

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

Automne 2016

**Module 08 - Création d'un diagramme de classes
de conception pour *Robo sapiens*
Architecture logique et diagramme de packages UML**

Martin.Savoie@ift.ulaval.ca

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

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
arreteTout
```

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

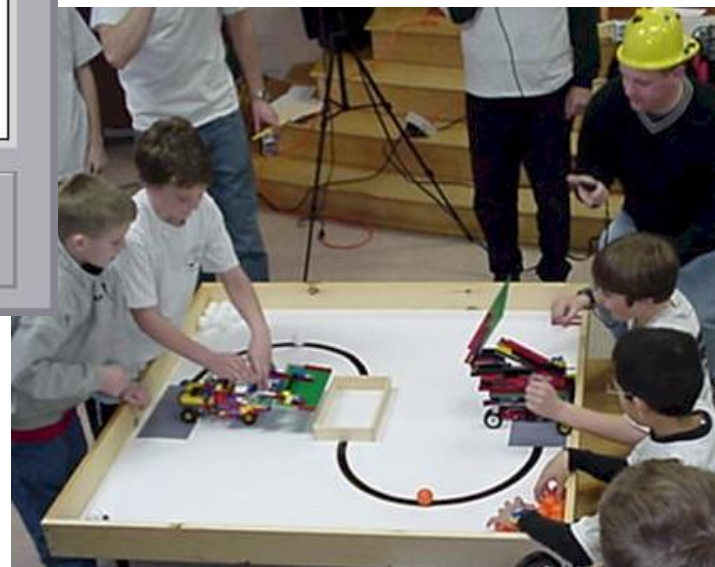
