

R2.01: **Développement orienté objet : Conception**

U.E.2.1 et U.E. 2.2

Janvier avril 2022

Sophie Voisin



Introduction

► Séquence pédagogique:

- 4 cours de 1H: 1 x 4 semaines
- 6 TD de 1H30: 1 x 6 semaines
- 9 TP de 1H30: 1 x 5 semaines + 2 x 2 semaines
- 1 contrôle (1H30) la semaine du 28 mars

► 4 intervenants:

- Sophie Laplace (cours TD1 TP1)
- Mickaël Berger (TD2 TP3 TP4)
- Yann Carpentier (TD3 TP5)
- Yon Dourisboure (TP2)

► Objectif du PPN: Initiation à la programmation objet



Introduction

► Savoirs de référence à étudier:

- Concepts fondamentaux de la programmation orientée objets
- Application orientée objets des algorithmes sur des structures de données (ex : collections...)
- Illustration de l'exécution d'un algorithme dans un schéma mémoire (pile et tas)
- Lecture d'une conception orientée objet détaillée (ex : diagramme de classes)
- Bases de la modélisation objet pour l'analyse et la conception détaillée (ex : diagramme de classes, diagramme des cas d'utilisation, diagramme de séquences...)



Introduction

► Analyse:

- Choix de l'approche objet
- Autres possibilités: entité-association (MERISE)

► Langage de modélisation: UML

- Unified Modeling Language (langage unifié de modélisation)
- modélisation de systèmes et de processus
- langage graphique: meilleure perception des abstractions par les hommes
- basé sur l'approche objet
- UML 2



Approche objet

Historique des techniques de programmation

► Complexification des applications:

- **Algorithmique:**

- > Programmation en langage binaire (1800)
- > Programmation en assembleur (1947)
- > Langages plus évolués Fortran Cobol (1956)
- > Programmation structurée Pascal C Ada (1970)

- **Génie Logiciel (1968):**

(optimisation du coût du développement logiciel)

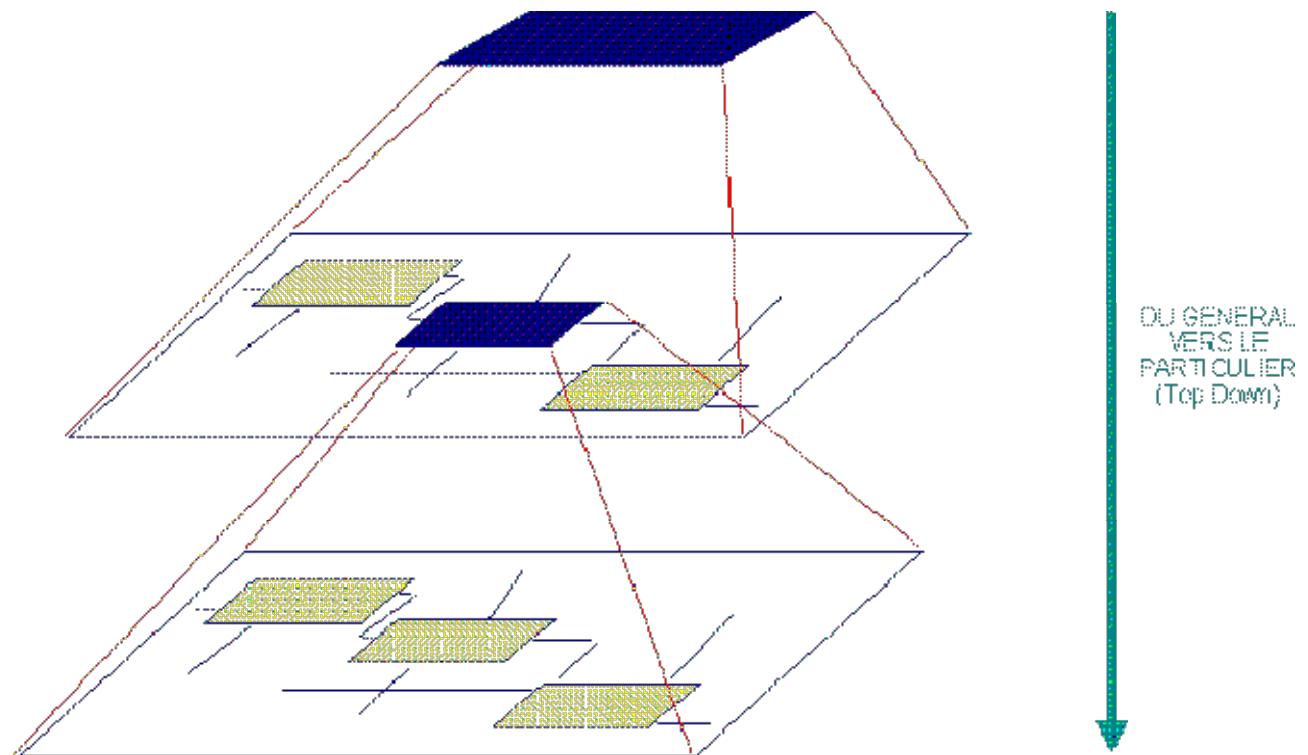
- > Méthode Merise (1978)
- > Programmation orientée objet (1967)



Approche objet

Méthode d'analyse et de conception

Exemple de méthode structurée ou procédurale: SADT

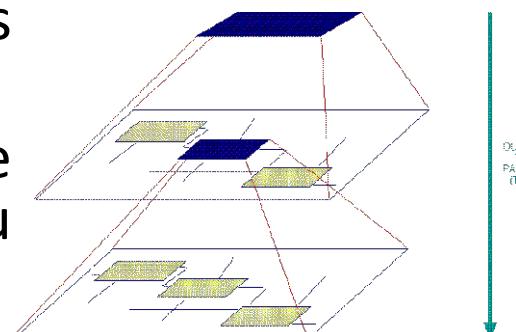


Approche objet

Méthode d'analyse et de conception

► Méthode structurée ou procédurale:

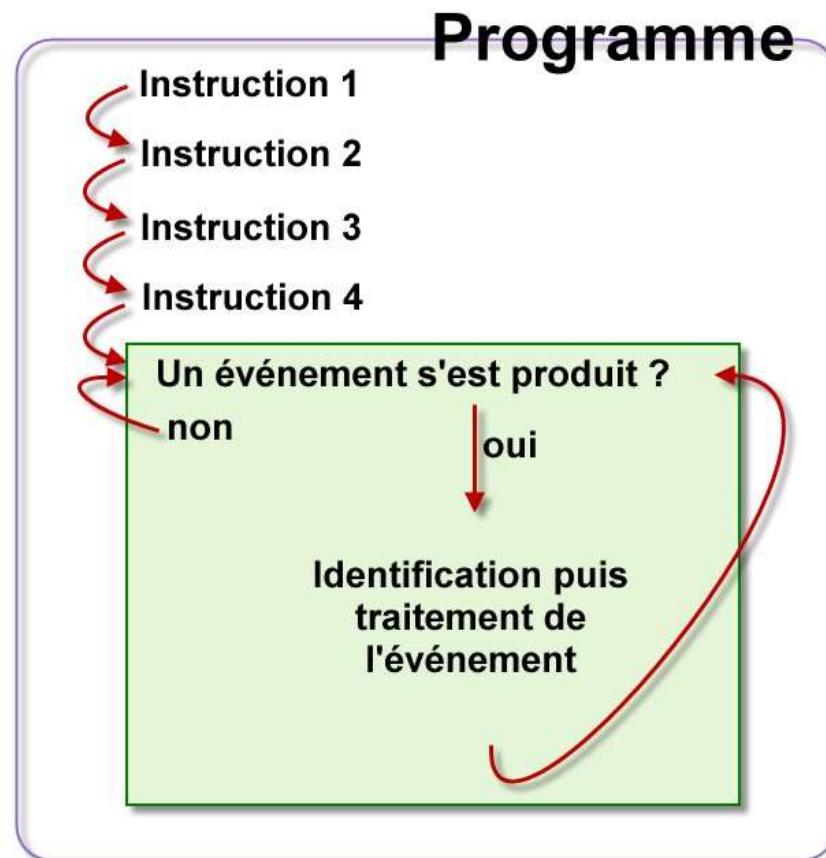
- Système = ensemble hiérarchique d'unités en interaction
- Séparation de la représentation des données et de leur traitement
- Décompositions successives du système en respectant les entrées/sorties du niveau supérieur
- Architecture du système dictée par la réponse au problème
- Traduction de l'évolution des besoins par une remise en question de la structure logicielle: **décomposition dictée par les besoins**



Approche objet

Méthode d'analyse et de conception

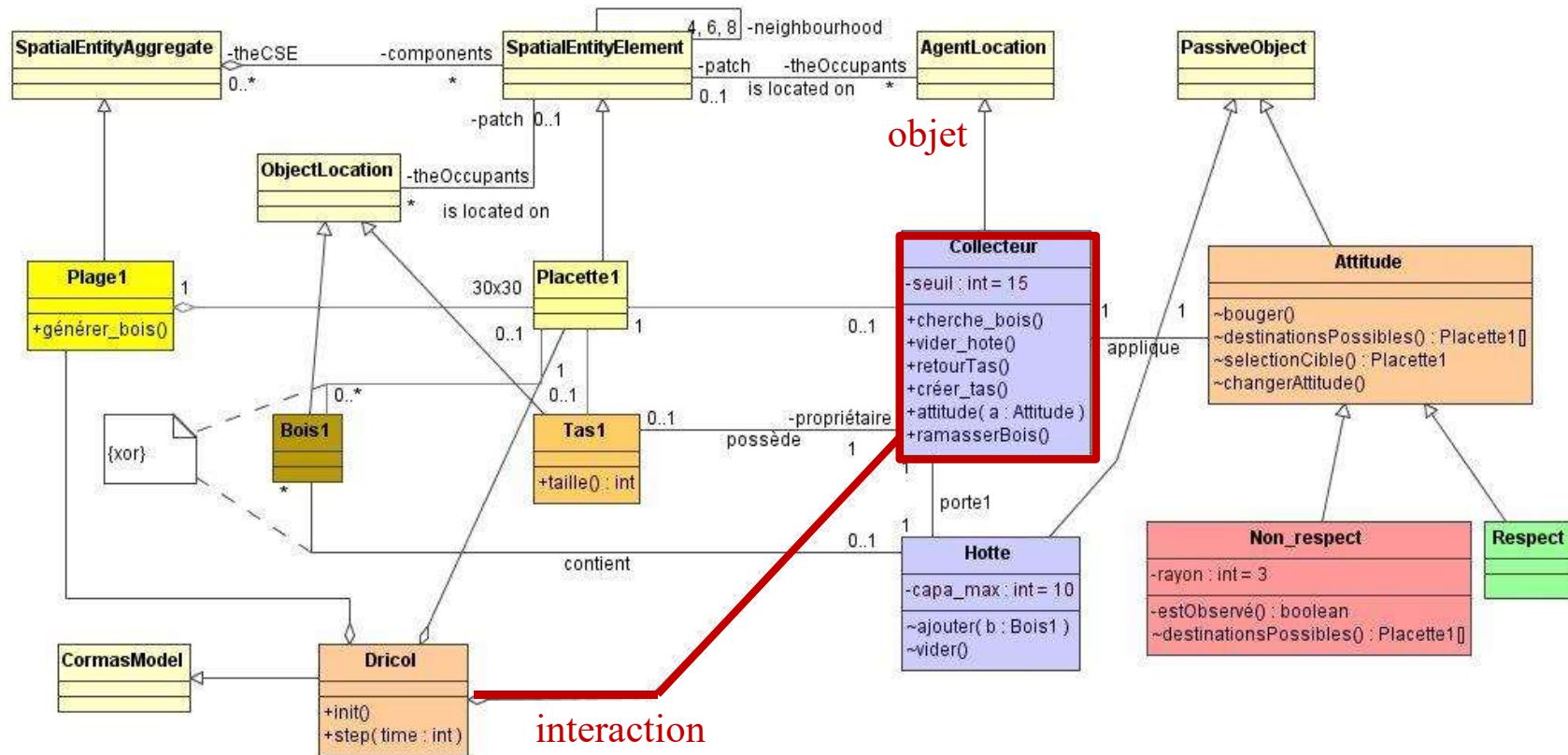
- Méthode structurée ou procédurale: programme



