Cours « Atelier UML »

Tarak Chaari

Maître assistant à l'institut supérieur d'électronique et de communication

tarak.chaari@gmail.com

Votre interlocuteur

- Tarak CHAARI
- Maître assistant à l'ISECS
- Membre de l'unité de recherche RedCad (ENIS)
- Recherche: l'adaptation dans les environnements dynamiques
- Enseignement: Ingénierie des systèmes d'information

Présentation générale du cours

Le nom du cours

- Atelier UML

Volume horaire

- 10.5 heures
- Cours

Objectifs

- Comprendre les fondements de base de UML
- Pouvoir utiliser et appliquer UML dans des cas réels

Contenu du cours

Introduction générale

- Rappels sur les fondements objet
- Importance de la modélisation
- Présentation générale d'UML

Modélisation objet avec UML

- Notations
- Diagrammes UML et leurs utilités
- Exemples
- Exercices

En rodage !!!

Introduction générale et rappels

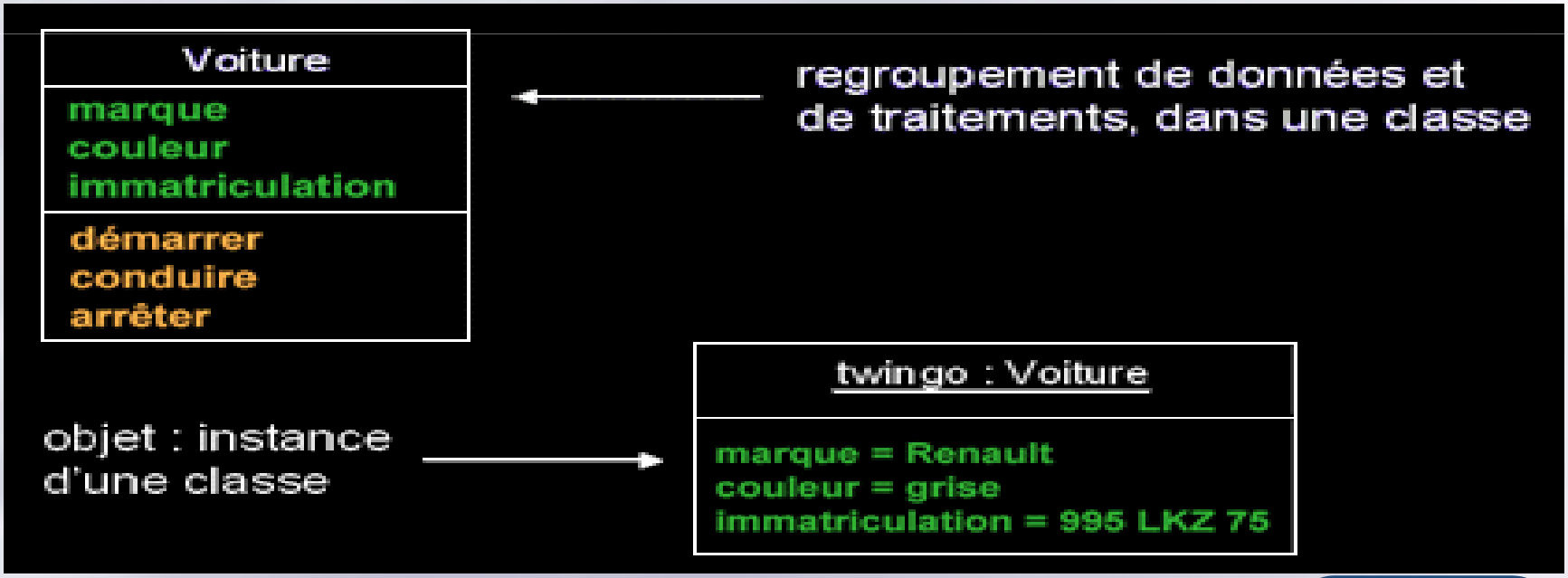
Vision objet d'un système d'information (1)

- Un SI = un ensemble d'objets qui collaborent entre eux

- Un objet représente une entité du système qui est caractérisée par:
  - Des frontières précises
  - Une identité (ou référence)
  - Un ensemble d'attributs (propriétés) décrivant son état
  - Un ensemble de méthodes (opérations) définissant son comportement

Vision objet d'un système d'information (2)

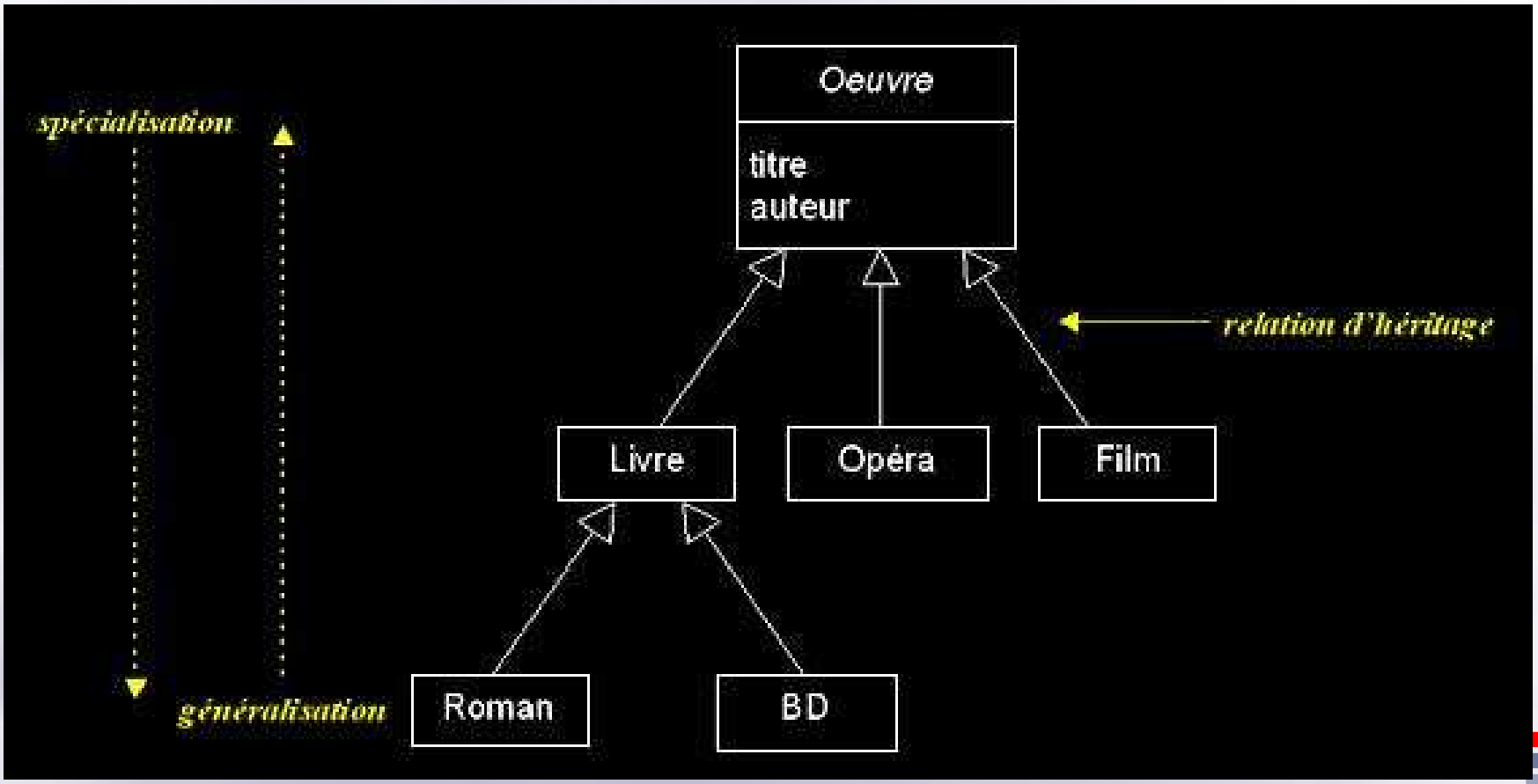
- Un objet est une instance de classe (une occurrence d'un type abstrait)
- Une **classe** est un type de données abstrait(modèle) , caractérisé par des propriétés (attributs et méthodes) communes à des objets et permettant de créer des objets possédant ces propriétés.



Vision objet d'un système d'information (3)

Héritage

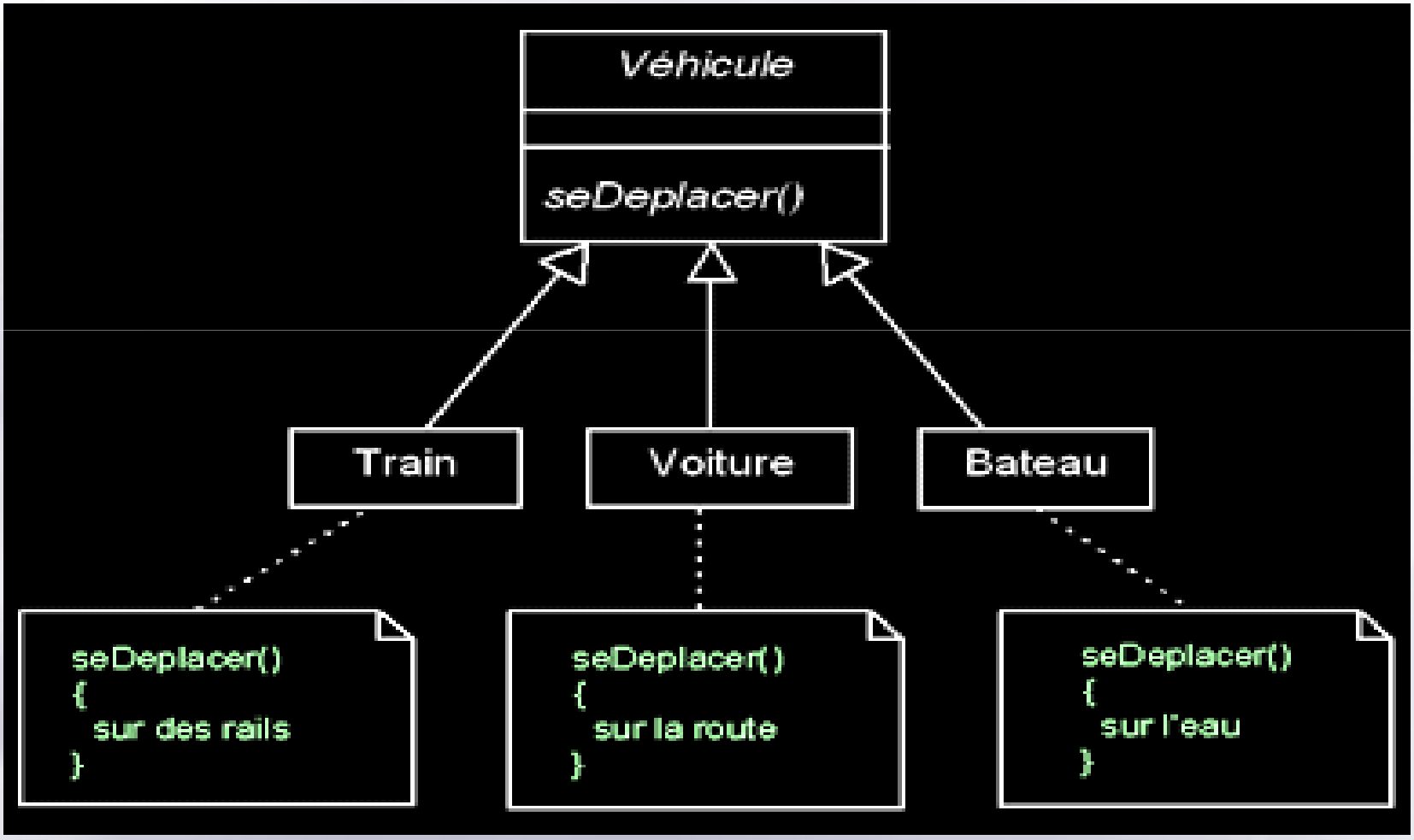
- **Transmission de propriétés (attributs et méthodes) d'une classe à une sous classe d'objets**



Vision objet d'un système d'information (4)

Polymorphisme

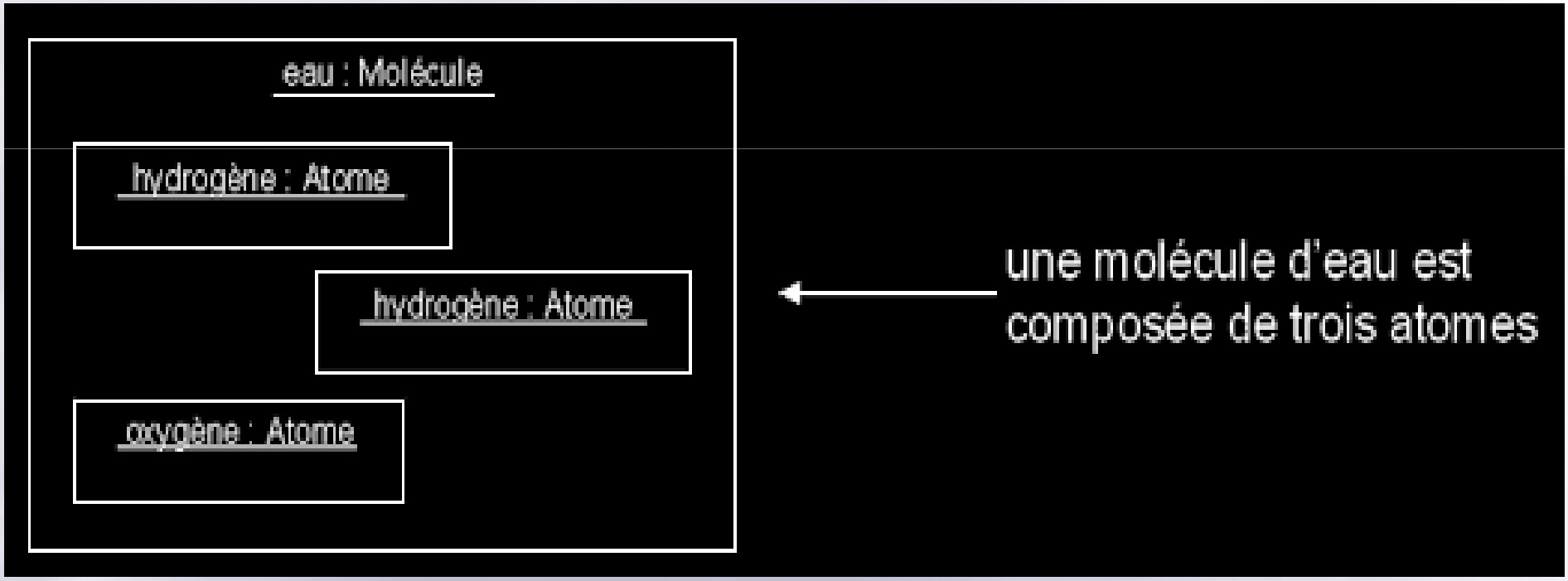
- Factorisation de comportement (méthodes) commun d'objets



Vision objet d'un système d'information (5)

L'agrégation

- Une relation d'agrégation permet de définir des objets composés d'autres objets.
- L'agrégation permet d'assembler des objets de base, afin de construire des objets plus complexes.



Résumé des concepts fondateurs de l'approche objet (1)

- L'héritage est un mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe.

- Une classe peut être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques ou d'en adapter certaines.

- Plusieurs classes peuvent être généralisées en une classe qui les factorise afin de regrouper les caractéristiques communes d'un ensemble de classes.

Résumé des concepts fondateurs de l'approche objet (2)

- La spécialisation et la généralisation permettent de construire des hiérarchies de classes. L'héritage peut être simple ou multiple.
- L'héritage évite la duplication et encourage la réutilisation.
- Le polymorphisme représente la faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes.
- Le polymorphisme augmente la généricité du code.

L’approche objet : une solution parfaites?

- L'approche objet est moins intuitive que l'approche fonctionnelle
  - Quel moyen utiliser pour faciliter l’analyse objet?
  - Quels critères identifient une conception objet pertinente ?
- L'application des concepts objets nécessite une grande rigueur
  - Le vocabulaire présente des d'ambiguïtés et des d’incompréhension
  - Comment décrire la structure objet d'un système de manière pertinente?
  - Comment décrire l’interaction entre ces objets de manière précise?

Remèdes aux inconvénients de l’approche objet

- **Un langage (ou modele) pour exprimer les concepts objet qu'on utilise, afin de pouvoir :**
  - Représenter des concepts abstraits (graphiquement par exemple)
  - Limiter les ambiguïtés (parler un langage commun)
  - Faciliter l'analyse (simplifier la comparaison et l'évaluation de solutions)
- **Une démarche d'analyse et de conception objet pour :**
  - Ne pas effectuer une analyse fonctionnelle et se contenter d'une implémentation objet, mais penser objet dès le départ
  - Définir les vues qui permettent de couvrir tous les aspects d'un système, avec des concepts objets

Dans quel cas modéliser et analyser?

- **Construire**
  - **une niche: c'est facile**
  - **une maison: c'est un peu plus compliqué**
  - **un immeuble: bon là, comment s'y prendre?**

- **Nombreux fabricants de logiciels veulent construire « des immeubles » mais abordent le problème comme d'ils devaient bricoler une niche**

- ***Nous construisons des modèles pour les systèmes complexes parce que nous ne sommes pas en mesure d'appréhender de tels systèmes dans leur intégralité.***

  Le guide de l'utilisateur UML**,** Grady Booch, James Rumbaugh et Ivar Jacobson Eyrolles**,** Octobre 20002.

Mais pourquoi analyser et modéliser?

- **Pour éviter les résultats négatifs suivant:**
  - Les modules ne fonctionnent pas ensemble
  - Le produit est non conforme avec les besoins
  - Le produit est difficile à maintenir et à faire évoluer
  - Le produit présente beaucoup d'anomalies et de bugs
- **Mais aussi pour**
  - Simplifier la problématique posée par le client
  - « Visualiser » le système
  - Spécifier sa *structure* et son *comportement*
  - Aider à sa construction
  - Travailler en équipe

Et c’est quoi un modèle?

- Un modèle est une abstraction de la réalité
- Il s'agit d'un processus qui consiste à identifier les caractéristiques intéressantes d'une entité, en vue d'une utilisation précise.
- L'abstraction désigne aussi le résultat de ce processus, c'est-à-dire l'ensemble des caractéristiques essentielles d'une entité, retenues par un observateur.
- Un modèle est une vue subjective mais pertinente de la réalité
- Un modèle définit une frontière entre la réalité et la perspective de l'observateur. Ce n'est pas "la réalité", mais une vue très subjective de la réalité.
- Bien qu'un modèle ne représente pas une réalité absolue, un modèle reflète des aspects importants de la réalité, il en donne donc une vue juste et pertinente.

Et UML dans tout ça?