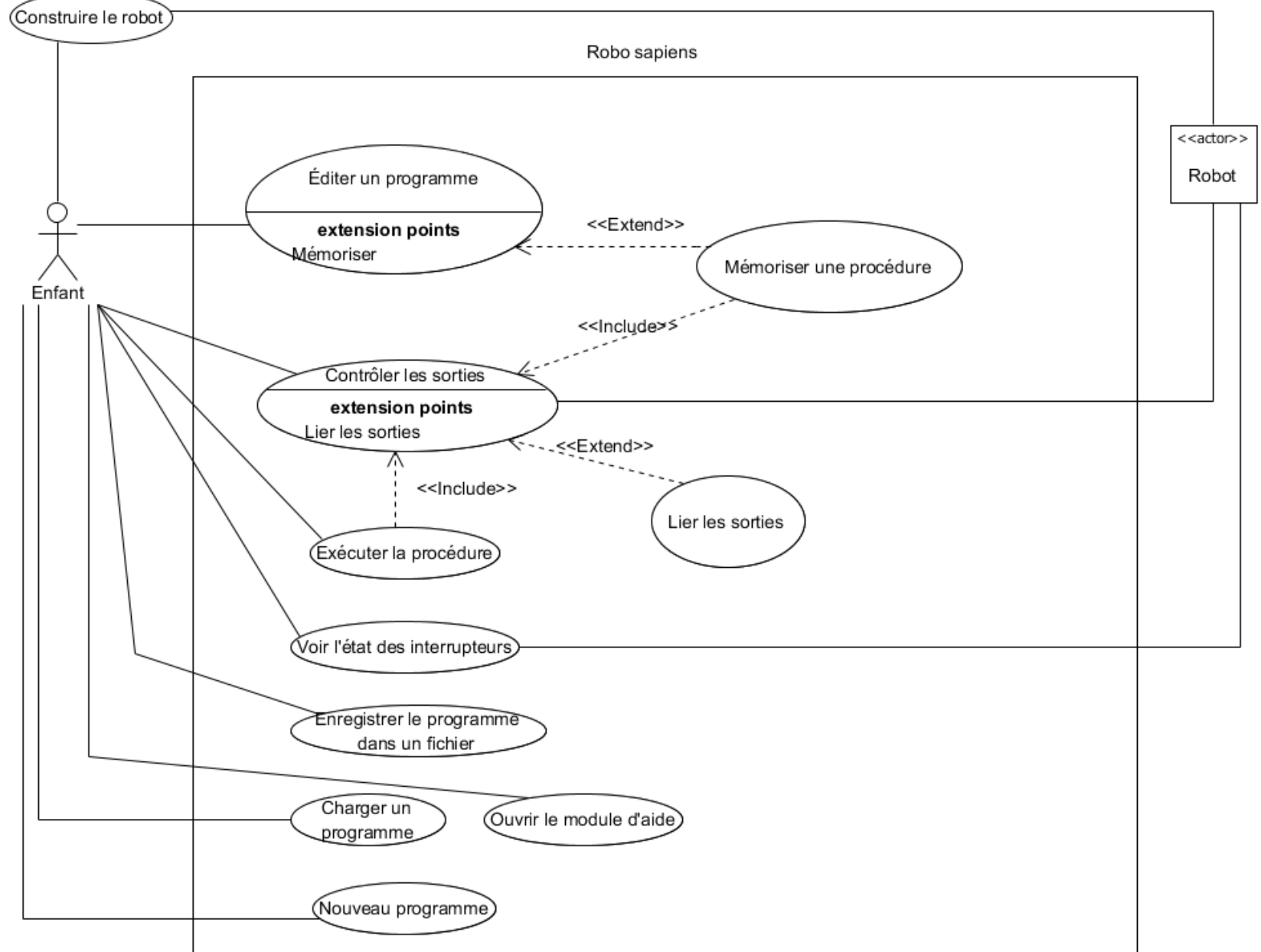


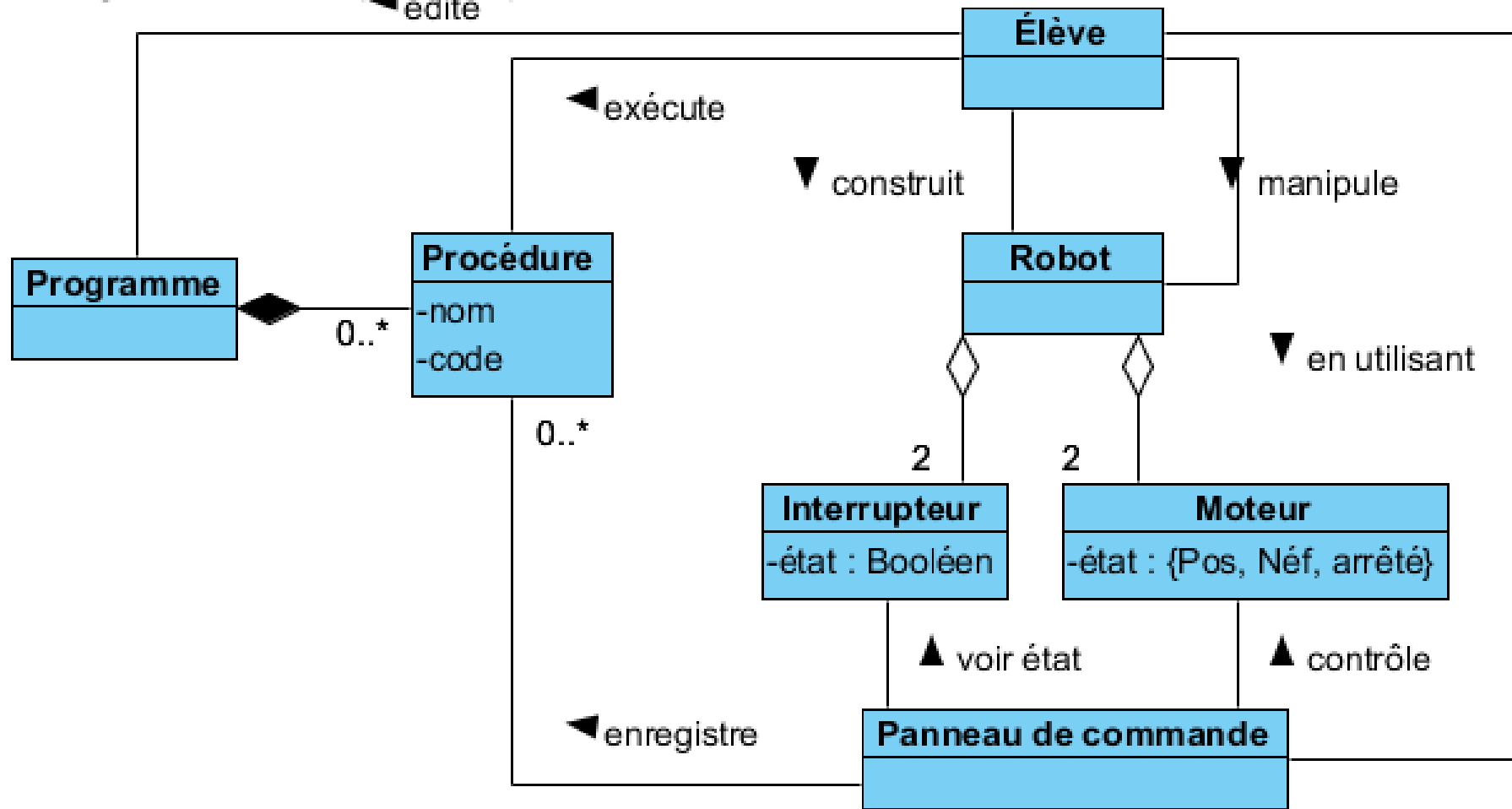
Diagramme des cas d'utilisation

Visual Paradigm Standard Edition (Université Laval)



Modèles du domaine / Diagramme de classes conceptuelles

Visual Paradigm for UML Standard Edition (Université Laval)



Création d'un diagramme de classe de conception pour Robo sapiens (avec Visual Paradigm)

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]

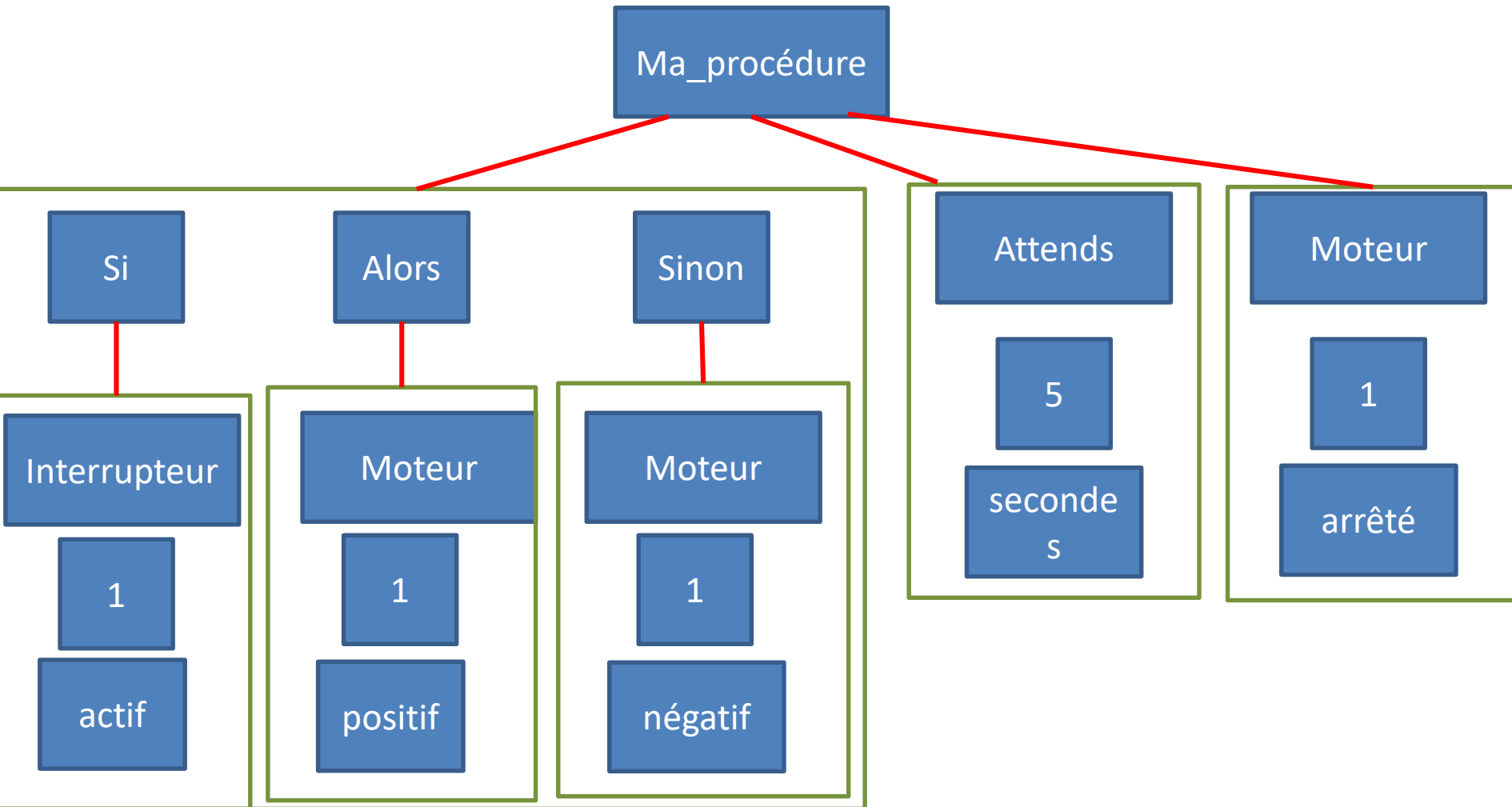
