

Unified Modeling Language

"langage de modélisation unifié"

F. Armetta (MCF) S. Hassas (PR)

UFR Informatique, Département informatique ISTIL
Université Claude Bernard Lyon1

Conventions typographiques

Exemple

- Ceci est un exemple
- ...

Attention

- Point important à retenir
- ...

Plan du cours

1 Introduction

2 Point de vue Fonctionnel (Client)

- Les diagrammes de cas d'utilisation
- Un diagramme de cas (version préliminaire)
- Expliciter les cas d'utilisation
- Approfondissements

3 Point de vue Statique

- Diagrammes de Classes
- Diagrammes d'Objets

4 Point de vue Dynamique

- Diagrammes de Séquence
- Diagrammes de Collaboration
- Diagrammes d'Etat

Plan du cours

1 Introduction

2 Point de vue Fonctionnel (Client)

- Les diagrammes de cas d'utilisation
- Un diagramme de cas (version préliminaire)
- Expliciter les cas d'utilisation
- Approfondissements

3 Point de vue Statique

- Diagrammes de Classes
- Diagrammes d'Objets

4 Point de vue Dynamique

- Diagrammes de Séquence
- Diagrammes de Collaboration
- Diagrammes d'Etat

Plan du cours

1 Introduction

2 Point de vue Fonctionnel (Client)

- Les diagrammes de cas d'utilisation
- Un diagramme de cas (version préliminaire)
- Expliciter les cas d'utilisation
- Approfondissements

3 Point de vue Statique

- Diagrammes de Classes
- Diagrammes d'Objets

4 Point de vue Dynamique

- Diagrammes de Séquence
- Diagrammes de Collaboration
- Diagrammes d'Etat

Plan du cours

1 Introduction

2 Point de vue Fonctionnel (Client)

- Les diagrammes de cas d'utilisation
- Un diagramme de cas (version préliminaire)
- Expliciter les cas d'utilisation
- Approfondissements

3 Point de vue Statique

- Diagrammes de Classes
- Diagrammes d'Objets

4 Point de vue Dynamique

- Diagrammes de Séquence
- Diagrammes de Collaboration
- Diagrammes d'Etat

Plan du cours

1 Introduction

2 Point de vue Fonctionnel (Client)

- Les diagrammes de cas d'utilisation
- Un diagramme de cas (version préliminaire)
- Expliciter les cas d'utilisation
- Approfondissements

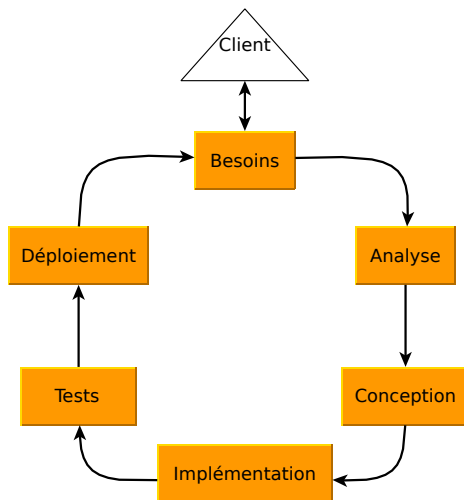
3 Point de vue Statique

- Diagrammes de Classes
- Diagrammes d'Objets

4 Point de vue Dynamique

- Diagrammes de Séquence
- Diagrammes de Collaboration
- Diagrammes d'Etat

Cycle de vie du logiciel



Développement Logiciel

Tâches à réaliser

- Comprendre et conceptualiser le problème (besoins, analyse)
- Résoudre le problème (conception)
- Donner une solution (implémentation)
- Documenter
- etc.

UML comme un outil

- Spécifier, visualiser et comprendre le problème
- Capturer, communiquer et utiliser des connaissances pour la résolution du problème
- Spécifier, visualiser et construire la solution
- Documenter la solution

Développement Logiciel

Tâches à réaliser

- Comprendre et conceptualiser le problème (besoins, analyse)
- Résoudre le problème (conception)
- Donner une solution (implémentation)
- Documenter
- etc.

UML comme un outil

- Spécifier, visualiser et comprendre le problème
- Capturer, communiquer et utiliser des connaissances pour la résolution du problème
- Spécifier, visualiser et construire la solution
- Documenter la solution

Développement Logiciel

Attention

- UML n'est cependant pas une méthodologie !
- Qu'est-ce qu'une méthodologie ?