- UML: *langage standard* de *modélisation* des systèmes d'information

- UML: langage visuel pour
  - comprendre le système
  - communiquer et travailler à plusieurs
  - aider à spécifier, concevoir et développer un système d'information

  → avec différents modèles et différentes vues

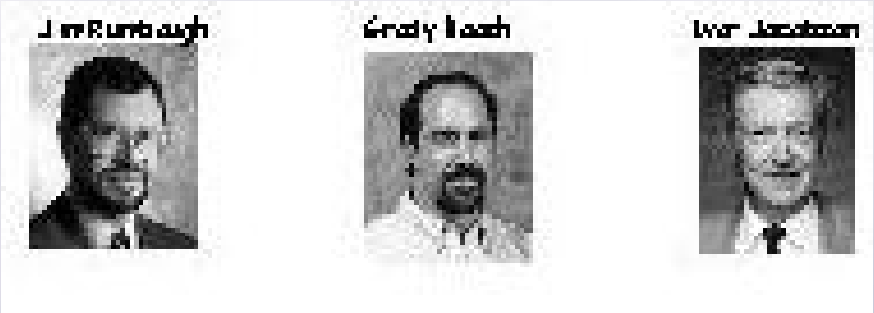
Et dans quel cas l'utiliser?

- **Systèmes à forte composante logicielle**
  - systèmes d'information pour les entreprises
  - les services bancaires et financiers
  - les télécommunications
  - les transports
  - la défense / l'aérospatiale
  - le commerce de détail
  - l'électronique médicale
  - les services distribués et les applications WEB
- **Pas limité à un domaine précis**

UML: un peu d'histoire

3 fondateurs principaux

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



Historique

- En 1995: Méthode unifiée 0.8 (intègrant les méthodes Booch'93 et OMT)
- En 1995: UML 0.9 (intègrant la méthode OOSE)
- En 1996: UML 1.0 (proposée à l'OMG)
- En 1997: UML 1.1 (standardisée par l'OMG)
- En 1998: UML 1.2
- En 1999: UML 1.3
- En 2000: UML 1.4
- En 2003: UML 1.5
- En 2004: UML 2.0
- …
- En 2010: UML 2.3 beta

Caractéristiques d'UML

- **UML cadre l'analyse objet, en offrant**
  - différentes vues (perspectives) complémentaires d'un système, qui guident l'utilisation des concept objets,
  - plusieurs niveaux d'abstraction, qui permettent de mieux contrôler la complexité dans l'expression des solutions objets.

- **UML est un support de communication**
  - Sa notation graphique permet d'exprimer visuellement une solution objet.
  - L'aspect formel de sa notation limite les ambiguïtés et les incompréhensions.
  - Son aspect visuel facilite la comparaison et l'évaluation de solutions.
  - Son indépendance (par rapport aux langages d'implémentation, domaine d'application, processus...) en font un langage universel.

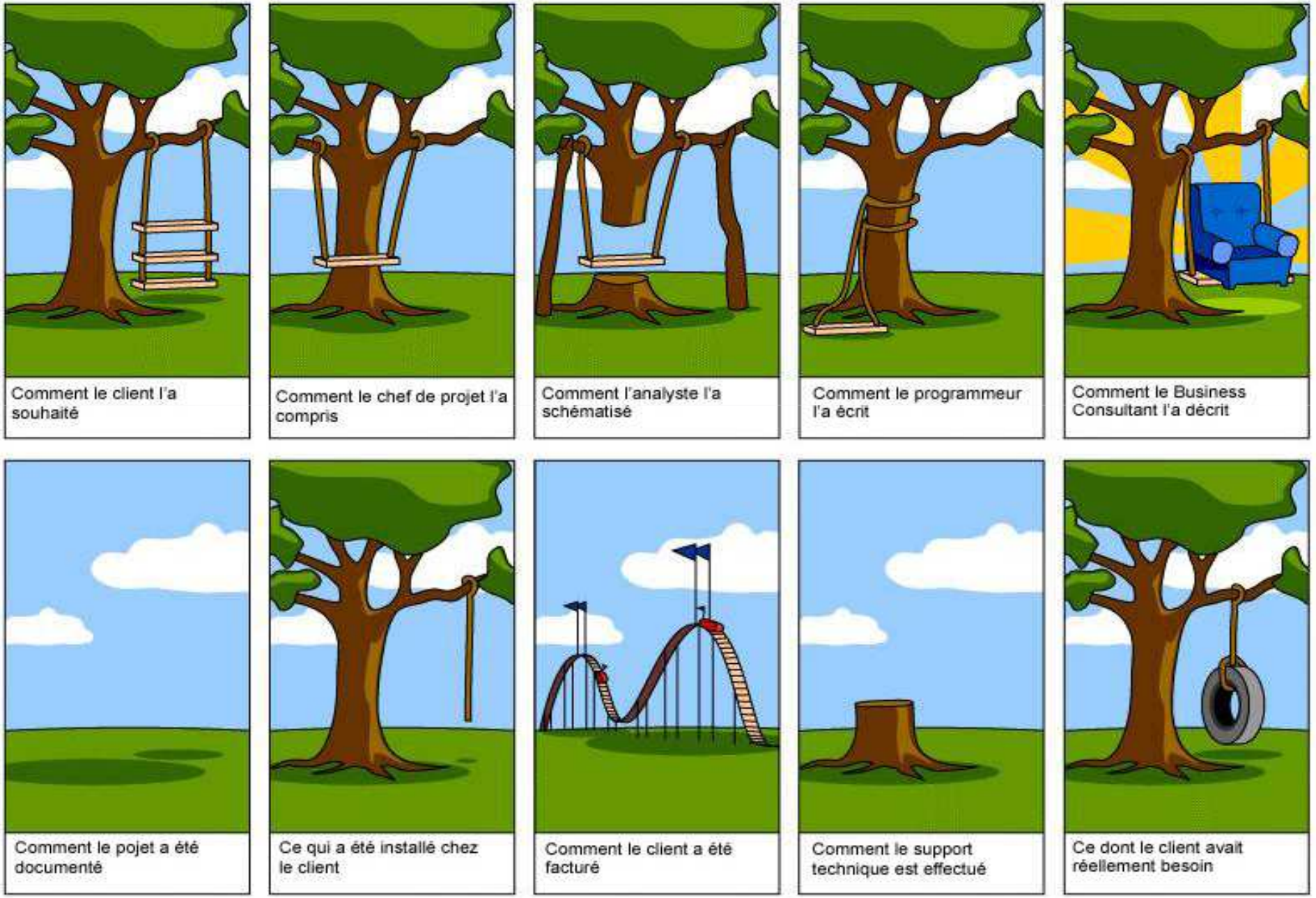
Alors UML, c'est la solution à tout?

- UML n'est qu'un ensemble de formalismes permettant d'appréhender un problème et de le modéliser
- Un formalisme n'est qu'un outil
- Le succès = savoir utiliser les outils
- Ce n'est pas un algorithme ou une méthode automatique à appliquer
- UML laisse la liberté de « penser » ☺
- IMAGINATION IS MORE IMPORTANT THAN KNOWLEGDE (A. Einstein)

Et comment l'utiliser alors?

- **Avant tout, une bonne démarche qui permet de :**
  - Bien comprendre les demandes et exigences des utilisateurs finaux
  - Bien communiquer avec le client
  - Tenir compte des changements du cahier des charges
  - Empêcher la découverte tardive de défauts sérieux dans le projet
  - Traiter au plus tôt tous les points critiques du projet
  - Bien maîtriser la complexité
  - Favoriser la réutilisation
  - Définir une architecture robuste
  - Faciliter le travail en équipe

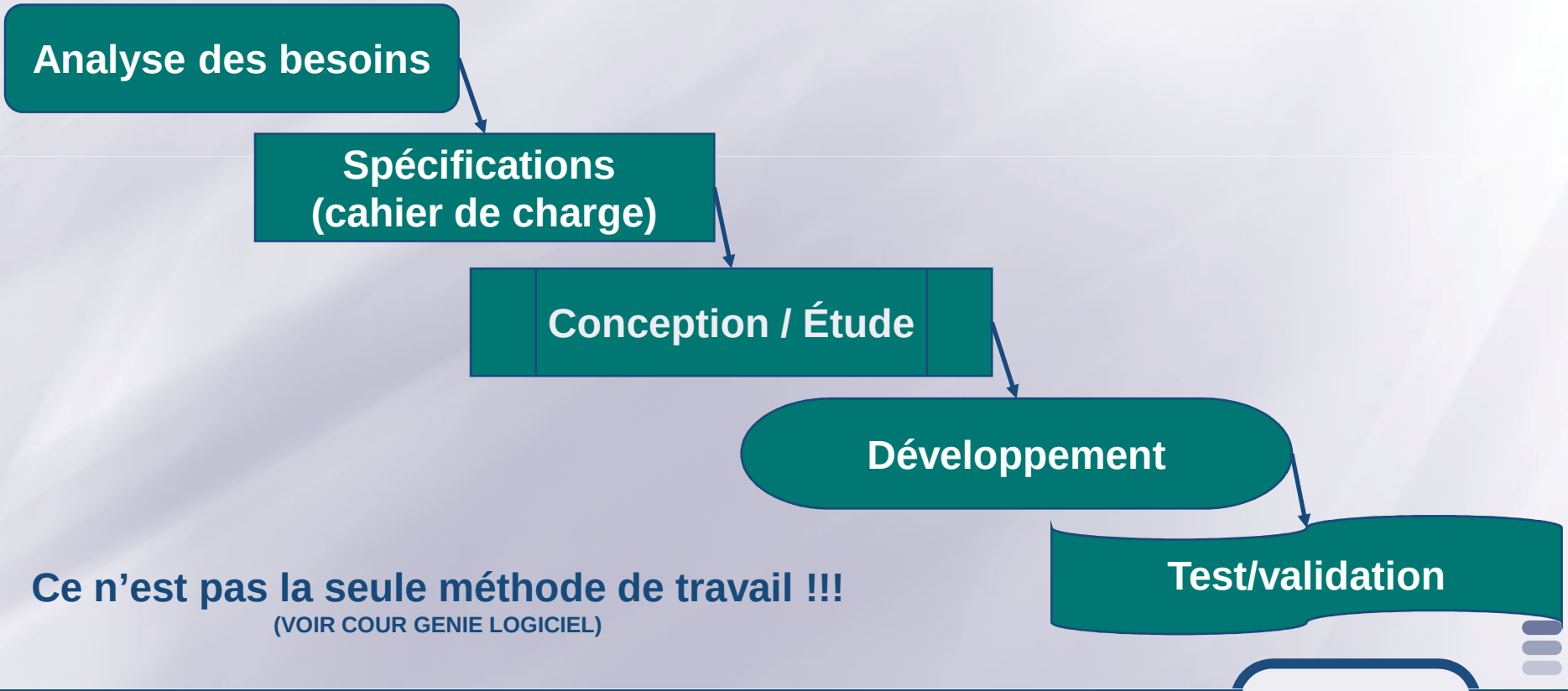
Et pourquoi une démarche?



Les démarches ou méthodes

3 types de démarches:

- Itérative et incrémentale
- guidée par l'utilisateur
- centrée sur l'architecture



Ce n'est pas la seule méthode de travail !!!
(VOIR COUR GENIE LOGICIEL)

Passons aux choses sérieuses !!!

MODELISATION OBJET AVEC UML

Les éléments de modélisation (1)

Les objets

- la description d'une entité du monde réel ou virtuel

Les classes

- la description d'un ensemble d'objets

Les états

- une étape de la vie d'un objet

Les acteurs

- utilisateurs finaux du système

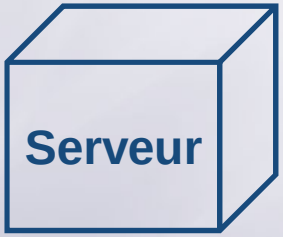
Les éléments de modélisation (2)

Les cas d'utilisation

- Une manière dont un acteur utilise le système

Les noeuds

- Un dispositif matériel



Les paquetages

- Une partition du modèle

Les notes

- Un commentaire, une explication ou une annotation

Les diagrammes UML

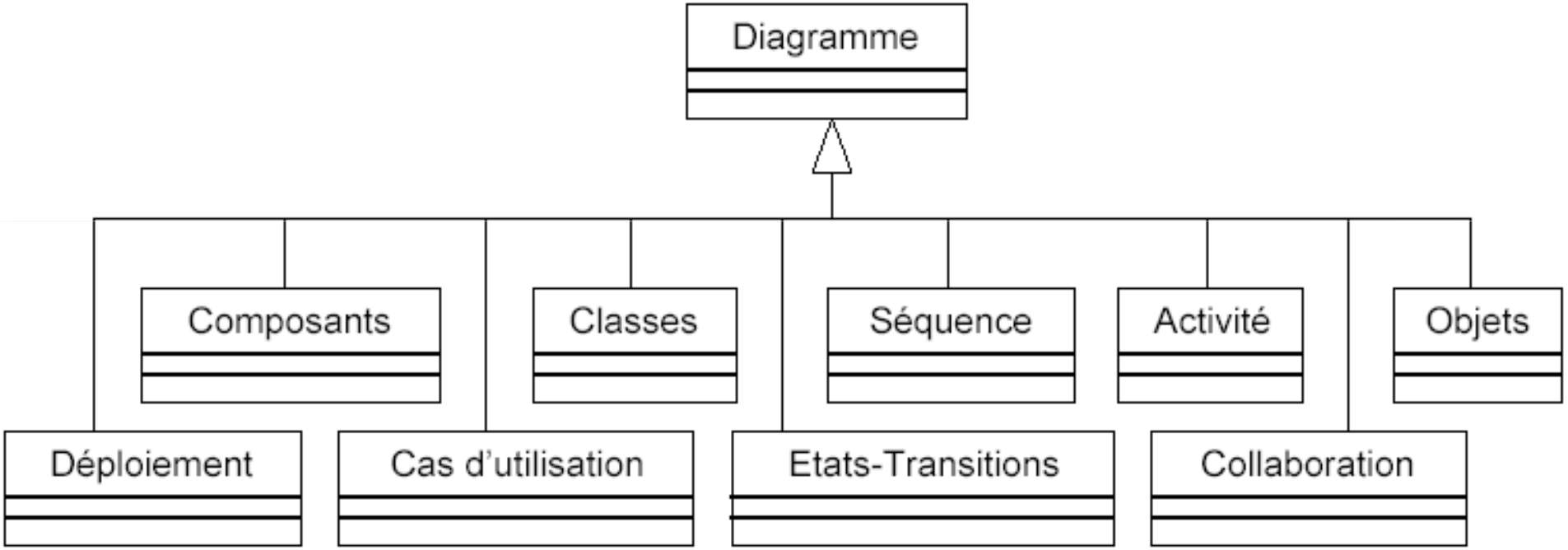


Diagramme des cas d'utilisation

(Ivar Jacobson)

Diagramme de cas d'utilisation (0)

- Une réflexion sur les fonctionnalités attendues du futur système avant la conception
- Avoir une idée (même assez vague) sur les grands modules du système
- Les fonctionnalités que doit fournir chaque composant
- Ces fonctionnalités vont aider les utilisateurs à effectuer leur « mission »

Diagramme de cas d'utilisation (1)

- La détermination et la compréhension des besoins sont souvent difficiles

➔ il faut clarifier et organiser les besoins des clients (les modéliser).

- Les use cases permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système
- Ils centrent l'expression des exigences du système sur ses utilisateurs
- Ils se limitent aux préoccupations "réelles" des utilisateurs ; ils ne présentent pas de solutions d'implémentation et ne forment pas un inventaire fonctionnel du système
- Ils identifient les utilisateurs du système (acteurs) et leur interaction avec le système
- Ils permettent de classer les acteurs et structurer les objectifs du système

Diagramme de cas d'utilisation (2)

- Le DCU décrit sous la forme d'*actions* et de *réactions* le comportement *externe* du système
- Le comportement externe = du point de vue des utilisateurs
- Il permet de définir les frontières du système et les relations entre le système et l'environnement
- Un DCU comprend les acteurs, le système et les ses cas d'utilisation.

Diagramme de cas d'utilisation (3)

Qu'est ce qu'un cas d'utilisation?

- Un cas d'utilisation (use case) représente un ensemble de séquences d'actions réalisées par le système et produisant un résultat observable intéressant pour un acteur particulier

- Un cas d'utilisation modélise un service rendu par le système. Il exprime les interactions acteurs/système et apporte une valeur ajoutée « notable » à l'acteur concerné

- Exemples: Achat billet, Ouverture compte, Livraison Client, Traiter commande, établir facture…

Diagramme de cas d'utilisation (4)

Qu'est ce qu'un acteur?

