



Prof. Khadija Bouzaachane

## Plan

2

- Notion de modélisation
- Techniques de modélisation orientée objets
- UML
- Diagrammes
- Axe Statique
- Axe Dynamique

## Notion de modélisation

### Qu'est-ce qu'un modèle ?

- Un **modèle** est une **représentation abstraite et simplifiée** d'une partie du système réel avec un but spécifique.
  - Le système réel est complexe, alors il est nécessaire d'une simplification.
- Un modèle est souvent une **représentation visuelle simplifiée** du système réel.
- Un **modèle** représente le système:
  - à certain niveau d'abstraction
  - par des moyens de description (texte, image, ...)
- La **modélisation** est le processus de représenter un système par des modèles.

## Modélisation: pour quoi?

- Mieux comprendre le système.
- Mieux construire le système.
- Facilite la communication
  - Fournir des moyens de communication entre des développeurs

### Un bon modèle devrait

- Utiliser une notation standardisée.
- Être compréhensible pour les clients et les utilisateurs.
- Permettre aux ingénieurs logiciels de bien saisir le système.

### En génie logiciel

- Modélisation = spécification + conception
- Aider la réalisation d'un logiciel à partir des besoins du client

## Naissance d' UML

5

- **1993-1994:** **Bloch' 93, OMT-2**
  - Les 2 méthodes sont **leaders** sur le marché
  - Elles sont de plus en plus **proches**
- **Octobre 1994:**
  - **J. Rumbaugh** (OMT) rejoint **G. Booch** chez **Rational Software**
  - Annonce de l' unification des deux méthodes
- **Octobre 1995:** La première version sort sous le nom **Unified Method v0.8**
- **Fin 1995:** le fondateur d' **Objectory, Ivar Jacobson**, rejoint à son tour **Rational Software**
- **Janvier 97 :** **Soumission** à l' **OMG** de la version UML 1.0
  - **OMG: Object Management Group ([www.omg.org](http://www.omg.org)) :**
    - Organisme à but non lucratif fondé en 1989.
    - l'objectif est de standardiser et promouvoir le modèle objet sous toutes ses formes.
    - Plus de 700 entreprises y adhèrent.
    - Connue pour la norme CORBA.
- **Septembre 97 : UML 1.1** standardisé par l'OMG

## Naissance d' UML

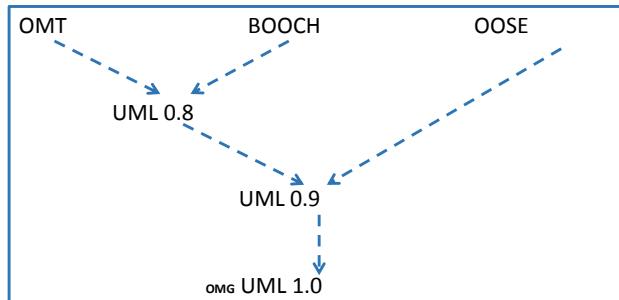
- **UML** : Prendre le **meilleur** de chacune des méthodes
  - **OOSE** (Jacobson) : **Use Cases**
  - **OMT** (Rumbaugh) : **Analyse**
  - **Bloch** (Booch) : **Conception, Architecture**
- UML est dans le **domaine public**
- Soutenu par le **marché**
  - Microsoft, HP, Oracle, IBM...

## Naissance d' UML

7

- Trois fondateurs principaux :

- Grady BOOCH (BOOCH)
- James Rumbaugh (OMT)
- Ivar Jacobson (OOSE)



## UML ?

- **UML : Unified Modeling Language**
- **UML est une notation, pas une méthode**
- **UML est un langage** composé du vocabulaire, de la syntaxe et de la sémantique
- **UML est un langage de modélisation objet**
- **UML convient à tous** les langages objets:
  - C++ (Héritage multiple, Template)
  - Java (Interface)
  - SmallTalk
- **UML est un langage de visualisation**
  - Utiliser des représentations graphiques
  - Apporter une meilleure vue du système (en utilisant des représentations graphiques)

## UML ?

9

- **UML** n'est pas une méthodologie ou un processus
- **UML** peut être combiné avec plusieurs processus de développement
- **UML est un langage de visualisation**
  - Utiliser des représentations graphiques
  - Apporter une meilleure vue du système (en utilisant des représentations graphiques)

## UML: Unified Modeling Language

- **Langage :**
  - **Syntaxe** et règles d'écriture
  - Notations graphiques **normalisées**
  - **de modélisation :**
  - **Abstraction** du fonctionnement et de la structure du système
  - **Spécification et conception**
  - **unifié :**
  - Fusion de plusieurs notations antérieures : Booch, OMT, OOSE
  - **Standard** défini par l'OMG (Object Management Group)
  - Dernière version : UML 2.5 (Mai 2015)
- En résumé :** Langage graphique pour visualiser, spécifier, construire et documenter un logiciel.

## UML : les modèles

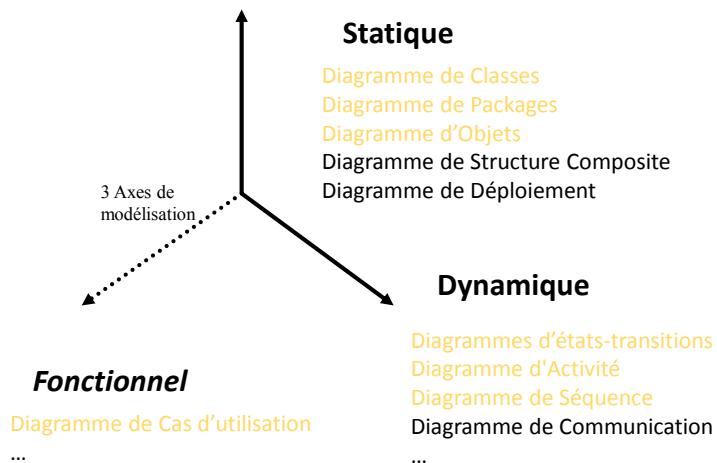
UM 2.0 se compose de 13 diagrammes

- **Modélisation des besoins**
  - Diagrammes de cas d'utilisation (Use case diagram)
- **Modélisation de la structure statique**
  - Diagrammes de classes (Class diagram)
  - Diagrammes d'objets (Object diagram)
- **Modélisation du comportement dynamique**
  - Diagrammes d'interaction (Interaction diagram)
  - Diagrammes de séquence (Sequence diagram)
  - Diagramme de communication (Communication diagram)
  - Diagrammes d'activité (Activity diagram)
  - Diagrammes d'états-transitions (State machine diagram)
  - Diagramme de temps (Timing diagram)

## UML : les modèles

- **Modélisation de l'architecture**
  - Diagrammes de composants (Component diagram)
  - Diagrammes de déploiement (Deployment diagram)
  - **Diagramme de packages (Package diagram)**
  - Diagramme de structures composites (Composite structure diagram)

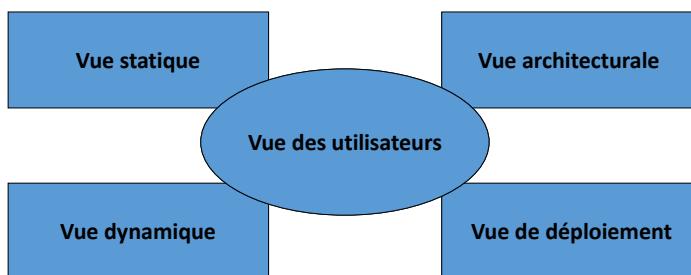
## Axe de Modélisation



## Les Vues

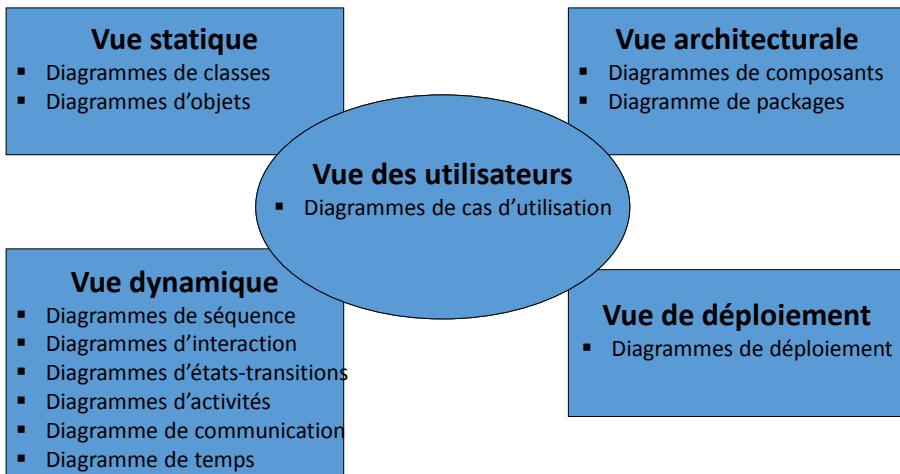
14

Un système se modélise en 5 vues différentes



## Les 4+1 Vues

### Diagrammes & Vues



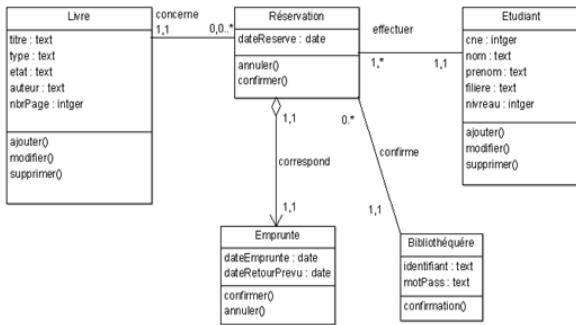
L' Axe Statique

## Diagrammes de classes/Class diagrams

- Décrire des classes et leurs relations.
- Décrire la vue statique du système.

### Exemple:

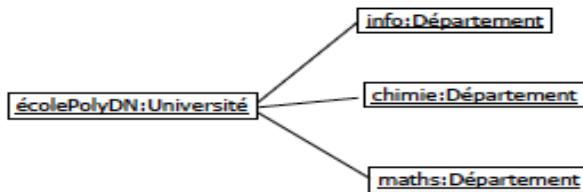
Dans cet exemple on a cinq classes : Livre, Réservation, Etudiant, Bibliothécaire, et Emprunte.



## Diagrammes d'objets/Object diagrams

- Décrire un ensemble des objets et leurs interactions.
- Un diagramme d'objets représentent les mêmes informations qu'un diagramme de classes mais en vue des instances des classes.

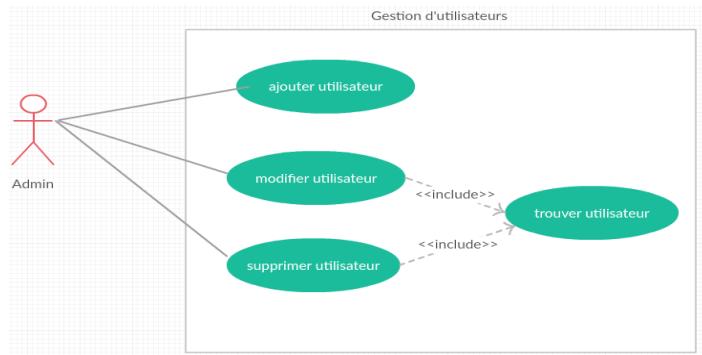
### Exemple



## Diagramme de cas d'utilisation/Use case diagrams

- Montrer les utilisations possibles d'un système.
- Décrire la vue statique du système selon **le point de vue des utilisateurs**.
- Très important pour saisir les fonctions du système.

**Exemple:**



## L' Axe Dynamique

## Diagrammes d'interaction/Interaction diagrams

- Décrire le comportement du système par les interactions entre des objets qui le composent.
- Montrer **la vue dynamique** du système.
- Les diagrammes d'interactions sont une extension des diagrammes d'objets en précisant les interactions entre les objets.
- Composer de deux types de diagramme:
  - **Diagrammes de séquence**
  - **Diagrammes de collaboration**

## Diagrammes d'interaction/Interaction diagrams

### Diagrammes de séquence

Décrire les interactions entre les objets en mettant l'accent sur le séquencement des messages.

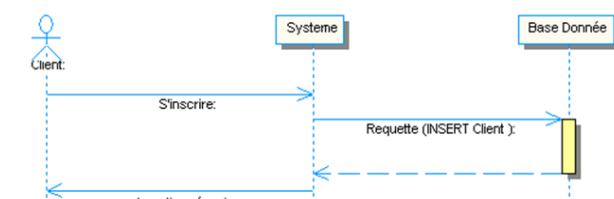
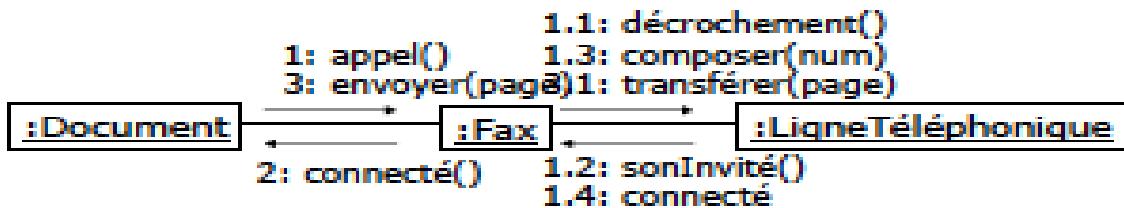


Diagramme de séquence pour l'inscription d'un client

## Diagrammes d'interaction/Interaction diagrams

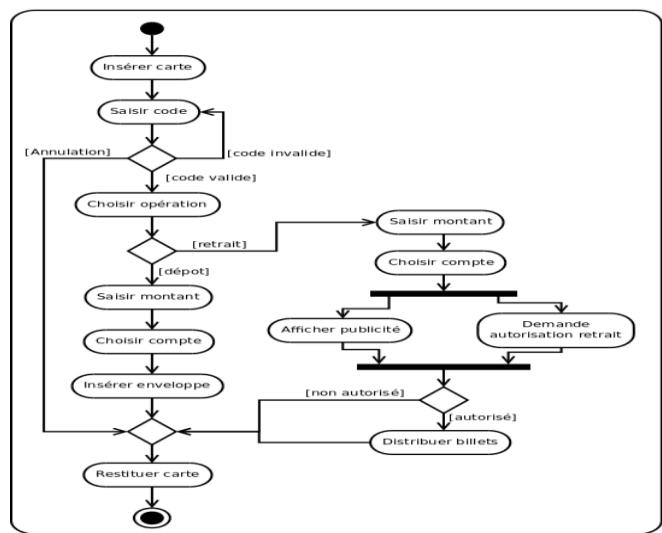
### Diagrammes de collaboration (ou communication)

Décrire les interactions entre les objets en mettant l'accent sur la structure des objets. permet de mettre en évidence les échanges de messages entre objets. Cela nous aide à voir clair dans les actions qui sont nécessaires pour produire ces échanges de messages. Et donc de compléter, si besoin, les diagrammes de séquence et de classes.



## Diagrammes d'activités/Activity diagrams

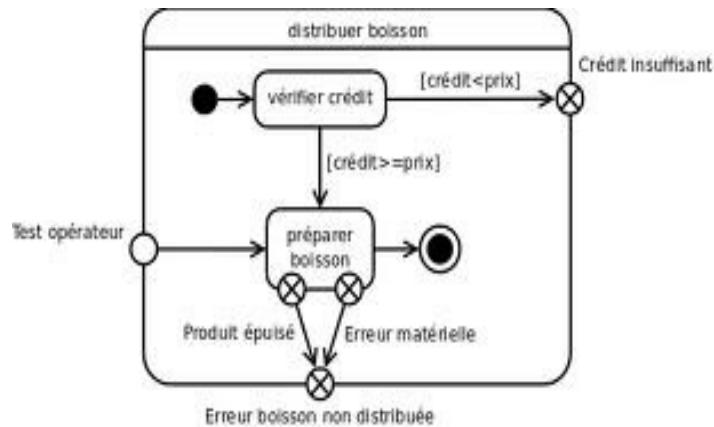
- Décrire les flots de l'information dans le système.
- Représenter le déroulement des actions, sans utiliser les objets.
- La vue dynamique du système.



## Diagrammes d'états/State diagrams

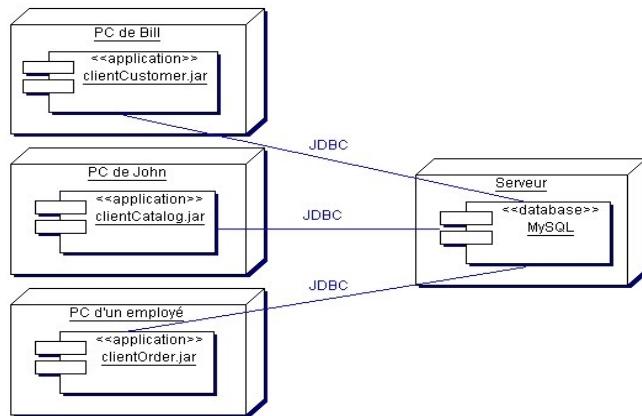
- Décrire comment le système se comporte de façon interne.
- La vue statique du système.
- permet de décrire le cycle de vie des objets d'une classe.

### Exemple



## Diagrammes de déploiement/Deployment diagrams

- Décrire l'organisation physique de différents composants (machines) du système (matériel).
- correspond à la description de l'environnement d'exécution du système (matériel, réseau...) et de la façon dont les composants y sont installés.



# Diagramme de cas d'utilisation

## Diagramme de cas d'utilisation

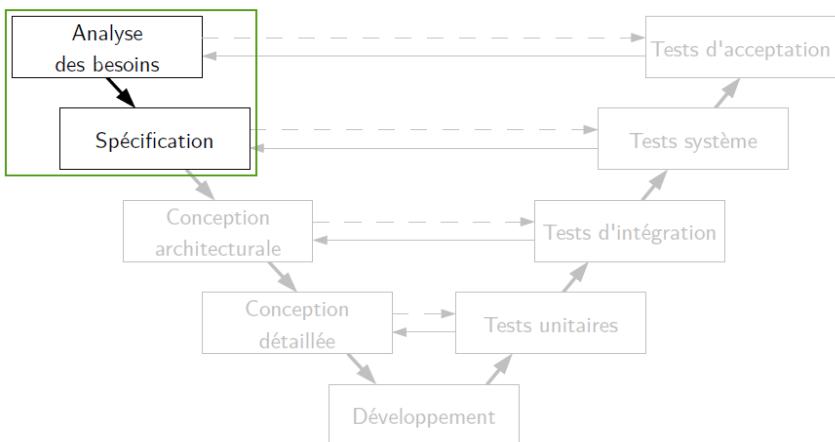
Les **diagrammes de cas d'utilisation** sont créés pour :

- Visualiser les interactions entre le système et le monde extérieur.
- Présenter les concepts UML relatifs à la vue fonctionnelle.
- Modéliser très tôt les processus métiers et l'organisation de l'entreprise.
- Ils se basent sur des diagrammes où des acteurs interagissent avec le système de l'extérieur ou de l'intérieur.

## Processus de développement logiciel

29

### Diagramme de cas d'utilisation



## Diagramme de cas d'utilisation

### Acteurs

- **Un acteur** représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié.
- **Un acteur** peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.
- **Un acteur** est identifié par le **nom du rôle**.

## Acteurs

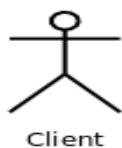
### Comment les identifier ?

Les acteurs candidats sont systématiquement :

- **Les utilisateurs humains** directs : faites donc en sorte d'identifier tous les profils possibles, sans oublier l'administrateur, l'opérateur de maintenance, etc. ;
- **Les autres systèmes** connexes qui interagissent aussi directement avec le système étudié, souvent par le biais de protocoles bidirectionnels(systèmes informatiques ou hardware).

## Acteurs

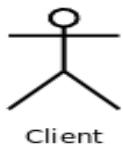
- Chaque acteur doit être nommé. Ce nom doit refléter son rôle.
- Représentation graphique standard de l'acteur en UML est l'icône appelée **stick man**, avec le nom de l'acteur sous le dessin.
- On peut également figurer un acteur sous un **classeur stéréotype**, avec le mot-clé `<<actor>>`.



## Acteurs

33

- Une bonne recommandation consiste à faire prévaloir l'utilisation de la forme graphique du **stick man** pour les acteurs humains et une représentation rectangulaire, **classeur stéréotypé**, pour les systèmes connectés.



## Types d'acteur

- **Les acteurs principaux** : entités qui utilisent les fonctions du système et agissent directement sur ce dernier. Il s'agit d'entités qui ont des besoins d'utilisation du système. On peut donc considérer que les futurs utilisateurs du logiciel sont les acteurs principaux.
- **Les acteurs secondaires** : entités qui effectuent des tâches administratives ou de maintenance et qui n'ont pas de besoin direct d'utilisation. Cela est généralement un autre système (logiciel) avec lequel le nôtre doit échanger des informations.

**Acteur principal** = acteur déclenchant le cas d'utilisation.

**Acteur secondaire** = acteur sollicité par le cas d'utilisation.

## Diagramme de cas d'utilisation

### Cas d'utilisation (use case)

- Les **cas d'utilisations (use-case)** sont des séquences d'actions menées par le système qui doit donner un résultat observable pour un acteur.
- Il est recommandé que l'intitulé du **cas d'utilisation** respecte **le pattern « verbe + compléments »**.
- Le verbe de l'intitulé permet de spécifier la nature de la fonctionnalité offerte par l'application, tandis que les compléments permettent de spécifier les données d'entrée ou de sortie de la fonctionnalité.

36

## Diagramme de cas d'utilisation

### Cas d'utilisation (use case)

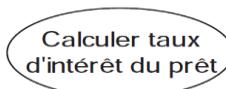
- **Fonctionnalités principales** du système du point de vue **extérieur**.
- **Action déclenchée** par un **acteur**.
- Identifié par une **action** (verbe à l'infinitif)

37

## Cas d'utilisation (use case)

### Exemple 1:

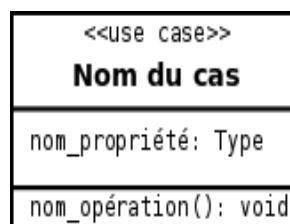
- Le cas d'utilisation "calculer taux d'intérêt du prêt" permet de comprendre d'une certaine manière que l'application permet à ses utilisateurs de calculer le taux d'intérêt d'un prêt.
- Un cas d'utilisation se représente par une ellipse contenant l'intitulé du cas d'utilisation.



## Cas d'utilisation (use case)

### Exemple 2:

- Dans le cas où l'on désire présenter les attributs ou les opérations du cas d'utilisation, il est préférable de le représenter sous la forme d'un classeur stéréotypé << use case >>.
- Représentation d'un cas d'utilisation sous la forme d'un classeur:



39

## Diagramme de cas d'utilisation

- ❑ **Acteur** : entité externe qui agit sur le système (opérateur, composant interne...).
- ❑ **Use case** : ensemble d'actions réalisées par le système, en réponse à une action d'un acteur. L'ensemble des use cases décrit les objectifs (le but) du système.



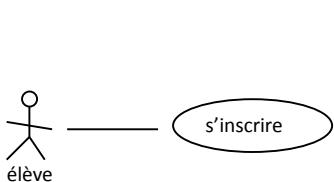
### ❑ Les relations de base entre cas d'utilisation

- ✓ « include »
- ✓ « extend »
- ✓ héritage

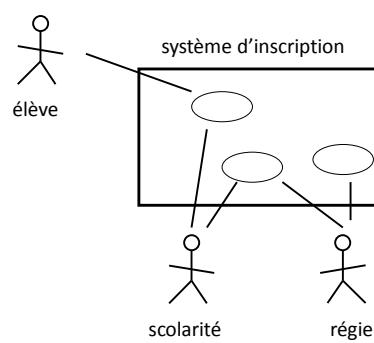
40

## Diagramme de cas d'utilisation

### Exemple:



Cas d'un élève qui s'inscrit



Plusieurs acteurs  
Plusieurs cas

41

## Diagramme de cas d'utilisation

### Système

- **Un système** représente une application dans le modèle UML. Il est identifié par un nom et regroupe un ensemble de cas d'utilisation qui correspondent aux fonctionnalités offertes par l'application à son environnement.
- **L'environnement** est spécifié sous forme d'acteurs liés aux cas d'utilisation.
- **Un système** se représente par un **rectangle** contenant le nom du système et les cas d'utilisation de l'application.