Si interrupteur 1 actif, alors moteur 1 positif,

Sinon, moteur 1 négatif

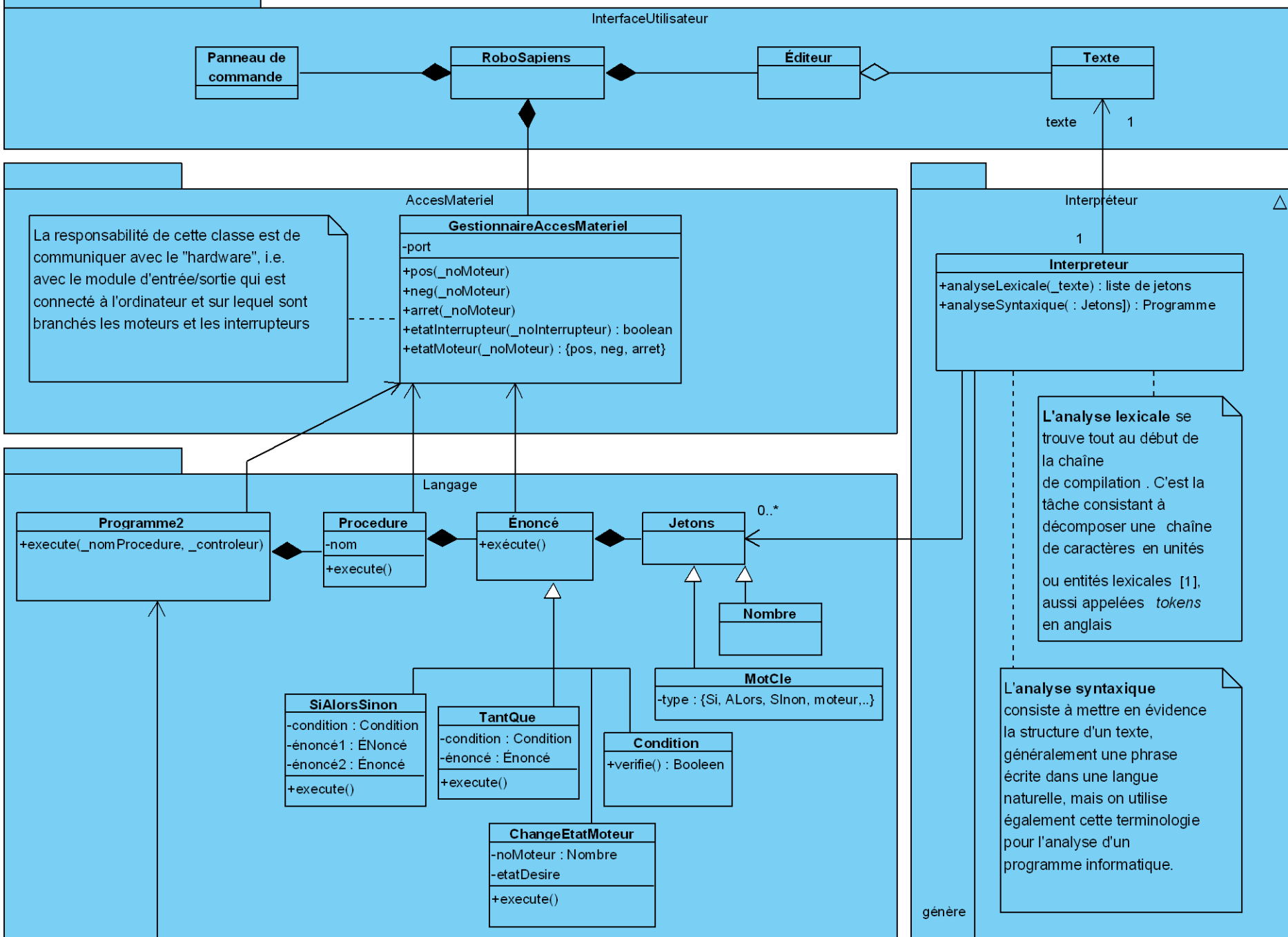
Attends 5 secondes

Moteur 1 arrêté

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

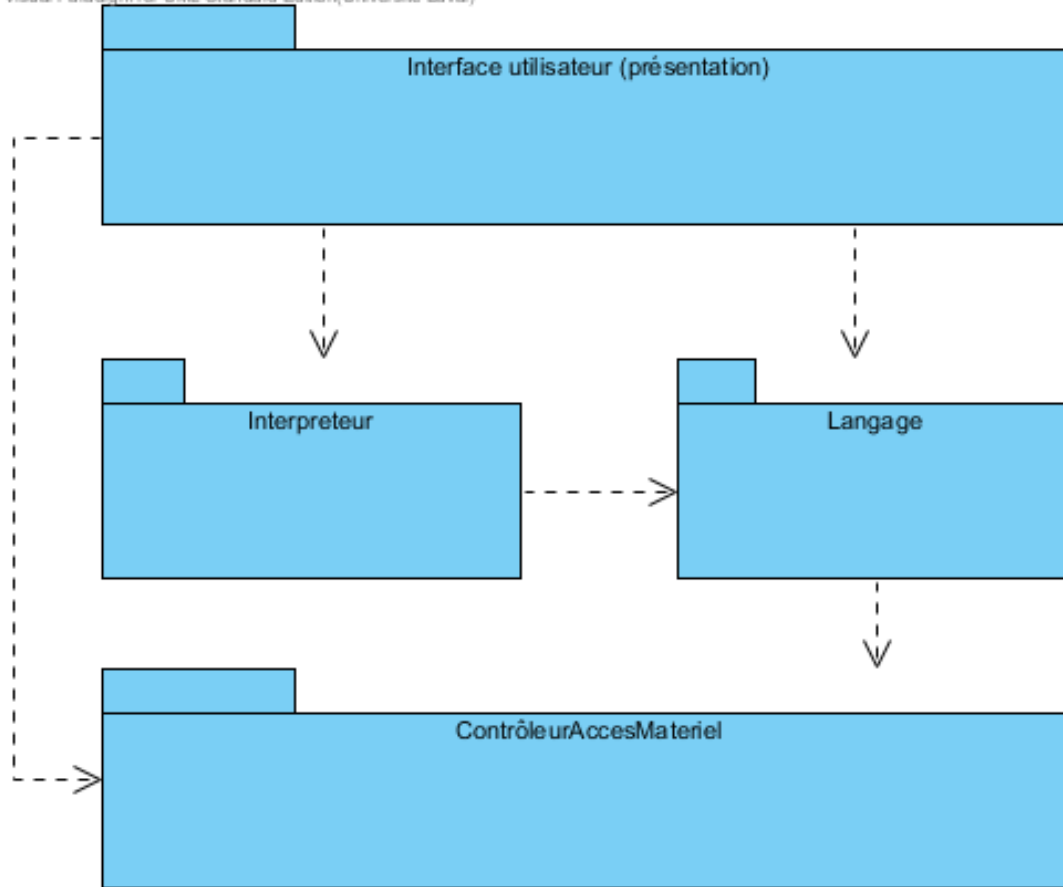


Solution - Création d'un diag. de classe de conception pour Robo sapiens



Architecture logique proposée

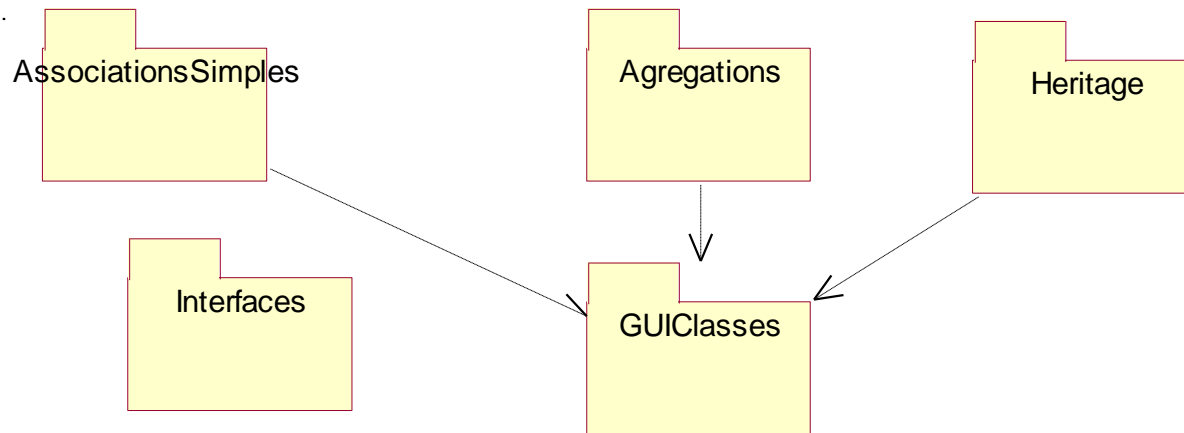
Visual Paradigm for UML Standard Edition (Université Laval)