Approche objet

Méthode d'analyse et de conception

- Exemple de Méthode orientée objet: diagramme des classes

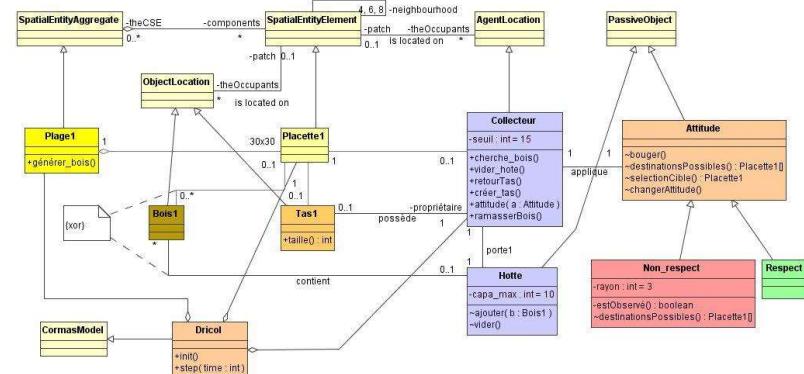


Approche objet

Méthode d'analyse et de conception

► Méthode orientée objet:

- Système = ensemble d'objets interagissant
- Réunion des données et de leur traitement au sein d'un unique objet
- Architecture logicielle fondée sur les objets du système plutôt que sur la fonction à réaliser
- **Architecture dictée par la structure du problème**
- Approche orientée donnée: fonctions déduites des données
- Traduction de l'évolution des besoins par une modification des interactions entre les objets



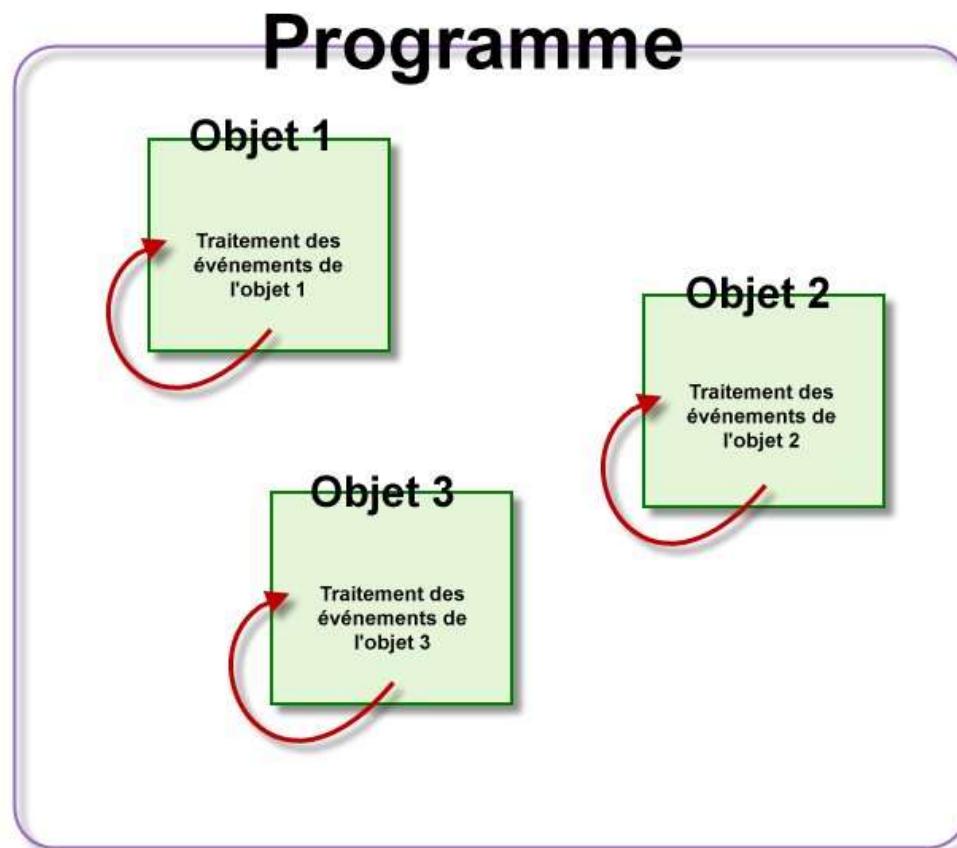
⇒ Logiciel plus adaptable,
plus évolutif



