

Diagramme UML des cas d'utilisation pour un système à l'étude. Le diagramme est divisé en deux zones par une ligne verticale. À gauche, les acteurs sont représentés par des figures d'homme : Acteur1, Acteur2 spécialisé (héritier d'Acteur1), et Acteur3. À droite, les cas d'utilisation sont représentés par des ovales : cas d'utilisation1, cas d'utilisation2, cas d'utilisation3 (avec la note "Point d'extension : xxx"), cas d'utilisation4, cas d'utilisation11, et cas d'utilisation12. Les relations sont les suivantes : Acteur1 est lié à cas d'utilisation1 et cas d'utilisation2. Acteur2 spécialisé est lié à cas d'utilisation3. Acteur3 est lié à cas d'utilisation1. Cas d'utilisation1 est une généralisation de cas d'utilisation11 et cas d'utilisation12. Cas d'utilisation2 inclut (indiqué par «<<include>>» et une flèche pointillée) sous-cas d'utilisation. Cas d'utilisation3 étend (indiqué par «<<extend>>» et une flèche pointillée) cas d'utilisation4. Une note de condition (indiquée par une flèche en pointillés) est associée à cas d'utilisation4, avec le contenu "condition: point d'extension:xxx". Une boîte à droite, intitulée «<<actor>> Système existant», est liée à cas d'utilisation2.

résumé		
état	activité	cas d'utilisation
adjectif	substantif	verbe
démarré	démarrage	démarrer

Diagramme de séquence

The image displays several UML sequence diagrams illustrating different interaction patterns:

- Top Diagram:** An **Acteur** (Actor) initiates a sequence by sending a message to **: ClasseA**. **: ClasseA** then sends a synchronous message, **appel (synchrone)**, to **objet : ClasseB**. A dashed return arrow indicates the completion of the synchronous call.
- Middle-Left Diagram:** **a : ClasseA** creates **b : ClasseB** (indicated by a **create** message). **a : ClasseA** then sends a message to **b : ClasseB**. Finally, **a : ClasseA** destroys **b : ClasseB** (indicated by a **destroy** message and a large 'X' over the **b : ClasseB** lifeline).
- Middle-Right Diagram:** **a : ClasseA** sends an asynchronous message, **signal (asynchrone)**, to **b : ClasseB**. The lifelines for both classes are shown, with **a : ClasseA** ending first.
- Bottom-Left Diagram:** **a : ClasseA** sends a message to **b : ClasseB** with a note indicating **(importance du temps)** (importance of time), suggesting a time constraint or priority.
- Bottom-Right Diagram:** A stack of UML diagram symbols is shown, representing control structures: **opt** (optional), **loop** (loop), **alt** (alternative), and **else** (else), each associated with a **[condition]** label.

```

sequenceDiagram
    participant A as a: ClasseA
    participant B as b: ClasseB
    A->>B: 1.message1
    B-->A: 2.message2
  
```

1, 2, 3, ... : simple
 1.1, 1.2, 1.2.1, ... : imbrication
 1.a, 1.b, ... : concurrence

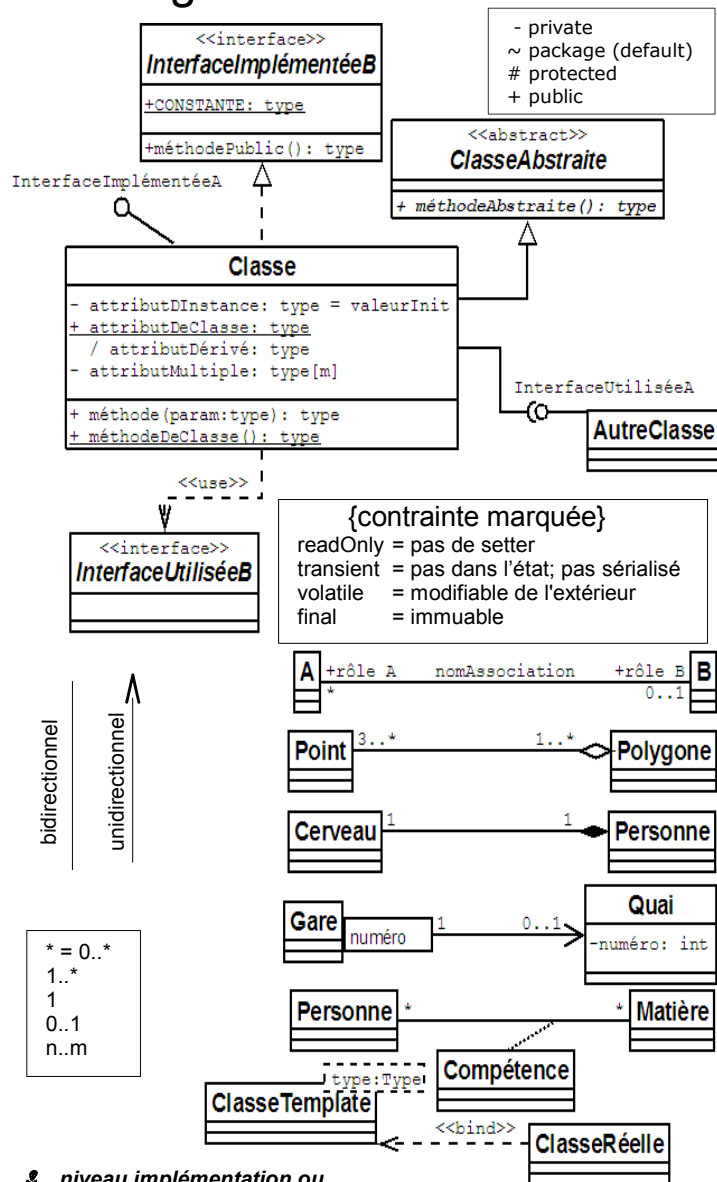
[illegible]