- Un acteur représente le rôle joué par quelque chose ou quelqu'un se trouvant dans l'environnement du système étudié
- Un acteur est en relation avec le métier de l'entreprise et interagit avec le système dans différents cas d'utilisation
- Il peut être un élément de la structure de l'entreprise tel qu'une direction, un service ou un poste de travail

Diagramme de cas d'utilisation (5)

Types d'acteurs

- Par défaut, le rôle d'un acteur est « principal ». Si ce n'est pas le cas on peut indiquer le rôle « secondaire » sur l'acteur
- Si un acteur a pour rôle unique de consommer des informations du système sans modifier l'état de celui-ci au niveau métier alors son rôle est secondaire
- Un acteur peut aussi être un périphérique externe ou un système externe

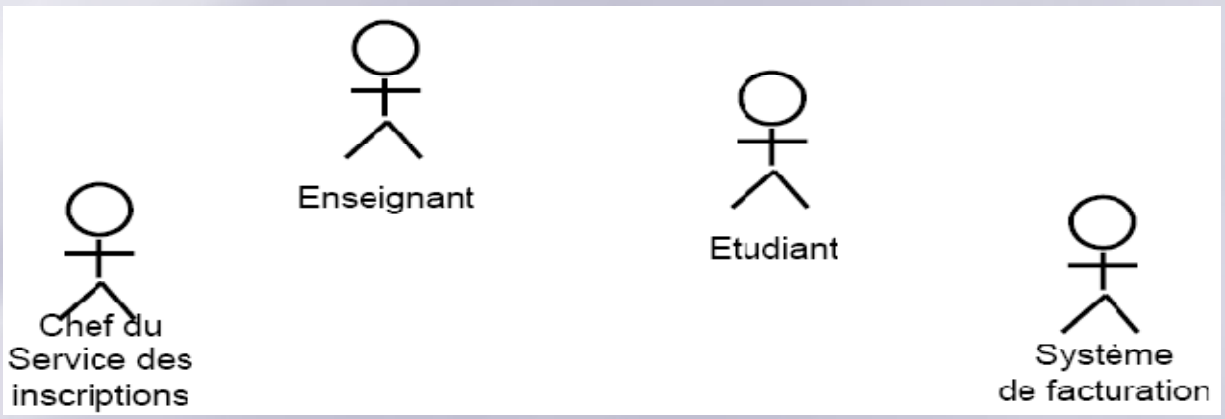
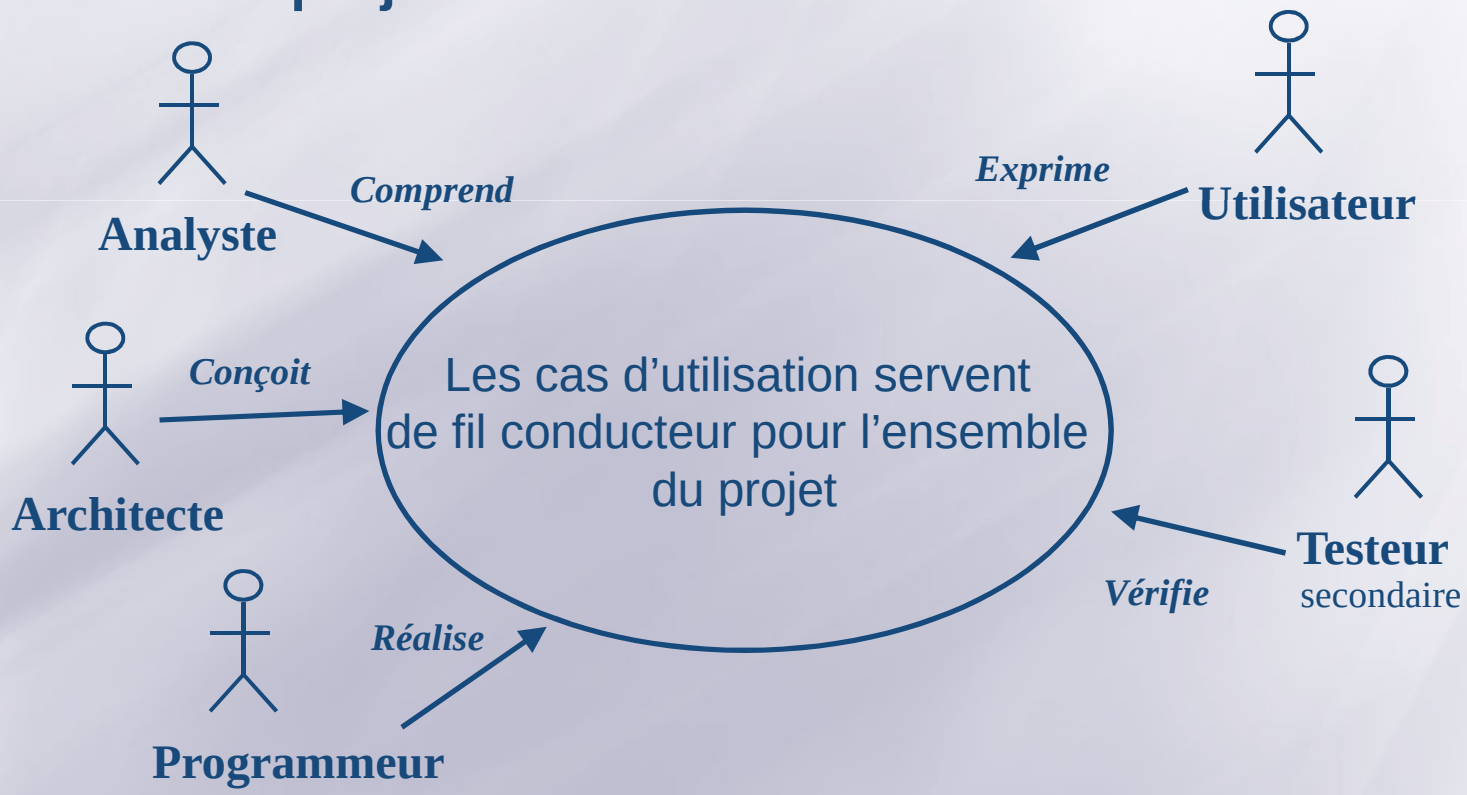


Diagramme de cas d'utilisation (6)

Les cas d'utilisation: portée

- **Les Cas d'utilisation interviennent tout au long du cycle de vie d'un projet**



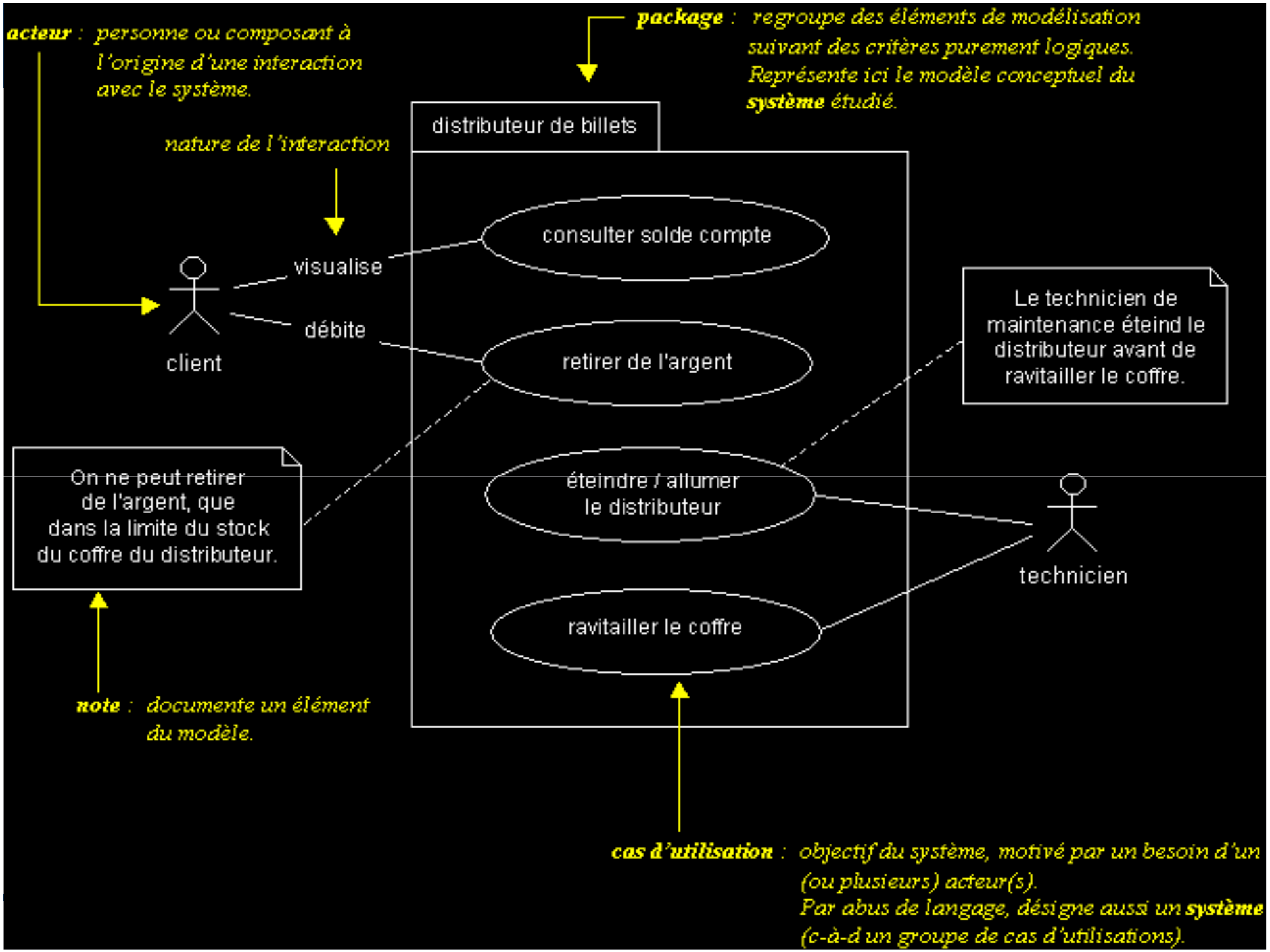

acteur : *personne ou composant à l'origine d'une interaction avec le système.*
package : *regroupe des éléments de modélisation suivant des critères purement logiques. Représente ici le modèle conceptuel du* ***système*** *étudié.*
distributeur de billets
nature de l'interaction
visualise
débite
client
consulter solde compte
retirer de l'argent
éteindre / allumer le distributeur
ravitailler le coffre
Le technicien de maintenance éteind le distributeur avant de ravitailler le coffre.
On ne peut retirer de l'argent, que dans la limite du stock du coffre du distributeur.
technicien
note : *documente un élément du modèle.*
cas d'utilisation : *objectif du système, motivé par un besoin d'un (ou plusieurs) acteur(s).*
Par abus de langage, désigne aussi un ***système*** *(c-à-d un groupe de cas d'utilisations).*

Diagramme de cas d'utilisation (8)

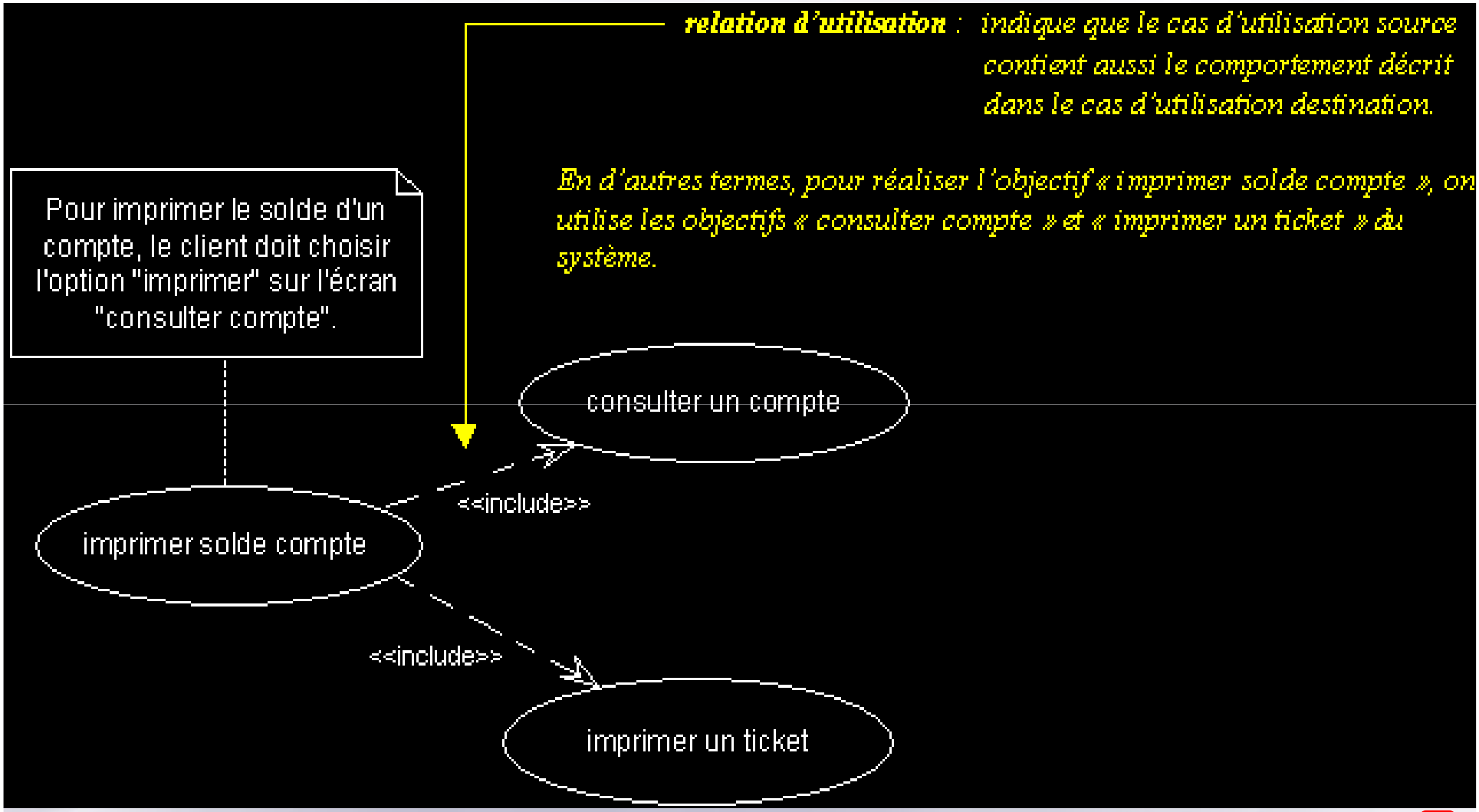
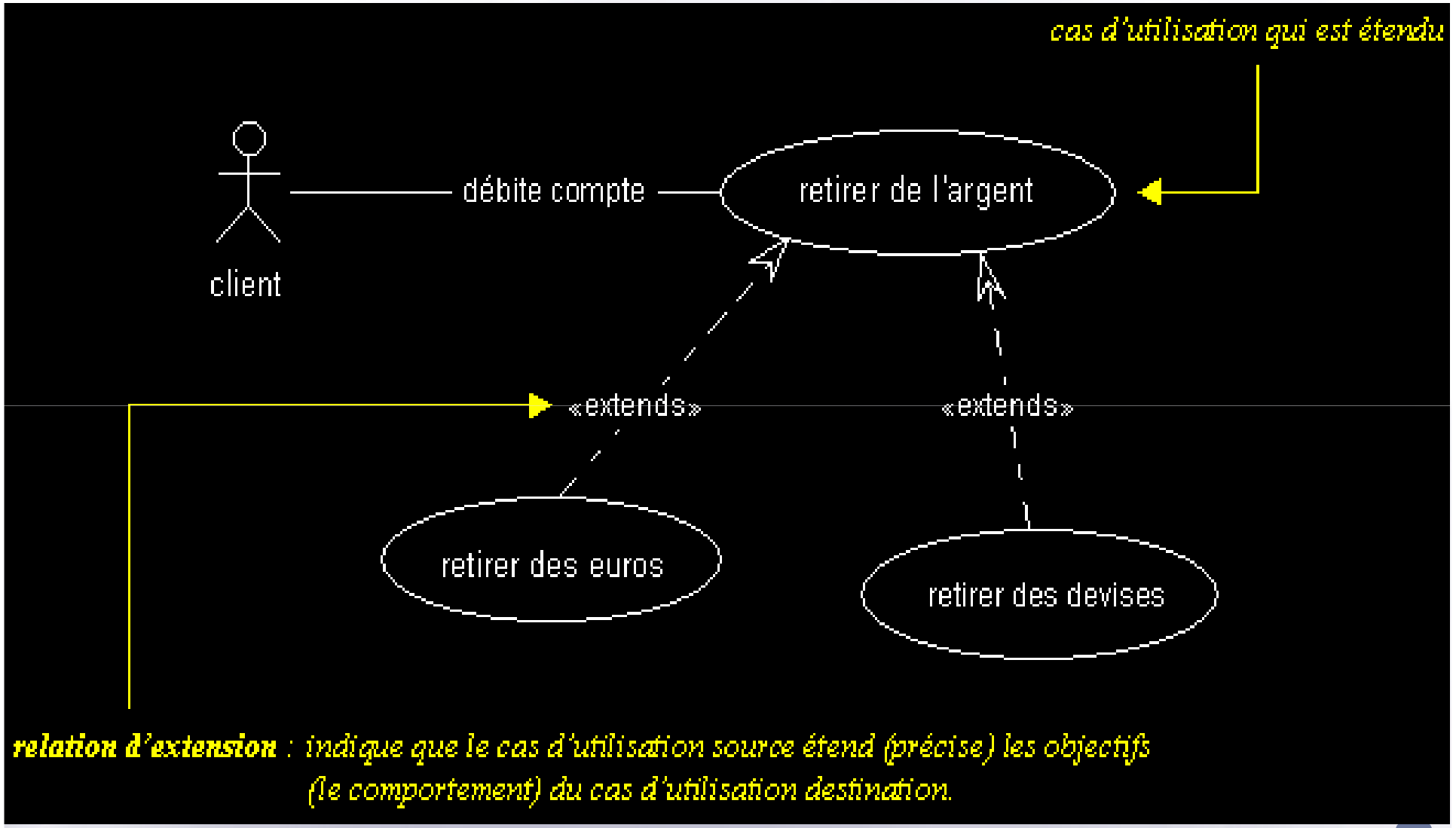


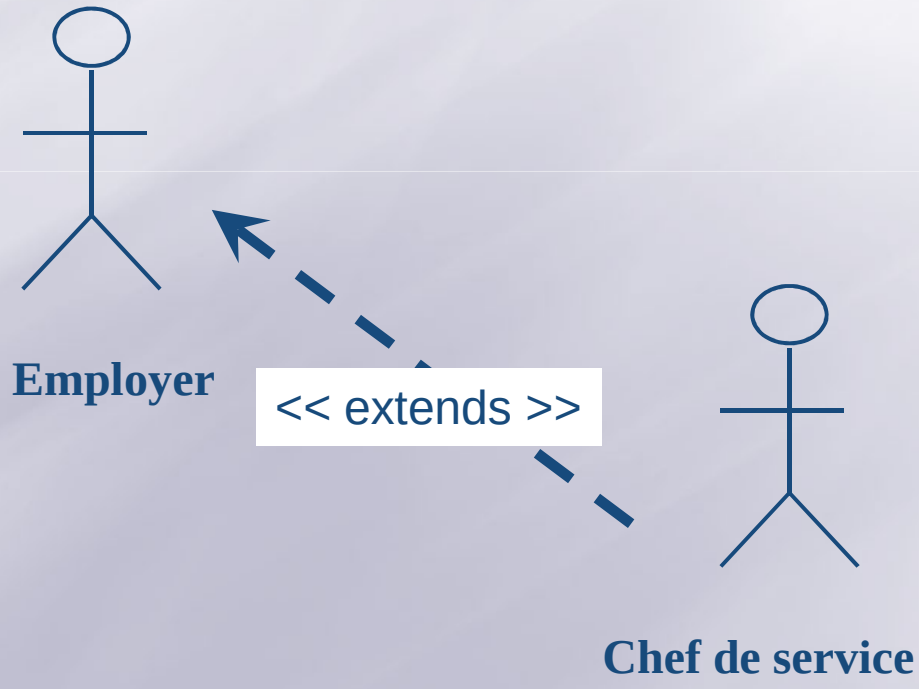
Diagramme de cas d'utilisation (9)