Exemple

Connaissez-vous des méthodologies ?

Développement Logiciel

Attention

- UML n'est cependant pas une méthodologie !
- Qu'est-ce qu'une méthodologie ?

Exemple

Connaissez-vous des méthodologies ?

Développement Logiciel

Définition Méthodologie

- Etude de la méthode
- Méthodologie \Leftrightarrow Méthode : abus de langage

Exemple : Méthodologie \Leftrightarrow Méthode

- RUP (*Rational Unified Process*)
- OOSE (*Object-Oriented Software Engineering, Jacobson*)
- OMT (*Object Modeling Technique, Rumbaugh*)

Développement Logiciel

Définition Méthodologie

- Etude de la méthode
- Méthodologie \Leftrightarrow Méthode : abus de langage

Exemple : Méthodologie \Leftrightarrow Méthode

- RUP (*Rational Unified Process*)
- OOSE (*Object-Oriented Software Engineering, Jacobson*)
- OMT (*Object Modeling Technique, Rumbaugh*)

UML : Unified Modeling Language

- **Langage de description objet** de modèles matériels ou logiciels
- Système de notation pour matérialiser les concepts orientés objet
- Visualisation sous **différents axes et vues**
- Utilisé pour **construire et documenter** la solution

Les vues d'UML

Nous verrons :

Vue Utilisateur Buts et objectifs des clients du systèmes,
Besoins requis par la solution

Vue structurelle Aspects statiques représentant la structure du
problème

Vue Comportementale Aspects dynamiques du comportement
du problème et de sa solution

Les vues d'UML

Il existe aussi :

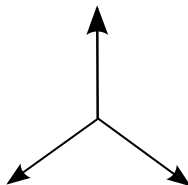
Vue implémentation Aspects de la structure et du comportement de la solution

Vue environnementale Aspects de la structure et du comportement du domaine dans lequel est réalisée la solution

3 axes de modélisation

Point de vue Fonctionnel (Client)

- Diagramme de cas d'utilisation
(diagramme de Séquences)
(diagramme d'Activité)



Point de vue Statique

- Diagramme de classes
- Diagramme d'objets

Point de vue Dynamique

- Diagramme de Séquence
- Diagramme de collaboration
- Diagramme d'Activité
- Diagramme d'Etats

Plan du cours

1 Introduction

2 Point de vue Fonctionnel (Client)

- Les diagrammes de cas d'utilisation
- Un diagramme de cas (version préliminaire)
- Expliciter les cas d'utilisation
- Approfondissements

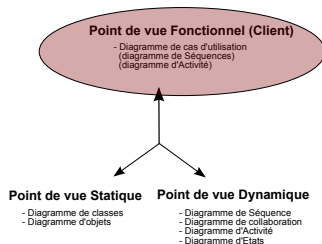
3 Point de vue Statique

- Diagrammes de Classes
- Diagrammes d'Objets

4 Point de vue Dynamique

- Diagrammes de Séquence
- Diagrammes de Collaboration
- Diagrammes d'Etat

Point de vue Utilisateur



- Pour **représenter les besoins**
- Principaux diagrammes associés :
 - **Diagramme de Cas d'Utilisation**
 - *Diagramme de Séquence*
 - *Diagramme d'Activité*

Point de vue Utilisateur

Attention !!

Point de vue dissocié du modèle d'implémentation objet :

- Décomposition fonctionnelle (et non objet)
- Description "externe" (Utilisateur) du comportement du système
- Le point de vue Utilisateur est un **point de vue descriptif "à part"**
 - Les entités de ses diagrammes ne seront pas toujours incarnées dans le modèle objet

Les diagrammes de cas d'utilisation

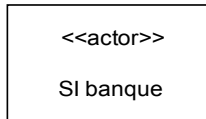
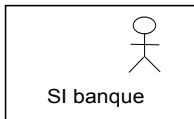
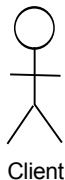
Exprime les **relations entre acteurs et cas d'utilisations** :