### Système

42

- **Les acteurs**, extérieurs au système, sont représentés et reliés aux cas d'utilisation qui les concernent. L'ensemble correspond à un diagramme de cas d'utilisation.

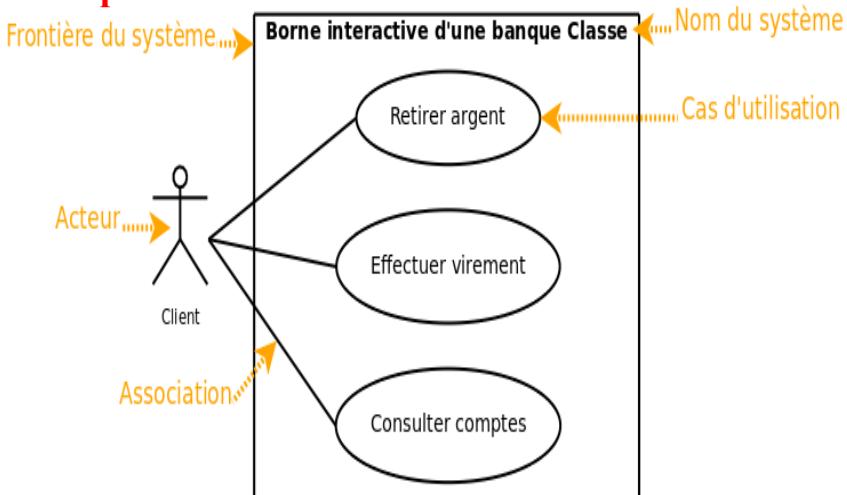
#### Exemple 1:

- La figure suivante représente un exemple simplifié de diagramme de cas d'utilisation modélisant une borne d'accès à une banque avec les cas d'utilisation offerts à l'acteur qui représente le client.

43

## Représentation d'un diagramme de cas d'utilisation

### Exemple :

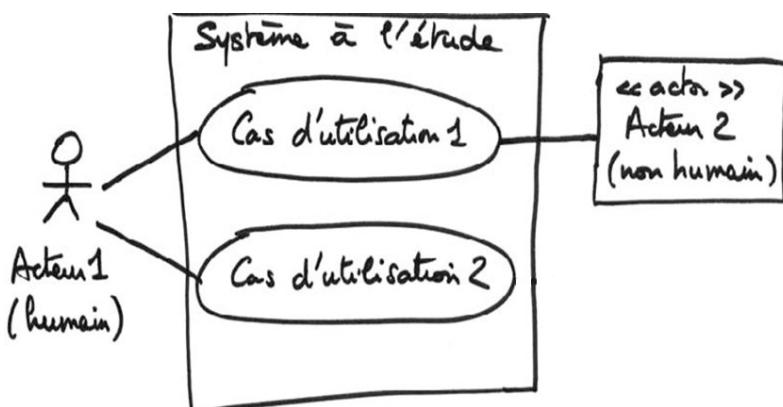


44

## Représentation d'un diagramme de cas d'utilisation

### Exemple 2:

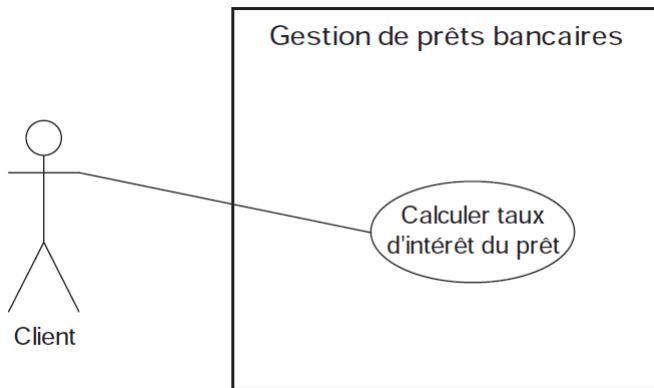
- Interactions fonctionnelles entre les acteurs.



45

## Représentation d'un diagramme de cas d'utilisation

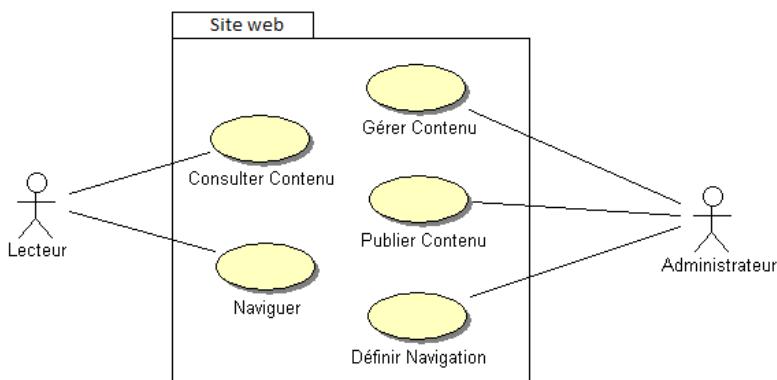
**Exemple :**



46

## Représentation d'un diagramme de cas d'utilisation

**Exemple :**



47

## Relations dans les diagrammes de cas d'utilisation

### Relations entre acteurs et cas d'utilisation

#### Relation d'association

- Une relation d'association est chemin de communication entre **un acteur** et **un cas d'utilisation**.
- Elle est représentée par **un trait continu**:



48

## Relations entre acteurs et cas d'utilisation

### Relation d'association

#### Multiplicité

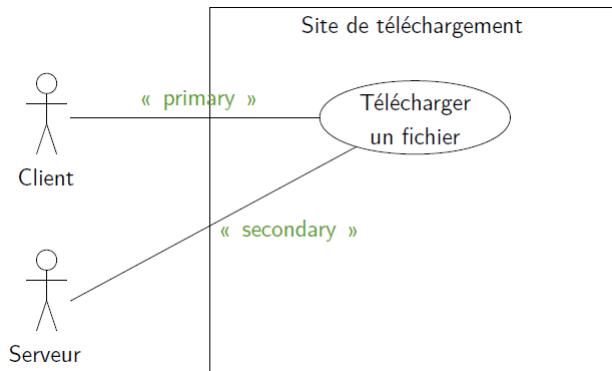
- Un acteur peut interagir plusieurs fois avec un cas d'utilisation,
- Il est possible d'ajouter une multiplicité sur l'association du côté du cas d'utilisation.
  - Le symbole **\*** signifie plusieurs,
  - **n...m** signifie entre **n** et **m**, etc.
  - Préciser une multiplicité sur une relation n'implique pas nécessairement que les cas sont utilisés en même temps.

49

## Relations entre acteurs et cas d'utilisation

### Relation d'association

**Exemple:**



50

## Relations dans les diagrammes de cas d'utilisation

### Relations entre Acteurs

- La seule relation possible entre **deux acteurs** est la généralisation :
- un acteur **A** est une généralisation d'un acteur **B** si l'acteur **A** peut être substitué par l'acteur **B**.
- Dans ce cas, tous les cas d'utilisation accessibles à **A** le sont aussi à **B**, mais l'inverse n'est pas vrai.

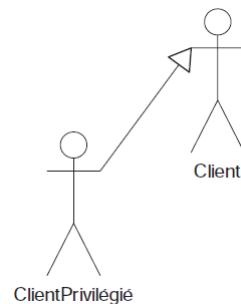
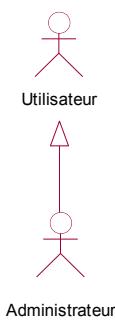
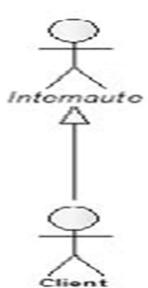
51

## Relations entre Acteurs

### Notation graphique:



- Le symbole utilisé pour la généralisation entre acteurs est une flèche avec un trait plein dont la pointe est un triangle fermé désignant l'acteur le plus général.



## Relations entre Acteurs

52

### Exemple 2:

Cette exemple représente une relation d'héritage entre l'acteur *Administrateur* et l'acteur *Utilisateur*.



53

## Relations entre cas d'utilisation

Il existe principalement deux types de relations :

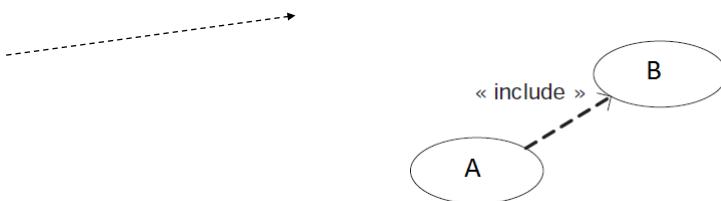
- Les dépendances (ou relations) stéréotypées, qui sont explicitées par un stéréotype (les plus utilisés sont l'**inclusion** et l'**extension**).
- **La généralisation/specialisation.**

54

## Relations entre cas d'utilisation

### *Représentation graphique inclusion/extension*

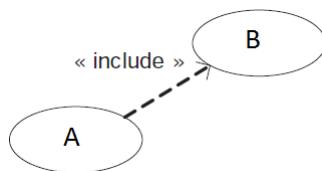
- Une dépendance se représente par **une flèche pointillée**. Si le cas A inclut ou étend le cas B, la flèche est dirigée de A vers B.



55

## Relations d'inclusion

- Un **cas A** inclut un **cas B** si le comportement décrit par **le cas A** inclut le comportement du **cas B** : **le cas A** dépend de **cas B**.  
Lorsque **A** est sollicité, **B** l'est obligatoirement, comme une partie de **A**.
- Cette dépendance est symbolisée par **le stéréotype <>include<>**.



## Relations d'inclusion

56

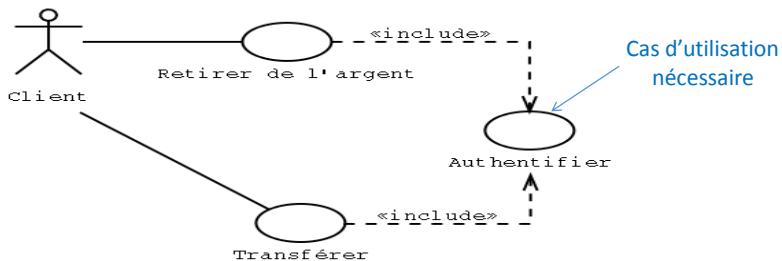
- Un **cas d'utilisation A** « **include** » un **cas d'utilisation B** c.à.d **cas d'utilisation A** implique **cas d'utilisation B**;
- ➔ **cas d'utilisation B** est **nécessaire** pour **cas d'utilisation A**.

57

## Relations d'inclusion

### Exemple :

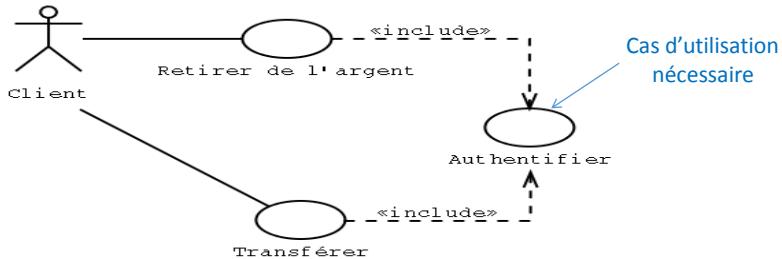
- Une opération de retrait et une opération de transfert nécessitent toutes deux une opération de vérification de l'identité du client.
- Effectuer un retrait **inclus** nécessairement une phase d'**authentification**.



## Relations d'inclusion

58

### Exemple :

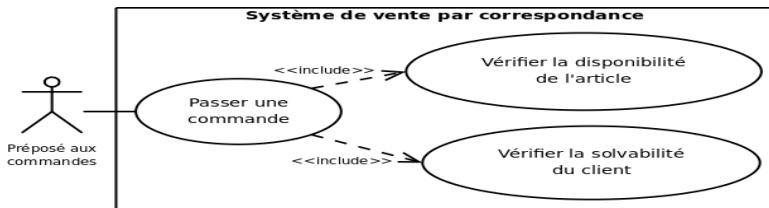


Les inclusions permettent essentiellement de factoriser une partie de la description d'un cas d'utilisation qui serait commune à d'autres cas d'utilisation (le cas **s'authentifier**).

59

## Relations d'inclusion

Les inclusions permettent également de décomposer un cas complexe en sous-cas plus simples:

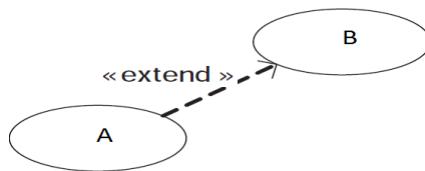


Cependant, il ne faut surtout pas abuser de ce type de décomposition : il faut éviter de réaliser du découpage fonctionnel d'un cas d'utilisation en plusieurs sous-cas d'utilisation pour ne pas retomber dans le travers de la décomposition fonctionnelle.

60

## Relations d'extension

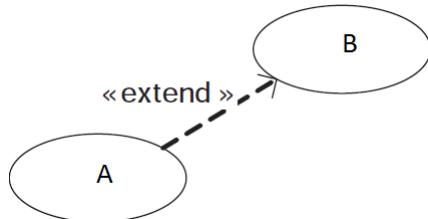
- La relation d'**extension** est probablement la plus utile car elle a une sémantique qui a un sens du point de vue métier au contraire des deux autres qui sont plus des artifices d'informaticiens.
- On dit qu'un **cas d'utilisation A** étend un **cas d'utilisation B** lorsque le **cas d'utilisation A** peut être appelé au cours de l'exécution du **cas d'utilisation B**. Exécuter B peut éventuellement entraîner l'exécution de A : contrairement à l'inclusion, *l'extension est optionnelle*.
- Cette dépendance est symbolisée par **le stéréotype <<extend>>**



## Relations d'extension

61

- On dit qu'un **cas d'utilisation A** étend un **cas d'utilisation B** lorsque le **cas d'utilisation A** peut être appelé au cours de l'exécution du **cas d'utilisation B**. Exécuter **B** peut éventuellement entraîner l'exécution de **A** : contrairement à l'inclusion, *l'extension est optionnelle*.
- Cette dépendance est symbolisée par le **stéréotype <>extend><**



## Relations d'extension

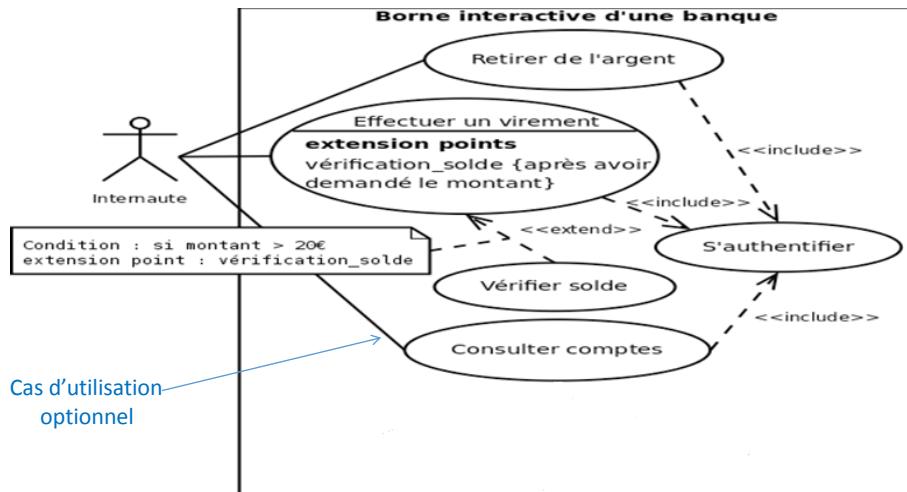
62

- Un **cas d'utilisation A** « extend » un **cas d'utilisation B** c.à.d **cas d'utilisation A** peut être provoqué par **cas d'utilisation B**;
- → **cas d'utilisation A** est **optionnel** pour **cas d'utilisation B**.

63

## Relations d'extension

**Exemple:**



## Relations d'extension

64

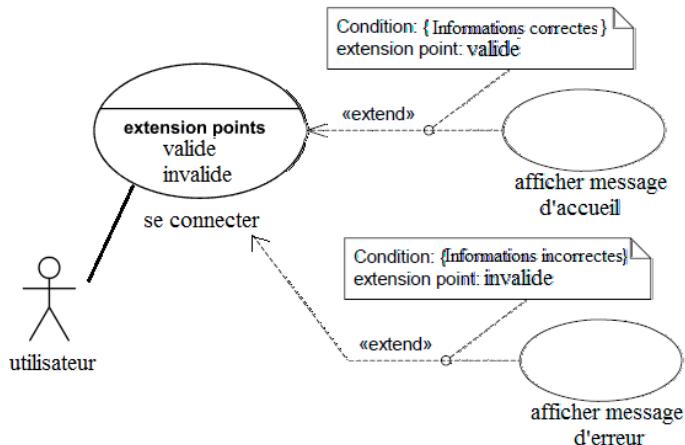
### Extension points (points d'extension)

- L'**extension** peut intervenir à un point précis du cas étendu. Ce point s'appelle le **point d'extension**.
- **Une extension est souvent soumise à une condition.**  
*Graphiquement, la condition est exprimée sous la forme d'une note.*
- Dans l'exemple d'une banque où la vérification du solde du compte n'intervient que si la demande de retrait dépasse 20 euros.

## Relations d'extension

65

### Exemple 2:



66

## Relations entre cas d'utilisation

### Cas d'utilisation interne

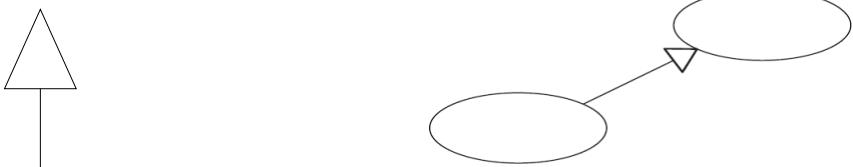
- Un cas d'utilisation est dit « interne » s'il n'est pas relié directement à un acteur.

67

## Relations entre cas d'utilisation

### ***Relation de généralisation***

- Un **cas A** est une généralisation d'**un cas B** si **B** est un cas particulier de **A**.
- Dans la figure précédente, la consultation d'un compte via Internet (**Consulter depuis internet**) est un cas particulier de la consultation (**Consulter comptes**).

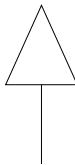


## Relation de généralisation

68

### ***Représentation graphique héritage***

- Le symbole utilisé pour la généralisation est une flèche pleine dont la pointe est un triangle fermé désignant le cas le plus général.

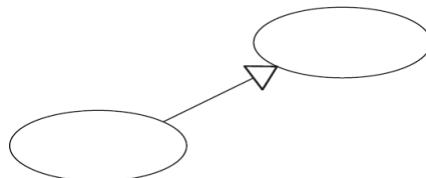


## Relation de généralisation

69

### **Notation graphique:**

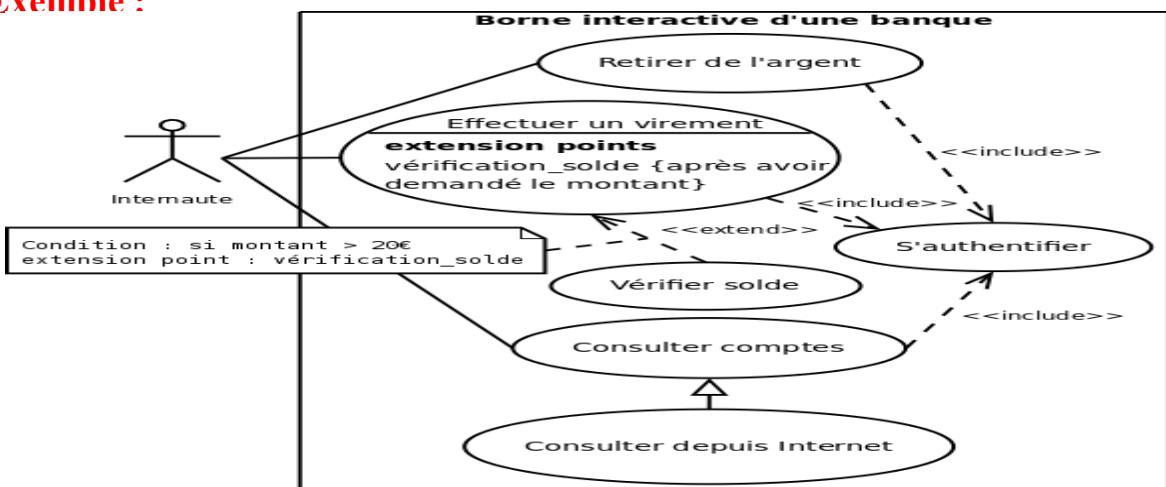
- Cette relation de généralisation/spécialisation est présente dans la plupart des diagrammes UML et se traduit par le concept d'héritage dans les langages orientés objet.



70

## Relation de généralisation

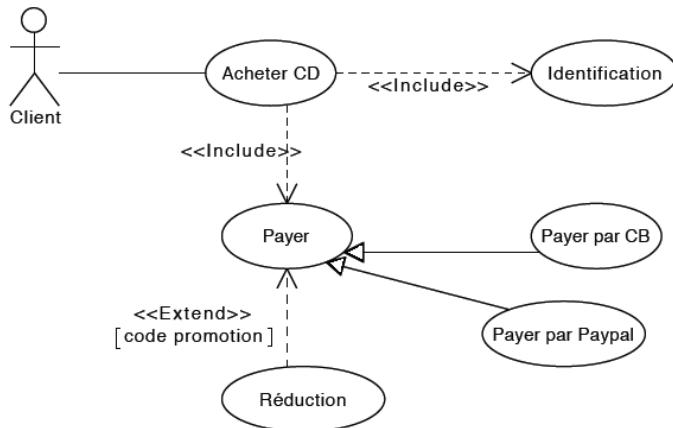
### **Exemple :**



## Relation de généralisation

71

### Exemple 2



72

## Modélisation des besoins avec UML

### Comment identifier les acteurs ?

- Les acteurs d'un système sont les entités externes à ce système qui interagissent (saisie de données, réception d'information, . . .) avec lui.
- Les acteurs sont à l'extérieur du système et dialoguent avec lui.
- Pour trouver les acteurs d'un système, il faut identifier quels sont les différents rôles que vont devoir jouer ses utilisateurs (ex : responsable clientèle, responsable d'agence, administrateur, . . .).

## Modélisation des besoins avec UML

73

### Comment identifier les acteurs ?

- Il faut également s'intéresser aux autres systèmes avec lesquels le système va devoir communiquer comme :
  - les périphériques manipulés par le système (imprimantes, hardware d'un distributeur de billet, ...);
  - des systèmes informatiques externes au système mais qui interagissent avec lui;
  - ...

## Modélisation des besoins avec UML

74

### Comment identifier les acteurs ?

- Pour faciliter la recherche des acteurs, on peut imaginer les frontières du système. Tout ce qui est à l'extérieur et qui interagit avec le système est un acteur, tout ce qui est à l'intérieur est une fonctionnalité à réaliser.

75

## Modélisation des besoins avec UML

### Comment recenser les cas d'utilisation ?

- L'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système.
- Pour identifier les cas d'utilisation, il faut se placer du point de vue de chaque acteur et déterminer comment et surtout pourquoi il se sert du système.
- Nommer les cas d'utilisation avec un verbe à l'infinitif suivi d'un complément en vous plaçant du point de vue de l'acteur et non pas de celui du système.

76

## Description textuelle des cas d'utilisation

Le diagramme de cas d'utilisation décrit les grandes fonctions d'un système du point de vue des acteurs, mais n'expose pas de façon détaillée le dialogue entre les acteurs et les cas d'utilisation.