- Minimise les dépendances
- Chaque package peut être développé / testé séparément
- Maximise la réutilisabilité

Les packages

- Un package est représenté par un **dossier** (folder)
- Les Packages ont eux aussi un **niveau de visibilité** est des **relations** qui sont ici: la **dépendance**, le **raffinement**, et la **généralisation**



Les Packages – En Java

- Pour les systèmes comprenant plusieurs classes, il convient de regrouper celles-ci en entités logiques, les **packages**
- Un package est un **ensemble de packages** et de **classes** reliés sémantiquement entre eux
- Chaque package est muni de **classes publiques** qui lui permet de communiquer sa fonctionnalité aux autres packages qui peuvent l'importer



Architecture logique et diagrammes de package UML

- Chapitre 12 (français)
- Chapitre 14 (anglais)

Architecture logique

- L'**architecture logique**, c'est l'organisation à grande échelle des classes logicielles en packages, sous-systèmes et couches.
- On la nomme architecture logique, car elle n'implique aucune décision quant à la façon dont ces éléments seront déployés physiquement.

Architecture en couche

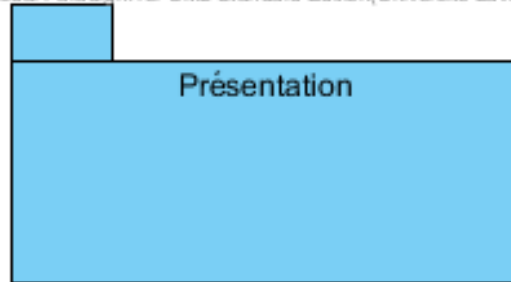
- Une **couche** est un regroupement à très forte granularité de classes, packages et sous-systèmes qui a une responsabilité de cohésion pour un aspect majeur du système.