Les acteurs Entités en interaction avec le système

Les cas d'utilisation Fonctionnalités du système pour les acteurs

Les acteurs

- **Externes au système** (humain, autre système informatique, etc.)
- **Principal ou secondaire** (respectivement à gauche et droite du diagramme)
- Différentes représentations :



Les cas d'utilisation

- **Un point de vue acteur** (des fonctions du métier)
- **Représentation exhaustive des fonctionnalités** (conformément au cahier des charges)

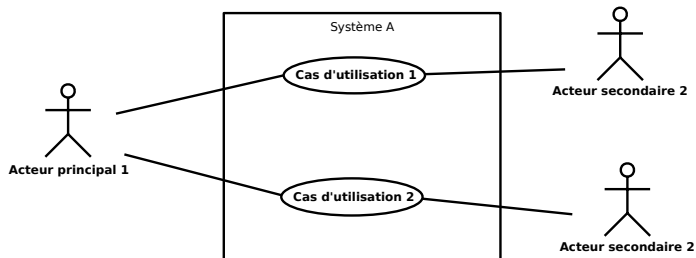


Diagramme de cas : le GAB

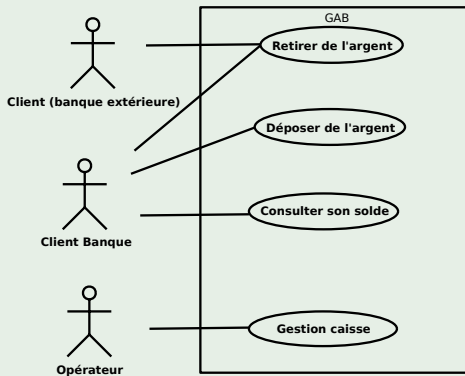
Exercice :

Réaliser un diagramme de cas d'utilisation pour un Guichet Automatique de Banque (GAB).

Méthode :

- 1 Identifier les acteurs
- 2 Identifier les cas d'utilisation
- 3 Réaliser les associations

cas d'utilisation d'un GAB



Expliciter les cas d'utilisation

Compléter le diagramme de cas, afin d'exprimer clairement le fonctionnement global.

Documenter

Décrire les acteurs (rôles, responsabilités) **et les cas d'utilisation** (phrases courtes et détaillées).

Les acteurs

Acteur	Description	Cas d'utilisation
Client banque	Client de la banque. Est autorisé à retirer de l'argent si le solde de son compte de dépôt le permet, etc.	1, 2, 3
	...	

Les cas d'utilisation

N°	Cas d'utilisation	Description
1	Retirer de l'argent	s'identifier, entrer le montant de la somme et récupérer les billets
	...	

Diagrammes complémentaires

- **Diagrammes de séquence et d'activité**
- Explicitent les diagrammes de cas
- Ces diagrammes sont issus de l'axe dynamique d'UML
- **Utilisation spécifique, simplifiée** pour le point de vue Utilisateur
 - Le point de vue Utilisateur est "à part"
 - Dans ce cadre, ces diagrammes sont décorrélés du modèle objet sous-jacent