Une description textuelle couramment utilisée se compose de deux parties principales.

77

## Description textuelle des cas d'utilisation

**La première partie** permet d'identifier le cas d'utilisation, elle doit contenir les informations suivantes.

**Nom :** Utiliser une tournure à l'infinitif (ex : Réceptionner un colis).

**Objectif :** Une description résumée permettant de comprendre l'intention principale du cas d'utilisation. Cette partie est souvent renseignée au début du projet dans la phase de découverte des cas d'utilisation.

**Acteurs principaux :** Ceux qui vont réaliser le cas d'utilisation (la relation avec le cas d'utilisation est illustrée par le trait liant le cas d'utilisation et l'acteur dans un diagramme de cas d'utilisation).

**Acteurs secondaires :** Ceux qui ne font que recevoir des informations à l'issue de la réalisation du cas d'utilisation.

**Dates :** Les dates de créations et de mise à jour de la description courante.

**Responsable :** Le nom des responsables.

**Version :** Le numéro de version.

78

## Description textuelle des cas d'utilisation

**La deuxième partie** contient la description du fonctionnement du cas d'utilisation sous la forme d'une séquence de messages échangés entre les acteurs et le système. Elle contient toujours une séquence nominale qui décrit le déroulement normal du cas d'utilisation. *À la séquence nominale s'ajoutent fréquemment des séquences alternatives* (des embranchements dans la séquence nominale) *et des séquences d'exceptions* (qui interviennent quand une erreur se produit).

- **Préconditions:** elles décrivent dans quel état doit être le système (l'application) avant que ce cas d'utilisation puisse être déclenché.
- **Scénarios :** Ces scénarios sont décrits sous la forme d'échanges d'évènements entre l'acteur et le système. On distingue le scénario nominal, qui se déroule quand il n'y a pas d'erreur, des scénarios alternatifs qui sont les variantes du scénario nominal et enfin les scénarios d'exception qui décrivent les cas d'erreurs.
- **Postconditions:** elles décrivent l'état du système à l'issue des différents scénarios.

79

## Description textuelle des cas d'utilisation

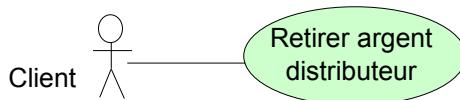
### La deuxième partie:

- **Préconditions:** Conditions au démarrage.
- **Scénarios :**
  - Étapes du déroulement normal (« nominal »),
  - Variantes possibles et les cas d'erreurs,
  - Informations échangées entre acteur et système,
- **Postconditions:** Conditions à la terminaison,
- **Contraintes non fonctionnelles** (performance, sécurité, disponibilité, confidentialité...).

80

## Description textuelle des cas d'utilisation

**Exemple :** cas d'utilisation "Retirer argent distributeur"



81

## Description textuelle des cas d'utilisation

### **Préconditions:**

Contient des billets; en attente d'une opération : ni en panne, ni en maintenance.

### **Postconditions:**

Si l'argent a pu être retiré, la somme sur le compte est égale à la somme qu'il y avait avant moins le retrait. Sinon, la somme sur le compte est inchangée.

### **Déroulement normal:**

1. le client introduit sa carte bancaire,
2. le système lit la carte et vérifie si la carte est valide,
3. le système demande au client de taper son code,
4. le client tape son code confidentiel,
5. le système vérifie que le code correspond à la carte,
6. le client choisit une opération de retrait,
7. le système demande le montant à retirer, etc.

82

## Description textuelle des cas d'utilisation

### **Variantes**

(A) Carte invalide : au cours de l'étape (2), si la carte est jugée invalide, le système affiche un message d'erreur, rejette la carte et le cas d'utilisation se termine.

(B) ...

### **Contraintes non fonctionnelles**

(A) Performance : le système doit réagir dans un délai inférieur à 4 secondes, quelque soit l'action de l'utilisateur.

(B) Sécurité ...

83

## Étude de cas: Guichet Automatique de Banque (GAB)

### Enoncé

Cette étude de cas concerne un système simplifié de Guichet Automatique de Banque (GAB). Le GAB offre les services suivants :

1. Distribution d'argent à tout Porteur de carte de crédit, via un lecteur de carte et un distributeur de billets.
  2. Consultation de solde de compte, dépôt en numéraire et dépôt de chèques pour les clients porteurs d'une carte de crédit de la banque adossée au GAB.
- N'oubliez pas non plus que :
3. Toutes les transactions sont sécurisées.
  4. Il est parfois nécessaire de recharger le distributeur, etc.

84

## Étude de cas: Guichet Automatique de Banque (GAB)

### Enoncé

À partir de ces quatre phrases, nous allons progressivement :

- identifier les acteurs ;
- identifier les cas d'utilisation ;
- construire un diagramme de cas d'utilisation ;
- décrire textuellement les cas d'utilisation ;

85

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 1 – Identification des acteurs du GAB

- *La phrase 1* nous permet d'identifier immédiatement un premier acteur évident : tout « **Porteur de carte** ». Il pourra uniquement utiliser le GAB pour *retirer de l'argent avec sa carte*.
- *La phrase 2* identifie des services supplémentaires qui ne sont proposés qu'aux **clients de la banque porteurs d'une carte de crédit** de cette dernière. Il s'agit donc d'un profil différent du précédent, que nous matérialisons par un deuxième acteur, appelé **Client banque**.

86

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 1 – Identification des acteurs du GAB

- *La phrase 3* nous incite à prendre en compte le fait que toutes les transactions sont sécurisées. Mais sécurisées par qui ? Pas par le GAB. Il existe donc d'autres entités externes qui jouent le rôle de Système d'autorisation et avec lesquelles le GAB communique directement. *Une interview de l'expert métier est nécessaire, pour nous permettre d'identifier deux acteurs différents :*
- ✓ **le Système d'autorisation global Carte Bancaire**, pour les transactions de retrait ;
- ✓ **le Système d'information de la banque**, pour autoriser toutes les transactions effectuées par un client avec sa carte de la banque, mais également pour accéder au solde des comptes.

87

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 1 – Identification des acteurs du GAB

- La phrase 4 nous rappelle qu'un GAB nécessite également des actions de maintenance, telles que le recharge en billets du distributeur, la récupération des cartes avalées, etc. Ces actions de maintenance sont effectuées par un nouvel acteur, que nous appellerons pour simplifier : **Opérateur de maintenance**.

88

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 2 – Identification des cas d'utilisation

Préparez une liste préliminaire des cas d'utilisation du GAB, par acteur. Reprenons un à un les cinq acteurs et listons les différentes façons qu'ils ont d'utiliser le GAB :

#### **Porteur de carte :**

- ✓ Retirer de l'argent.

#### **Client banque :**

- ✓ Retirer de l'argent (à ne pas oublier !).
- ✓ Consulter le solde de son compte courant.
- ✓ Déposer du numéraire.
- ✓ Déposer de l'argent (du numéraire ou des chèques).

89

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 2 – Identification des cas d'utilisation

#### *Opérateur de maintenance :*

- ✓ *Recharger le distributeur.*
- ✓ *Maintenir l'état opérationnel* (récupérer les cartes avalées, récupérer les chèques déposés, remplacer le ruban de papier, etc.).

#### *Système d'autorisation (Sys. Auto.) :*

- ✓ Néant.

#### *Système d'information (SI) banque :*

- ✓ Néant.

90

## Étude de cas: Guichet Automatique de Banque (GAB)

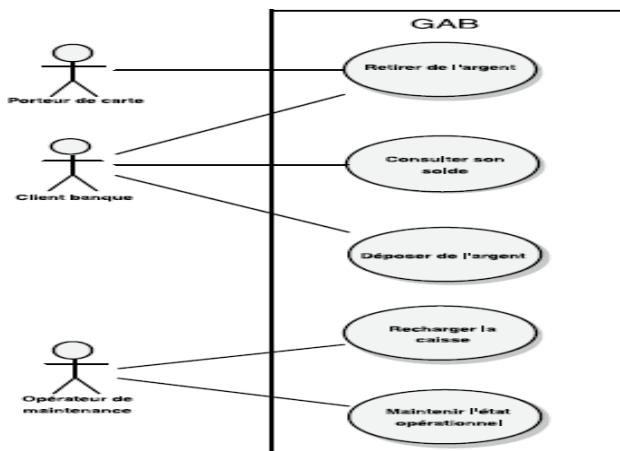
### Étape 3 – Réalisation de diagrammes de cas d'utilisation

On obtient sans difficulté un diagramme préliminaire en transcrivant la réponse précédente sur un schéma qui montre les cas d'utilisation (ovales) reliés par des associations (lignes) à leurs acteurs principaux (icône du « stick man »).

91

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 3 – Réalisation de diagrammes de cas d'utilisation

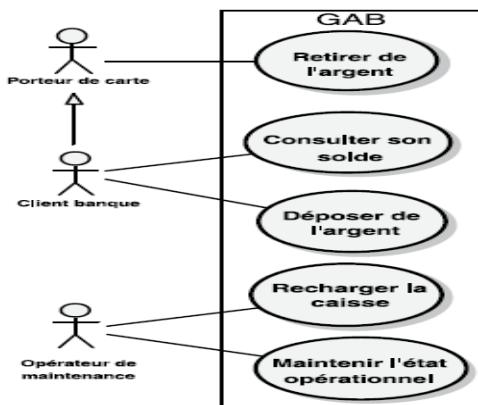


92

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 3 – Réalisation de diagrammes de cas d'utilisation

#### Généralisation entre acteurs

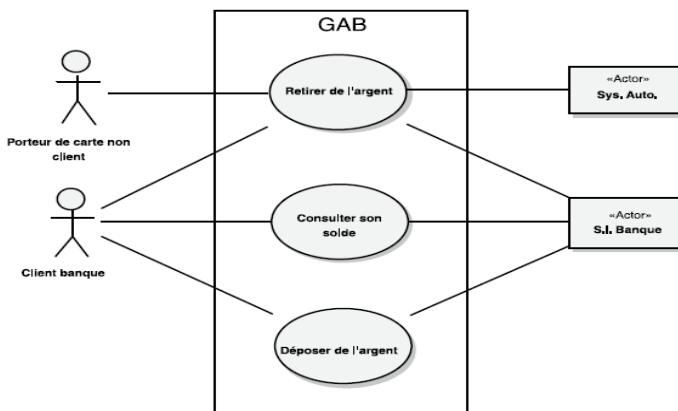


93

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 3 – Réalisation de diagrammes de cas d'utilisation

#### Acteurs secondaires



94

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 4 – Description textuelle des cas d'utilisation

Décrire la partie obligatoire du cas d'utilisation RETIRER DE L'ARGENT (pour l'acteur non client de la banque):



95

## Étude de cas: Guichet Automatique de Banque (GAB)

### Étape 4 – Description textuelle des cas d'utilisation

#### Sommaire d'identification:

**Titre :** Retirer de l'argent

**Résumé :** ce cas d'utilisation permet à un Porteur de carte, qui n'est pas client de la banque, de retirer de l'argent, si son crédit hebdomadaire le permet.

**Acteurs :** Porteur de carte (principal), *Système d'autorisation (secondaire)*.

**Date de création :** 02/12/2014

**Date de mise à jour :** 02/12/2017

**Version :** 4.0

**Responsable :** Mr X

96

### Étape 4 – Description textuelle des cas d'utilisation

#### Description des scénarios:

##### **Préconditions**

- La caisse du GAB est alimentée (il reste au moins un billet !).
- Aucune carte ne se trouve déjà coincée dans le lecteur.
- La connexion avec le Système d'autorisation est opérationnelle.

97

## Étape 4 – Description textuelle des cas d'utilisation

### Scénario nominal

1. Le Porteur de carte introduit sa carte dans le lecteur de cartes du GAB.
2. Le GAB vérifie que la carte introduite est bien une carte bancaire.
3. Le GAB demande au Porteur de carte de saisir son code d'identification.
4. Le Porteur de carte saisit son code d'identification.
5. Le GAB compare le code d'identification avec celui qui est codé sur la puce de la carte.
6. Le GAB demande une autorisation au Système d'autorisation.
7. Le Système d'autorisation donne son accord et indique le crédit hebdomadaire.
8. Le GAB demande au Porteur de carte de saisir le montant désiré du retrait.
9. Le Porteur de carte saisit le montant désiré du retrait.
10. Le GAB contrôle le montant demandé par rapport au crédit hebdomadaire.
11. Le GAB demande au Porteur de carte s'il veut un ticket.
12. Le Porteur de carte demande un ticket.
13. Le GAB rend sa carte au Porteur de carte.
14. Le Porteur de carte reprend sa carte.
15. Le GAB délivre les billets et un ticket.
16. Le Porteur de carte prend les billets et le ticket.

98

## Étape 4 – Description textuelle des cas d'utilisation

### Enchaînements alternatifs

#### *A1 : code d'identification provisoirement erroné*

- L'enchaînement A1 démarre au point 5 du scénario nominal.
6. Le GAB indique au Porteur de carte que le code est erroné, pour la première ou deuxième fois.
  7. Le GAB enregistre l'échec sur la carte.
- Le scénario nominal reprend au point 3.

#### *A2 : montant demandé supérieur au crédit hebdomadaire*

- L'enchaînement A2 démarre au point 10 du scénario nominal.
11. Le GAB indique au Porteur de carte que le montant demandé est supérieur au crédit hebdomadaire.
- Le scénario nominal reprend au point 8.

#### *A3 : ticket refusé*

- L'enchaînement A3 démarre au point 11 du scénario nominal.
12. Le Porteur de carte refuse le ticket.
  13. Le GAB rend la carte au Porteur de carte.
  14. Le Porteur de carte reprend sa carte.
  15. Le GAB délivre les billets.
  16. Le Porteur de carte prend les billets.

## Étape 4 – Description textuelle des cas d'utilisation

99

### Enchaînements d'erreur

#### *E1 : carte non valide*

L'enchaînement E1 démarre au point 2 du scénario nominal.

3. Le GAB indique au Porteur que la carte n'est pas valide (illisible, périmée, etc.), la confisque ; ***le cas d'utilisation se termine en échec.***

#### *E2 : code d'identification définitivement erroné*

L'enchaînement E2 démarre au point 5 du scénario nominal.

6. Le GAB indique au Porteur de carte que le code est erroné, pour la troisième fois.
7. Le GAB confisque la carte.
8. Le Système d'autorisation est informé ; ***le cas d'utilisation se termine en échec.***

#### *E3 : retrait non autorisé*

L'enchaînement E3 démarre au point 6 du scénario nominal.

100

## Étape 4 – Description textuelle des cas d'utilisation

### Enchaînements d'erreur (suite)

8. Le GAB éjecte la carte; ***le cas d'utilisation se termine en échec.***

#### *E4 : carte non reprise*

L'enchaînement E4 démarre au point 13 du scénario nominal.

14. Au bout de 10 secondes, le GAB confisque la carte.

15. Le Système d'autorisation est informé; ***le cas d'utilisation se termine en échec.***

#### *E5 : billets non pris*

L'enchaînement E5 démarre au point 15 du scénario nominal.

16. Au bout de 10 secondes, le GAB reprend les billets.

17. ***Le cas d'utilisation se termine en échec.***

#### *E6 : annulation de la transaction*

L'enchaînement E6 peut démarrer entre les points 4 et 12 du scénario nominal.

- 4 à 12. Le Porteur de carte demande l'annulation de la transaction en cours.

Le GAB éjecte la carte; ***le cas d'utilisation se termine en échec.***

101

## Étape 4 – Description textuelle des cas d'utilisation

### Postconditions

- La caisse du GAB contient moins de billets qu'au début du cas d'utilisation (le nombre de billets manquants est fonction du montant du retrait).
- Une transaction de retrait a été enregistrée par le GAB avec toutes les informations pertinentes (montant, numéro de carte, date, etc.). Les détails de la transaction doivent être enregistrés aussi bien en cas de succès que d'échec.

102

## Étape 4 – Description textuelle des cas d'utilisation

### Contraintes non fonctionnelles

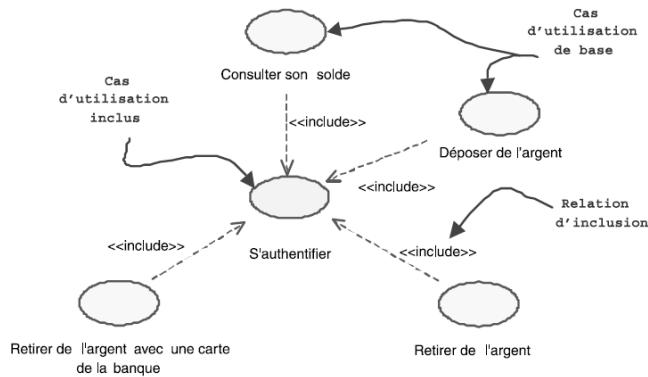
- (A) **Performance :** Le système doit réagir dans un délai inférieur à 4 secondes, quelque soit l'action de l'utilisateur.
- (B) **Disponibilité:** Le GAB est accessible 7 jours sur 7, 24 h sur 24. L'absence de papier pour imprimer les tickets ne doit pas empêcher les retraits.
- (C ) **Confidentialité:** La comparaison du code d'identification saisi sur le clavier du GAB avec celui de la carte doit être fiable à  $10^{-6}$ .
- (D) **Sécurité:**.....

## Étude de cas: Guichet Automatique de Banque (GAB)

103

### Relation d'inclusion entre cas d'utilisation

- Identifiez une partie commune aux différents cas d'utilisation et factorisez-la dans un nouveau cas inclus dans ces derniers.

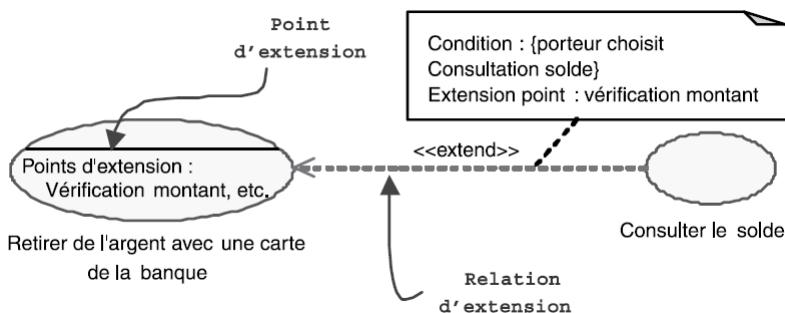


## Étude de cas: Guichet Automatique de Banque (GAB)

104

### Relation d'extension entre cas d'utilisation

- En extrapolant sur les besoins initiaux, identifiez une relation d'extension entre deux cas d'utilisation du client de la banque.

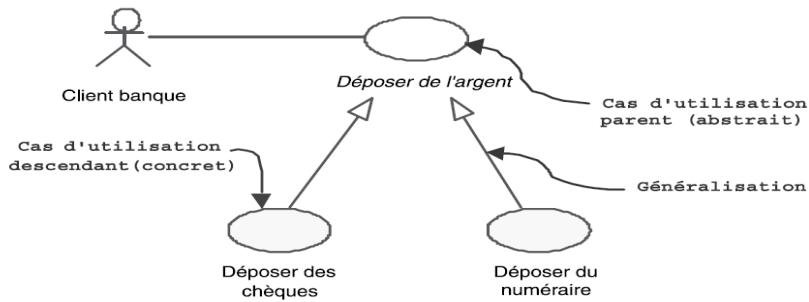


## Étude de cas: Guichet Automatique de Banque (GAB)

105

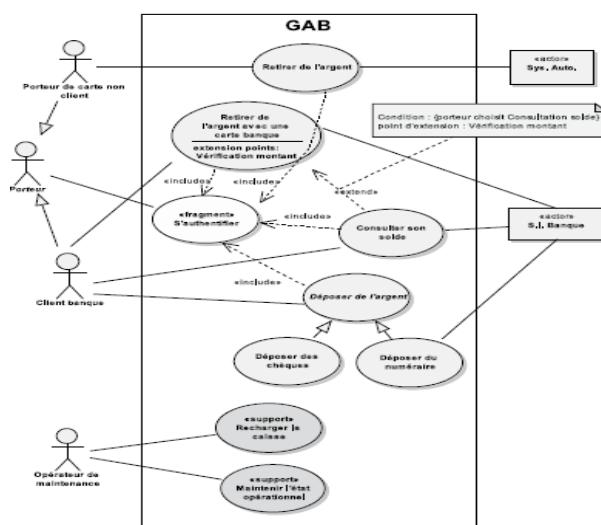
### Généralisation/spécialisation entre cas d'utilisation

- Identifiez une relation de généralisation/spécialisation qui implique un cas d'utilisation du client de la banque.



### Diagramme de cas d'utilisation complet du GAB

106



107

## Diagramme de packages

### Notion du package (paquetage)

- **Le package** est une sorte de dossier permettant de structurer un modèle en unités cohérentes.
- **Un package** est un regroupement d'éléments de modèle et de diagrammes. Il permet ainsi d'organiser des éléments de modélisation en groupes.
- Il peut contenir tout type d'élément de modèle : des classes, des cas d'utilisation, des interfaces, des diagrammes, . . . et même des packages imbriqués (décomposition hiérarchique).



## Diagramme de packages

108

### Représentations d'un package

**Exemple:**



109

## Diagramme de packages du GAB

### Structuration des cas d'utilisation en packages

- Proposez une structuration des cas d'utilisation du GAB en packages.

110

## Diagramme de packages du GAB

### Structuration des cas d'utilisation en packages

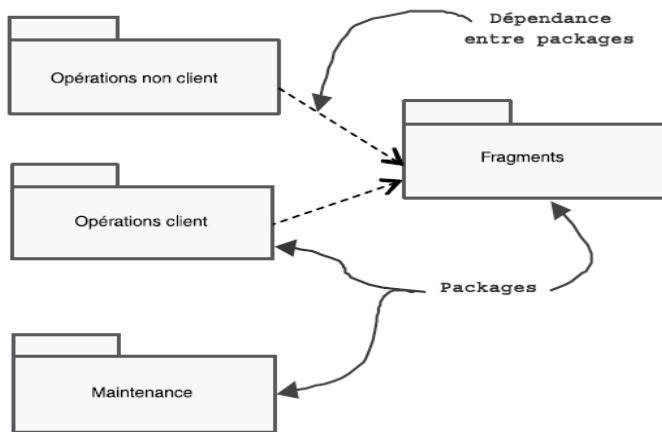
- Plusieurs stratégies sont possibles : procéder au regroupement par acteur, par domaine fonctionnel, etc.
- Dans notre exemple, un regroupement des cas d'utilisation par acteur principal s'impose, car cela permet également de répartir les acteurs secondaires.
- Le cas d'utilisation inclus S'authentifier est mis dans un package à part, en tant que fragment commun, pour bien le distinguer des vrais cas fonctionnels qui l'incluent.
- Les flèches de dépendance entre packages de cas d'utilisation synthétisent les éventuelles relations entre les cas, c'est-à-dire ici les inclusions.

111

## Diagramme de packages du GAB

### Diagramme de packages des cas d'utilisation du GAB

Le schéma suivant présente la structuration proposée des cas d'utilisation.

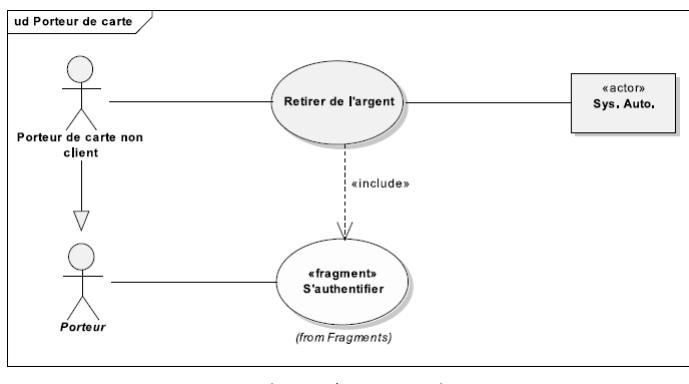


112

## Diagramme de packages du GAB

### Diagramme de packages des cas d'utilisation du GAB

Il est maintenant possible de dessiner un diagramme de cas d'utilisation par packages. Cela ne présente aucune difficulté et nous donnerons uniquement le diagramme du premier package:



## Diagrammes de classes

- Introduction
- Diagramme de Classes
- Relations dans les diagrammes de classes
- Types de classes

### Introduction

- Les diagrammes de classes se composent d'un ensemble des classes, des interfaces et leurs relations.
- Les diagrammes de classes représentent la **vue statique** du système.
- Les diagrammes de classes permettent de produire/construire le **squelette** du système.
- La modélisation des diagrammes de classes est **l'étape essentielle** dans **la conception orientée objets**.

