bien réussir l'examen

Pas seulement pour avoir une bonne note: mais aussi pour bien intégrer les concepts, bien réussir son projet, gagner le prix du meilleur projet, avoir une super job et devenir un super analyste.