Approche objet

Méthode d'analyse et de conception

- ▶ Méthode orientée objet: programme



Approche objet

Méthode d'analyse et de conception

	Procédural	Objet
Système	Ensemble hiérarchique d'unités en interaction	Ensemble d'objets interagissant
Données et traitement	Séparation de la représentation des données et de leur traitement	Réunion des données et de leur traitement au sein d'un unique objet
Architecture	Dictée par la réponse	Dictée par la structure du problème
Evolution des besoins	Modification de la structure logicielle	Modification des interactions

⇒ Plus évolutif, plus adaptatif



Approche objet

Objet

► Définition d'un objet:

- objet identifiable du monde réel
- avec ou sans existence physique
- avec: carte de crédit
- sans: panier virtuel

► Objet = abstraction d'un objet réel

► Abstraction:

concernant un objet, entité retenant uniquement les propriétés **pertinentes** pour un problème précis



Approche objet

Objet

- Variété des choix possibles de propriétés
- ⇒ Existence de différents modèles pour un problème donné
- ⇒ Difficulté à choisir le modèle le plus efficace:
 - Simple
 - Facile à implémenter
 - Retenant toutes les caractéristiques utiles



Approche objet

Objet

► Exemple: commerce électronique

DÉTAILS DE VOTRE PANIER

< Retourner au rayon

4 ARTICLES DANS VOTRE PANIER **5,10€**

VALIDER LE PANIER

EPICERIE SUCRÉE

Produit	Prix unitaire	Quantité	Total
HARIBO - Tagada 300g	0,97€	- 1 +	Total 0,97€
HARIBO - Dragibus 250g	1,61€	- 1 +	Total 1,61€
HARIBO - Schtroumpf 300g	1,09€	- 1 +	Total 1,09€
HARIBO - Chamallows 300g	1,43€	- 1 +	Total 1,43€

4 ARTICLES DANS VOTRE PANIER **5,10€**

VALIDER LE PANIER

- Objet: Panier virtuel

Identifiant (numéro)

Structure (nombre d'articles, montant total, date...)

Comportement (annuler, valider, payer...)

- Objet: Article dans le panier

Identifiant (nom)

Structure (quantité, prix unitaire...)

Comportement (ajouter, supprimer, remplacer...)



Approche objet

Objet

- ▶ Différences entre les approches:
 - Approche procédurale:
 - une fonction valide un panier.
 - une fonction ajoute un article au panier.
 - Approche objet:
 - le panier « est validé » à la demande du système, d'un autre objet...
 - un article du panier « est rajouté».
- ▶ En objet, tout système est composé d'objets:
 - interagissant entre eux
 - effectuant des opérations propres à leur comportement
- ▶ Comportement global d'un système objet: réparti entre les différents objets



Approche objet

Mise en oeuvre

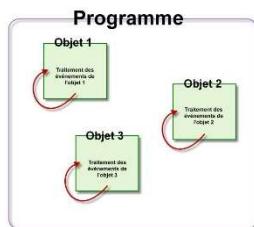
- ▶ Méthode de conception: Approche objet
- ▶ Langage de modélisation: UML
- ▶ Support de conception: ArgoUML ou Modelio
- ▶ Langage de programmation: C++



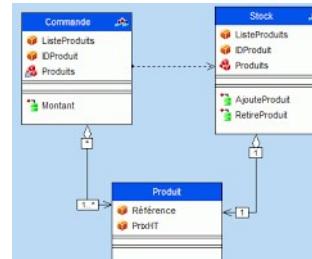
Approche objet

Objectifs

- ▶ Objectif : construire un programme en objet



- ▶ Nécessité de définir ces objets:
 - Combien? Lesquels?
 - Leurs comportements?



⇒ Diagramme de classe

- ▶ Avant cela: définir les limites
les services rendus par le système



Approche objet

Objectifs

- Première étape: définir le système en modélisant les exigences
 - Cas d'utilisation
 - Scénarios

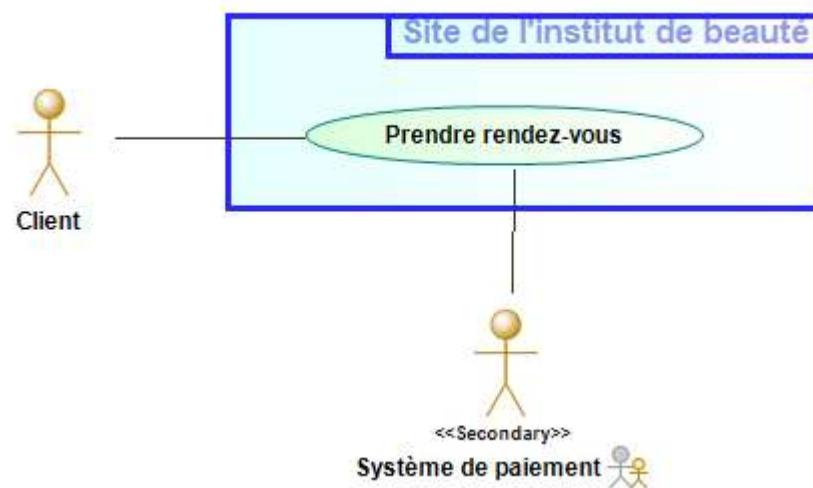


Modélisation des exigences

Cas d'utilisation

► Objectifs: description

- Des **exigences fonctionnelles** attendues lors de la rédaction d'un cahier des charges d'un système
- Des **fonctionnalités** d'un système existant



Modélisation des exigences

Cas d'utilisation

► Contenu:

- Exigences fonctionnelles attendues ou existantes
- Acteurs (utilisateurs) du système
- Relations entre acteurs et exigences

► Intérêt:

- Support pour la modélisation, le développement et la validation
- Référentiel normé de dialogue entre informaticiens et clients
- Base pour l'élaboration du cahier des charges fonctionnel