## Introduction

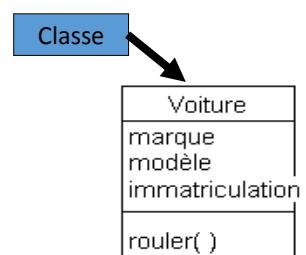
Les éléments de base dans les diagrammes de classes sont:

- **Classes** (Encapsulation des données et des fonctions)
- **Attributs** (Des données simples dans les classes)
- **Opérations /Méthodes** (Des fonctions dans les classes/ interfaces)
- **Relations** (Représentent les liens entre les instances des classes)
- **Interfaces** (Déclaration des fonctions)

## Notation de base

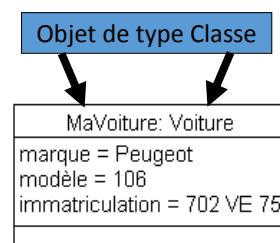
### • Classe

- Une description d' un ensemble d'objets qui partage les mêmes attributs, opérations, méthodes, relations et contraintes.

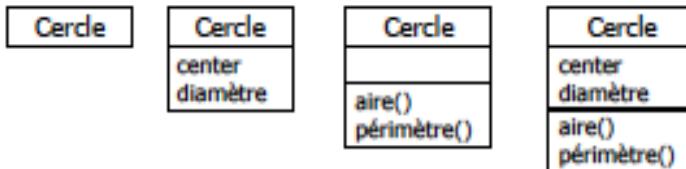


### • Objet

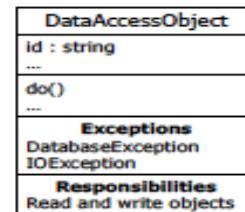
- Une entité avec une limite et une identité bien définies qui encapsule de l'état et du comportement. L' état est représenté par des attributs et des relations, le comportement est représenté par des opérations et des méthodes. ***Un objet est une instance d 'une classe.***



## Classe



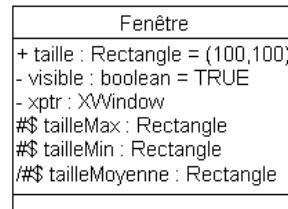
Une classe peut avoir des compartiments supplémentaires:



## Attributs

### Attributs de la classe

- **Attribut = propriété** nommée d'une classe
- **Syntaxe**
  - visibilité *nom* : *type* = *valeur initiale*
- **Visibilité**
  - + public (publique):
  - # protected (protégé)
  - - private (privé)
  - ~ ou rien package



## Attributs

### Attributs de classe (variable de classe)

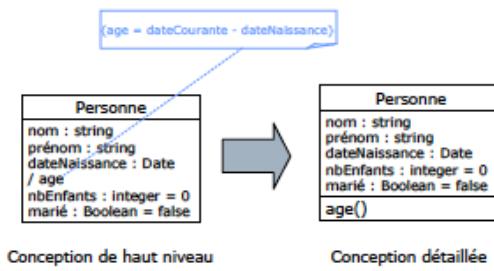
- la portée standard d'un attribut est limité à un **objet**
- quand cette portée s'applique à la **classe** elle-même, on parle d'**attribut de classe** (représenté par le symbole \$ ou souligné)

Fenêtre
+ taille : Rectangle = (100,100)
- visible : boolean = TRUE
- xptr : XWindow
## tailleMax : Rectangle
## tailleMin : Rectangle
## tailleMoyenne : Rectangle

## Attributs

### Attributs dérivés

- Attributs qui peuvent être calculés à partir d'autres attributs et de formules de calcul.
- Attributs qui peuvent être **déduit** d'un ou plusieurs **autres attributs** (représenté par le symbole « / » devant leur nom).
- L'âge d'une personne peut être dérivé de la date de naissance.



## Méthodes

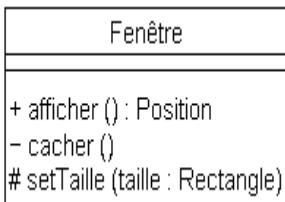
□ **Méthode** = service que l' on peut demander à un objet pour réaliser un comportement.

□ **Syntaxe**

- visibilité **nomDeLaMethode** (*paramètres: leurs types*) : *type retour*

□ Mêmes notions que les attributs

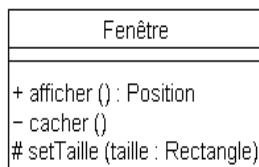
- **visibilité**



## Méthodes

### Méthode de classe

- Comme pour les attributs de classe, il est possible de déclarer des **méthodes de classe**.
- Une **méthode de classe** ne peut manipuler que des attributs de classe et ses propres paramètres.
- Cette méthode n'a pas accès aux attributs de la classe (i.e. des instances de la classe).
- L'accès à une **méthode de classe** ne nécessite pas l'existence d'une instance de cette classe.
- Graphiquement, une **méthode de classe** est **soulignée** ou représenté par le symbole \$.



## Méthodes

### Classes et Méthodes abstraites

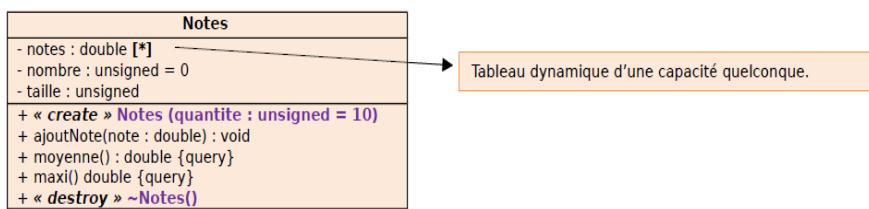
- ❑ Une méthode est dite **abstraite** lorsqu'on connaît son entête mais pas la manière dont elle peut être réalisée.
- ❑ Une classe est dite **abstraite** lorsqu'elle définit au moins **une méthode abstraite** ou lorsqu'une classe parent contient **une méthode abstraite** non encore réalisée.
- ❑ On ne peut instancier **une classe abstraite** : elle est vouée à se spécialiser.
- ❑ Une classe abstraite peut très bien contenir des méthodes concrètes.
- ❑ Une interface = une classe abstraite pure ne comporte que des **méthodes abstraites**.
- ❑ Pour indiquer qu'**une classe est abstraite**, il faut ajouter le mot-cléf **abstract** derrière son nom.

## Méthodes

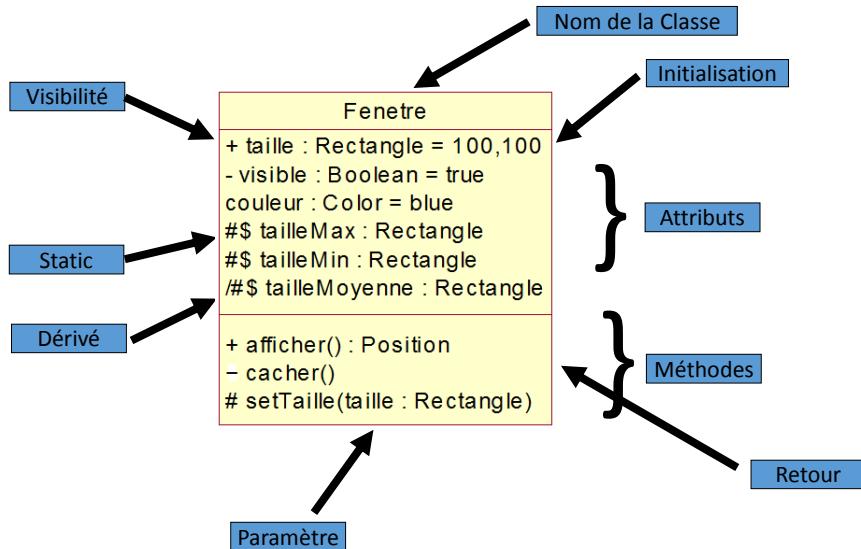
### Constructeur et destructeurs

- ❑ Les stéréotypes peuvent être utilisés pour identifier des opérations particulières comme les constructeurs (stéréotype « **create** ») et le destructeur (stéréotype « **destroy** »).

#### ❑ Exemple:

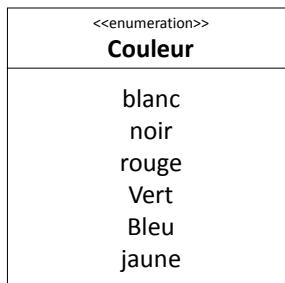


## Notation Complète



## Enumération

Une enumération ou un type énuméré est un type possédant un nombre fini et arbitraire de valeurs possibles.



# Relations entre classes

## Types de relation

Les relations entre les classes:

### **Association**

- Une association est une relation entre deux classes (association binaire) ou plus (association n-aire), qui décrit les connexions structurelles entre leurs instances.
- Une association indique donc qu'il peut y avoir des liens entre des instances des classes associées.

### **Héritage**

- Une classe peut hériter d'une ou plusieurs classes.

## Types de relation

### Agrégation

- Une association montre qu'une classe est une partie d'une autre classe.

### Composition

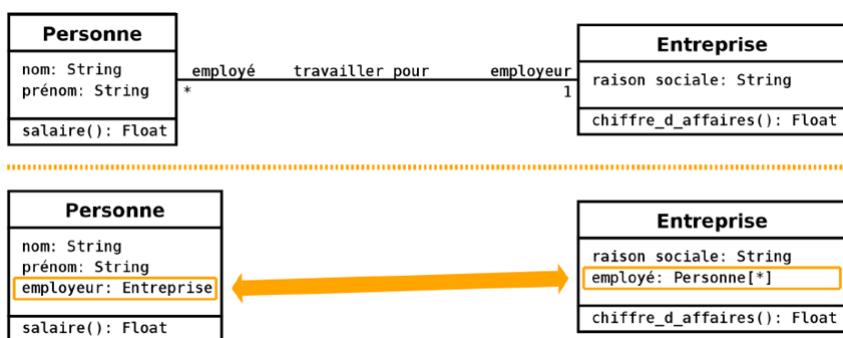
- Une forme forte d'agrégation.

### Dépendance

- Montre la dépendance entre les classes.

## Notion d'association

### Deux façons de modéliser une association



## Notion d'association

### Une terminaison d'association est une propriété

Une propriété est une caractéristique structurelle. Dans le cas d'une classe, les propriétés sont constituées par les attributs et les éventuelles terminaisons d'association que possède la classe. Elle peut être paramétrée par les éléments suivants:

#### nom :

- Comme un attribut, une terminaison d'association peut être nommée. Le nom est situé à proximité de la terminaison, mais contrairement à un attribut, ce nom est facultatif.
- Le nom d'une terminaison d'association est appelée ***nom du rôle***.
- Une association peut donc posséder autant de noms de rôle que de terminaisons (deux pour une association binaire et n pour une association n-aire).

## Notion d'association

### Une terminaison d'association est une propriété

Elle peut être paramétrée par les éléments suivants:

#### Visibilité :

- Comme un attribut, une terminaison d'association possède une visibilité.
- La visibilité est mentionnée à proximité de la terminaison, et plus précisément, le cas échéant, devant le nom de la terminaison.

## Notion d'association

### **Une terminaison d'association est une propriété**

Elle peut être paramétrée par les éléments suivants:

#### **Multiplicité/cardinalité :**

- ❑ Comme un attribut, une terminaison d'association peut posséder une multiplicité.
- ❑ Elle est mentionnée à proximité de la terminaison. Il n'est pas impératif de la préciser, mais, contrairement à un attribut dont la multiplicité par défaut est 1, la multiplicité par défaut d'une terminaison d'association est non spécifiée.
- ❑ L'interprétation de la multiplicité pour une terminaison d'association est moins évidente que pour un attribut.

## Notion d'association

### **Une terminaison d'association est une propriété**

Elle peut être paramétrée par les éléments suivants:

#### **Navigabilité :**

Pour un attribut, la navigabilité est implicite, navigable, et toujours depuis la classe vers l'attribut. Pour une terminaison d'association, la navigabilité peut être précisée.

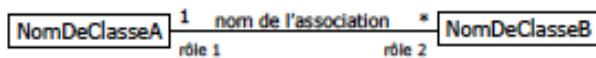
## Relation d'association

### Association binaire

- **Définition**

- Exprime une **connexion** sémantique bidirectionnelle entre deux classes.
- Une association est une abstraction des liens entre des instances des classes (objets).
- Le **sens** d'une association peut-être précisé par une **flèche**. ( $\blacktriangleright$  ou  $\blacktriangleleft$ )
- Elle peut être ornée d'un nom, avec éventuellement une précision du sens de lecture
- Une association est utilisée pour montrer comment deux classes sont liées entre elles.

- **Notation**



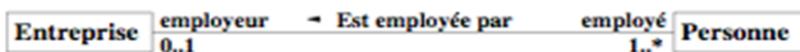
## Relation d'association

136

**Rôle** = rôle joué par une classe dans une association.

- Chaque extrémité d'une association est appelée **un rôle**.
- Un rôle montre le but de l'association.
- Un rôle peut avoir (**un nom**, **une expression de multiplicité**).

### Exemple:



## Relation d'association

137

- **Multiplicité ou cardinalité** = indique le **nombre** d' instances d' une classe qui peut être mise en relation avec une seule instance de la classe associée.
- **Multiplicité ou cardinalité** = définir combien d'instances de la classe A est associé à une instance de la classe B.



- Différentes expressions de multiplicité
  - 1 : obligatoire (un et un seul)
  - 0..1 : optionnel (zéro ou un seul)
  - 0..\* ou \* : quelconque (zéro ou plusieurs)
  - 1..\* : au moins 1(d'un à plusieurs)
  - 1..5, 10 : entre 1 et 5, ou 10
  - n : exactement n (entier naturel)
  - m..n : de m à n (entiers naturels)

## Relation d'association

138

### Navigabilité

- ❑ La navigabilité indique s'il est possible de traverser une association.
- ❑ On représente graphiquement la navigabilité par une flèche du côté de la terminaison navigable et on empêche la navigabilité par une croix du côté de la terminaison non navigable .
- ❑ **Par défaut, une association est navigable dans les deux sens.**

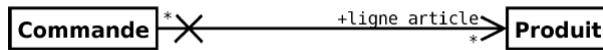
## Relation d'association

139

### Navigabilité

#### Exemple

- La terminaison du côté de la classe **Commande** n'est pas navigable : cela signifie que les instances de la classe **Produit** ne stockent pas de liste d'objets du type **Commande**.
- Inversement, la terminaison du côté de la classe **Produit** est navigable : chaque objet commande contient une liste de produits.



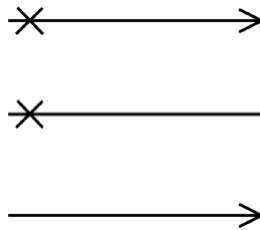
## Relation d'association

140

### Navigabilité

#### Remarque

- Implicitement, ces trois notations ont la même sémantique.

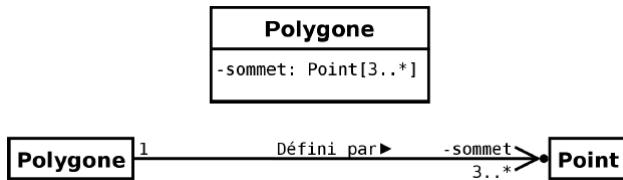


## Relation d'association

141

### Navigabilité

- Deux modélisations équivalentes:



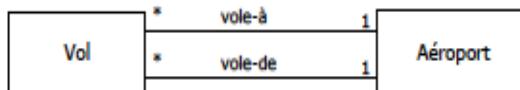
## Relation d'association

142

### Associations multiples

- Deux classes peuvent avoir plusieurs associations entre elles.

### Exemple 1



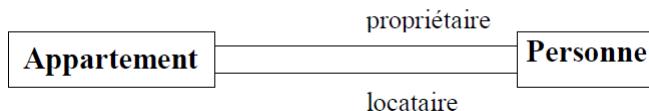
## Relation d'association

143

### Associations multiples

- Associations multiples entre 2 classes.

#### Exemple 2



## Relation d'association

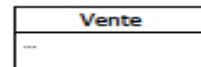
144

### Association directionnelle et Attributs

- Par défaut, les associations sont bidirectionnelles.
- Pourtant, les associations peuvent être directionnelles.
- La navigabilité pointant de **Caisse** à **Vente** montre que **Caisse** possède un attribut de type **Vente**.
- Cet attribut s'appelle *venteCourante*



- Autre façon de représentation: utilisation de l'attribut



## Relation d'association

145

### Quand utilise-t-on l'association directionnelle ou l'attribut?

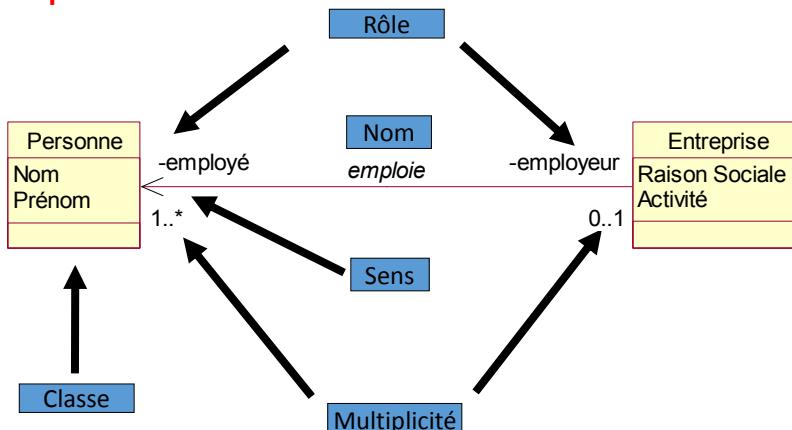
- On utilise l'**attribut** pour les types de données, comme *Boolean, Integer, Number, Real, Time, Date, ...*
- On utilise l'**association directionnelle** pour d'autres classes:  
Pour voir mieux les connexions entre les classes.
- C'est juste pour mieux représenter, ces deux façons sont sémantiquement équivalentes.
- **Exemple:**



## Relation d'association

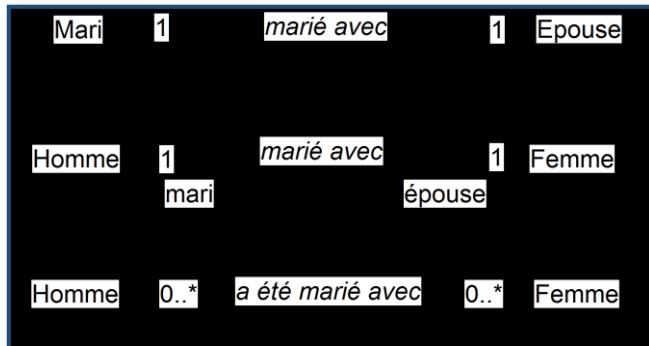
146

### Exemple:



## Sémantique

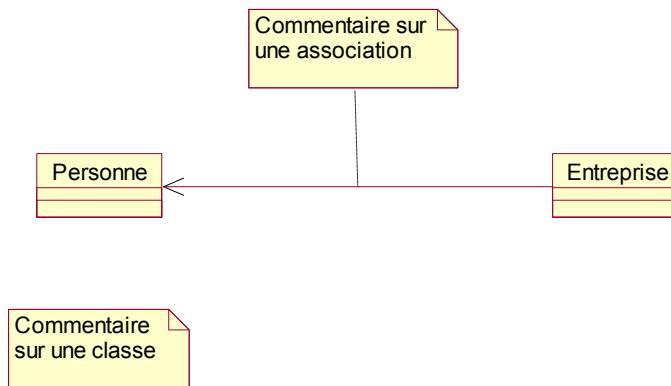
147



## Note

148

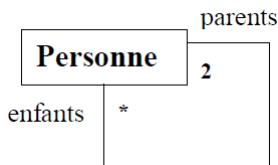
- **Note** = Commentaire placé sur un diagramme.



## Association réflexive

149

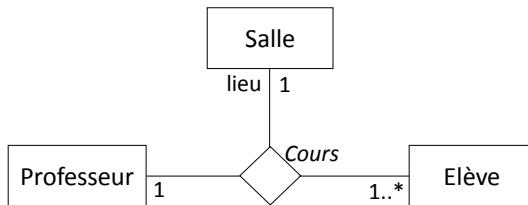
- Quand les deux extrémités de l'association pointent vers la même classe, l'association est dite réflexive.



## Association n-aire

150

- Association n-aire** = une association parmi 3 classes ou plus.
- Chaque instance de l' association est un **n-tuple** de valeurs des classes respectives.



## Classe d' association/Classe-association

151

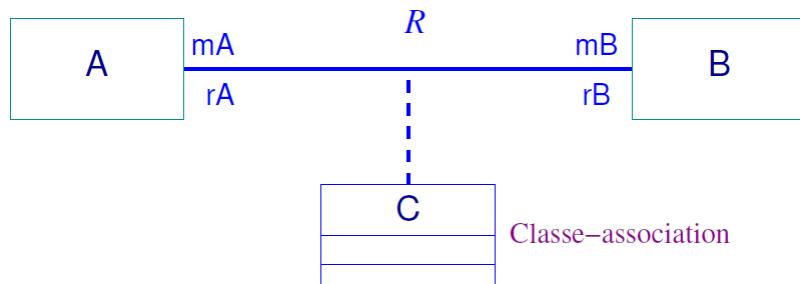
### Définition et représentation

- Une **classe-association** possède les caractéristiques des associations et des classes : elle se connecte à deux ou plusieurs classes et possède également des attributs et des opérations.
- Une **classe-association** est caractérisée par un trait discontinu entre la classe et l'association qu'elle représente.

## Classe d' association/Classe-association

152

### Définition et représentation

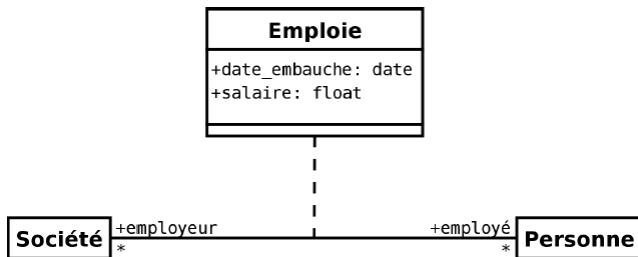


- ❑ L'association entre les classes **A** et **B** est réalisée par **un objet** de la classe **C** (sous sa responsabilité).
- ❑ La classe **C** a des propriétés qui lui sont propres (attributs, opérations...).

## Classe d' association/Classe-association

153

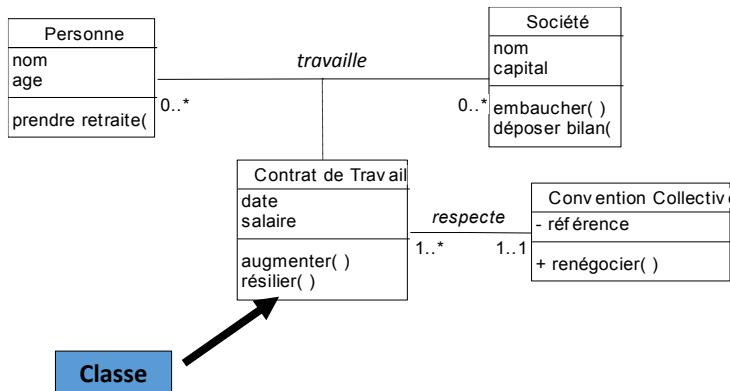
### Exemple de classe-association



## Classe d' association/Classe-association

154

- **Classe d' association/classe-association** = Elément ayant à la fois les propriétés d' une classe et d' une association



Classe

## Classe d' association/Classe-association

155

### Auto-association sur classe-association

- Imaginons que nous voulions ajouter *une association Supérieur de* dans le diagramme précédent pour préciser qu'une personne est le supérieur d'une autre personne.
- On ne peut simplement ajouter *une association réflexive* sur la classe **Personne**.