Diagramme de séquence

Intérêt : Déroulement chronologique d'un scénario

Entités : Les acteurs et le système

Messages : Echanges d'informations entre entités

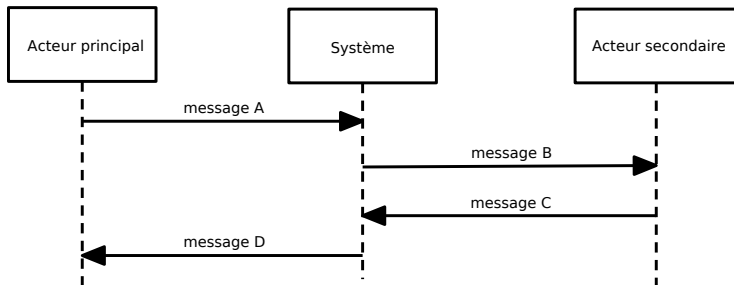


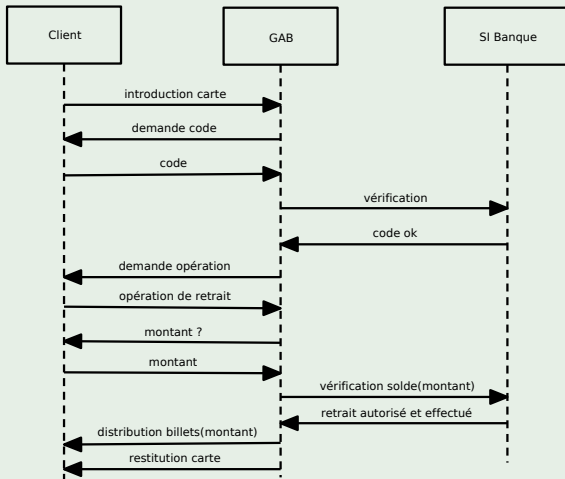
Diagramme de séquence

Représenter le scénario suivant :

Un client se présente au GAB, et y retire une somme d'argent.

Diagramme de séquence

Une possibilité :



- Permet de représenter les activités et leurs enchainements possibles
- Représentation proche du langage algorithmique

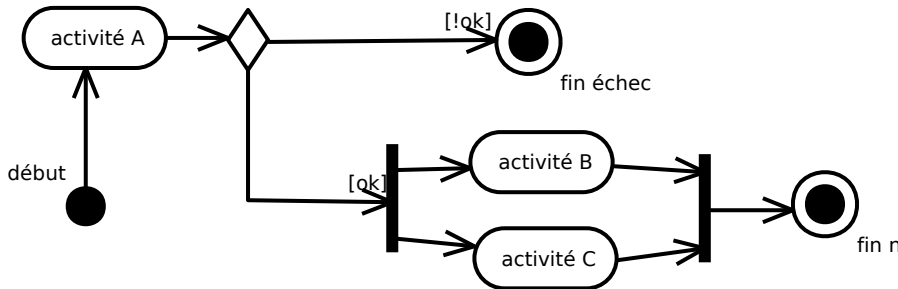


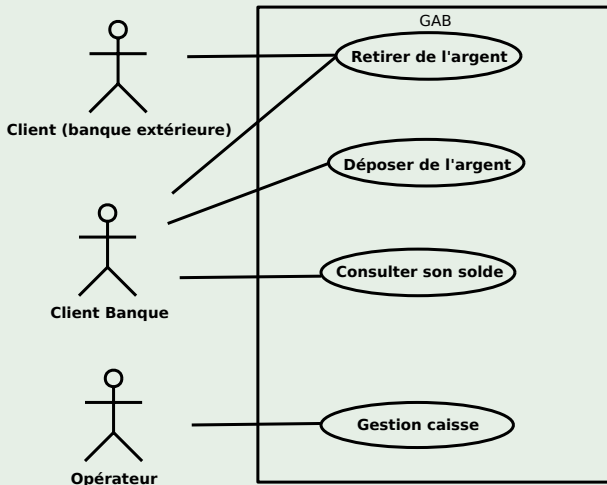
Diagramme de processus de transaction avec carte bancaire. Le processus commence par un état initial (cercle noir) menant à l'activité "début transaction".

Le processus se déroule comme suit :

- Activité "reconnaissance carte" mène à un point de décision.
- Si la carte est valide ([carte valide]), le processus passe à l'activité "vérification code".
- Après "vérification code", un point de décision dirige le processus :
 - Si la carte n'est pas valide ([carte non valide]), le processus se termine à l'état final "fin : transaction annulée".
 - Si la carte est valide, le processus passe à l'activité "demande opération".
- Après "demande opération", un point de décision dirige le processus :
 - Si l'opération est un retrait ([opération retrait]), le processus passe à l'activité "demande montant".
 - Si l'opération n'est pas un retrait, le processus se termine à l'état final "fin : transaction annulée".
- Après "demande montant", un point de décision dirige le processus :
 - Si le montant est supérieur au solde ([montant > solde]), le processus se termine à l'état final "fin : transaction annulée".
 - Si le montant est inférieur ou égal au solde ([montant <= solde]), le processus passe à un point de synchronisation.
- Le point de synchronisation mène à deux activités parallèles : "distribution billets" et "restitution carte".
- Après ces activités, un point de synchronisation mène à l'état final "fin nominale".