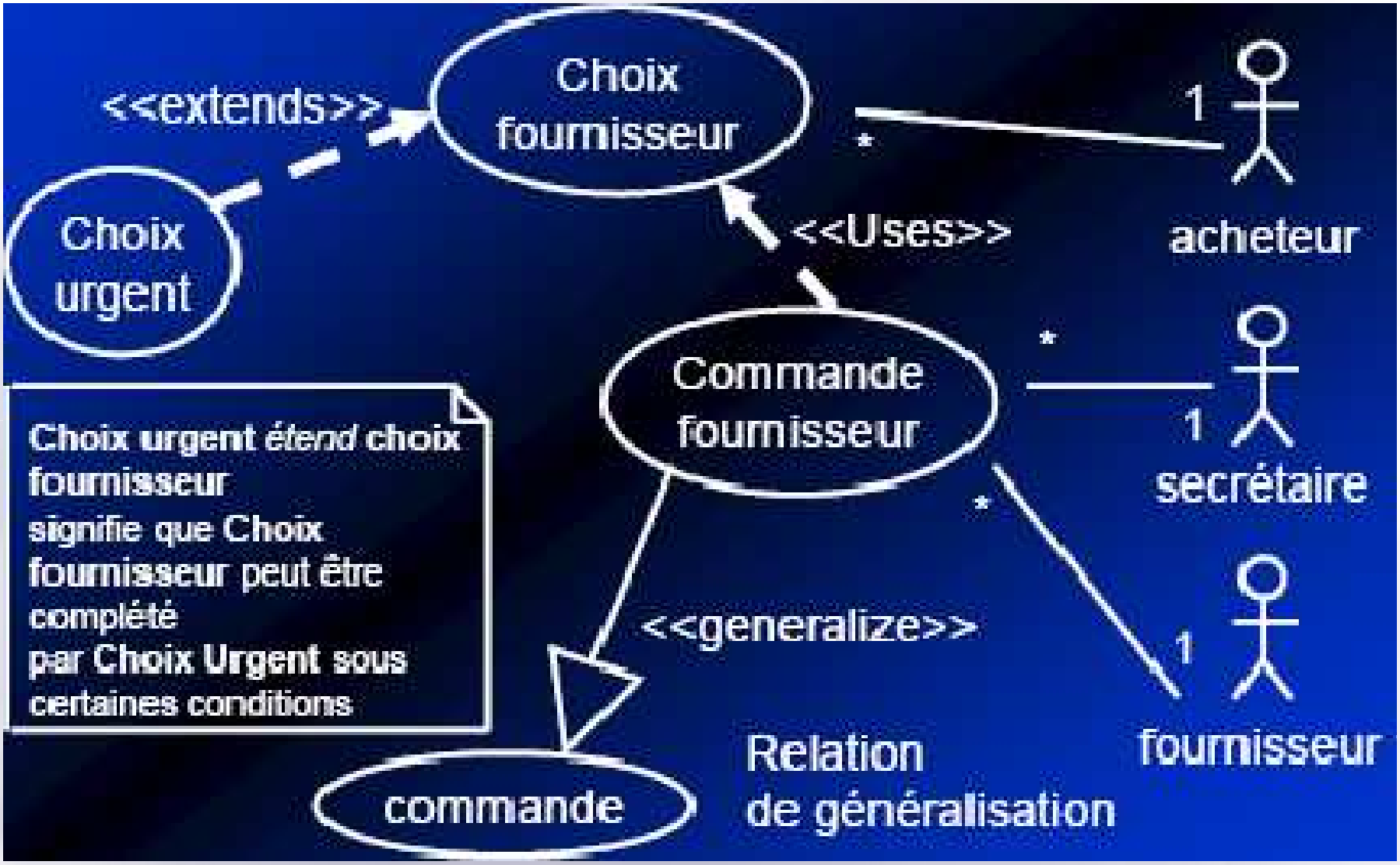
relation d'extension : *indique que le cas d'utilisation source étend (précise) les objectifs (le comportement) du cas d'utilisation destination.*

Diagramme de cas d'utilisation (10)

Extension des acteurs



Exemple d'un diagramme de cas d'utilisation



Synthèse sur les diagrammes de cas d'utilisation

- Première étape à faire (capture des besoins)
- Cas d'utilisation: chaque comportement du système attendu par l'utilisateur
  - Définir le périmètre du système : Paquetage
  - Inventorier les acteurs (ceux qui utiliseront le futur logiciel)
  - Évaluer les besoins de chaque acteur en CU
  - Regrouper ces CU dans un diagramme présentant une vue synthétique du paquetage
  - Possibilité de documentation des CU complexes (cahier de charge)
  - Na pas descendre trop bas et réinventer l'analyse fonctionnelle

A vous de jouer

- **Élaborez le diagramme des cas d'utilisation d'une agence de voyage. Pour organiser un voyage, l'agent doit réserver au client une chambre d'hôtel, lui réserver un billet d'avion ou de train, lui réserver un taxi pour venir de l'aéroport ou de la gare et enfin, lui établir une facture. Certains clients peuvent demander une facture plus détaillée**

Diagramme de classes

Diagramme de classes (1)

Les classes (1)

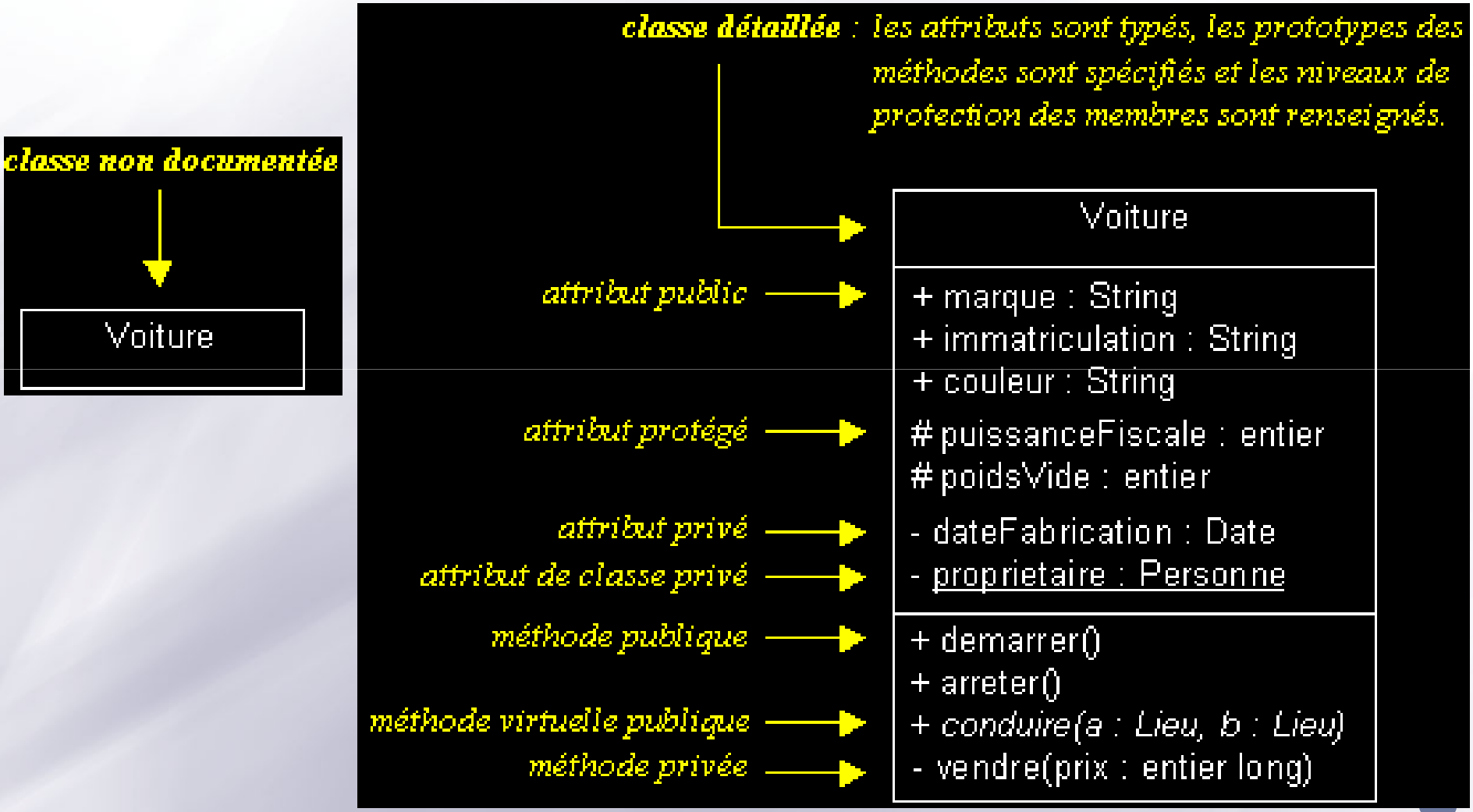


Diagramme de classes (2)

Les classes (2)

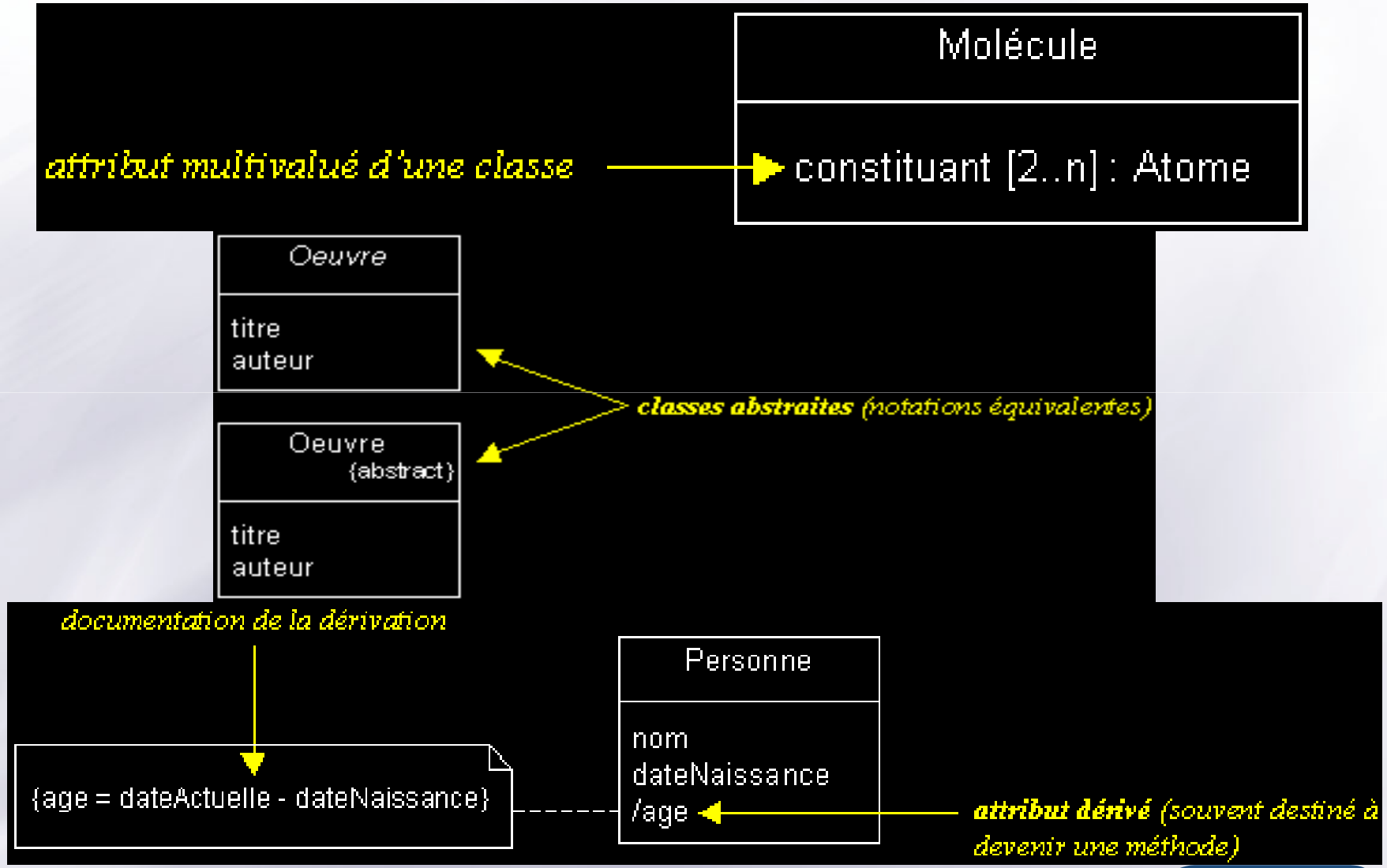


Diagramme de classes (3)

Associations entre les classes (généralités)

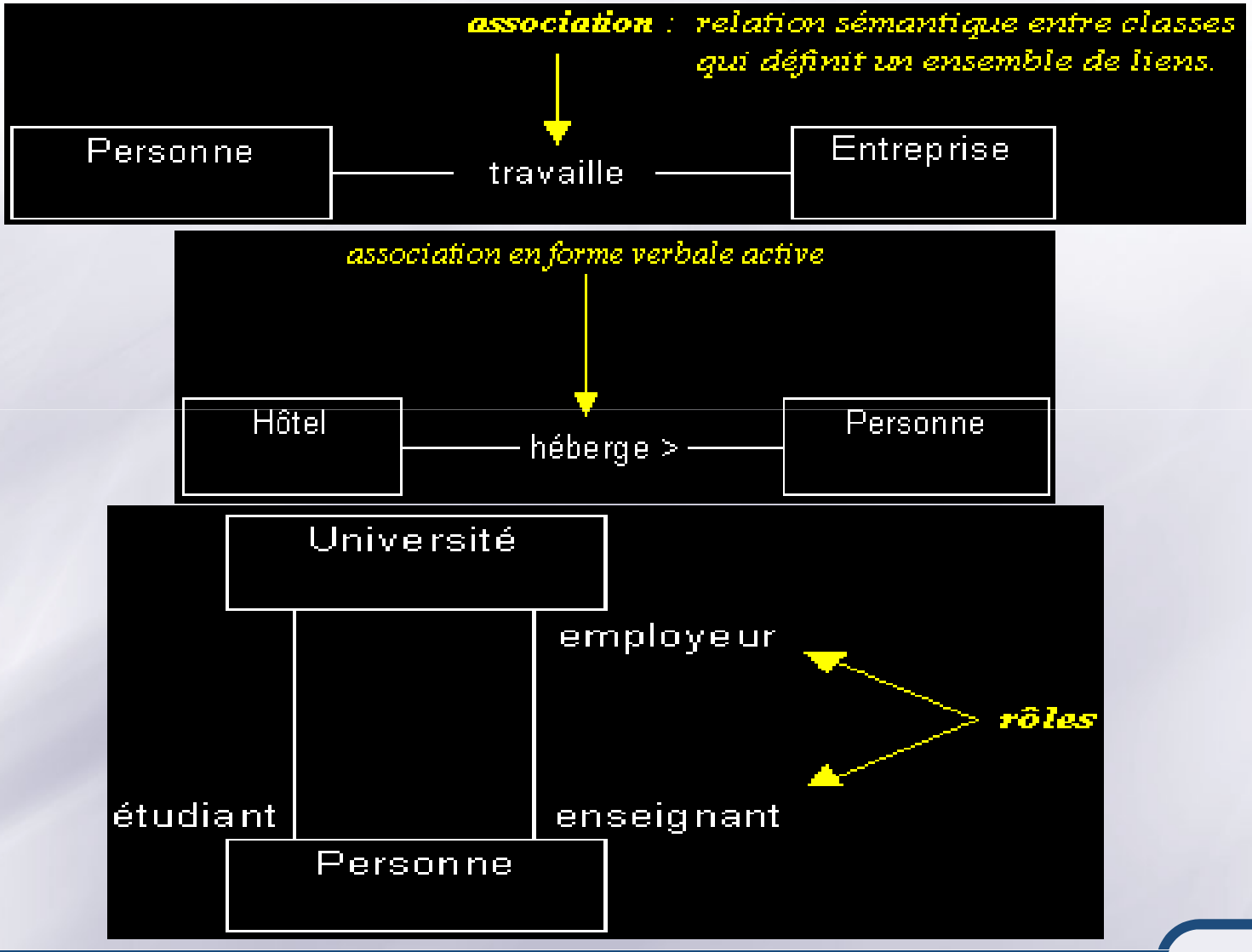


Diagramme de classes (4)

Associations entre les classes (cardinalités)

- n : exactement n instances de la classe (n entier >= 0)
- 0..1 : zéro ou une instance
- m..n : de m à n instances
- * : zéro à plusieurs instances
- 1..* : un à plusieurs instances (maximum non fixé)

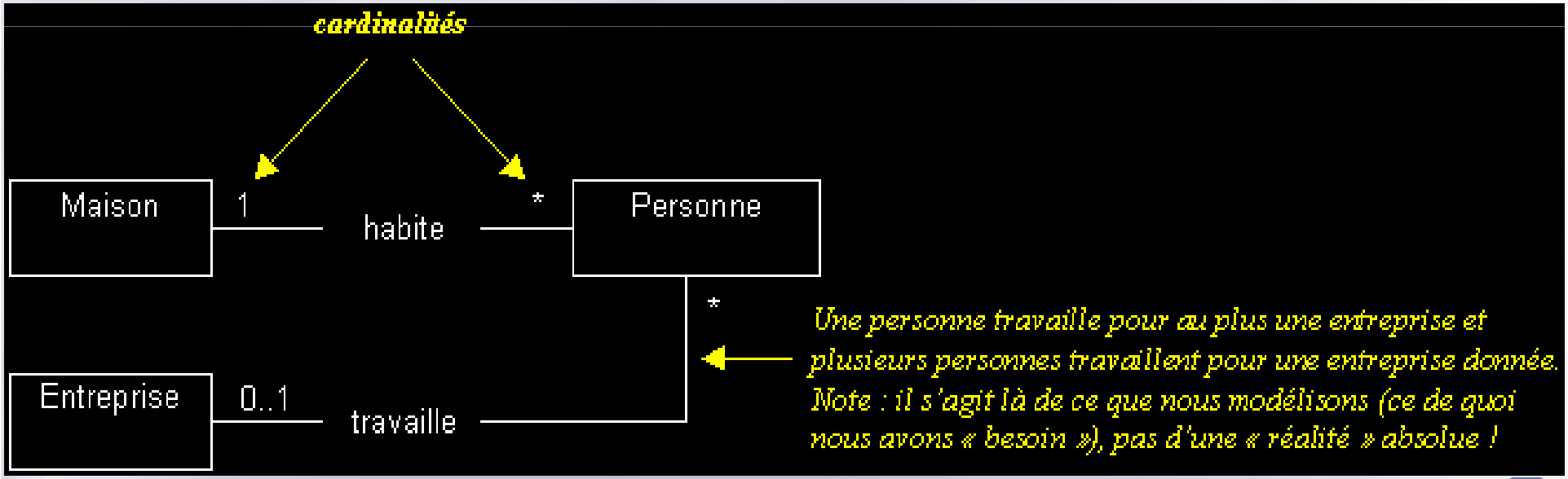
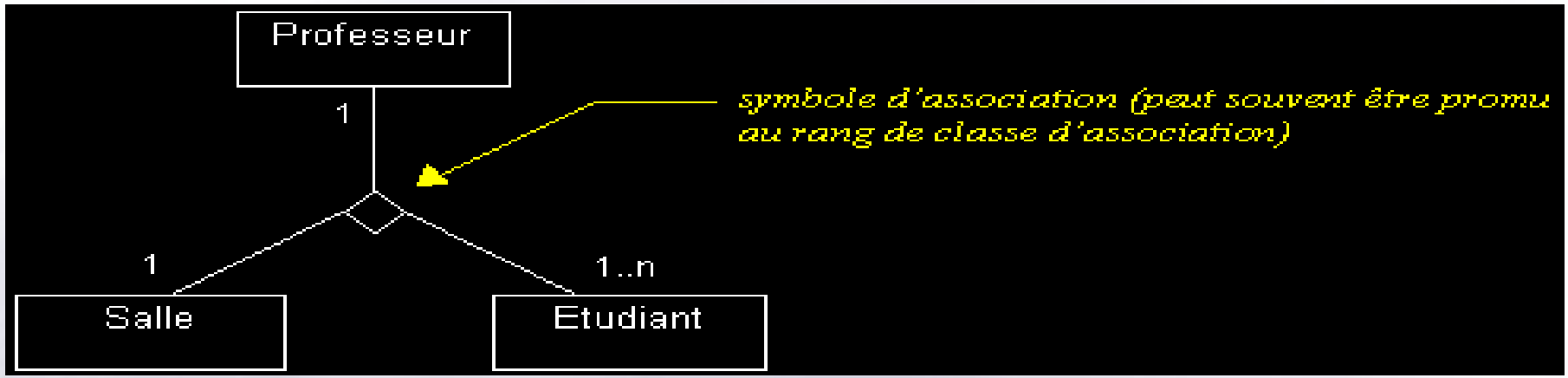