## Classe d' association/Classe-association

156

### Auto-association sur classe-association

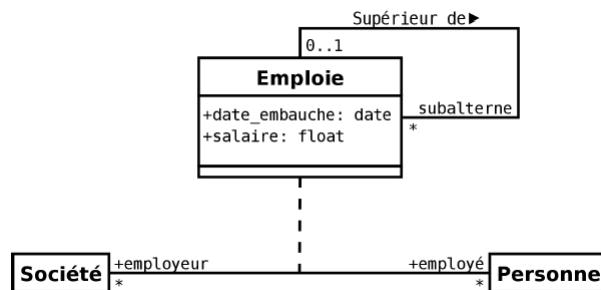
- En effet, une personne n'est pas le supérieur d'une autre dans l'absolu.
- Une personne est, en tant qu'employé d'une entreprise donné, le supérieur d'une autre personne dans le cadre de son emploi pour une entreprise donné (généralement, mais pas nécessairement, la même).
- Il s'agit donc d'une association réflexive, non pas sur la classe **Personne** mais sur *la classe-association Emploie*.

## Classe d' association/Classe-association

157

### Auto-association sur classe-association

**Exemple:** auto-association sur classe-association



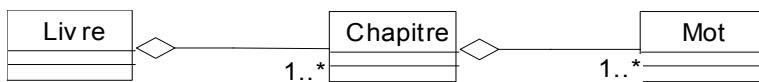
Agrégation  
et  
composition

## Agrégation

159

### Définition

- **Agrégation = association particulière** spécifiant ***une relation 'tout - partie'*** entre l'agrégat et un composant où une classe constitue un élément plus grand (tout) composé d'éléments plus petit (partie), il faut utiliser une agrégation.
  - Inclusion
  - Propagation



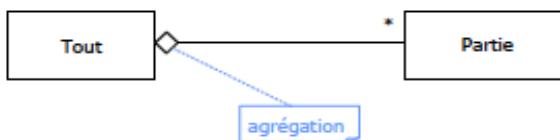
- Une agrégation est utilisée entre deux classes lorsque la relation qu'elle représente est de type
  - Maître et esclaves : « appartient à »
  - Tout et parties : « est une partie de »

## Agrégation

160

### Notation graphique:

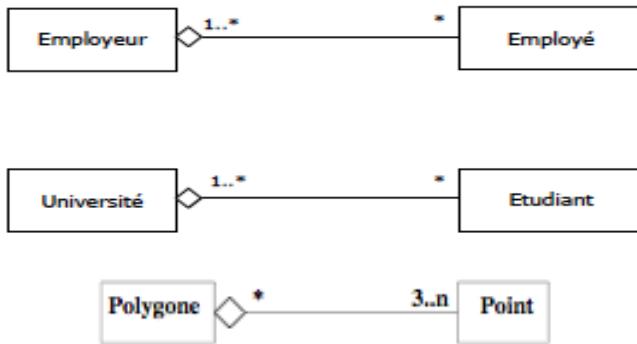
- Le symbole, un petit losange blanc , désignant l'agrégation se place du côté de l'agrégat (tout).



## Agrégation

161

### Exemples



## Composition

162

### Définition

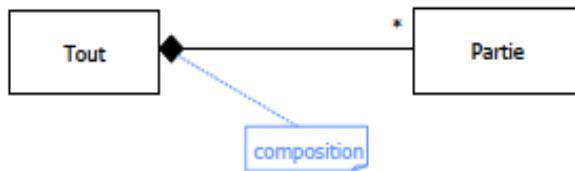
- **Composition** = forme **forte d' agrégation** avec un **cycle de vie** des parties lié à celui du composite.
- Une composition est aussi une relation « tout-partie » mais l'agrégat est plus important.
- Si le tout est détruit, les parties sont aussi détruites.

## Composition

163

### Notation graphique

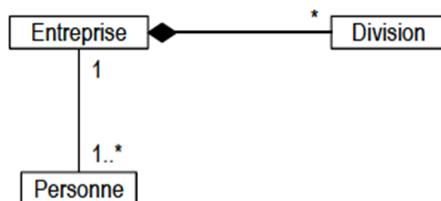
- Le symbole, un petit losange noir plein, désignant la composition se place du côté du composite (tout).



## Composition

164

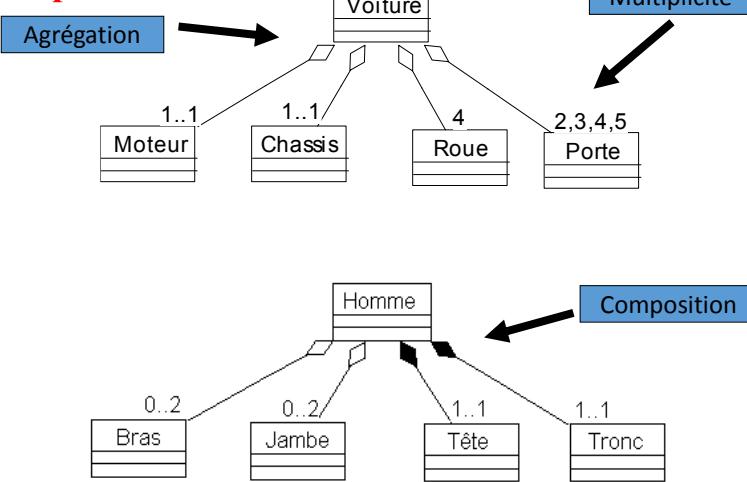
### Exemple



## Agrégation/Composition

165

### Exemples



## Agrégation/Composition

166

### Agrégation ou composition?

- La partie peut-elle faire partie de plus d'un tout?

- ✓ Oui → agrégation
- ✓ Non → question suivante

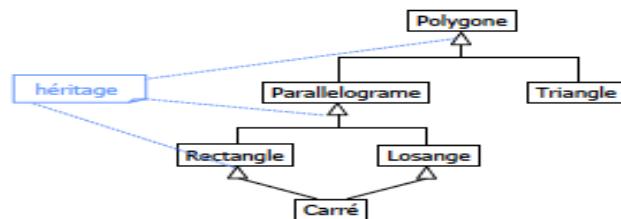
- La destruction du tout entraîne-t-elle celle des parties?

- ✓ Oui → composition
- ✓ Non → agrégation

## Héritage

167

- **Généralisation** = relation entre un élément plus général et un élément plus spécifique qui est entièrement conforme avec le premier élément, et qui ajoute de l'information supplémentaire.
- **Spécialisation** = mécanisme par lequel des éléments plus spécifiques incorporent la structure et le comportement d'éléments plus généraux (notion d'**héritage**).

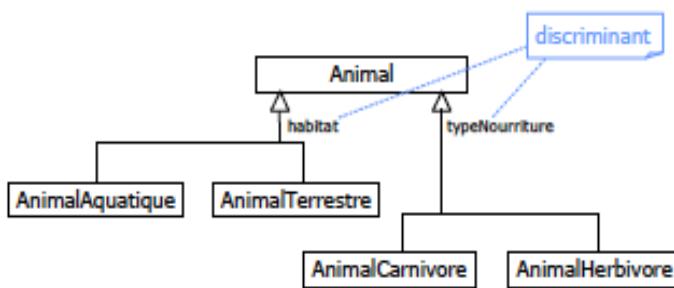


## Héritage

168

- Le **discriminant** (optionnel) est une étiquette décrivant le critère suivant lequel se base la spécialisation

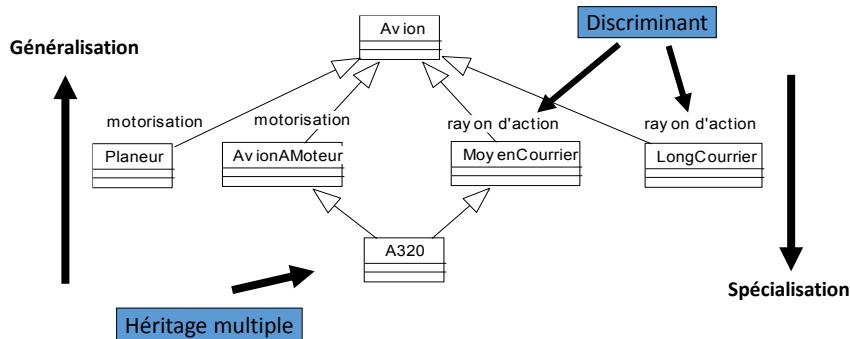
### Exemple 1



## Héritage

169

### Exemple 2



## Relation de dépendance

170

Une classe peut dépendre d'une autre classe.

Plusieurs types de dépendance:

- Avoir un attribut de type d'une autre classe.
- Envoyer un message en utilisant un attribut, une variable locale, une variable globale d'une autre classe ou des méthodes statiques.
- Recevoir un paramètre de type d'une autre classe.

## Relation de dépendance

171

### Représentation graphique

- La dépendance est représentée par un trait discontinu orienté.
- Elle est souvent stéréotypée, comme **use**, pour mieux expliciter le lien sémantique entre les éléments du modèle.



## Relation de dépendance

172

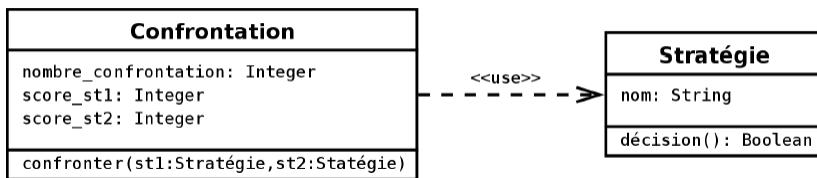
### Exemple 1



## Dépendance

173

### Exemple 2



## Stéréotypes de dépendance entre classes

174

Stéréotype	Définition
<i>access</i>	import du contenu d'un autre package
<i>create</i>	la classe crée des instances d'une autre classe
<i>instantiate</i>	la méthode d'une classe crée des instances d'une autre
<i>permit</i>	donne accès aux éléments privés
<i>use</i>	un élément requiert un autre élément

## Types de relation entre classes

175



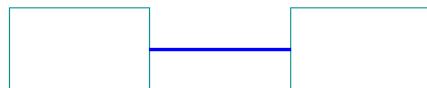
Dépendance



Généralisation



Réalisation



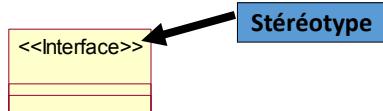
Association

## Types de Classes

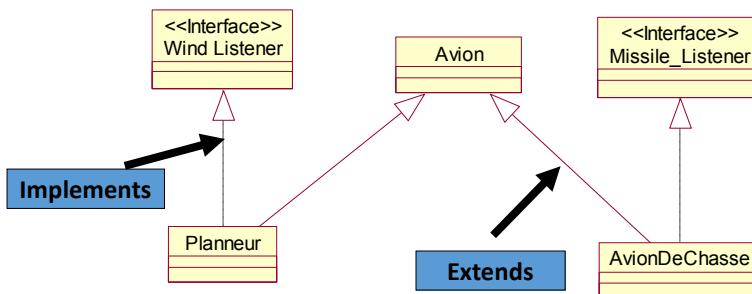
## Interface

177

- Notations



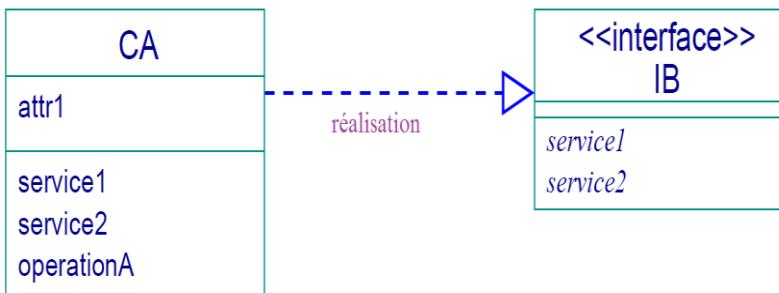
- Hériter d'une interface



## Interface

178

### Relation de réalisation entre classes et interfaces :



## Interface

179

**Deux notations pour la réalisation  
(l'implémentation):**

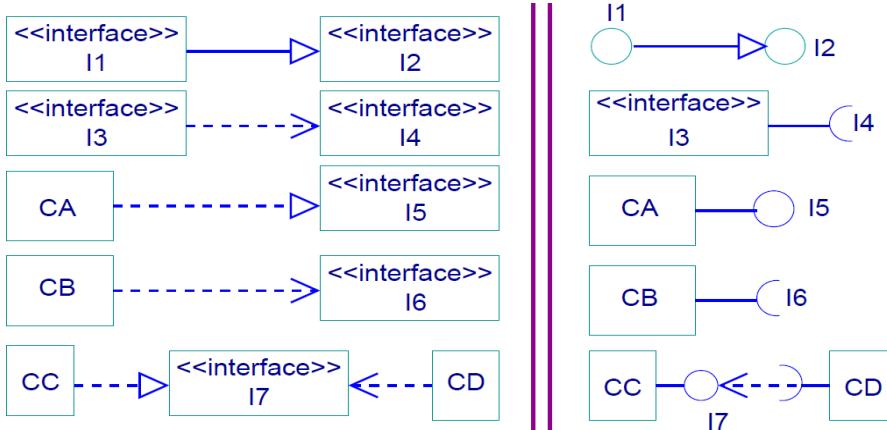
—○ ou - - - - ▷

## Interface

180

**Deux notations pour la réalisation:**

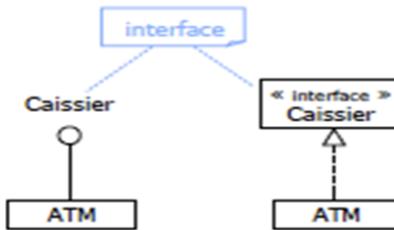
**Exemple 1:**



## Interface

181

**Deux notations pour la réalisation  
(l'implémentation):**



## Interface

182

**Notions sur l'interface**

- **Une interface** doit être réalisée par au moins **une classe** et peut l'être par plusieurs.
- La réalisation est représentée par un trait discontinu terminé par une flèche triangulaire et **le stéréotype « realize »**.
- **Une classe** peut très bien réaliser plusieurs **interfaces**.
- **Une classe** (classe cliente de l'interface) peut dépendre d'une interface (interface requise).
- On représente cela par une relation de dépendance et **le stéréotype « use »**.

## Interface

183

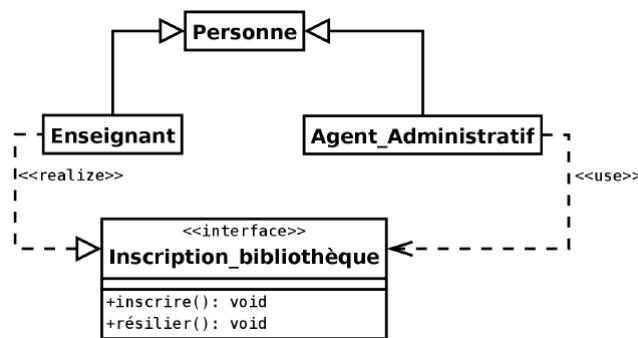
### Utilisation d'une interface:

— C ou - - - «use» - - - >

## Interface

184

### Exemple



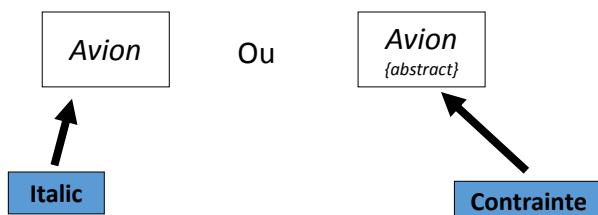
## Classe Abstraite

185

### Definition:

- **Classe Abstraite** = classe que l'on ne peut pas **instancier**.

### Notation :



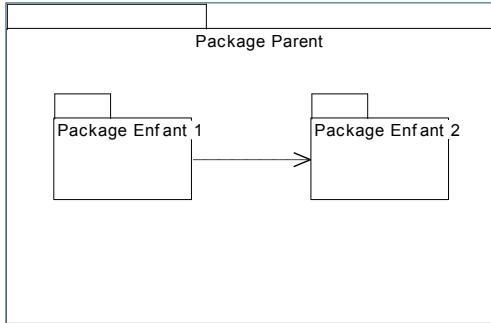
## Packages

## Packages

187

### Pour structurer

- **Package = Regroupement** d' éléments de modèle et de diagrammes.
- Les Packages divisent et organisent les modèles de la même manière que les répertoires organisent les systèmes de fichiers
- Les Packages eux-mêmes peuvent être **imbriqués** à l' intérieur d' autres Packages.



## Relation entre packages

188

### Dépendance :



Au moins un élément du package source utilise les services d'au moins un des éléments du package destination.

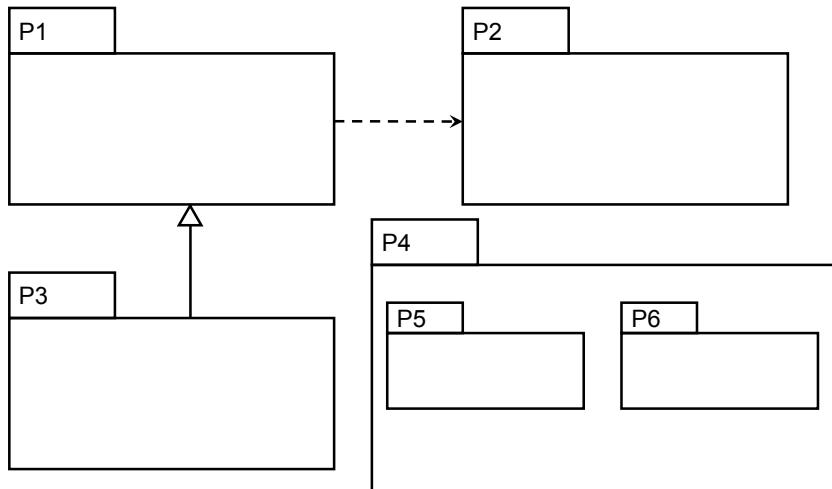
### Héritage :



Au moins un élément du package source spécialise (est dérivé d') au moins un des éléments du package destination.

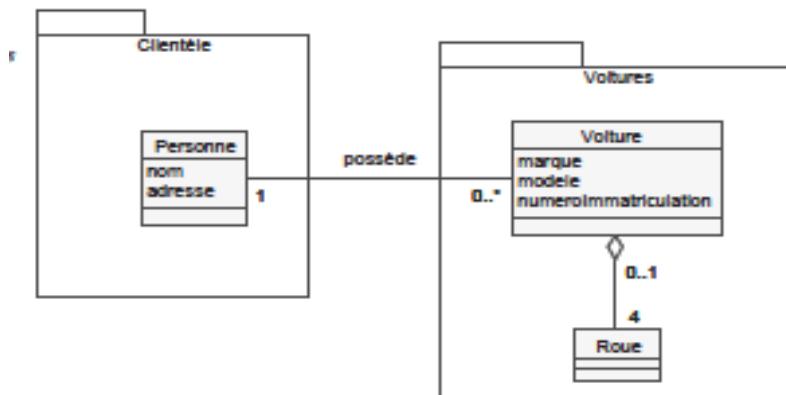
## Convention graphique

189



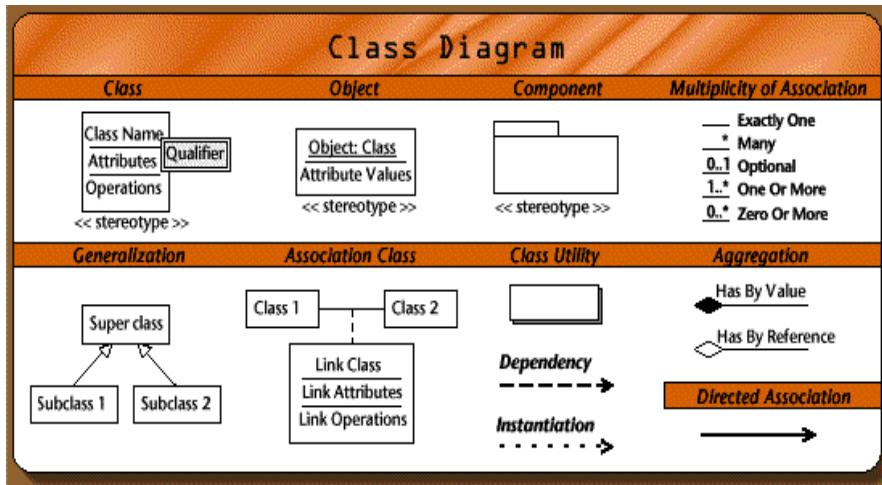
## Exemple de package

190



## Notation

191



## Élaboration d'un diagramme de classes

192

Une démarche couramment utilisée pour bâtir un diagramme de classes consiste à :

### 1. Trouver les classes du domaine étudié:

Cette étape empirique se fait généralement en collaboration avec un expert du domaine. Les classes correspondent généralement à des concepts ou des substantifs du domaine.

### 2. Trouver les associations entre classes:

Les associations correspondent souvent à des verbes, ou des constructions verbales, mettant en relation plusieurs classes, comme « est composé de », « pilote », « travaille pour ».

## Élaboration d'un diagramme de classes

193

### 3. Trouver les attributs des classes:

- Les attributs correspondent souvent à des substantifs, ou des groupes nominaux, tels que «la masse d'une voiture » ou « le montant d'une transaction ».
- Les adjectifs et les valeurs correspondent souvent à des valeurs d'attributs.

Vous pouvez ajouter des attributs à toutes les étapes du cycle de vie d'un projet (implémentation comprise).

N'espérez pas trouver tous les attributs dès la construction du diagramme de classes.

## Élaboration d'un diagramme de classes

194

### 4. Organiser et simplifier le modèle:

- En éliminant les classes redondantes et en utilisant l'héritage.

### 5. Itérer et raffiner le modèle:

- Un modèle est rarement correct dès sa première construction.  
La modélisation objet est un processus non pas linéaire mais itératif.

## Diagrammes d'objets

195

- Présentation
- Objectifs
- Classe/Objet
- Liens
- Relation de dépendance d'instanciation
- Exemples

## Présentation

196

- **Un diagramme d'objets** représente des objets (i.e. instances de classes) et leurs liens (i.e. instances de relations) pour donner une vue figée de l'état d'un système à un instant donné.
- Ils ont des **vues statiques des instances** des éléments qui apparaissent dans les **diagrammes de classes**.
- **Un diagramme d'objets** présente la **vue de conception d'un système**, exactement comme les diagrammes de classes, mais à partir de **cas réel** ou de prototypes.
- Il est **une instance d'un diagramme de classes**.
- Les diagrammes de classes peuvent aussi contenir des objets.

## Objectifs

197

- Un diagramme d'objets peut être utilisé pour :
  - illustrer le modèle de classes en montrant un exemple qui explique le modèle ;
  - préciser certains aspects du système en mettant en évidence des détails imperceptibles dans le diagramme de classes ;
  - prendre une image (snapshot) d'un système à un moment donné.
- Le diagramme de classes modélise des règles et le diagramme d'objets modélise des faits.