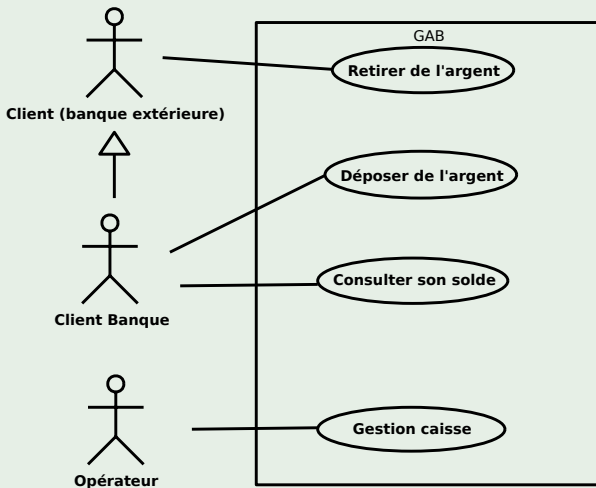
Généralisation des acteurs

Généraliser le diagramme suivant :

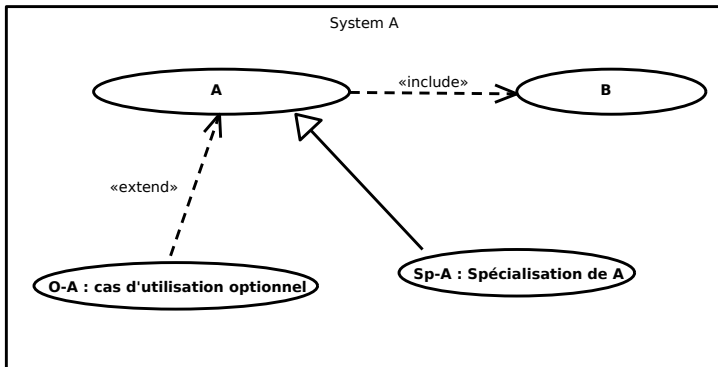


Généralisation des acteurs

Généralisation :



Relations entre cas d'utilisation



Inclusion («include») : A implique B

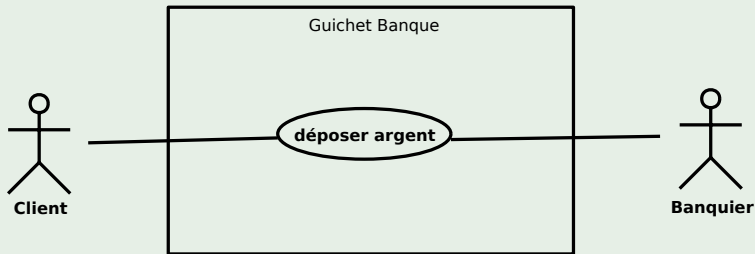
Extension («extend») : A peut provoquer O-A

Généralisation : Sp-A est un cas particulier de A

Relations entre cas d'utilisation

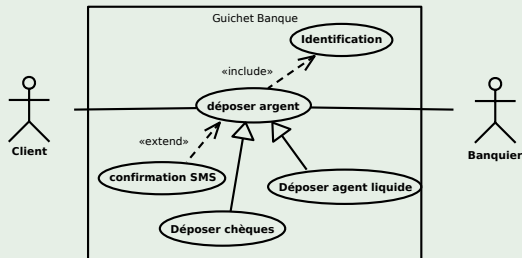
Compléter le diagramme :

- 1 Ajouter le cas d'utilisation "Identification"
- 2 Ajouter les cas d'utilisation "Déposer chèques" et "Déposer argent liquide"
- 3 Ajouter le cas d'utilisation "Confirmation SMS"



Relations entre cas d'utilisation

Diagramme approfondi :



Le modèle objet dans tout ça ?

Rappel :

- Les diagrammes fonctionnels (cas d'utilisations, etc.) sont "à part"
- A utiliser pour documenter, vue client
- Même si la représentation approfondie s'inspire du modèle objet, l'**implémentation objet concrète** est souvent **différente et dissociée**

Plan du cours

1 Introduction

2 Point de vue Fonctionnel (Client)

- Les diagrammes de cas d'utilisation
- Un diagramme de cas (version préliminaire)
- Expliciter les cas d'utilisation
- Approfondissements