- **Architecture en couche**: organisée de manière à ce que les couches accèdent aux couches inférieures, mais pas l'inverse.

Architecture en couche - Avantages

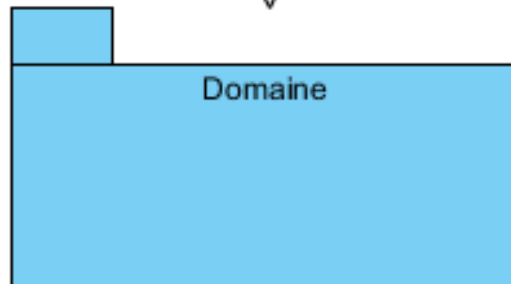
- Réduction du couplage et des dépendances
- Maintenance facilitée
- Plus facile de remplacer certaines couches par de nouvelles implémentations
- Les couches les plus basses contiennent des fonctionnalités réutilisables
- La segmentation facilite le développement en équipes

Architecture en couche « classique »

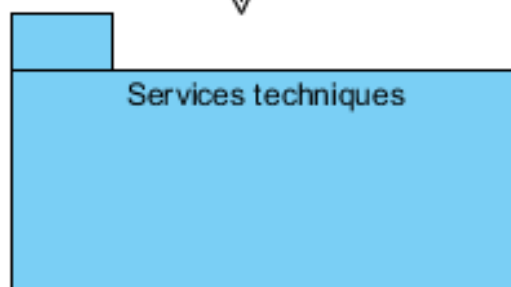
Visual Paradigm for UML Standard Edition (Université Laval)



Interface utilisateur

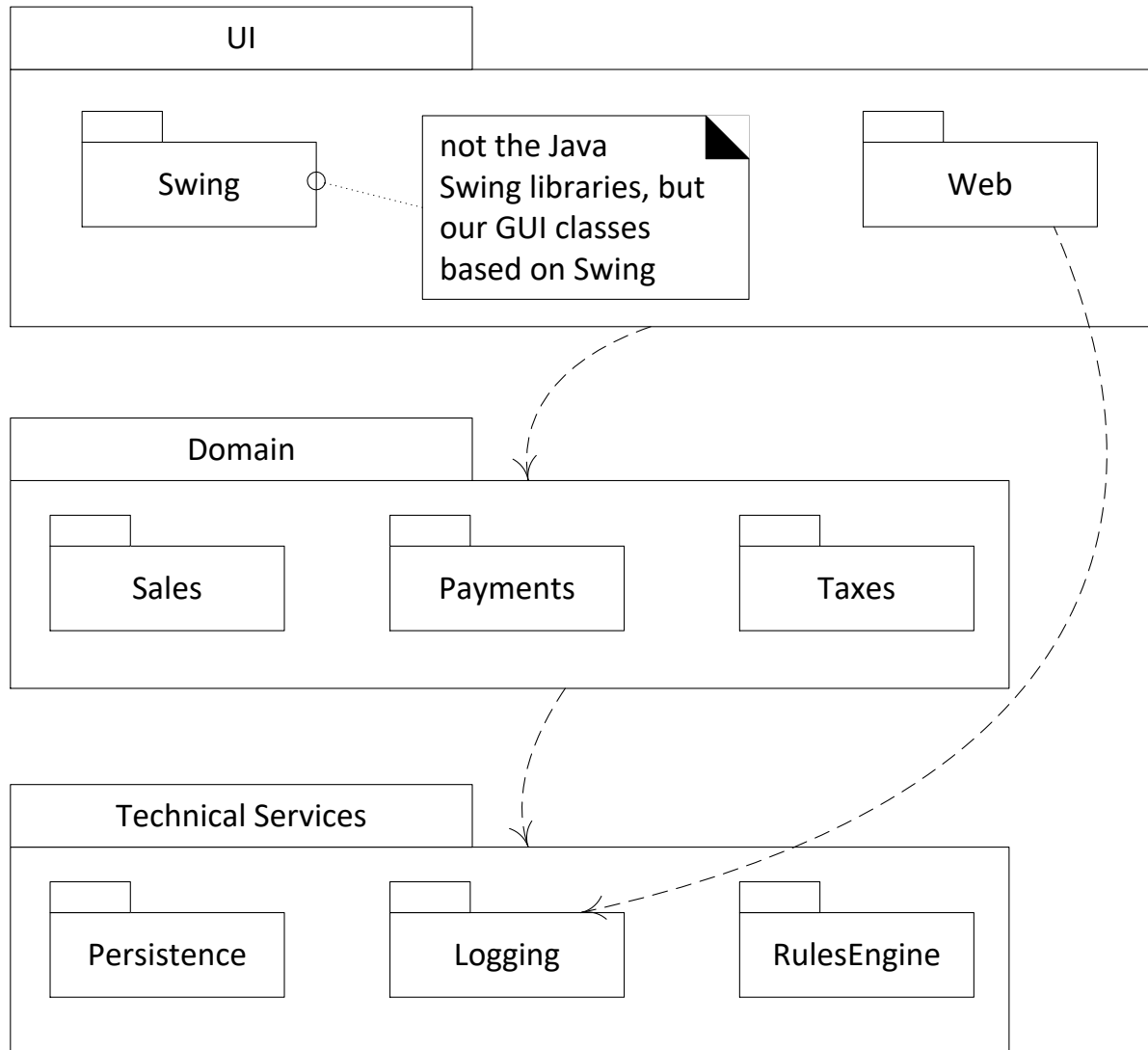


Logique applicative et objets
du domaine



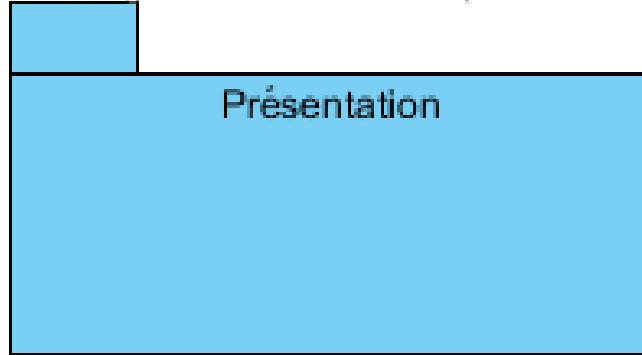
Objets à usage général (ex:
classes accès à une BD) –
généralement indépendant de
l'application

Architecture en couche – Exemple NexGen POS