Modélisation des exigences

Cas d'utilisation

- ▶ Définition d'un cas d'utilisation:
 - Description sous la forme d'actions et de réactions du comportement du système **du point de vue des utilisateurs**
- ▶ Cas d'utilisation avec objectif utilisateur:
 - Description des interactions entre un utilisateur et le système, interactions **liées à un objectif fonctionnel** de l'utilisateur
 - Explication de la partie des exigences fonctionnelles du système qui concerne **l'un des objectifs de l'utilisateur**
 - Réalisation d'un service de bout en bout: déclenchement, déroulement, fin
- ▶ Utilité: Définition
 - Des frontières du système (fonctionnalités qu'il remplit)
 - De ses relations avec son environnement



Modélisation des exigences

Cas d'utilisation

► Définition des cas d'utilisation d'un système:

- Définition des fonctionnalités du système: Diagramme des cas d'utilisation
 - > Définition des cas d'utilisation du système: nommage
 - > Définition des acteurs
 - > Définition des relations entre les acteurs et les cas d'utilisation
- Définition de la réalisation des fonctionnalités: Scénarii
 - > Interactions entre le système et les utilisateurs pour réaliser un objectif fonctionnel de l'utilisateur
 - > Réalisation d'un service de bout en bout



Modélisation des exigences

Cas d'utilisation

► Choix du nom du cas d'utilisation:

- **Verbe à l'infinitif** avec complément
- Du point de vue de l'acteur et non pas du système
- Ex: distributeur automatique bancaire
 - > **Retirer de l'argent**: point de vue de l'acteur
 - > **Distribuer de l'argent**: point de vue du système

► Granularité des cas d'utilisation:

- Éviter la décomposition fonctionnelle \Rightarrow grand nombre de cas
- Cas d'utilisation
 - = une fonction métier du système selon le point de vue d'un acteur
 - = un service attendu par un acteur pour effectuer son travail
- Ne peut se réduire à une seule interaction (ex: déclencher le chauffage plutôt gérer le chauffage)
- Réponse à la question

Comment et pourquoi l'acteur se sert-il du système?



Modélisation des exigences

Acteur

- ▶ Acteur = Cas **d'utilisateur externe** au système jouant un rôle vis-à-vis du système
- ▶ Définition d'un acteur:
 - couple (utilisateur, rôle)
- ▶ Désignation: par le nom du rôle
- ▶ Exemple: autres systèmes interagissant avec le système étudié
- ▶ Deux catégories:
 - **acteurs primaires:** l'objectif du cas d'utilisation est essentiel pour eux
 - > Ex: client du distributeur automatique
 - **acteurs secondaires:** interagissent avec le cas d'utilisation mais il n'est pas essentiel pour eux
 - > Ex: base de donnée des interdits bancaires consultée pour le retrait



Modélisation des exigences

Acteur

► Acteur ≠ utilisateur

- Même rôle pour plusieurs utilisateurs ⇒ même acteur
- Plusieurs rôles par une même personne physique ⇒ plusieurs acteurs

► Acteurs:

- Communication directe avec le système
- Émission et réception de message du ou vers le système
 - > Hôtesse de caisse: acteur pour la caisse
 - > Client: non car pas d'interaction avec le système
 - > Client acteur d'une caisse automatique



Modélisation des exigences

Relation de communication

- ▶ Définition de la relation de communication :
relation **liant un acteur à un cas d'utilisation**
- ▶ Différents modèles de communication:
 - services que le système doit fournir à chacun des acteurs du cas d'utilisation;
 - informations du système qu'un acteur peut introduire, consulter ou modifier;
 - changements intervenant dans l'environnement dont un acteur informe le système;
 - changements intervenant au sein du système dont ce dernier informe un acteur.



Modélisation des exigences

Relation de communication

► Exemple:

- Système: Site web d'un institut de beauté
- Cas d'utilisation: Prendre rendez-vous
- Acteur primaire: Client
- Acteur secondaire: Système de paiement
- Scénario: Mme Voisin prend rendez-vous pour une manucure
- Relations de communication:
 - > Mme Voisin reçoit des informations de l'institut
 - > Mme Voisin fournit des informations de l'institut