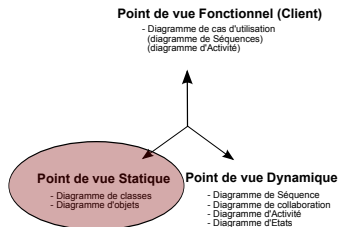
3 Point de vue Statique

- Diagrammes de Classes
- Diagrammes d'Objets

4 Point de vue Dynamique

- Diagrammes de Séquence
- Diagrammes de Collaboration
- Diagrammes d'Etat

Point de vue Statique



- Pour **Représenter la structure**
- Principaux diagrammes associés :
 - **Diagramme de Classes**
 - *Diagramme d'Objets*

Diagrammes de Classes

- Décrivent la **structure statique** du système
- Les diagrammes doivent être **consistants** entre eux : la structure statique sera compatible avec les autres diagrammes (statiques ou dynamiques)

Rappel modèle Objet

Modèle objet \Rightarrow on cherche à :

- 1 Identifier les types d'objet manipulés : **classes**
- 2 Identifier leurs données et leurs fonctions : **attributs** et **méthodes**
- 3 Identifier les liens entre les types d'objets (ou classes) : **associations**
- 4 Identifier leur points communs et leurs différences : **généralisation** et **spécialisation** par le graphe d'héritage

Rappel modèle Objet

Modèle objet \Rightarrow on cherche à :

- 1 Identifier les types d'objet manipulés : **classes**
- 2 Identifier leurs données et leurs fonctions : **attributs** et **méthodes**
- 3 Identifier les liens entre les types d'objets (ou classes) : **associations**
- 4 Identifier leur points communs et leurs différences : **généralisation** et **spécialisation** par le graphe d'héritage

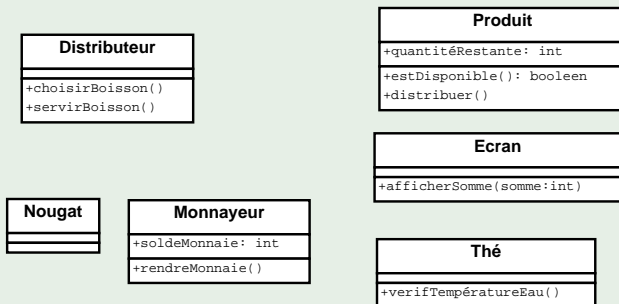
Rappel modèle Objet

Modèle objet \Rightarrow on cherche à :

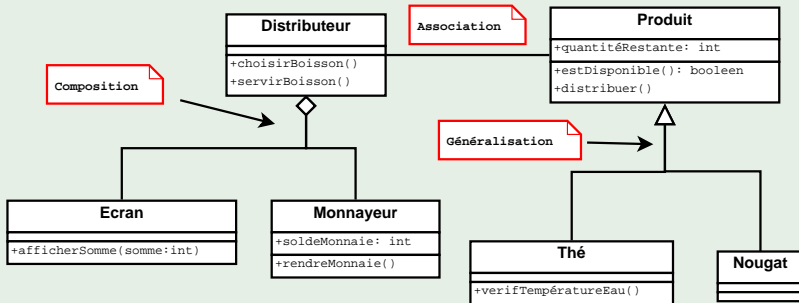
- 1 Identifier les types d'objet manipulés : **classes**
- 2 Identifier leurs données et leurs fonctions : **attributs** et **méthodes**
- 3 Identifier les liens entre les types d'objets (ou classes) : **associations**
- 4 Identifier leur points communs et leurs différences : **généralisation** et **spécialisation** par le graphe d'héritage

Etablir les relations

Identifier les relations d'**association** et les relations d'**héritage**



Etablir les relations



Formalisme

Les Classes

- Nom de la classe
- Attributs typés (exemple : "*solde : int*")
- méthodes typées (exemple : "*estDisponible(qt : int) : booléen*")
- Visibilité et Encapsulation (exemple : " ?? *solde : int*")
 - + \Rightarrow public
 - - \Rightarrow privé
 - # \Rightarrow protected
- Valeur initiale : "*+ solde : int = 12*")

Formalisme

Les Classes

- Nom de la classe
- Attributs typés (exemple : "*solde : int*")
- méthodes typées (exemple : "*estDisponible(qt : int) : booléen*")
- Visibilité et Encapsulation (exemple : " ?? *solde : int*")
 - + \Rightarrow public
 - - \Rightarrow privé
 - # \Rightarrow protected
- Valeur initiale : "*+ solde : int = 12*")