Diagramme de classes (5)

Associations entre les classes (classes d'association)



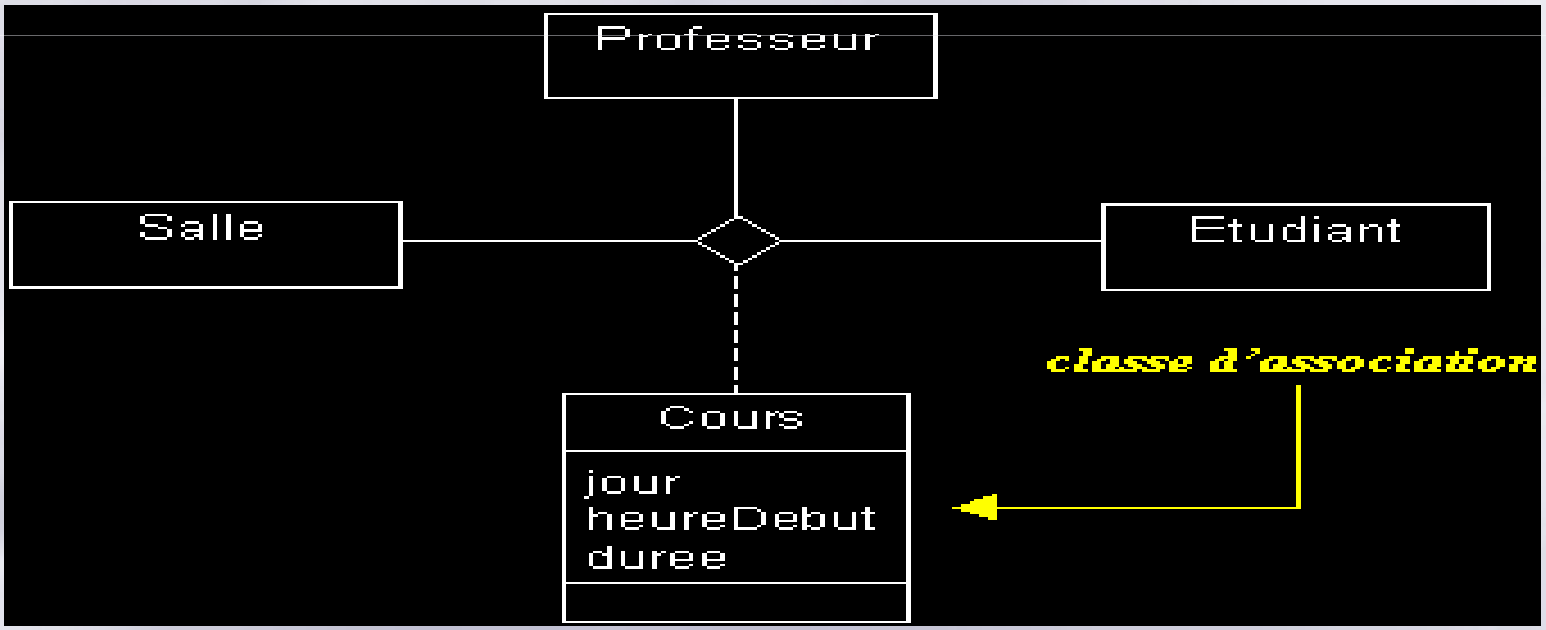


Diagramme de classes (6)

Associations entre les classes (qualifications)

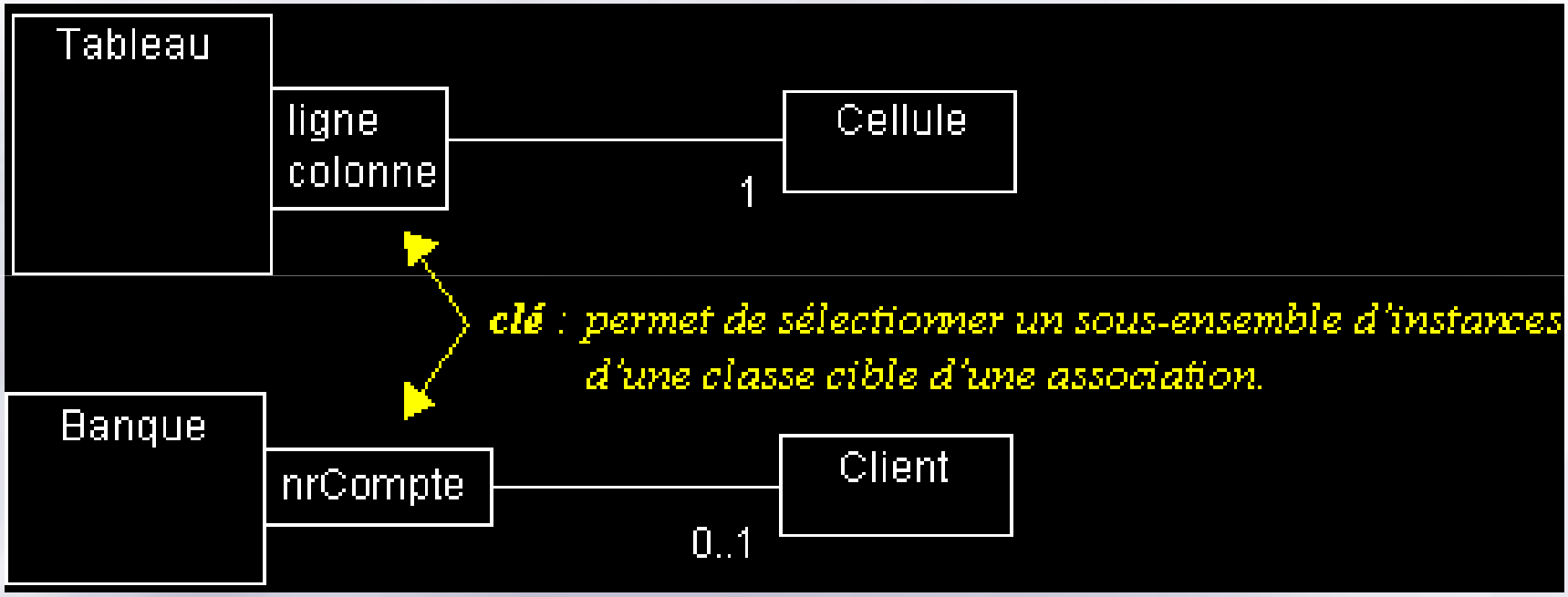


Diagramme de classes (7)

Associations entre les classes (héritage)

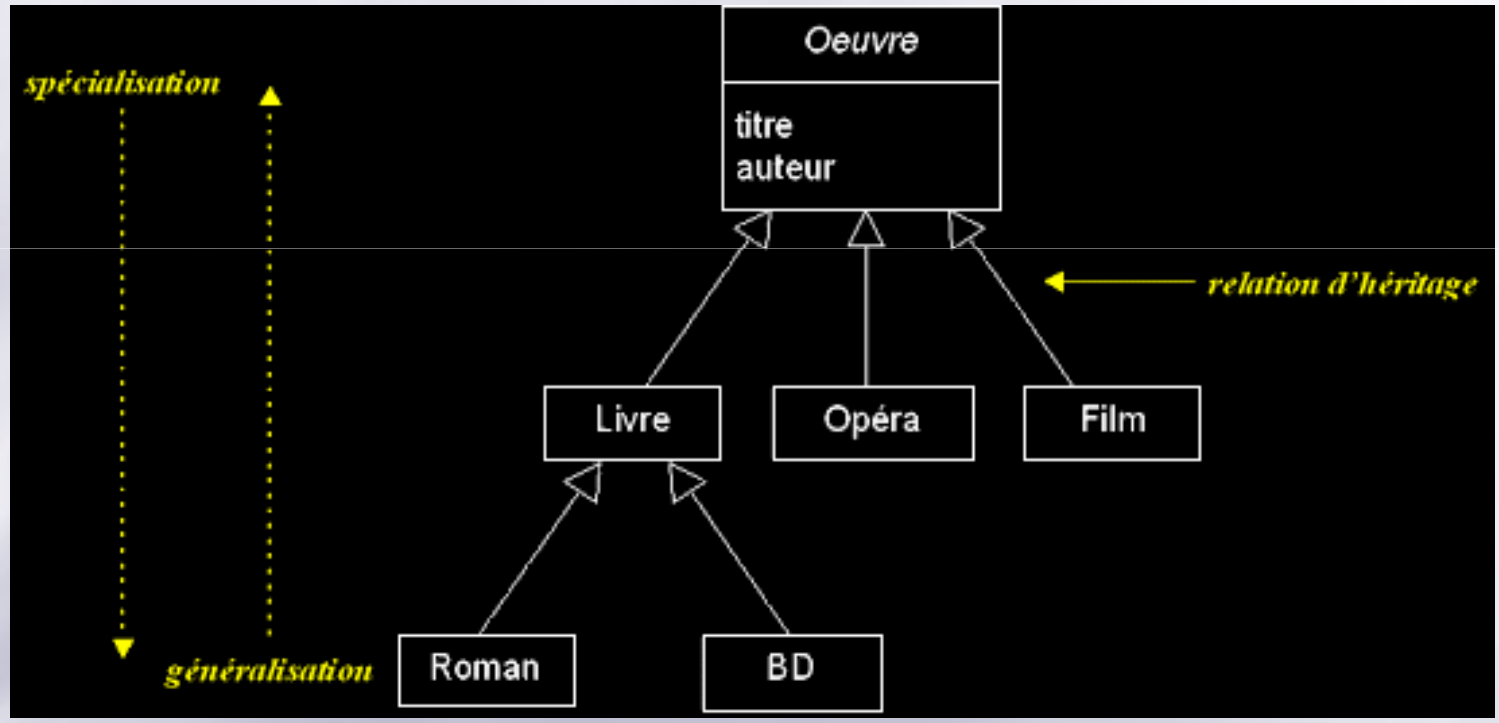


Diagramme de classes (8)

Associations entre les classes (agrégation)

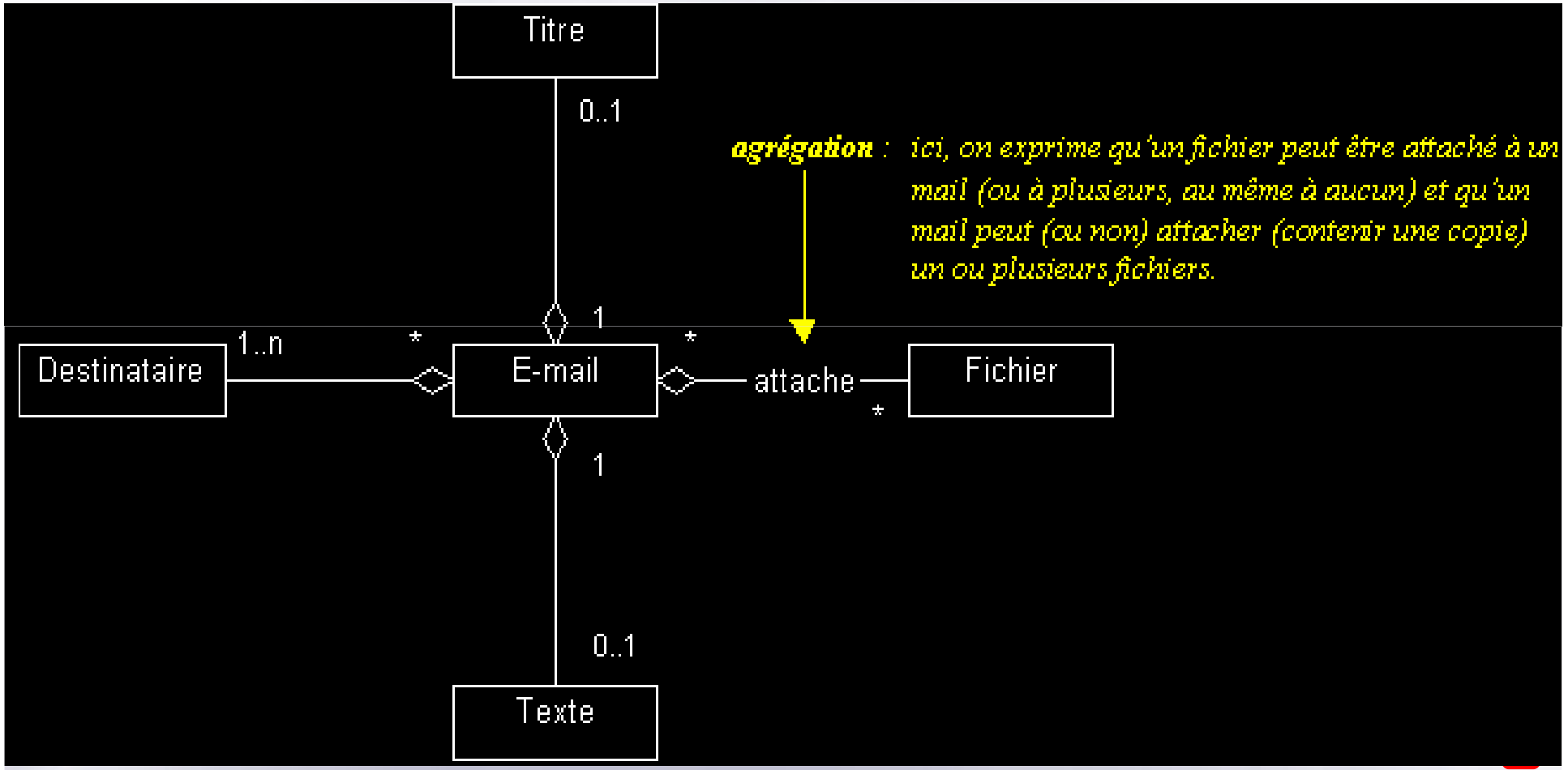


Diagramme de classes (9)

Associations entre les classes (composition)

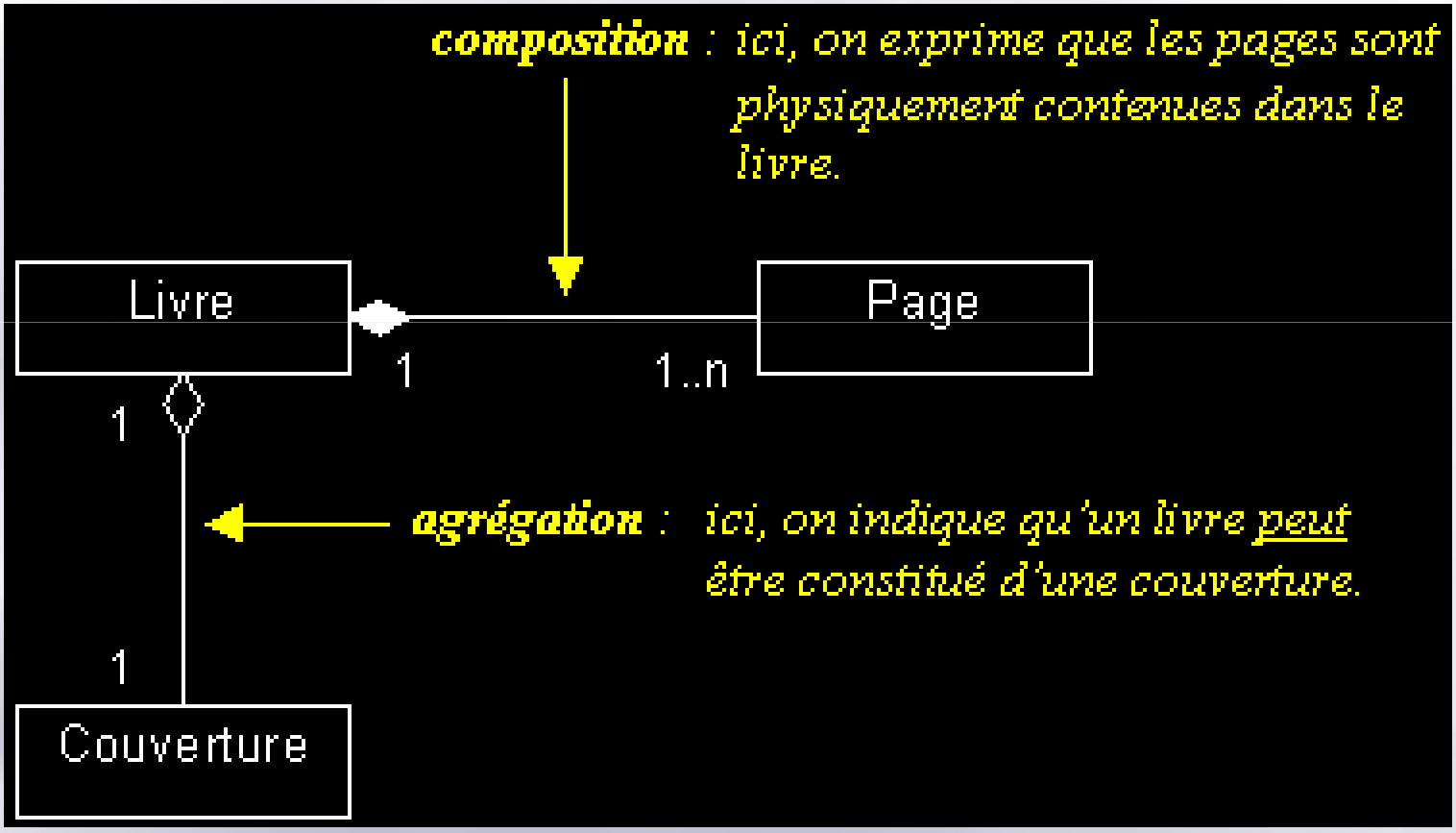


Diagramme de classes (10)

Associations entre les classes (interfaces)