The diagram illustrates various state machine notations and transitions:

- Top Row:** A linear sequence of states: `●` → `état1` → `[garde] événement / action` → `état2` → `●`.
- Left Column:** A vertical sequence of states: `●` → `A` → `●`.
- Top Left:** A box labeled `état` containing a table:

entry/ action
do/ activité
exit/ action
- Top Center:** A box labeled `super état` containing states `B` and `C`. `B` transitions to `C`. The `super état` box transitions to state `D`.
- Bottom Left:** A box containing states `B` and `C`. `B` transitions to `C`. `C` transitions to state `A`. State `A` transitions to a final state `●`. There is a feedback loop from `A` to `B` labeled `(H)`.
- Bottom Center:** A box containing a vertical sequence of states: `●` → `A` → `B` → `C` → `●`.
- Bottom Right:** A box containing a vertical sequence of states: `●` → `E` → `F` → `●`. A dashed vertical line separates the two boxes in the bottom row.

Diagramme de classes



bidirectionnel
unidirectionnel

* = 0..*
1..*
1
0..1
n..m

✂ niveau implémentation ou conceptuel

Niveau conceptuel: uniquement les concepts, l'ge prog ignoré, type ignoré, interface ignorée

Niveau implémentation: l'ge prog connu, types et interfaces précisés.

type au niveau implémentation (l'ge Java):

- soit primitif
- soit de référence
- soit void (sauf attribut)

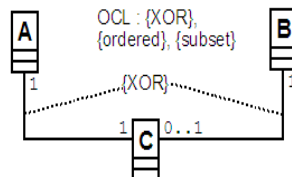


Diagramme d'objets

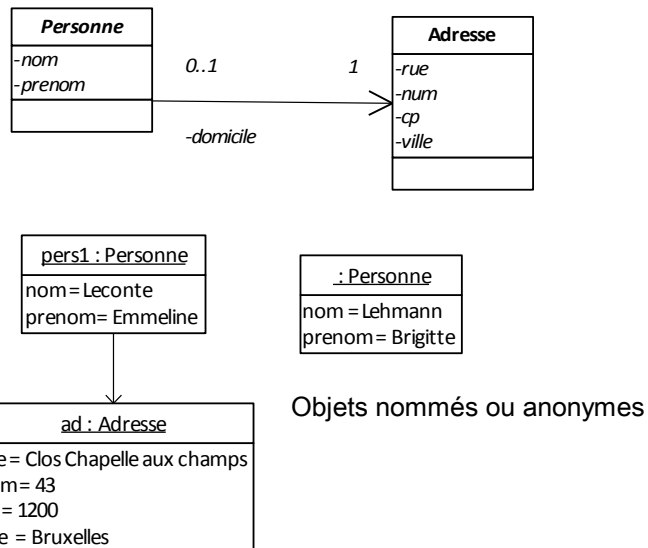
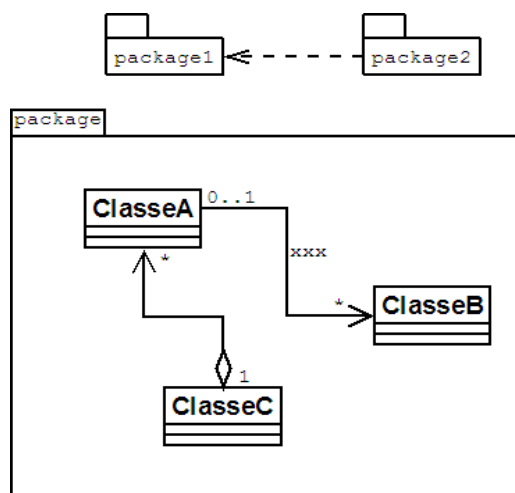


Diagramme de packages



Class	Superclass(es)	Subclasses
Responsibility	Collaborators	

CRC



Mémento des notations UML

Ce document résume les notations UML 2.0 utilisées dans le cours d'UML de l'Institut Paul Lambin.

Les diagrammes présentés sur fond gris ne font pas partie d'UML.