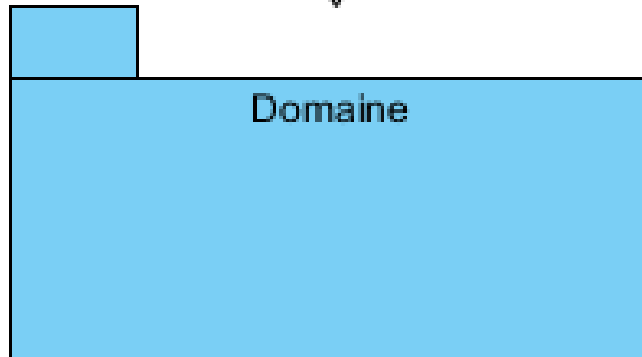


Grand principe: la séparation modèle-vue

Visual Paradigm for UML Standard Edition(Université Laval)



Vue



Modèle

Quel est le contraire d'une architecture basée le principe Modèle-Vue ?

- Une interface-moteur!
 - La logique applicative et l'interface utilisateur sont mélangées
 - Button1.onClick() fait des traitements intelligents!
 - Rien de réutilisable!



Péché mortel !

À faire cette semaine

- Lecture des chapitres
 - Version anglaise: 12,16,13,34,35
 - Version française: 15,12,28,29
- Comprendre la relation entre les concepts suivants:
 - Analyse; Design
 - Diagramme des classes; Diagramme des classes conceptuelles (modèle du domaine); Diagramme des classes de conception
 - Architecture logique / architecture logicielle
 - Architecture en couche;
 - Architecture en couche classique
 - Modèle-Vue
 - Diagramme de package