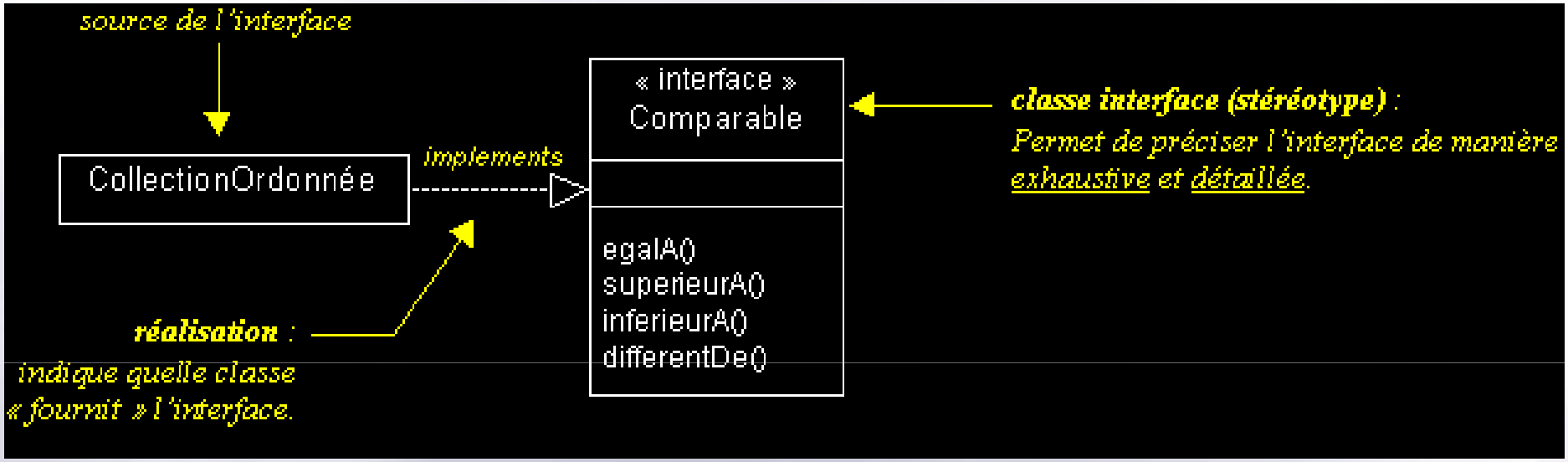


Diagramme de classes (11)

Associations entre les classes (dépendances)

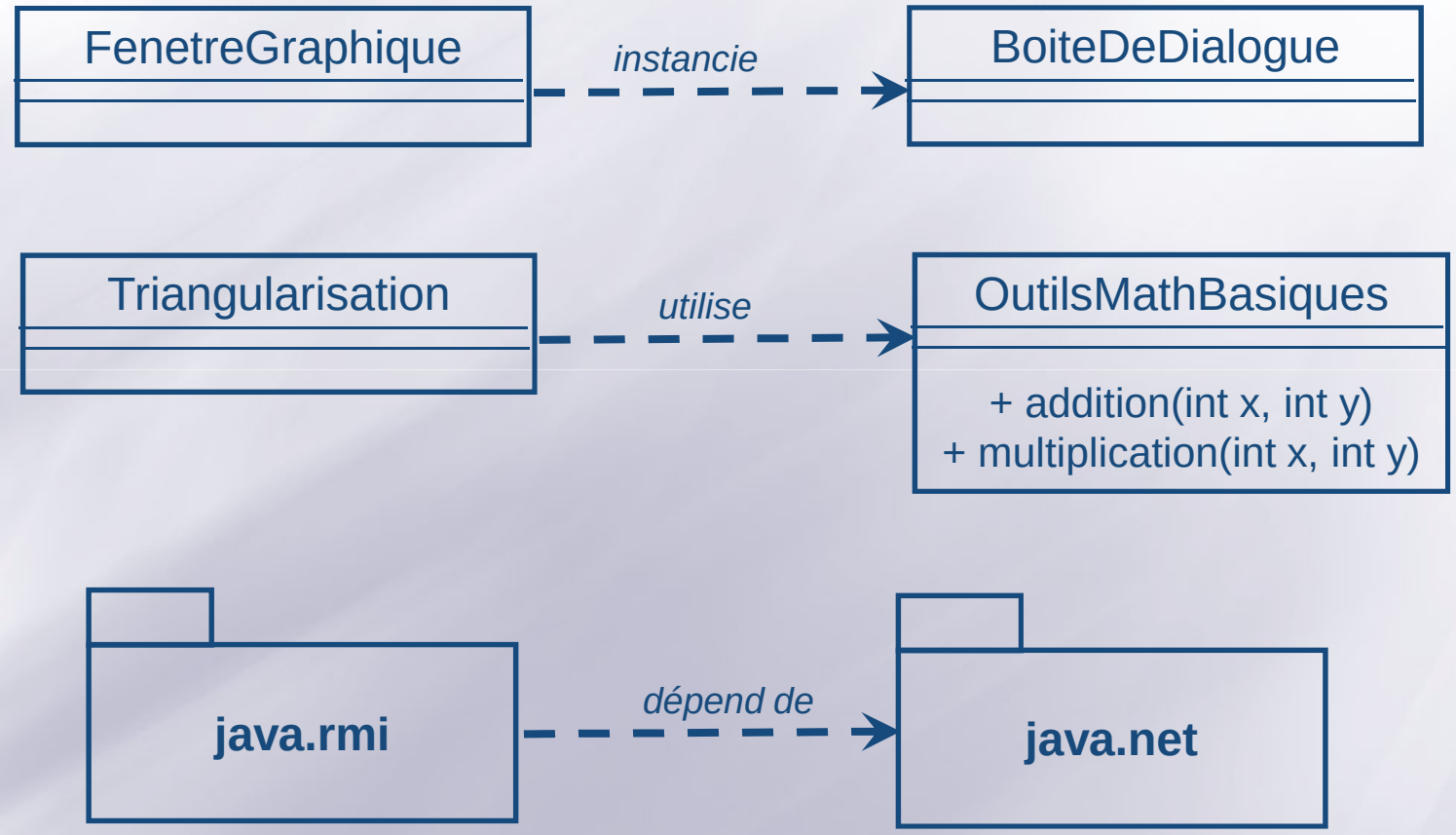
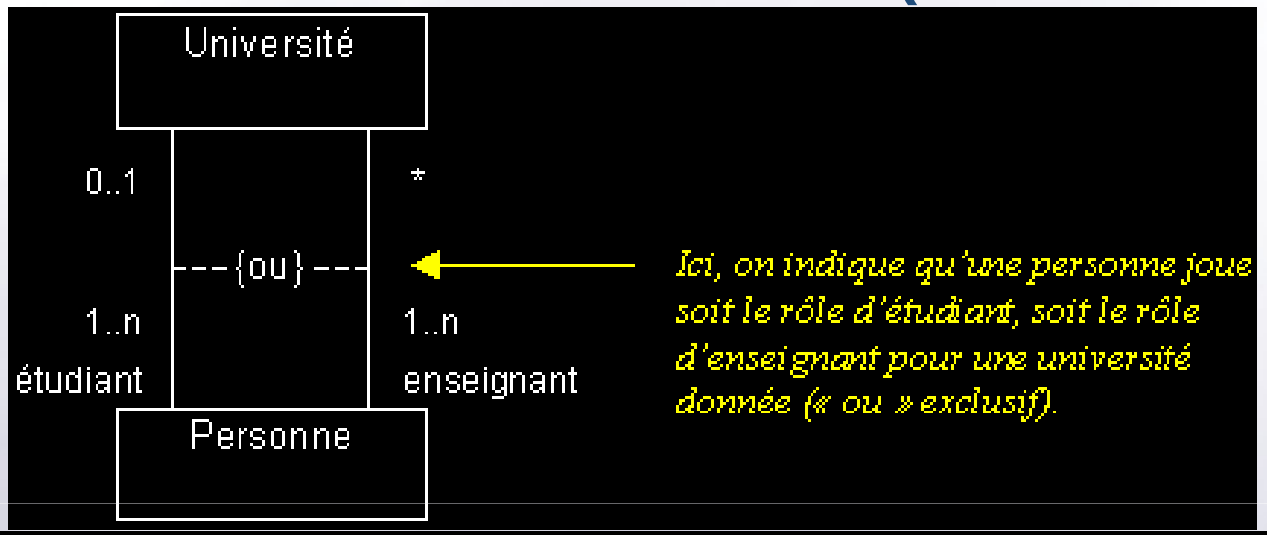


Diagramme de classes (12)

Associations entre les classes (contraintes)



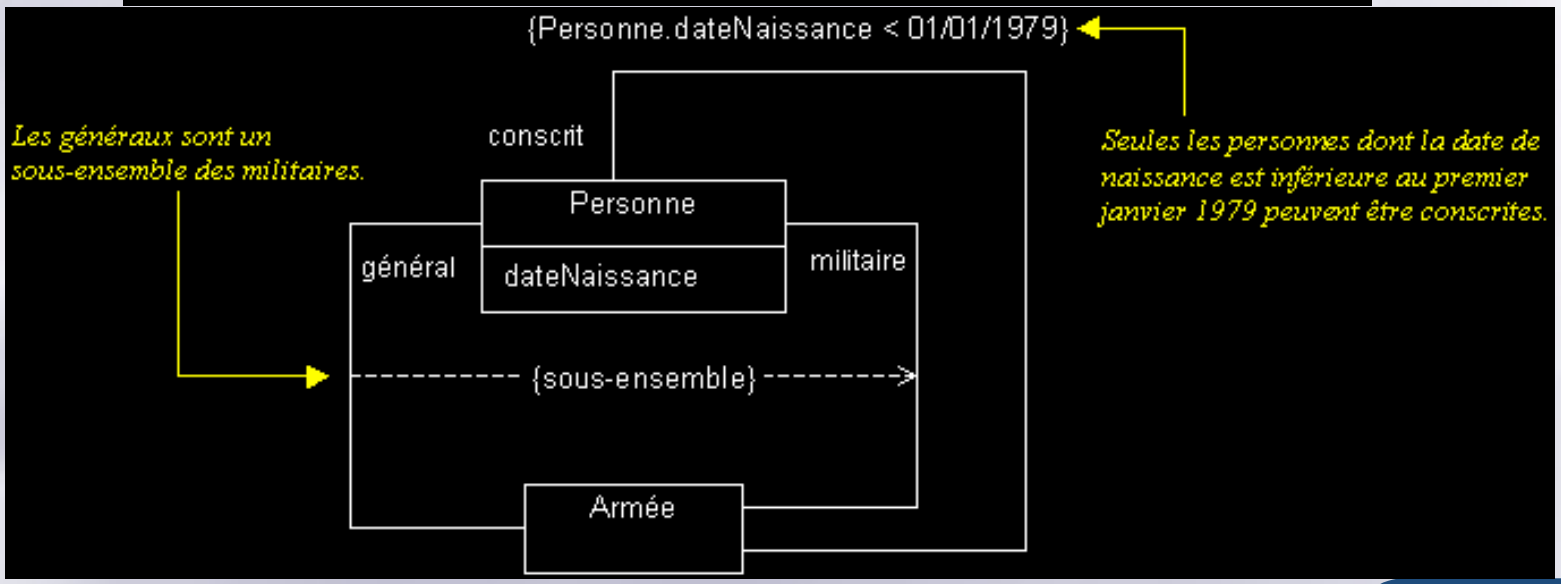


Diagramme de classes (Synthèse)

- Un diagramme de classes est une collection d'éléments de modélisation statiques (classes, paquetages...), qui montre la structure d'un modèle de données
- Un diagramme de classes fait abstraction des aspects dynamiques et temporels
- Pour un modèle complexe, plusieurs diagrammes de classes complémentaires doivent être construits
- Pour réussir un diagramme de classes:
  - identifier les entités (ou classes) pertinentes
  - identifier leurs interactions (relations et cardinalités)
  - utiliser les designs patterns (singleton, héritage…)

A vous de jouer (1)

- **Gestion de la cité U (diagramme de classes)**

Une Cité U est constituée d'un ensemble de bâtiments. Un bâtiment comporte un certain nombre de chambres. *La cité peut employer du personnel et est dirigé par l'un des employés*. Chaque chambre de la cité se loue à un prix donné (suivant ses prestations).

L'accès aux salles de bain est compris dans le prix de la location d'une chambre. Certaines chambres comportent une salle de bain, mais pas toutes. Les hôtes de chambres sans salle de bain peuvent utiliser une salle de bain sur le palier. Ces dernières peuvent être utilisées par plusieurs hôtes.

Une personne peut louer une et une seule chambre et une chambre peut être loué par une ou deux personnes.

Les pièces de la Cité U qui ne sont ni des chambres, ni des salles de bain (hall d'accueil, cuisine...) ne font pas partie de l'étude (hors sujet).

A vous de jouer (2)

- élaborer le diagramme de classes impliquant les entités suivantes:
  - Forme graphique (toute forme graphique est dessinable)
  - Forme euclidienne (on peut calculer son aire)
  - Ellipse, trapèze, cercle, rectangle

Diagramme de séquence

Diagramme de séquence (1)

- Les diagrammes de séquences permettent de représenter des collaborations entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages
- Contrairement au diagramme de collaboration, on n'y décrit pas le contexte ou l'état des objets, la représentation se concentre sur l'expression des interactions
- Les diagrammes de séquences peuvent servir à illustrer un cas d'utilisation
- L'ordre d'envoi d'un message est déterminé par sa position sur l'axe vertical du diagramme ; le temps s'écoule "de haut en bas" de cet axe
- La disposition des objets sur l'axe du temps n'a pas de conséquences pour la sémantique du diagramme
- Les diagrammes de séquences et les diagrammes d'état-transitions sont les vues dynamiques les plus importantes d'UML

Diagramme de séquence (2)

- Présente les interactions chronologiques entre les objets

- Focalise sur les
  - objets (les classes)
  - les acteurs (en partie)
  - échange de messages
  - scénarii d'exécution
  - ligne de vie des objets

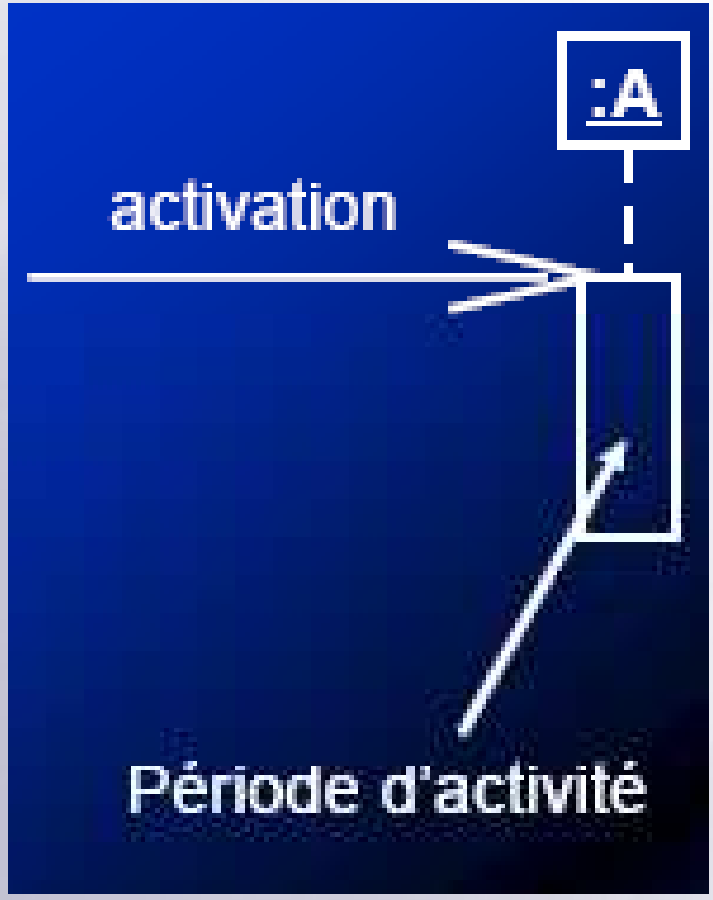
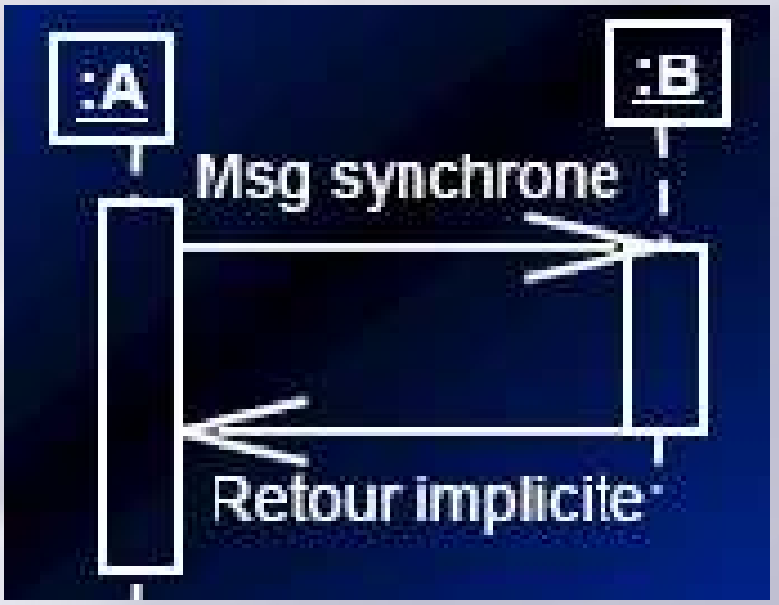


Diagramme de séquence (3)

Messages synchrones et asynchrones

- messages synchrones
  - l'émetteur est bloqué jusqu'au traitement effectif du message

- Messages asynchrones
  - l'émetteur n'est pas bloqué et il peut poursuivre son exécution