Formalisme

Les Classes

- Nom de la classe
- Attributs typés (exemple : "*solde : int*")
- méthodes typées (exemple : "*estDisponible(qt : int) : booléen*")
- Visibilité et Encapsulation (exemple : " ?? *solde : int*")
 - + \Rightarrow public
 - - \Rightarrow privé
 - # \Rightarrow protected
- Valeur initiale : "*+ solde : int = 12*")

Formalisme

Les **Relations** entre classes

- Généralisation : 
- Agrégation : 
- Composition : 
Lien plus fort que l'agrégation :
destruction de l'élément composé => destruction des composants
- Association : 

Formalisme

La **cardinalité** des associations

1 un et un seul

0..1 zéro ou un

M..N de M à N (M et N entiers naturels)

* ou 0..* de zéro à plusieurs

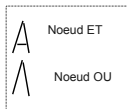
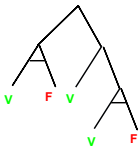
1..* de 1 à plusieurs

N exactement N (N entier naturel)

Application

Exemple

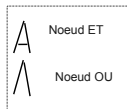
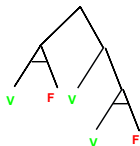
- Proposer un modèle objet (axe statique), concernant la manipulation d'un arbre ET/OU
- - 1 Identifier les **classes** à instancier
 - 2 Identifier les **relations l'héritage** entre ces classes (généralisation/spécialisation)
 - 3 Identifier **attributs**



Application

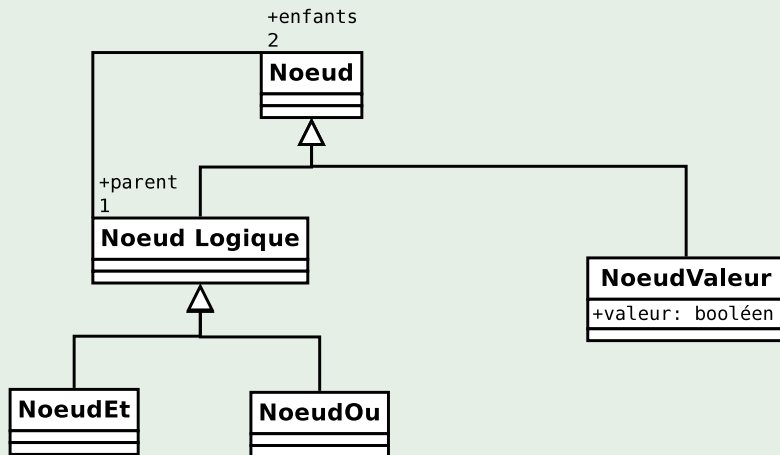
Exemple

- Proposer un modèle objet (axe statique), concernant la manipulation d'un arbre ET/OU
- - 1 Identifier les **classes** à instancier
 - 2 Identifier les **relations l'héritage** entre ces classes (généralisation/spécialisation)
 - 3 Identifier **attributs**



Application

Exemple



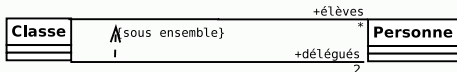
Les contraintes

- Une contrainte porte sur une relation ou un groupe de relations
- Notation : {*description contrainte*}

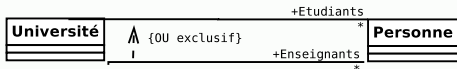
Exemple : Ordre



Exemple : Sous-ensemble



Exemple : Exclusivité



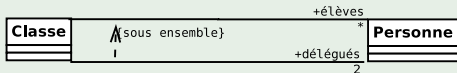
Les contraintes

- Une contrainte porte sur une relation ou un groupe de relations
- Notation : {*description contrainte*}

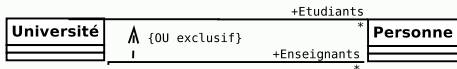
Exemple : Ordre



Exemple : Sous-ensemble



Exemple : Exclusivité



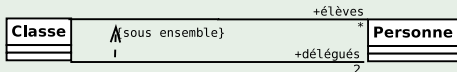
Les contraintes

- Une contrainte porte sur une relation ou un groupe de relations
- Notation : {*description contrainte*}