## Objets

:Voiture

**Instance anonyme de la classe voiture**

golf:Voiture

**Instance nommée de la classe voiture**

golf

**Instance nommée d'une classe anonyme**

golf:Voiture

Couleur = "bleu M"  
Puissance = 7ch**Spécification des attributs**

golf:Vehicule::Voiture::Berline

**Spécification du chemin complet**

:Voiture

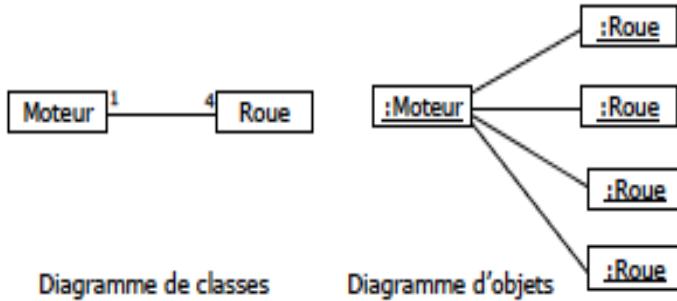
**Collection d'instance  
(groupe d'objets instances d'une même classe)**

## Classe/objet

199

### Représentation

- Un diagramme d'objets est une instance d'un diagramme de classe



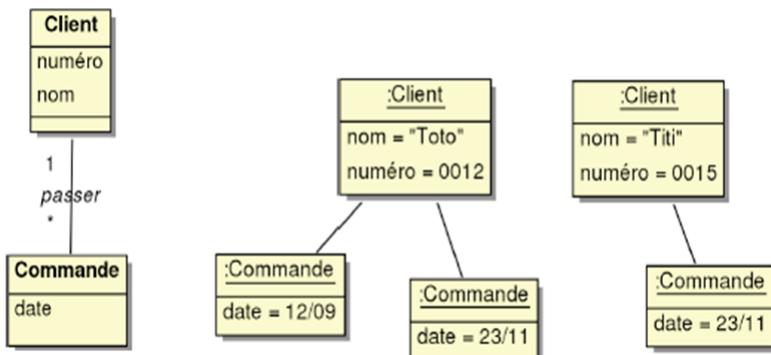
## Classe/objet

200

### Représentation

- Le diagramme de classes contraint la structure et les liens entre les objets.

*Diagramme cohérent avec le diagramme de classes:*



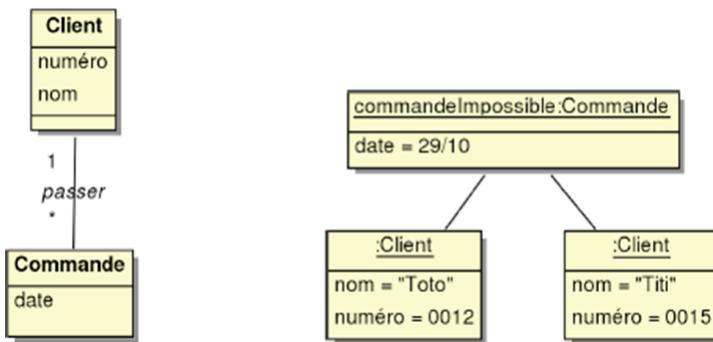
## Classe/objet

201

### Représentation

- Le diagramme de classes constraint la structure et les liens entre les objets.

*Diagramme incohérent avec le diagramme de classes:*



## Liens

202

- Un **lien** est une instance d'une relation (association, agrégation ou composition).
- Un **lien** se représente comme une association mais s'il a un nom, il est souligné.
- Naturellement, on ne représente pas les multiplicités qui n'ont aucun sens au niveau des objets.
- Dans un diagrammes d'objets, les relations du diagramme de classes deviennent des liens.
- Les relations relient les classes par contre les liens relient les objets.**

## Liens

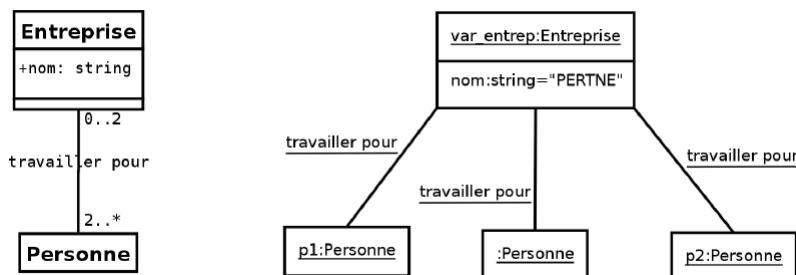
203

- Un lien entre deux objets implique obligatoirement une relation entre les classes des deux objets
- La relation de généralisation/specialisation ne possède pas d'instance, elle n'est donc jamais représentée dans un diagramme d'objets.

## Liens

204

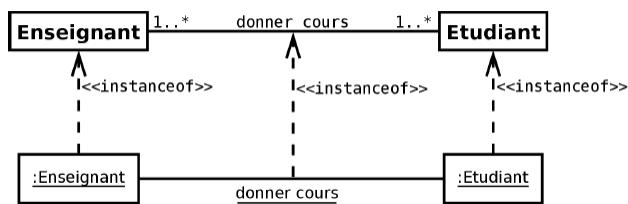
### Exemple



## Relation de dépendance d'instanciation

205

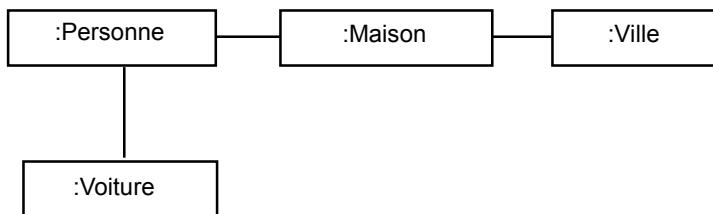
- La relation de dépendance d'instanciation (stéréotypée « `<<instanceof>>` ») décrit la relation entre un classeur et ses instances.
- Elle relie, en particulier, les liens aux associations et les objets aux classes.



## Diagrammes d'objets

206

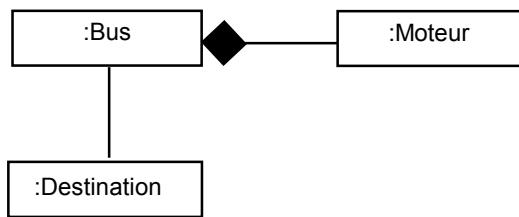
### Association



## Diagrammes d'objets

207

### Composition



## Plan

208

- Diagrammes d'interaction
- Diagramme de séquence
- Diagramme de communication

# DiAGRAMMEs d'interaction

## Diagrammes d'interaction

210

- **Objectif :**

Représenter les **communications** avec le logiciel et au sein du logiciel.

- **Diagramme de communication**

- Représentation **spatiale** des objets et de leurs interactions.
- **Diagramme d'objets** dont les associations sont étiquetées par les messages envoyés.

- **Diagramme de séquence**

- Représentation **temporelle** des interactions entre les objets
- **Chronologie** des messages échangés entre les objets et avec les acteurs.

## Diagrammes d'interaction

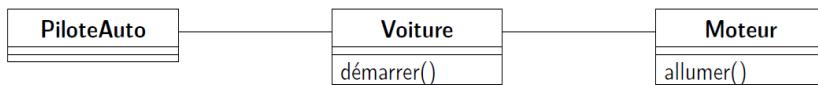
211

- **Diagramme de communication**
- Le **diagramme de communication** se focalise sur la **représentation spatiale**.
- **Diagramme de séquence**
- Le **diagramme de séquence** se focalise sur les **aspects temporels**.
- **Diagrammes équivalents** en phase de conception : Description du lien entre cas d'utilisation et diagramme de classes.

## Diagrammes d'interaction

212

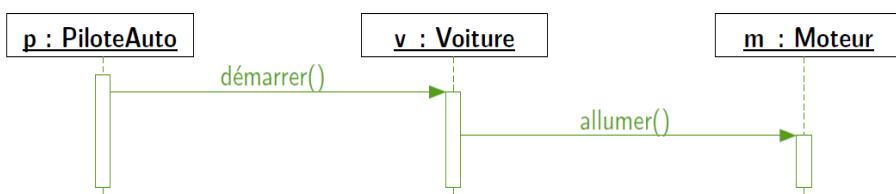
- **Exemple :** À partir d'un diagramme de classes et d'un cas d'utilisation



- **Diagramme de communication**



- **Diagramme de séquence**



## Diagrammes d'interaction

213

- **Objectif :** Décrire la réalisation des cas d'utilisation sur le système décrit par le diagramme de classes
  - Point de vue **interne** sur le fonctionnement du système
  - Description au niveau de l'**instance** (état du système à un instant)
  - Description de **scénarios** particuliers
  - Représentation des **échanges de messages**
    - ✓ Entre les acteurs et le système, entre les objets du système
    - ✓ De façon chronologique

## Diagrammes d'interaction

214

### Scénario

- Il y a autant de diagrammes de séquence qu'il y a de scénarios.
- Un **scénario** montre une séquence particulière d'interactions entre objets, dans un seul contexte d'exécution du système.
- Un **scénario** peut être vu comme une réponse à un besoin ou une partie d'un besoin du diagramme des **cas d'utilisation**.
- On y fait intervenir des **objets**, des **messages** et des **événements**.

## Scénario

215

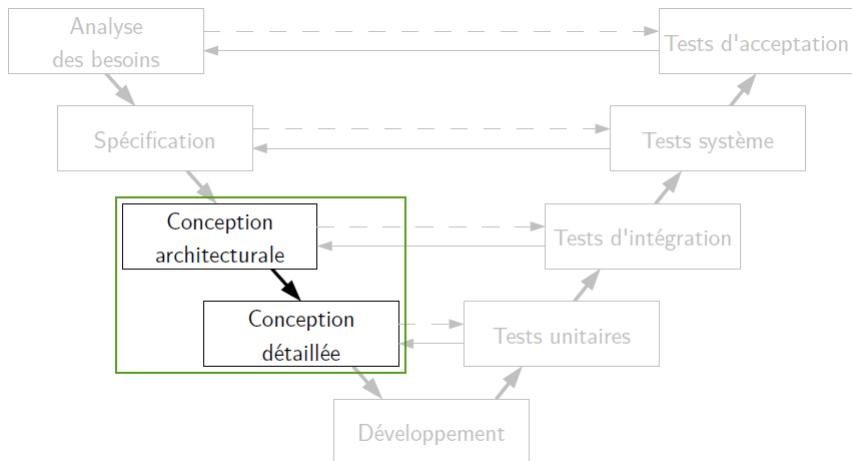
- Un **scénario** représente une succession particulière d'enchaînements, s'exécutant du début à la fin du cas d'utilisation.
- Un cas d'utilisation contient en général:
  - un scénario nominal
  - plusieurs scénarios alternatifs (qui se terminent de façon normale)
  - plusieurs scénarios d'erreur (qui se terminent par un échec).

## DiAGRAMME de sequence

## Processus de développement logiciel

217

### Diagramme de séquence



## Diagramme de séquence/Sequence diagram

218

- Un **diagramme de séquence** représente une interaction entre objets en insistant sur la chronologie des envois de messages.
- Structuration en termes de:
  - **temps**: axe vertical
  - **objets**: axe horizontal
- Le **diagramme de séquence** permet de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs.
- Le **diagramme de séquence** fait partie des diagrammes d'interactions.
- **Les objets** au cœur d'un système interagissent en s'échangeant des messages.
- **Les acteurs** interagissent avec le système au moyen d'IHM.

## Diagramme de séquence

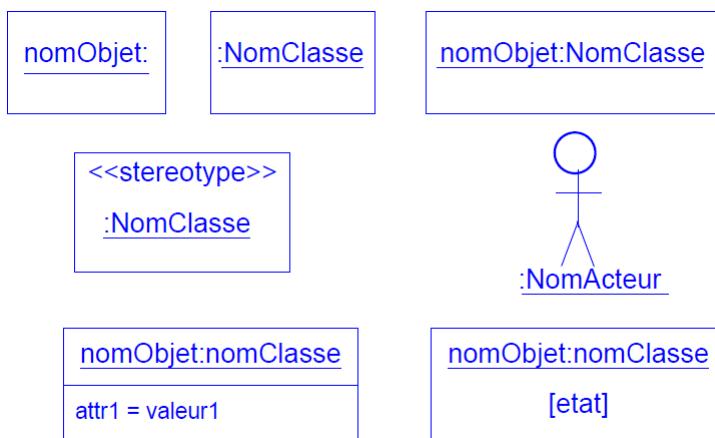
219

- Éléments du diagramme de séquence
  - Acteurs
  - Objets (instances)
  - Messages (cas d'utilisation, appels d'opération)
- **Principes de base** : Représentation graphique de la **chronologie** des **échanges de messages** avec le système ou au sein du système
  - « Vie » de chaque entité représentée verticalement
  - Échanges de messages représentés horizontalement

## Diagramme de séquence

220

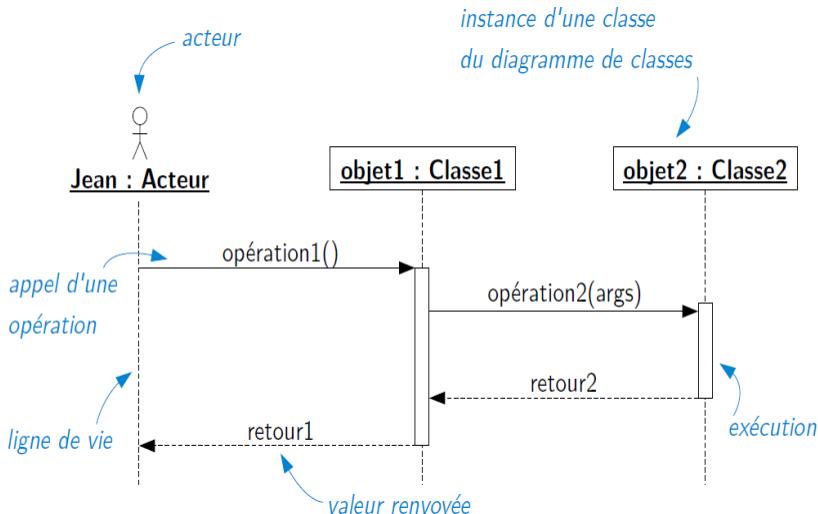
### Objets: instances de classifier



## Diagramme de séquence

221

### Éléments de base

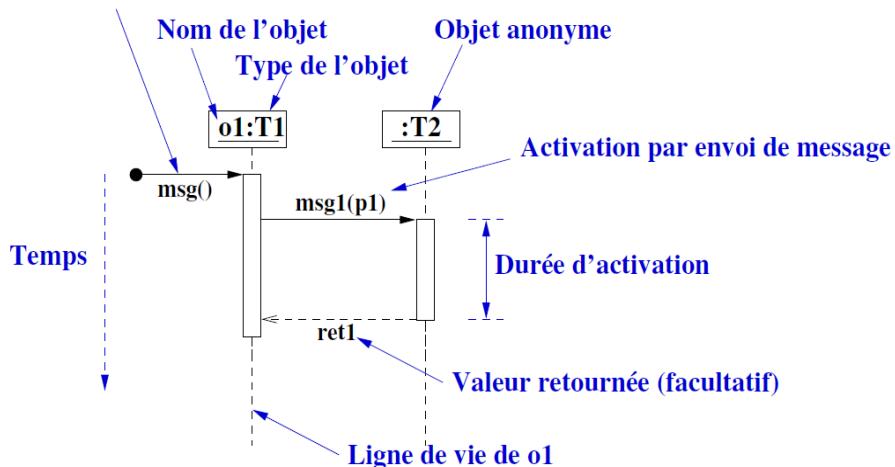


## Diagramme de séquence

222

### Ligne de vie et activation

Message dont l'émetteur n'est pas connu



## Diagramme de séquence

223

### Ligne de vie

- Une ligne de vie représente un participant à une interaction (objet ou acteur).
- La syntaxe de son libellé est :  
**nomDuRole : NomClasseOuActeur**
- Une ligne de vie est une instance, donc il y a nécessairement les deux points (:) dans son libellé.
- Ligne verticale
- Au moins un des deux noms doit être spécifié dans l'étiquette, les deux points (:) sont, quant à eux, obligatoires.

## Diagramme de séquence

224

### Messages

Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie :

- Ils sont représentés par des flèches;
- Ils sont présentés du haut vers le bas le long des lignes de vie, dans un ordre chronologique.

## Diagramme de séquence

225

### Messages

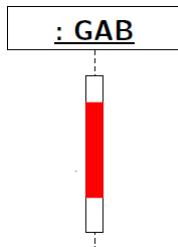
- ❑ Un message définit une communication particulière entre des lignes de vie (objets ou acteurs).
- ❑ Plusieurs types de messages existent, dont les plus courants :
  - l'envoi d'un signal ;
  - l'invocation d'une opération (appel de méthode) ;
  - la création ou la destruction d'une instance.

## Diagramme de séquence

226

### Activité

- ❑ La réception des messages provoque une période d'activité (rectangle vertical sur la ligne de vie) marquant le traitement du message (spécification d'exécution dans le cas d'un appel de méthode).



## Types de messages

227

### Message synchrone

- Un message synchrone bloque l'expéditeur jusqu'à la réponse du destinataire. Le flot de contrôle passe de l'émetteur au récepteur.
- Si un objet A envoie un message synchrone à un objet B, A reste bloqué tant que B n'a pas terminé.
- On peut associer aux messages d'appel de méthode un message de retour (en pointillés) marquant la reprise du contrôle par l'objet émetteur du message synchrone.

message synchrone



## Types de messages

228

### Message synchrone

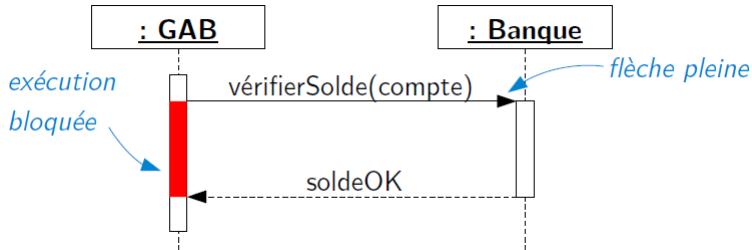
message de retour



## Types de messages

229

### Messages synchrones



## Types de messages

230

### Messages asynchrones

- Un message asynchrone n'est pas bloquant pour l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.

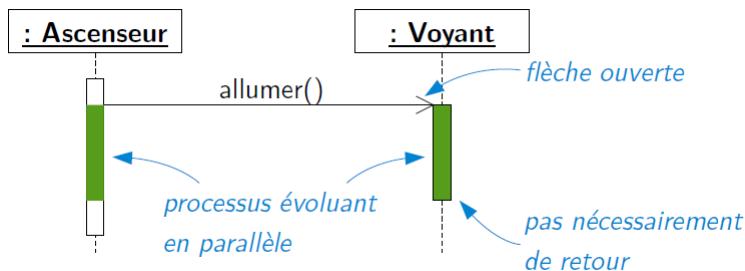
message asynchrone



## Diagramme de séquence

231

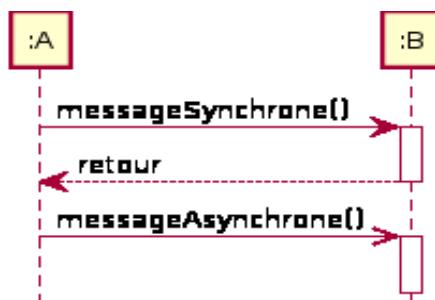
### Messages asynchrones



## Diagramme de séquence

232

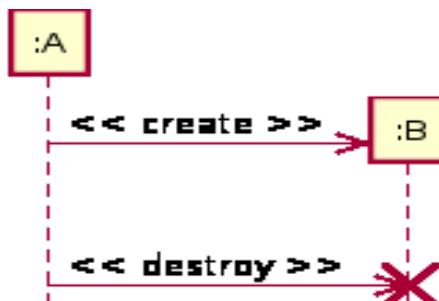
### Messages synchrones & asynchrones



## Création et destruction d'objet

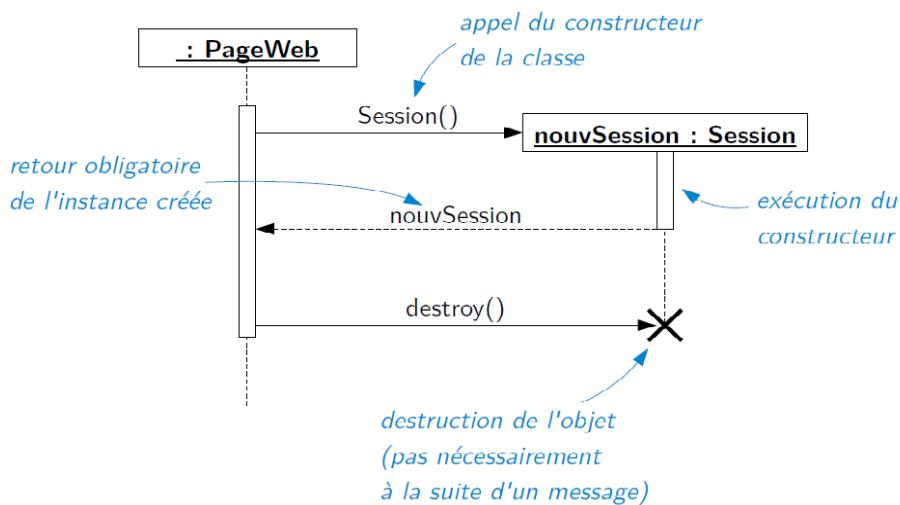
233

- **Création** : message asynchrone stéréotypé `<< create >>` pointant vers le rectangle en tête de la ligne de vie.
- **Destruction** : message asynchrone stéréotypé `<< destroy >>` précédant une croix sur la ligne de vie.



## Création et destruction d'objet

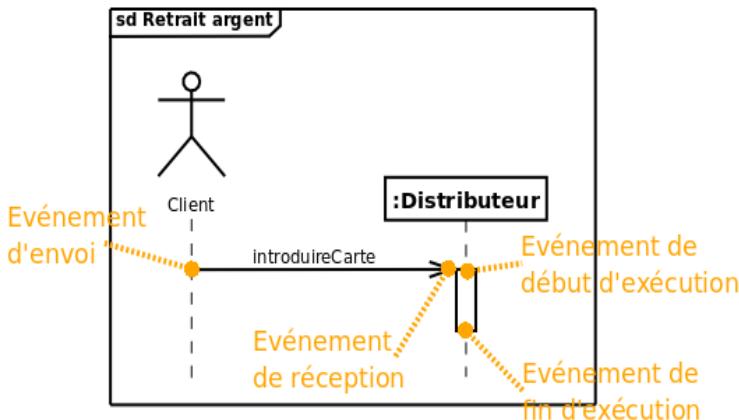
234



## Événements et messages

235

### Exemple



## Syntaxe des messages et des réponses

236

- La syntaxe des messages est :
 

```
nomSignalOuOperation(ListeDesParams)
```
- **ListeDesParams:** une liste de paramètres séparés par des virgules.
- Dans la liste des paramètres, on peut utiliser les notations suivantes :
  - pour donner une valeur à un paramètre spécifique:  
`nomParametre = valeurParametre`
  - pour préciser que l'argument est modifiable:  
`nomParametre : valeurParametre`

## Syntaxe des messages et des réponses

237

### Exemple

- appeler("MrInfo2", 5)
- afficher(x,y)
- initialiser(x=100)
- f(x:12)

## Syntaxe des messages et des réponses

238

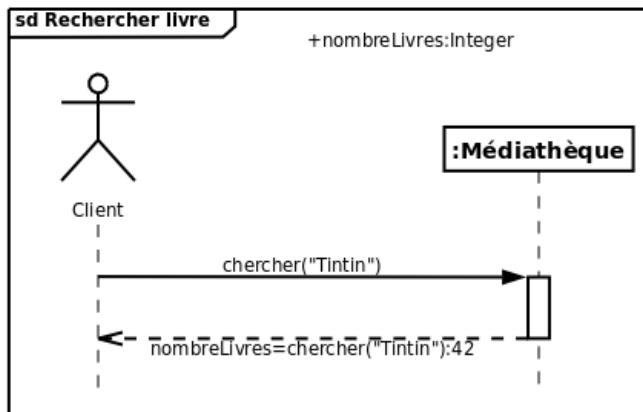
### Messages de retour

- ❑ Le récepteur d'un message synchrone rend la main à l'émetteur du message en lui envoyant un message de retour.
  - ❑ Les **messages de retour** sont **optionnels** : *la fin de la période d'activité marque également la fin de l'exécution d'une méthode.*
  - ❑ Ils sont utilisés pour spécifier le résultat de la méthode invoquée.
  - ❑ La syntaxe est la suivante :
- [attributCible =] nomMessageSynchroneInitial(ListeParams) : valeurRetour
- ❑ Les messages de retour sont représentés en pointillés.