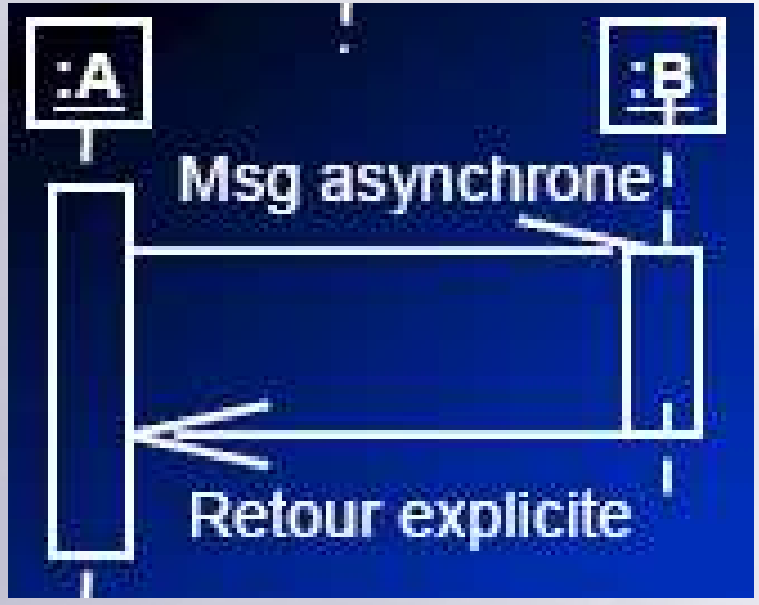
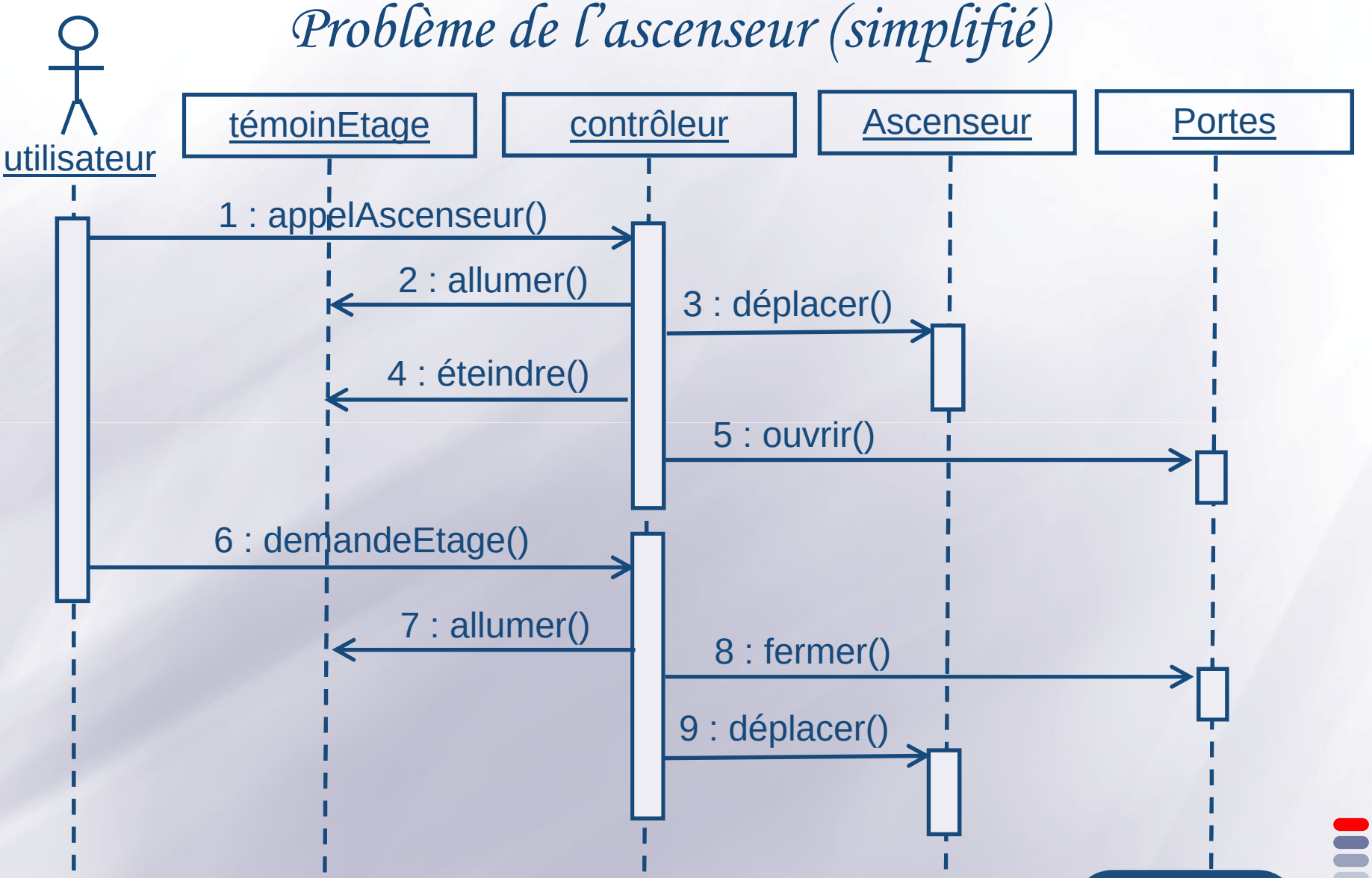


Diagramme de séquence (4)

Problème de l'ascenseur (simplifié)



A vous de jouer

- Une entreprise désire développer un logiciel de gestion de formation de ses employés. Un employé saisie sa demande de formation dans un formulaire après avoir consulté le catalogue des formations proposées par l'entreprise. Le module de gestion des formations, ajoute cette demande dans la base de données et envoie une notification à la direction par email (date de saisie, code de l'employé et le code de la formation). La direction consulte les détails de la formation à travers le même catalogue et répond à l'email du module de gestion par un refus ou un accord. Cet email est transmis par ce module à l'employé. En cas d'accord, le module envoie les formulaires d'inscription nécessaires avec le planning des sessions de formation à l'employé.

Diagramme d’états - transitions

Diagramme d’état – transition (1)

- Un diagramme d’états – transitions décrit l’évolution dans le temps d’un objet et son comportement en réponse aux interactions avec les objets qui peuplent son environnement
- Les diagrammes d'états - transitions permettent de décrire les changements d'états d'un objet, en réponse aux interactions avec d'autres objets ou avec des acteurs
- Un état se caractérise par sa durée et sa stabilité, il représente une conjonction instantanée des valeurs des attributs d'un objet
- Une transition représente le passage instantané d'un état vers un autre

Diagramme d’état – transition (2)

- Associé à chaque classe « intéressante » du diagramme de classes
- Il permet de visualiser l’évolution d’un objet au cours de sa vie sous la forme d’un automate.
- Il est une abstraction des comportements possibles à l’image des diagrammes de classes qui sont les abstractions de la structure statique

Diagramme d'état – transition (3)

- Chaque état possède un nom qui l'identifie
- Les états se caractérisent par la notion de durée et de stabilité. Un Objet est toujours dans un état donné pour un certains temps et un objet ne peut pas être dans un état inconnu ou non défini
- Un état est toujours l'image de la conjonction instantané des valeurs contenues par les attributs de l'objet, et de la présence ou non de ses liens avec d'autres objets

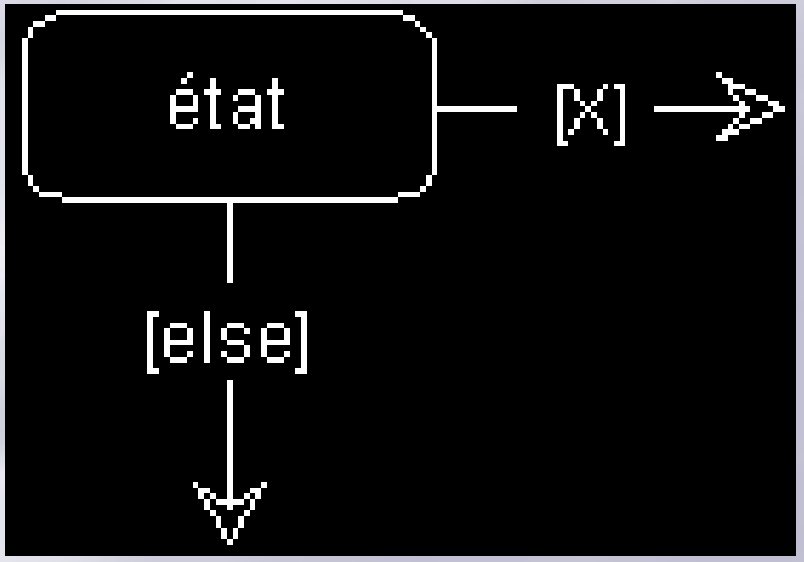
Diagramme d’état – transition (4)

- Une transition est déclenchée par un événement. En d'autres termes : c'est l'arrivée d'un événement qui conditionne la transition
- Les transitions peuvent aussi être automatiques, lorsqu'on ne spécifie pas l'événement qui la déclenche
- En plus de spécifier un événement précis, il est aussi possible de conditionner une transition, à l'aide de "gardes" : il s'agit d'expressions booléennes, exprimées en langage naturel (et encadrées de crochets)

Diagramme d'état – transition (5)

- **états, transition et événement, notation :**

- **transition conditionnelle :**



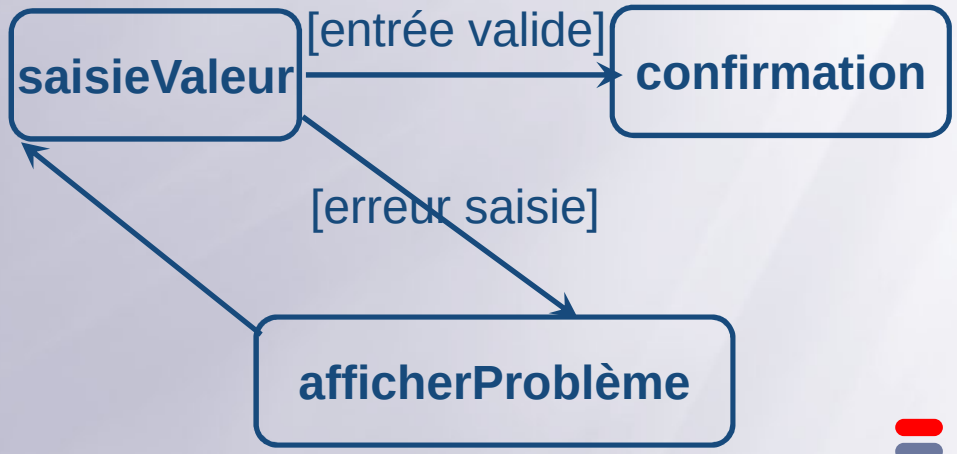
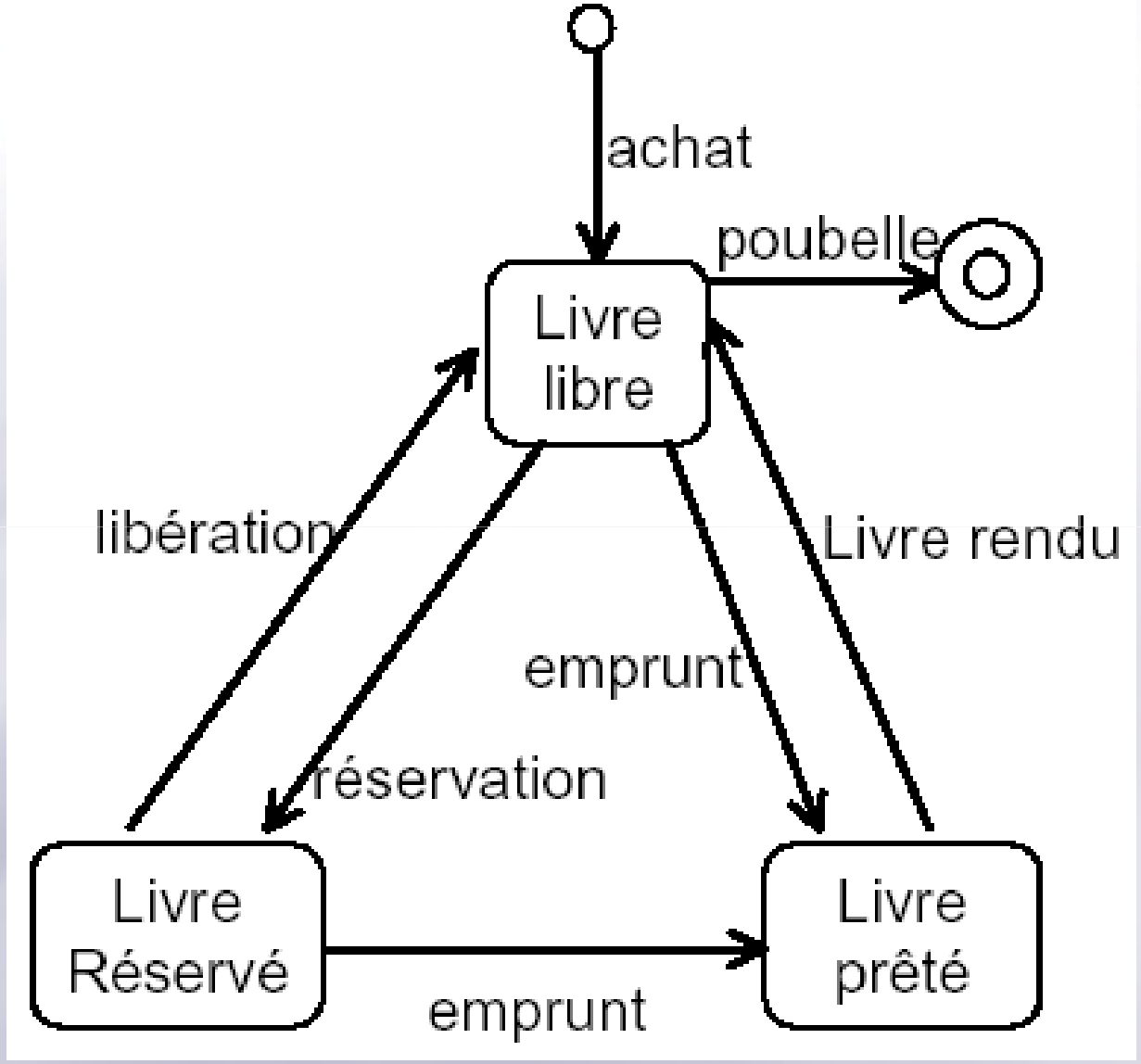


Diagramme d'état – transition (6)



A vous de jouer

- Représentez par un diagramme d'états – transitions les états que peu prendre un individu vis-à-vis la sécurité sociale: vivant, décédé, mineur, majeur, célibataire, marié, veuf et divorcé

- Une porte munie d'une serrure offre les opérations ouvrir, fermer, verrouiller (simple tour et double tour) et déverrouiller
  - représentez le diagramme états-transitions d'une serrure
  - représentez le diagramme états-transitions d'une porte avec verrou

Diagramme d’activités

Diagramme d’activités (1)

- UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation, à l'aide de diagrammes d'activités (une variante des diagrammes d'états - transitions).
- Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles.
- Le passage d'une activité vers une autre est matérialisé par une transition.
- Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre (elles sont automatiques).

Diagramme d'activités (2)

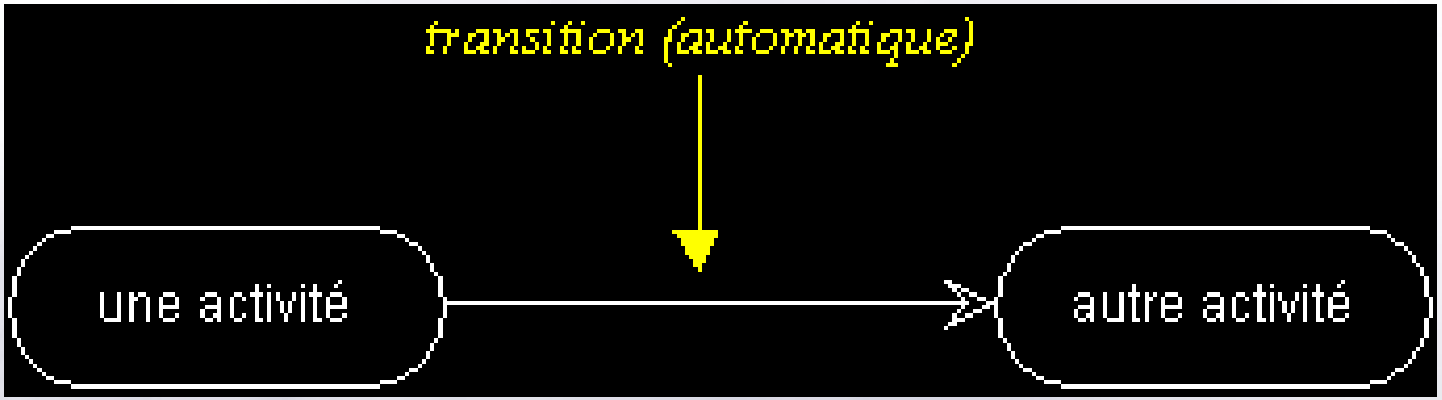
- Les diagrammes d'activités offrent un pouvoir d'expression très proche des langages de programmation objet
  - déclarations de variables, affectation…
  - structures de contrôles (conditionnelles, boucles…)
  - appel d'opérations, exceptions…
- Il peuvent aussi être utilisés pour des descriptions détaillées de cas d'utilisation
- En théorie, tous les mécanismes dynamiques pourraient être décrits par un diagramme d'activités, mais seuls les mécanismes complexes ou intéressants méritent d'être représentés.

Diagramme d’activités (3)

- **Les diagrammes d’activités permettent de décrire le comportement de:**
  - **une opération**
  - **une classe**
  - **cas d’utilisation**

Diagramme d'activités (4)