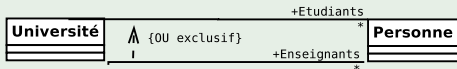
Exemple : Ordre



Exemple : Sous-ensemble



Exemple : Exclusivité

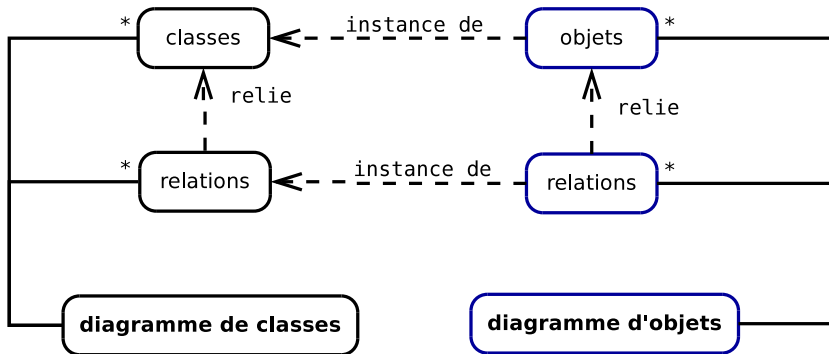


Diagrammes de classes et diagrammes d'objets

Les diagrammes de classes Structure du modèle sous la forme de **classes** et de **relations entre classes**

Les diagrammes d'objets Illustration par une instance du modèle (**objets** et **liens** qui les unissent)

Diagrammes de classes et diagrammes d'objets



(extrait simplifié du méta-modèle d'UML)

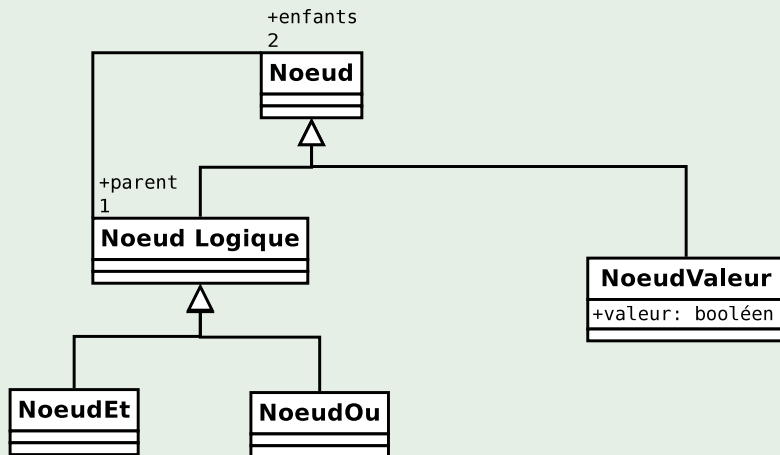
Diagrammes de classes et diagrammes d'objets

Attention !

- Toutes les classes ne sont pas toujours instanciées en objets
- Une classe peut être instanciée plusieurs fois dans le diagramme objet

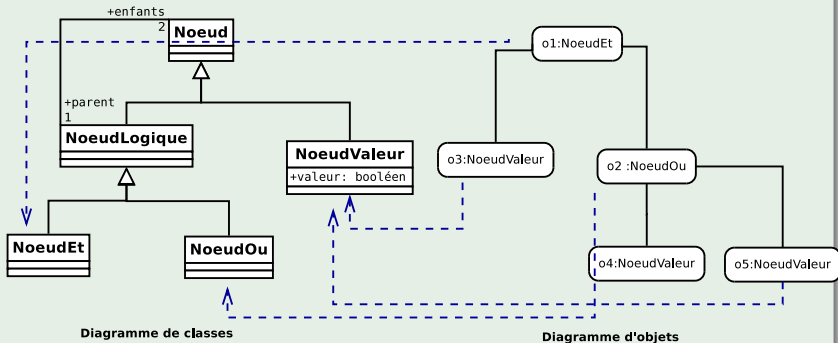
Application

Donner un diagramme d'objets



Application

Donner un diagramme d'objets



Diagrammes d'Objets

- Un diagramme d'objet décrit aussi un "état mémoire" du système
- \Rightarrow on peut choisir de représenter certains attributs d'objet pertinents