## Syntaxe des messages et des réponses

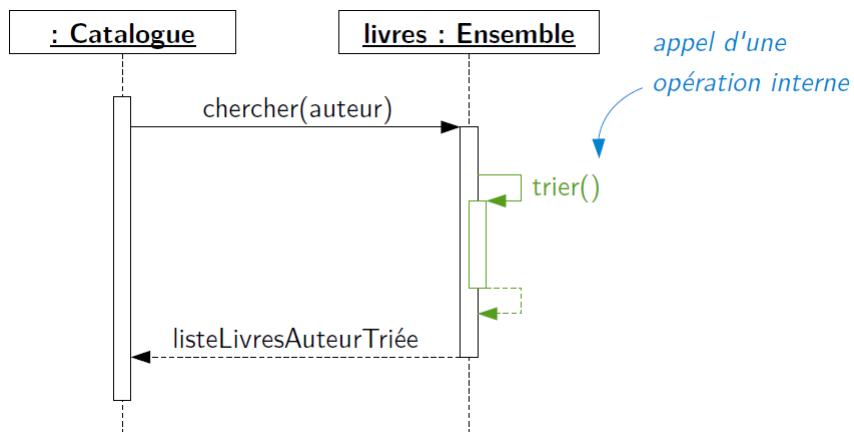
239

### Exemple:



## Message réflexif

240



## Message complet

241

- ❑ Un message complet est tel que les événements d'envoi et de réception sont connus.
- ❑ Un message complet se représente par une simple flèche dirigée de l'émetteur vers le récepteur.

## Message trouvé

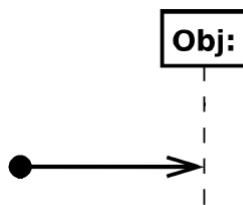
242

- ❑ Un message trouvé est tel que l'événement de réception est connu, mais pas l'événement d'émission.
- ❑ Une flèche partant d'une petite boule noire représente un message trouvé.

# Message trouvé

243

## Représentation d'un message trouvé



## Message perdu

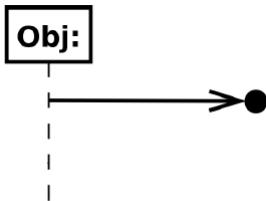
244

- Un message perdu est tel que l'événement d'envoi est connu, mais pas l'événement de réception.
  - Un message perdu se représente par une flèche qui pointe sur une petite boule noire

## Message perdu

245

### Représentation d'un message perdu



## Fragments d'interaction combinés

246

- Un **fragment combiné** représente des articulations d'interactions.
- Il est défini par un **opérateur** et des **opérandes**. L'opérateur conditionne la signification du **fragment combiné**.
- Les **fragments combinés** permettent de décrire des diagrammes de séquence de manière compacte.

## Fragments d'interaction combinés

247

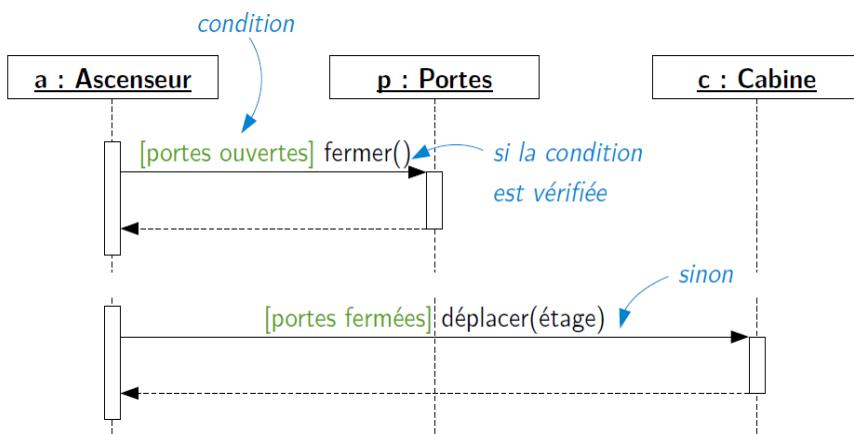
- Un **fragment combiné** se représente de la même façon qu'une interaction.
- Il est représenté dans un rectangle dont le coin supérieur gauche contient un pentagone. Dans le pentagone figure le type de la combinaison, appelé *opérateur d'interaction*.
- Les opérandes d'un opérateur d'interaction sont séparés par une ligne pointillée.
- Les conditions de choix des opérandes sont données par des expressions booléennes entre crochets ([ ]).

## Opérateur Alternative (alt)

248

**Principe :** Condition à l'envoi d'un message

**Notation :** Deux diagrammes



## Opérateur Alternative (alt)

249

### Fragment alt : opérateur conditionnel

Les différentes alternatives sont spécifiées dans des zones délimitées par des pointillés.

- Les conditions sont spécifiées entre crochets dans chaque zones.
- On peut utiliser une clause [else]

## Opérateur Alternative (alt)

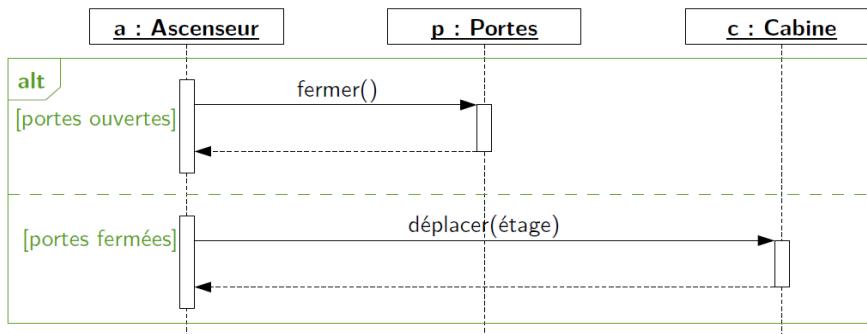
250

### Fragment alt : opérateur conditionnel

**Principe :** Condition à l'envoi d'un message

**Notation :**

- Deux diagrammes
- Bloc d'alternative alt



## Opérateur *opt*

251

### Fragment *opt* : opérateur d'option

- ❑ L'opérateur *option*, ou *opt*, comporte un opérande et une condition de garde associée.
- ❑ Le sous-fragment s'exécute si la condition de garde est vraie et ne s'exécute pas dans le cas contraire.

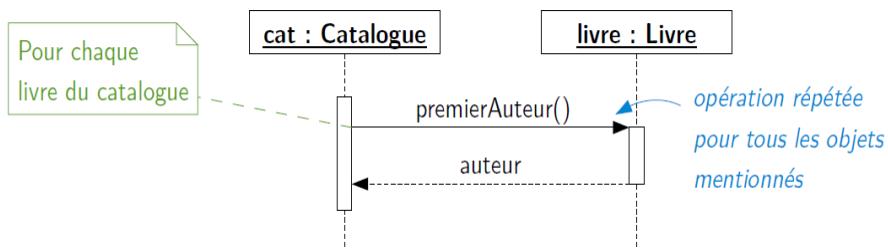
## Opérateur *loop*

252

**Principe :** Répéter un enchaînement de messages

**Notation :**

- Note

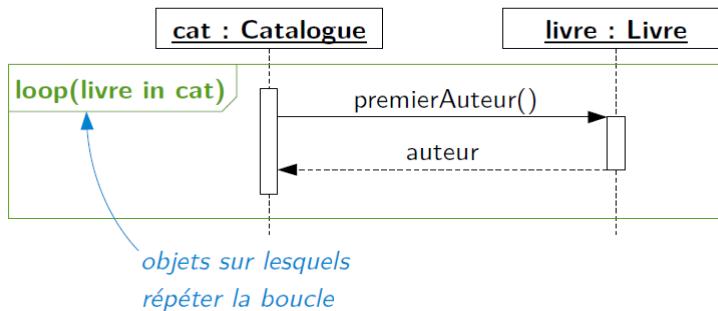


## Opérateur *loop*

253

### Fragment *loop* : opérateur d’itération

- Le fragment ***loop*** permet de répéter ce qui se trouve en son sein.
- On peut spécifier entre crochets à quelle condition continuer.



## Opérateur *loop*

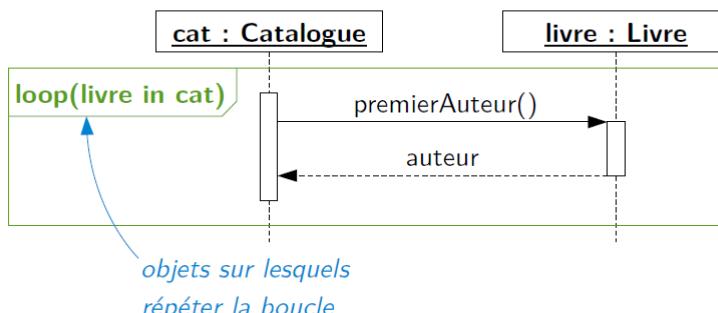
254

### Fragment *loop* : opérateur d’itération

Principe : Répéter un enchaînement de messages

Notation :

- Note
- Bloc de boucle ***loop***



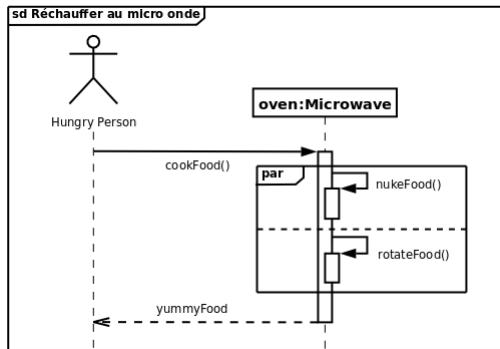
## Opérateur *par*

255

### Fragment par : opérateur de parallèle

L'opérateur *parallel*, ou *par*, possède au moins deux sous-fragments exécutés simultanément.

#### Exemple:



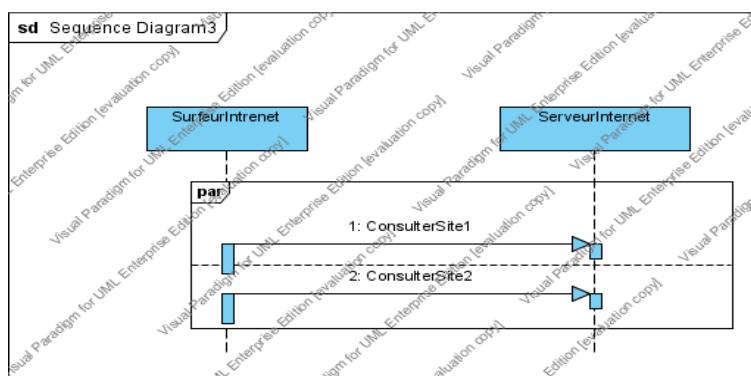
## Opérateur *par*

256

### Fragment par : opérateur de parallèle

Il est utilisé pour représenter des interactions en parallèle.

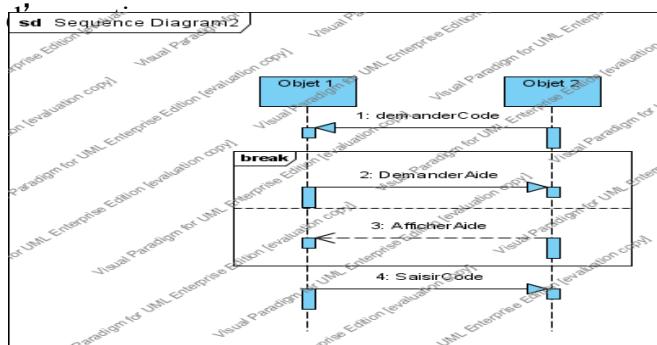
#### Exemple:



## Opérateur break

257

- Le fragment **break** utilisé pour représenter des scénarios



L'exemple montre un opérateur "break" : L'utilisateur (Objet 1), lorsque le distributeur (Objet 2) lui demande son code, peut choisir de rentrer son code ou de consulter l'aide. Si il choisit de consulter l'aide, le flot d'interaction relatif à la saisie du code est interrompu. Les interactions de l'opérateur break sont "exécutées".

## Référence à un autre

258

### Réutilisation de séquences

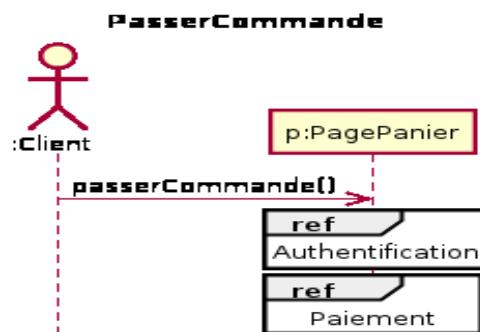
Un fragment **ref** permet d'indiquer la réutilisation d'un diagramme de séquences défini par ailleurs.

## Référence à un autre

259

### Réutilisation de séquences

En supposant qu'il existe un diagramme intitulé *Authentification* et un autre *Paiement*, on peut établir le diagramme suivant :

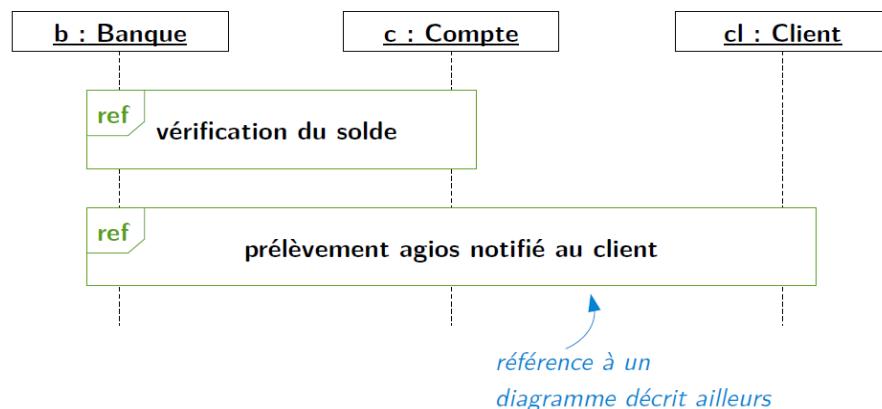


## Référence à un autre

260

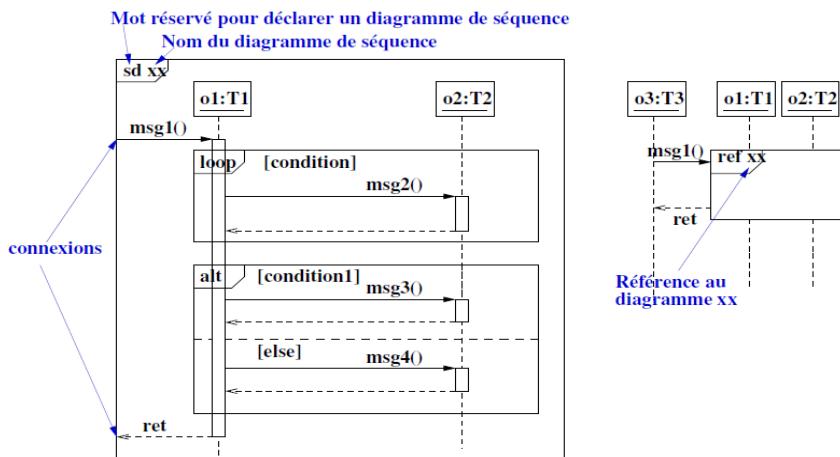
### Réutilisation de séquences

**Référence:** un pointeur ou un raccourci vers un autre diagramme de séquences existant.



## Diagrammes de séquence : Cadres

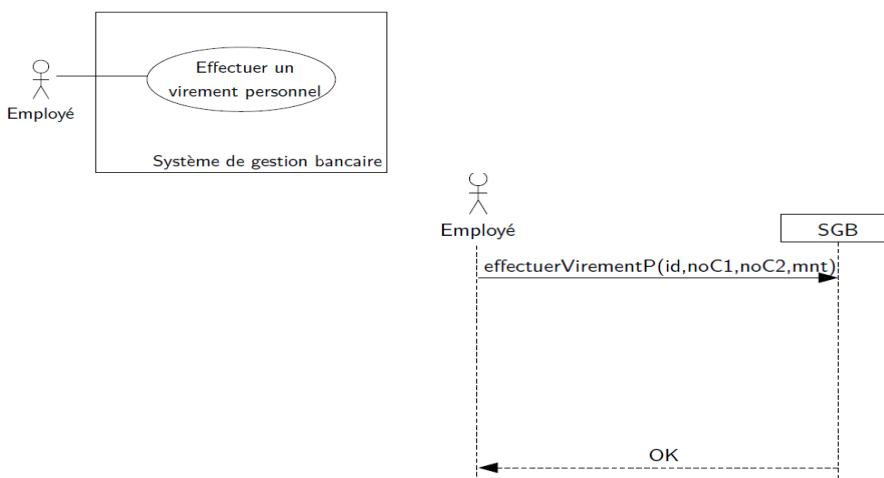
261



## Référence à un autre

262

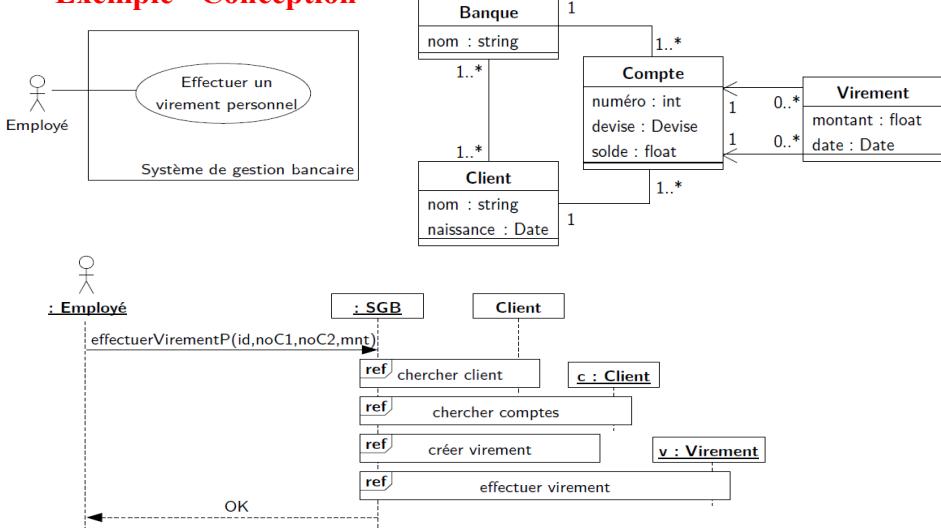
### Exemple – Analyse



## Référence à un autre

263

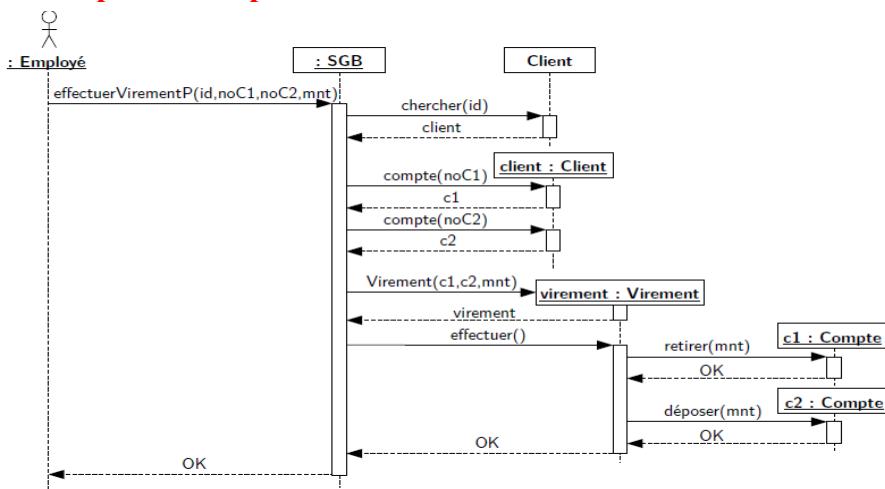
## Exemple - Conception



## Référence à un autre

264

## Exemple - Conception

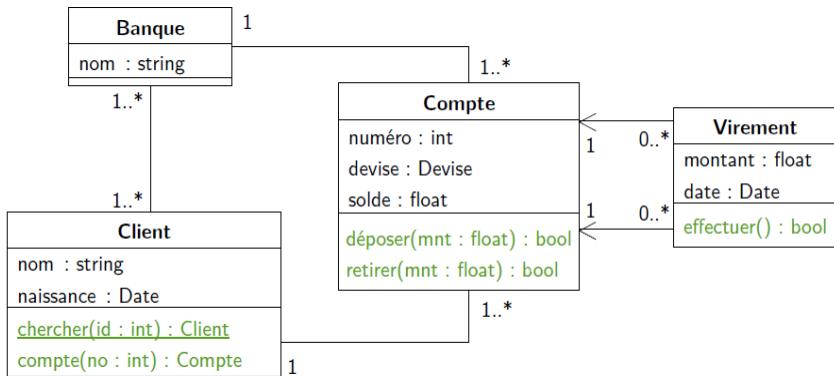


## Référence à un autre

265

### Exemple – Conception

Diagramme de classes **complété** avec les opérations nécessaires



## Quelques règles

266

### Messages au sein du système

- **Opérations** du diagramme de classes
- Si message de **objA : ClasseA** vers **objB : ClasseB**, alors
  - **ClasseA et ClasseB liées par une association**
  - Opération du message dans ClasseB

## Diagramme de séquence

267

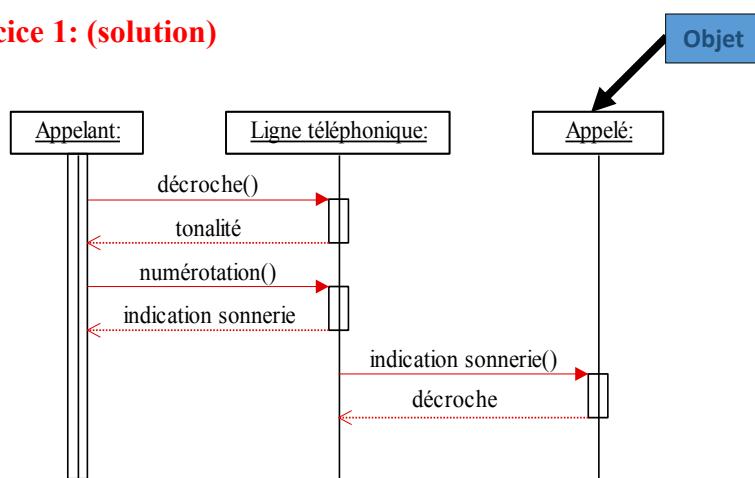
### Exercice 1:

Réaliser un diagramme de séquence pour l'appel d'un utilisateur à un autre par téléphone.