Notations



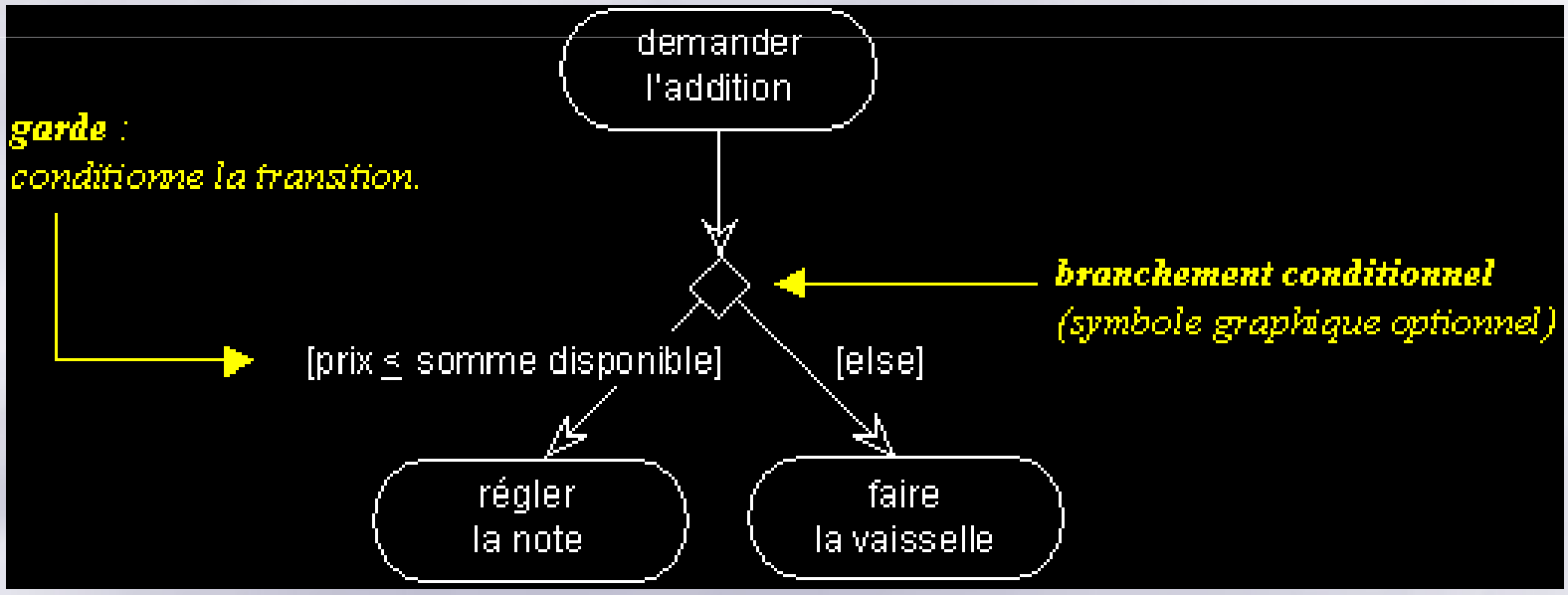
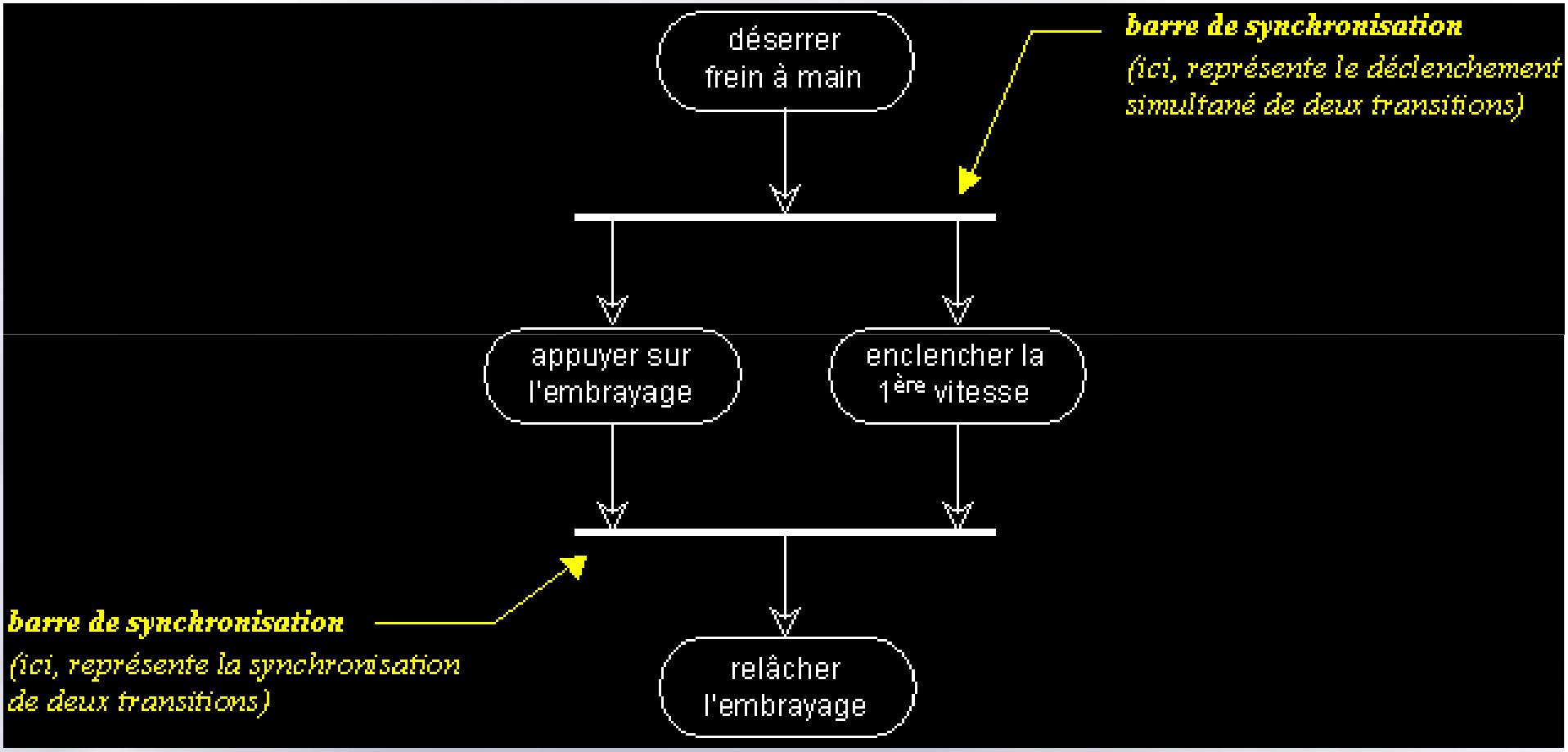
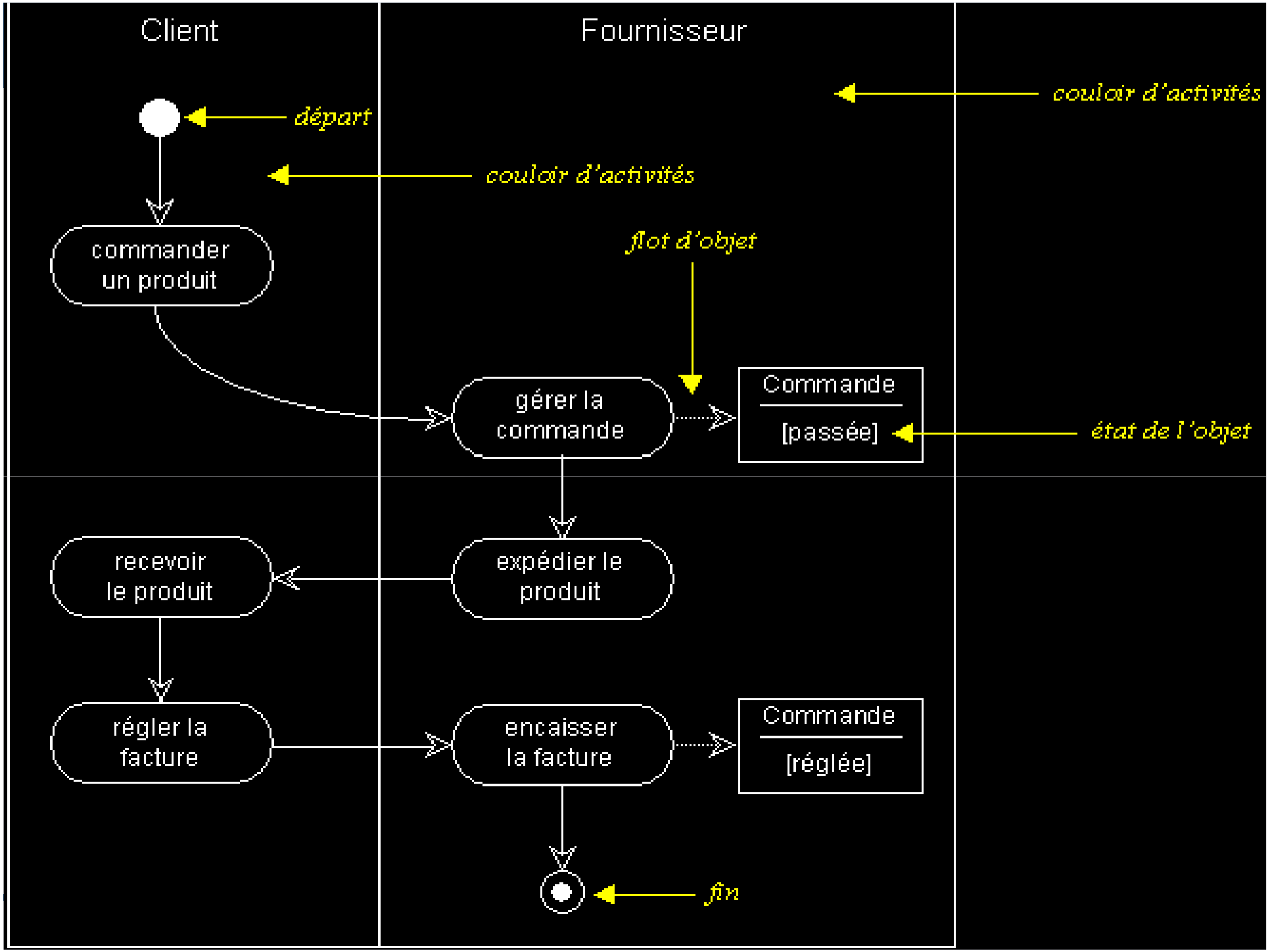


Diagramme d'activités (5)

Synchronisation




Client
Fournisseur
couloir d'activités
départ
couloir d'activités
commander un produit
flot d'objet
gérer la commande
Commande
[passée]
état de l'objet
recevoir le produit
expédier le produit
régler la facture
encaisser la facture
Commande
[réglée]
fin

A vous de jouer

- **Diagramme d'activités de validation d'achat en ligne**
  - Un site de vente en ligne propose des produits placés dans un panier virtuel. Pour valider ses achats, l'utilisateur clique sur le bouton valider. On lui propose alors de se connecter à un compte existant ou d'en créer un s'il en a pas encore.
  - Pour créer un nouveau compte, l'utilisateur doit fournir une adresse de messagerie, qui sert également de login, son nom et son adresse, éventuellement une adresse de livraison et ses coordonnées bancaires. On prévoit le cas où l'adresse de messagerie est déjà associée à un compte.
  - Si la validation de ces information réussit, on propose à l'utilisateur une confirmation définitive de l'achat

Diagramme de composants

Diagramme de composants (1)

- Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules : fichiers sources, librairies, exécutables, etc
- Les dépendances entre composants permettent notamment d'identifier les contraintes de compilation et de mettre en évidence la réutilisation de composants
- Le composants peuvent être organisés en paquetages, qui définissent des sous-systèmes

Diagramme de composants (2)

- **Les diagrammes de composants sont généralement utilisés pour identifier les différents modules d'un système d'information et leurs interactions**

- **Notations**

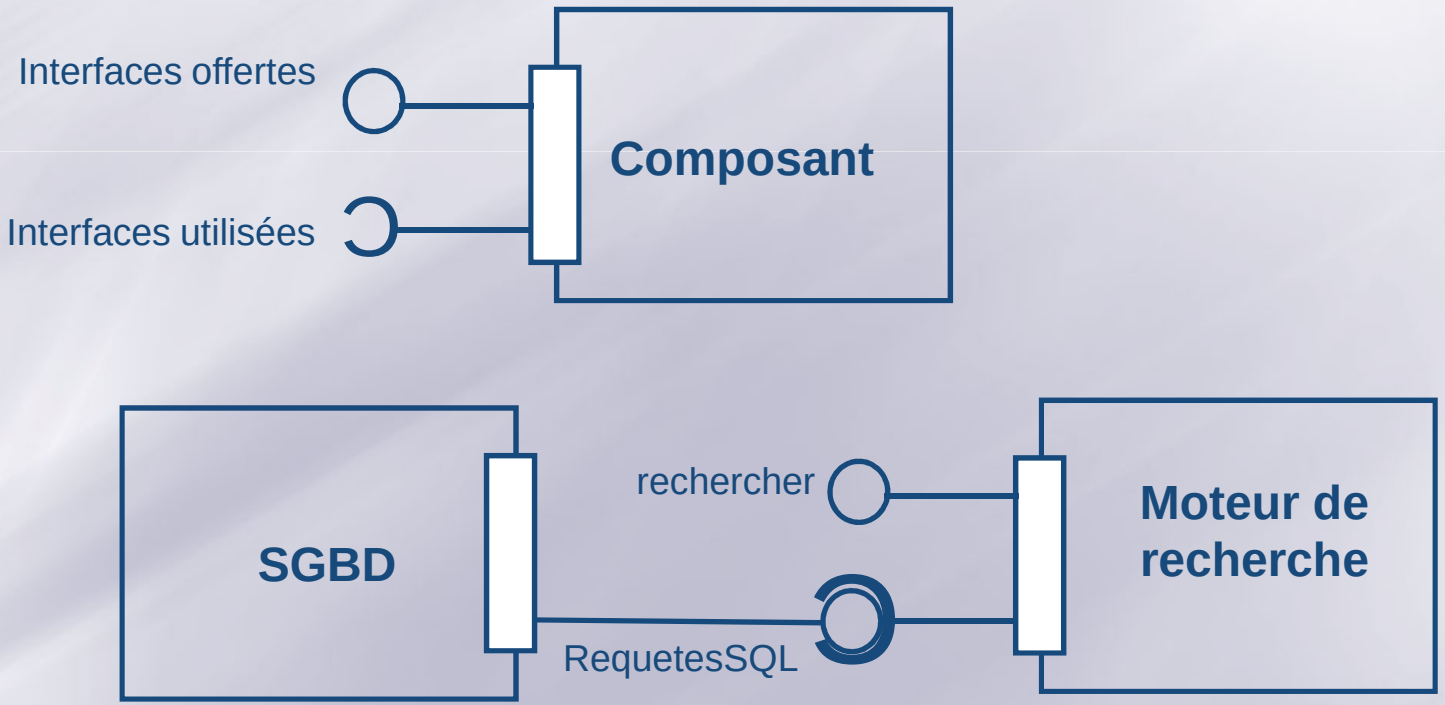


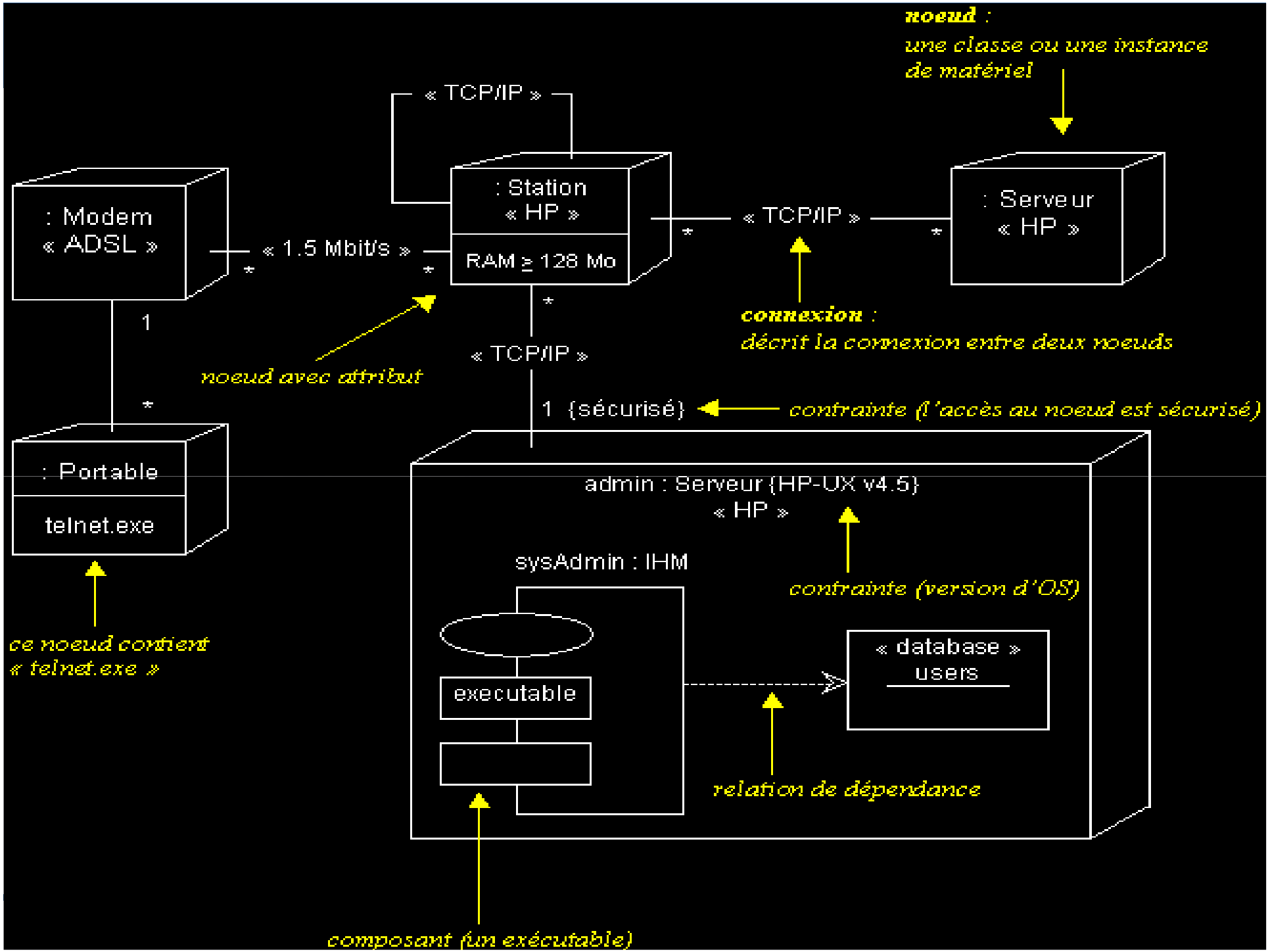
Diagramme de déploiement

Diagramme de déploiement (1)

- Les diagrammes de déploiement montrent la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels.
- Les ressources matérielles sont représentées sous forme de noeuds.
- Les noeuds sont connectés entre eux, à l'aide d'un support de communication.
- Les diagrammes de déploiement peuvent montrer des instances de noeuds (un matériel précis), ou des classes de noeuds.

Diagramme de déploiement (2)

- Les diagrammes de déploiement servent à donner une idée sur l'architecture physique (matérielle) et logique (logicielle) d'un système d'information
- Ils décrivent sur quels dispositifs matériels on va déployer les composants logiciels (les différents modules de l'application)
- Les natures des lignes de communication entre les dispositifs matériels peuvent être précisées


noeud :
une classe ou une instance de matériel
« TCP/IP »
: Station « HP »
RAM ≥ 128 Mo
: Modem « ADSL »
« 1.5 Mbit/s »
« TCP/IP »
: Serveur « HP »
connexion :
décrit la connexion entre deux noeuds
noeud avec attribut
« TCP/IP »
1
*
1 {sécurisé}
contrainte (l'accès au noeud est sécurisé)
: Portable
telnet.exe
admin : Serveur {HP-UX v4.5} « HP »
sysAdmin : IHM
contrainte (version d'OS)
« database » users
executable
ce noeud contient « telnet.exe »
relation de dépendance
composant (un exécutable)

A vous de jouer

- **Elaborez le diagramme de déploiement d'une application impliquant une machine (serv1) hébergeant un système de gestion de base de données (mysql), une machine (serv2) hébergeant un serveur HTTP (tomcat) et une application en JSP et une machine cliente disposant d'un navigateur WEB (internet explorer). Les clients peuvent se connecter directement aux bases de données sans passer par le serveur HTTP en utilisant l'application (mysql control center) via le protocole TCP/IP.**

Synthèse rapide

- **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

Une classification

- Vue statique du système
  - cas d'utilisation
  - classes
  - composants
  - déploiement
- Vue dynamique du système
  - séquence
  - états - transitions
  - activités

Des redondances? Pas vraiment…

- **Diagrammes de composants et de déploiement**
  - **ils sont vraiment complémentaires (même inséparables)**
- **Diagrammes d'activités et diagramme d'états-transitions**
  - **niveaux d'expressions différents mais un diagramme d'états – transitions est généralement suffisant**

Conclusion: les diagrammes principaux

Diagramme de cas d'utilisation

- c'est là où on assimile les fonctionnalités demandées par le client

Diagramme de classes

- le cœur de la conception d'un système

Diagramme de séquence

- indispensable pour comprendre l'interaction entre les classes

Diagramme états- transitions et diagramme d'activités

- Toute la dynamique du système est là

Mais le diagramme de déploiement reste très utile aussi !!!