## Diagramme de séquence

268

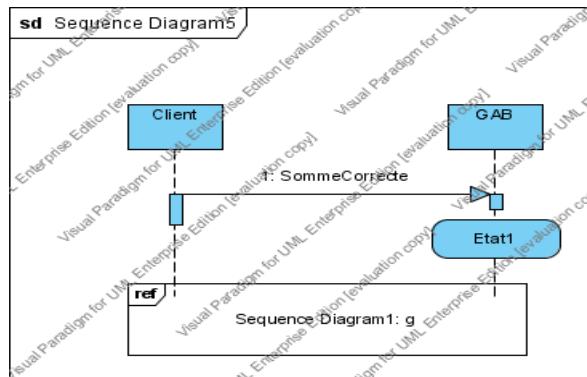
### Exercice 1: (solution)



## Diagramme de séquence

269

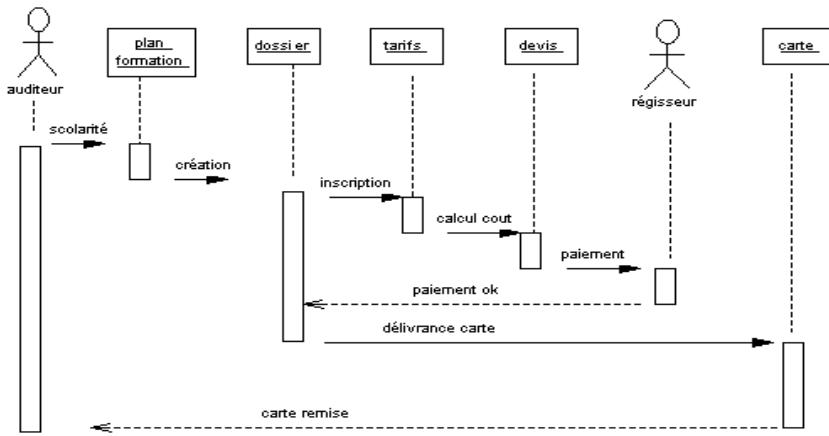
- Etat précise l' état dans lequel doit se trouver l' instance de classe concernée.



## Diagramme de séquence

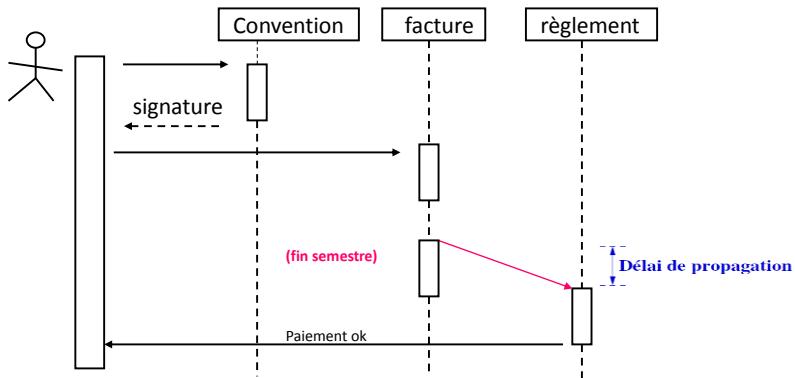
270

### Exemple:



## Diagramme de séquence : durée

271



## Diagramme de séquence

272

### Diagramme de séquence du système GAB

#### Exercice 2:

Réaliser un diagramme de séquence du système GAB qui décrit le scénario nominal du cas d'utilisation RETIRER DE L'ARGENT.

## Diagramme de séquence

273

### Diagramme de séquence du système GAB

#### Solution:

Il suffit de transcrire sous forme de diagramme de séquence les interactions citées dans le scénario nominal de ce cas d'utilisation:

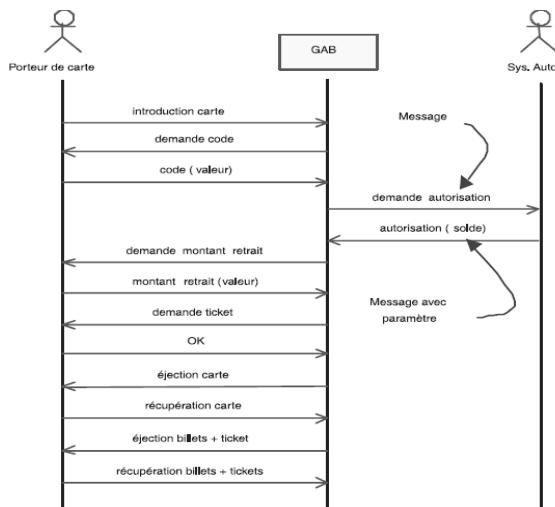
- l'acteur principal Porteur de carte à gauche;
- un participant représentant le GAB au milieu;
- l'acteur secondaire Sys. Auto. à droite du GAB.

## Diagramme de séquence

274

### Diagramme de séquence du système GAB

#### Solution:(suite)



## DiAGRAMME de communication

### Diagramme de communication

276

- ❑ Représentation **spatiale** des objets et de leurs interactions.
- ❑ **Diagramme d'objet** dont les ***associations*** sont étiquetées par les ***messages envoyés***.
- ❑ Structuration en multigraphe: numérotation des arcs pour modéliser l'**ordre des interactions**.

## Diagramme de communication

277

### Représentation des lignes de vie

- Les **lignes de vie** sont représentées par des **rectangles** contenant une étiquette dont la syntaxe est :  
[<nomDuRôle>] : [<NomDuType>]
- Au moins un des deux noms doit être spécifié dans l'étiquette, les deux points ( :) sont, quand à eux, obligatoire.

## Diagramme de communication

278

### Représentation des connecteurs

- Les **relations** entre les **lignes de vie** sont appelées **connecteurs** et se représentent par un **trait plein reliant deux lignes de vies** et dont les extrémités peuvent être **ornées de multiplicités**.

## Diagramme de communication

279

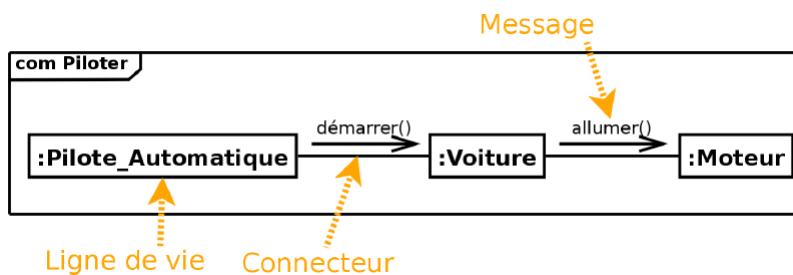
### Représentation des messages

Dans un **diagramme de communication**, les **messages** sont généralement **ordonnés** selon un **numéro de séquence croissant**.

## Diagramme de communication

280

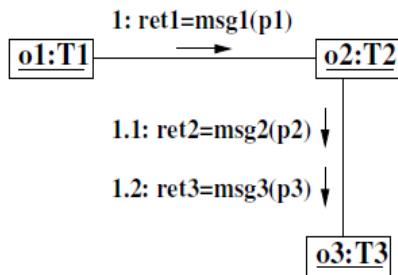
### Exemple 1:



## Diagramme de communication

281

### Exemple 2:

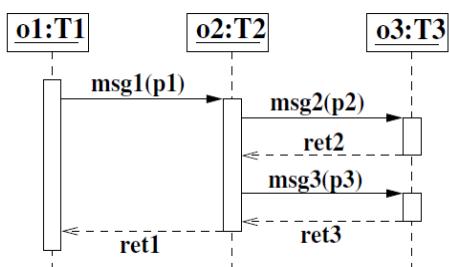


## Diagramme de communication

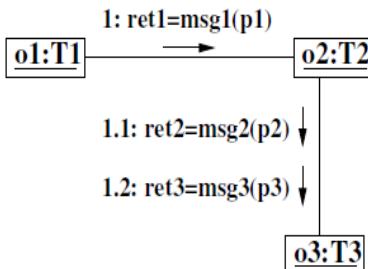
282

### Exemple 3:

#### Diagramme de séquence



#### Diagramme de communication



## Plan

283

- Diagramme états-transitions
- Diagramme d'activités

## L' Axe Dynamique

## Que représente-t-on dans le modèle dynamique ?

285

- Le modèle dynamique représente les séquences **d' événements, d' états et de réactions** qui doivent survenir dans le système.
- Il est intimement lié au modèle objet et décrit les aspects de contrôle d' un système en prenant compte du **temps, du séquencement des opérations** et des **interactions** entre objets
- Deux diagrammes fondamentaux :
  - **Diagramme d' activités**
  - **Diagramme Etats-Transitions**

## Définition

286

- Un diagramme **Etats-Transitions** (ou **Automate**) :
  - décrit l' **évolution** au cours du **temps** d' une instance d' une classe en réponse aux interactions avec d' autres objets
  - est forcément associé à **une classe**, mais toutes les classes n' en ont pas besoin
  - est un graphe orienté d' **états** (noeuds) connectés par des **transitions** (arc orientés)
- Source: Les **Statecharts** de **David Harel**

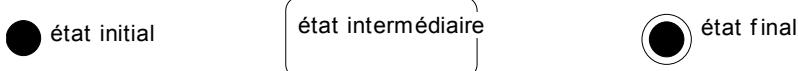


David Harel

## Etats

287

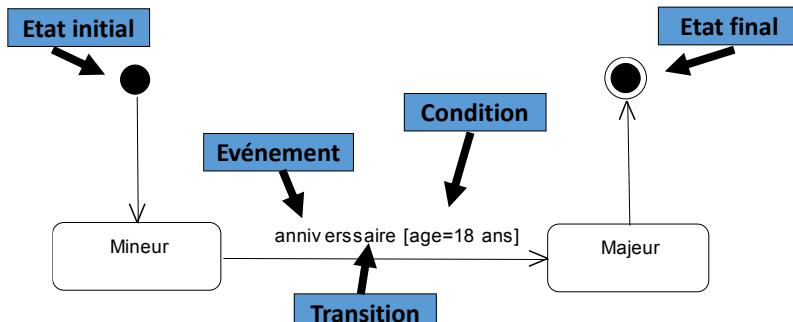
- Chaque objet est à un moment donné dans un **état** particulier :
  - **État initial** : état d' une instance juste après sa création (un seul état initial)
  - **État intermédiaire** : un objet est toujours dans un état donné pour un certain temps
  - **État final** : état d' une instance juste avant sa destruction (un automate infini peut ne pas avoir d' état final)



## Transition, Condition

288

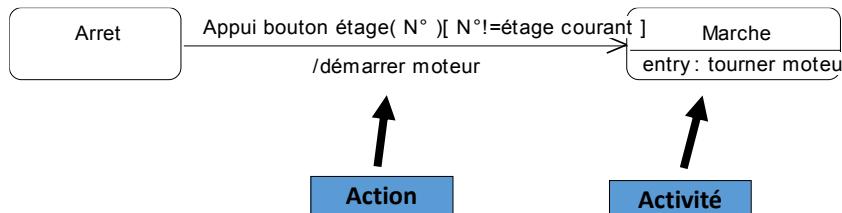
- **Transition** : relation entre 2 états indiquant qu' un objet dans le premier état va exécuter une **action** et entrer dans le deuxième état quand un **événement** apparaîtra
- **Condition** : expression booléenne devant être vérifiée pour permettre la transition



## Action, Activité

289

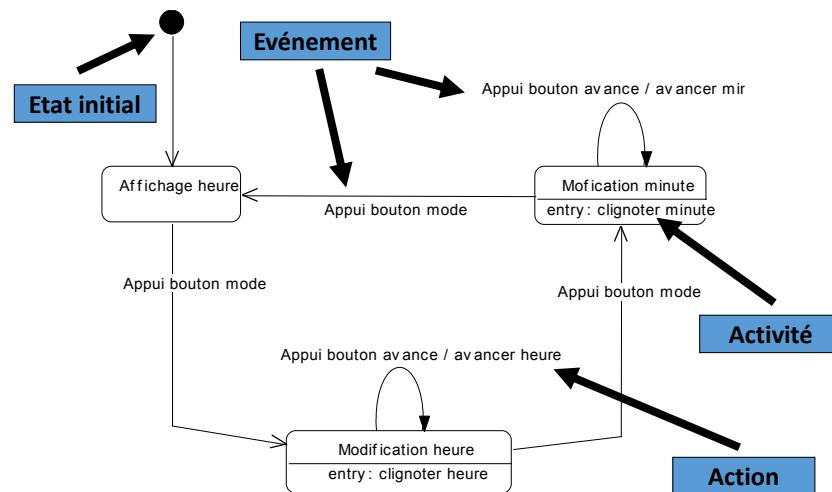
- **Action** : opération **atomique** (non interruptible) déclenchée par une transition
- **Activité** : opération qui **dure** un certain temps (interruptible) dans un état particulier
  - **entry** : action exécutée chaque fois que l'on **rentre** dans l'état
  - **exit** : action exécutée chaque fois que l'on **quitte** l'état



## Notation Complète

290

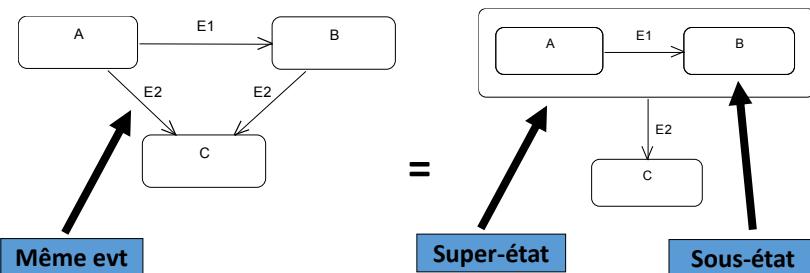
- **Exemple :** fonctionnement d'une montre digitale



## Généralisation d' états

291

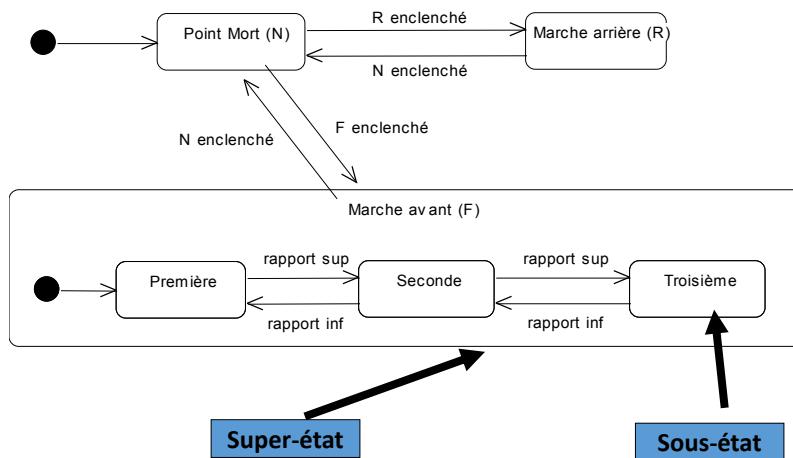
- Dans le cas d'un comportement dynamique **complexe**, les diagrammes d'états sur **un** niveau deviennent rapidement **illisibles**
- Pour éviter ce problème, il est nécessaire de **structurer** les diagrammes d'états en:
  - super-états** : états généraux
  - sous-états** : héritent des caractéristiques des états généraux



## Notation Complète

292

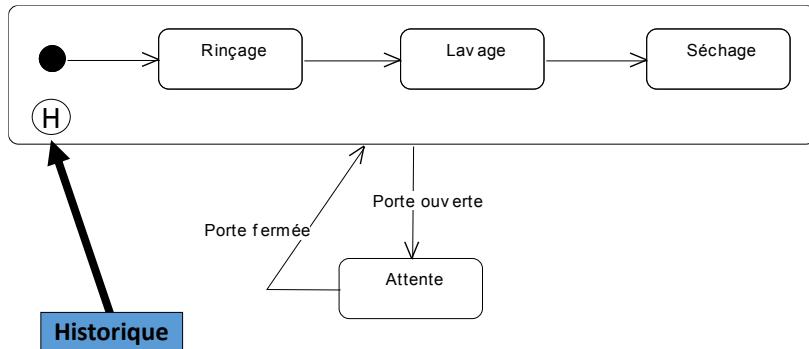
- Exemple :** transmission d'une automobile



## Historique

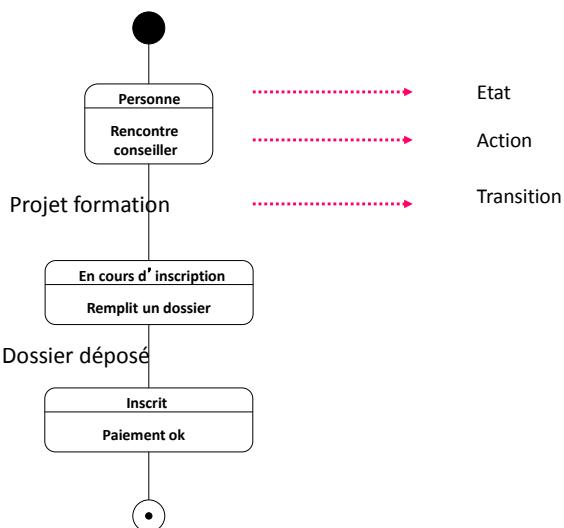
293

- Par défaut, un automate n'a **pas de mémoire**
- La notation **H** offre un **mécanisme pour mémoriser** le dernier sous-état qui l'englobe
- **Exemple :** cycle de lavage d'un lave vaisselle



## Exemple d' états-transitions

294



294

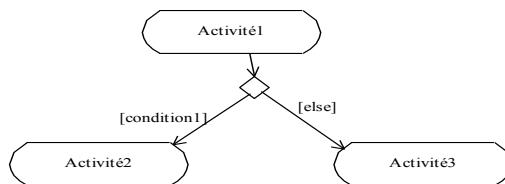
# Diagramme d'activités

## Diagramme d'activités

296

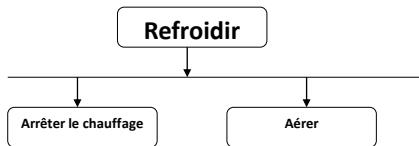
- ❑ Utilisé pour étiqueter les autres diagrammes (traitements associés aux messages des diagrammes de séquences, transitions des diagrammes d'états-transitions, activité d'un état...).
- ❑ Sert à spécifier un traitement à priori séquentiel en offrant un pouvoir d'expression très proche des algorithmes.

### Branche conditionnelle

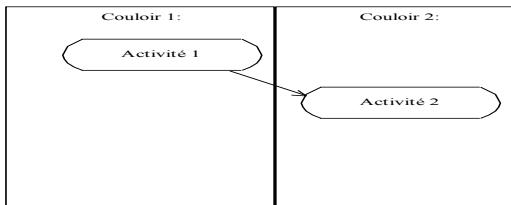


297

### Barre de synchronisation :

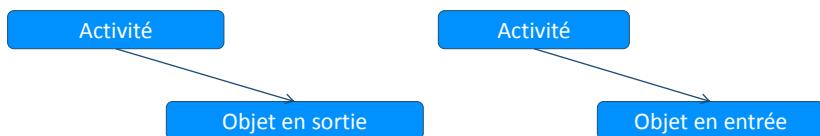


### Couloir d'activités :

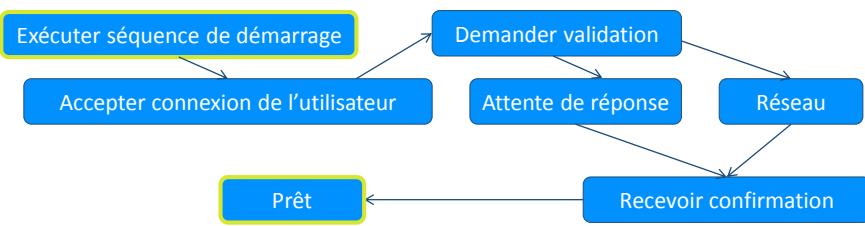


298

### Flux d'objets



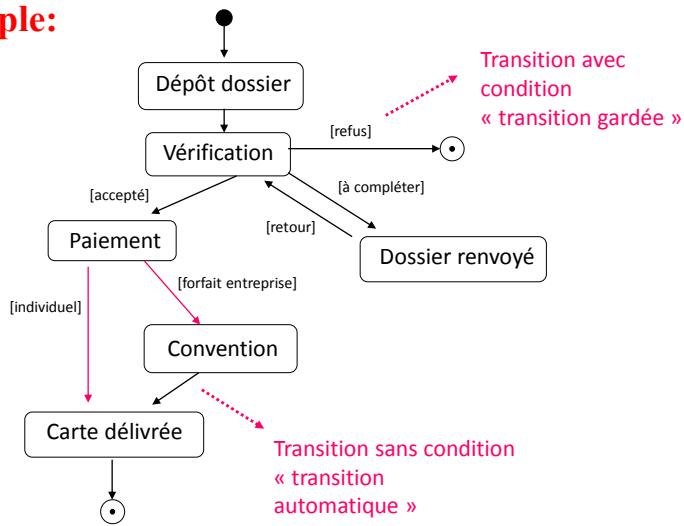
### Signal



## Diagramme d'activités

299

### Exemple:



## Les apports d'UML 2

## Diagramme de collaboration/communication

301

- Relations entre les **objets** et **messages** échangés
- Diverses informations mentionnées sur le diagramme
  - synchronisation : les préalables à l'envoi du message;
  - n° du message : ordre chronologique du message;
  - condition du déclenchement de l'envoi;
  - type d'envoi: séquentiel ou parallèle;
  - résultat : valeur renournée...

### Exemples:

[activité professionnelle=Ok] 1: établir plan()

Un élève ne peut s'inscrire que si il a une activité professionnelle

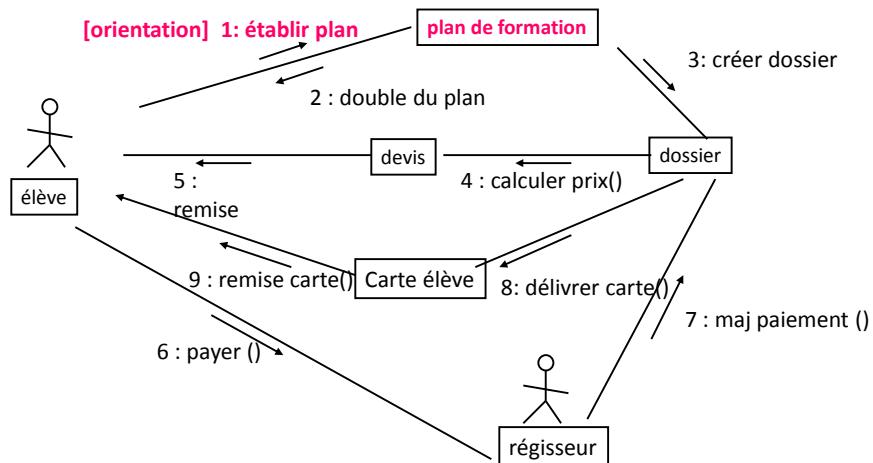
2 / || [j=1...n] 3 : inscrireUE()

Le message 3 ne sera envoyé qu'après le message 2

Envoi en // de n messages

## Exemple : Diagramme de collaboration

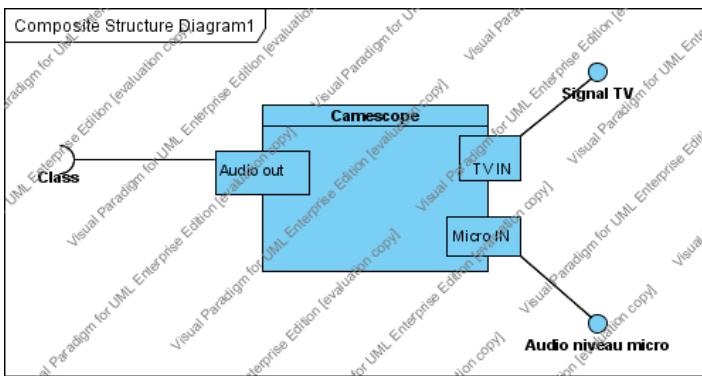
302



## Diagramme de structure composite

303

Spécifier la connectique interne et externe entre ces classes :



## Synthèse rapide

304

- **Diagramme de cas d'utilisation**
  - *Ce qu'on attend du système*
- **Diagramme de classes**
  - *Les entités du système*
- **Diagrammes de séquence et de collaboration**
  - *Comment les entités interagissent*
- **Diagrammes d'états - transitions et d'activités**
  - *Les états et les fonctionnalités de chaque entité*
- **Diagrammes de composants et de déploiement**
  - *L'architecture du système*

## Les outils de modélisation UML

305

- Plusieurs logiciels, citons :
  - Rational Rose, IBM
  - Together, Borland
  - Rhapsody Modeler, I-Logix
  - Win Design
  - Visio, Microsoft
  - Poseidon UML , Gentleware
  - PowerAMC/PowerDesigner , Sybase
  - Plugin Omondo , Eclipse
  - ArgoUML
  - Visual Paradigm
  - ...