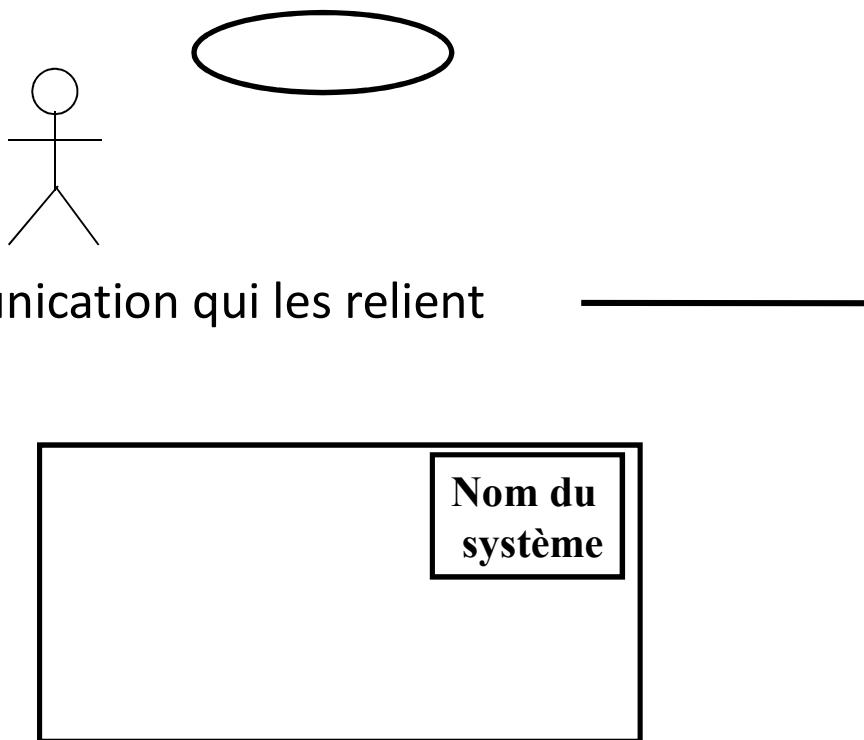


Modélisation des exigences

Diagramme des cas d'utilisation

- ▶ Définition du diagramme des cas d'utilisation :
diagramme représentant

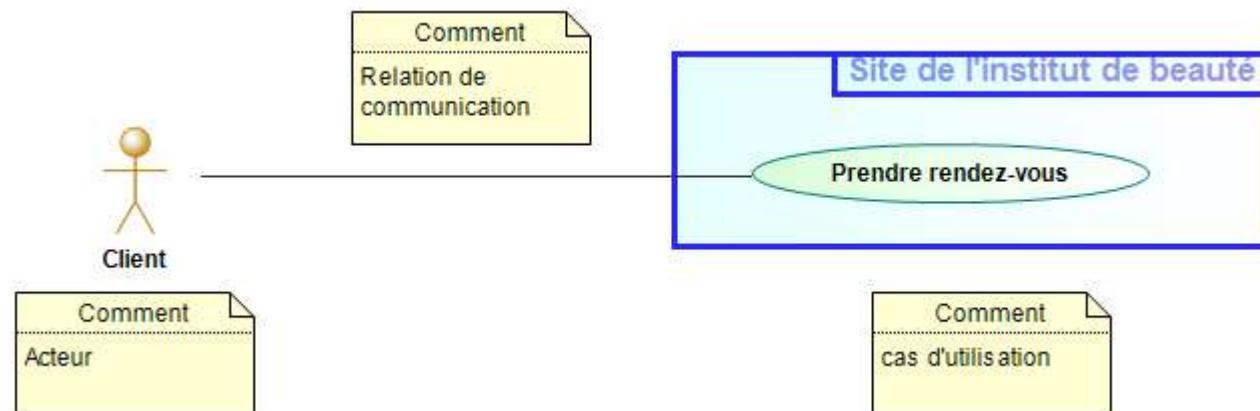
- les cas d'utilisations
- les acteurs
- les relations de communication qui les relient
- le système



Modélisation des exigences

Diagramme des cas d'utilisation

► Exemple :

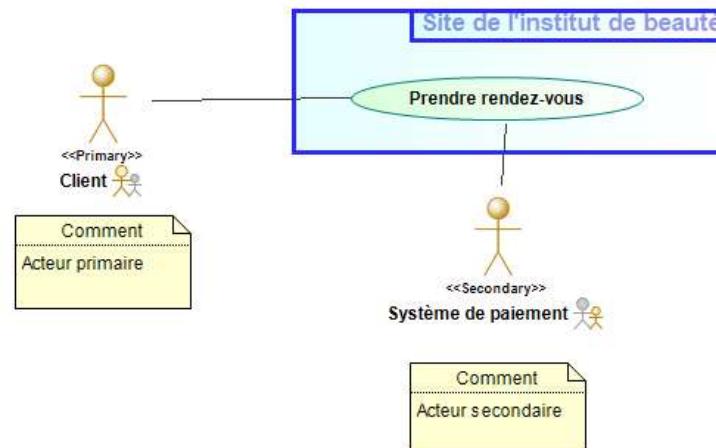


Modélisation des exigences

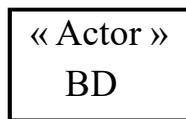
Diagramme des cas d'utilisation

► Acteurs primaire et secondaire :

- représentation identique
- souvent sens de communication inverse: système initie communication avec acteur secondaire
- ex: BD interrogée, Réseau énergétique (électrique, télécommunications...)



- Possibilité de différencier les acteurs non humains (système informatique)



Modélisation des exigences

Relations entre cas d'utilisation: Inclusion

- ▶ **Inclusion:** **enrichissement systématique** d'un cas d'utilisation par un autre
- ▶ cas inclus:
 - sous-fonction (*obligatoirement* utilisée par la fonction)
 - ne répond pas à un objectif d'un acteur primaire
- ▶ intérêt:
 - partager une fonctionnalité commune entre plusieurs cas d'utilisation
 - structuration d'un cas d'utilisation en sous-fonction
- ▶ Danger: décomposition fonctionnelle
- ▶ Notation UML:

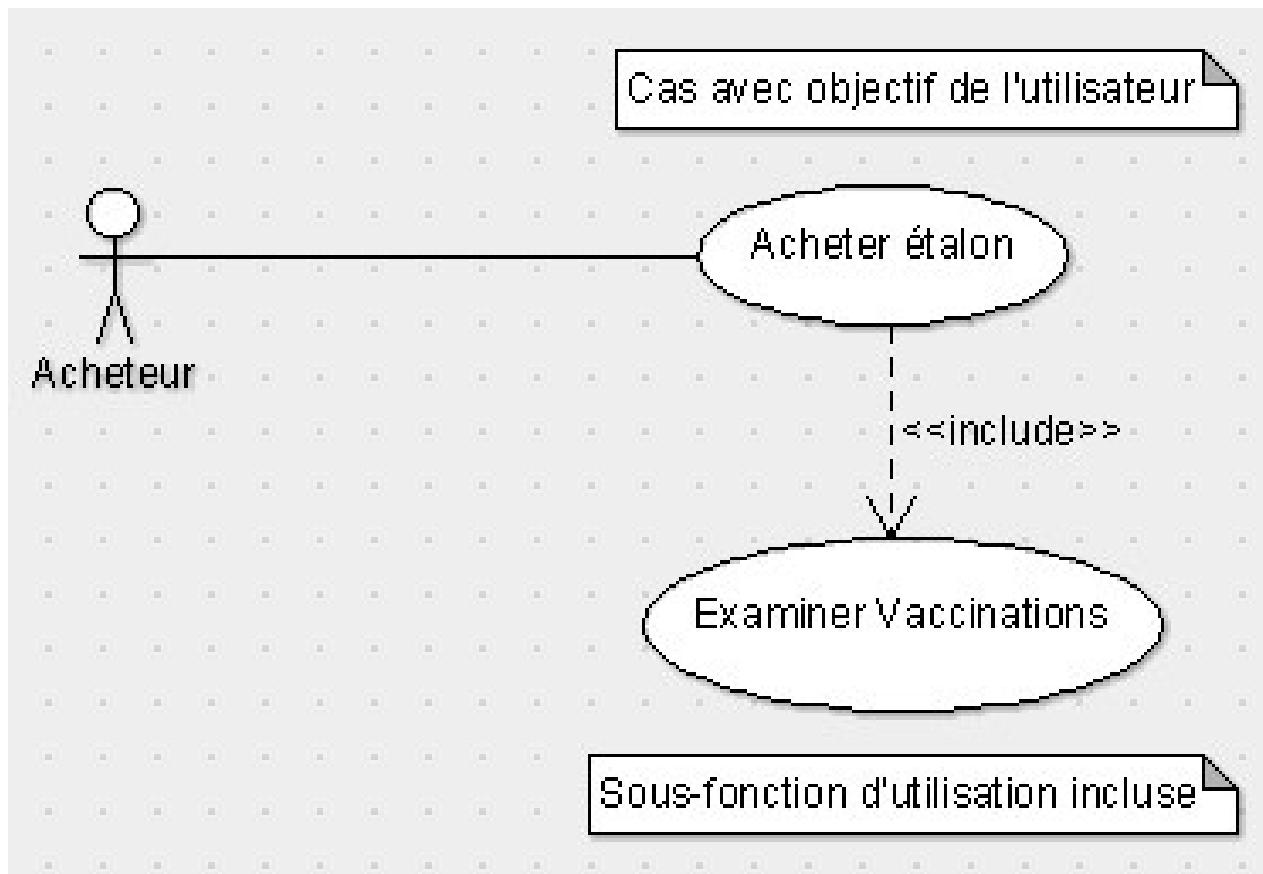
« « include » »
→



Modélisation des exigences

Relations entre cas d'utilisation: Inclusion

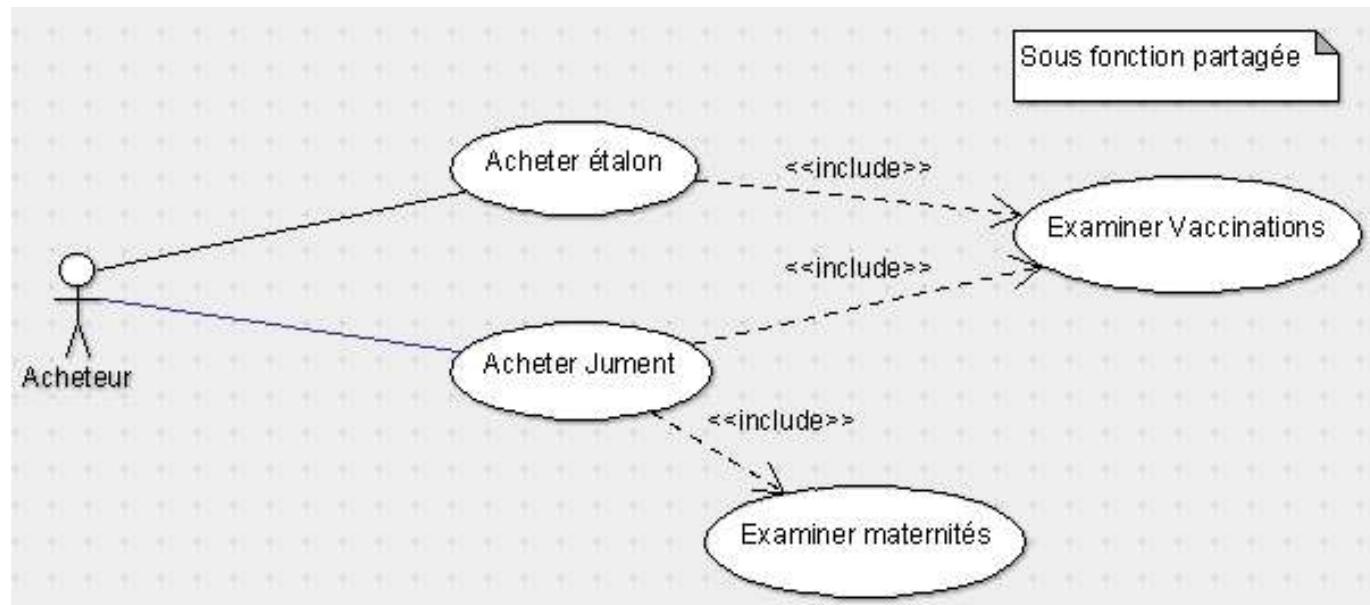
► Exemple:



Modélisation des exigences

Relations entre cas d'utilisation: Inclusion

- Exemple: décomposition d'un cas d'utilisation par inclusion



Modélisation des exigences

Relations entre cas d'utilisation: Extension

- ▶ Extension: enrichissement optionnel d'un cas d'utilisation par un autre (sous-fonction)
- ▶ réalisée en des points précis du cas d'utilisation de base: points d'extension
- ▶ application décidée lors du déroulement d'un scénario
- ▶ possibilité d'employer le cas d'utilisation de base sans l'étendre (*≠ de l'inclusion*)



Modélisation des exigences

Relations entre cas d'utilisation: Extension

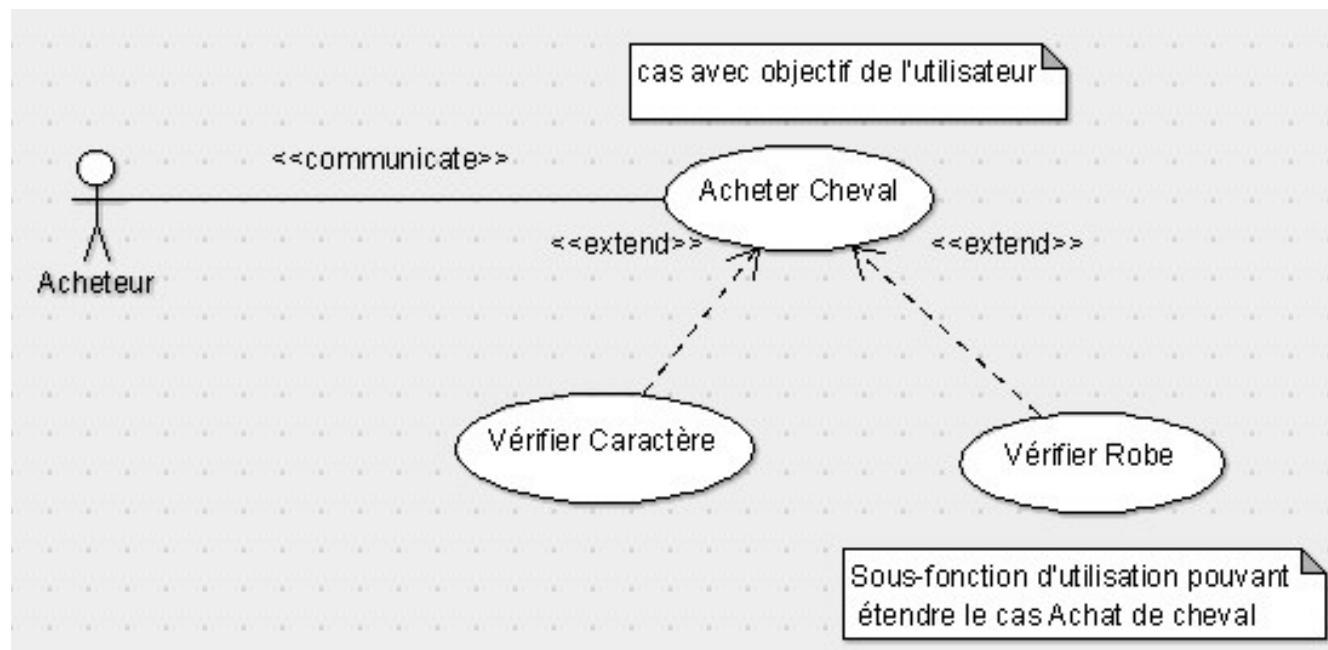
- ▶ Intérêt (identique à celui de l'inclusion):
 - partager une fonctionnalité commune entre plusieurs cas d'utilisation
 - structuration d'un cas d'utilisation en sous-fonction
- ▶ Danger (identique à celui de l'inclusion): décomposition fonctionnelle
- ▶ Notation UML:
 « « extend » »
 -----→



Modélisation des exigences

Relations entre cas d'utilisation: Extension

► Exemple:



Modélisation des exigences

Spécialisation et généralisation des cas d'utilisation

► Spécialisation d'un cas d'utilisation:

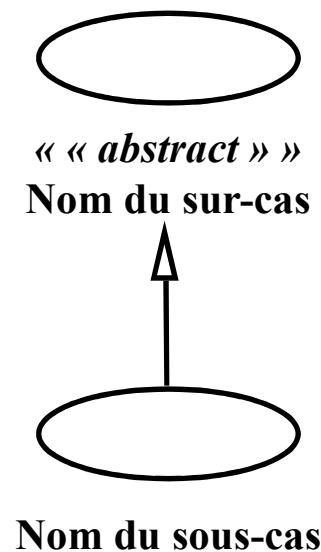
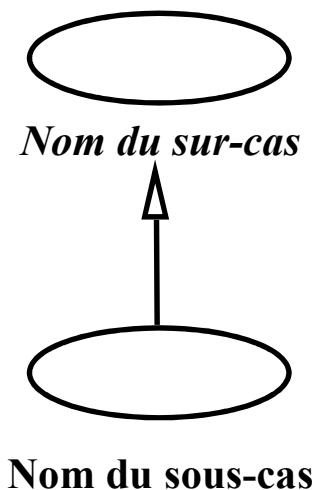
- Obtention d'un sous-cas d'utilisation
- Sous-cas hérite du sur-cas d'utilisation:
 - > comportement
 - > relations de communication
 - > inclusion
 - > extension
- Langage commun: cas général (sur-cas) et cas particulier (sous-cas) donc mêmes propriétés
- Souvent sur-cas d'utilisation abstrait: comportement partiel complété dans les sous-cas d'utilisation
- Même niveau pour le sous et sur cas d'utilisation:
 - > cas avec objectif utilisateur
 - > cas de sous-fonction



Modélisation des exigences

Spécialisation et généralisation des cas d'utilisation

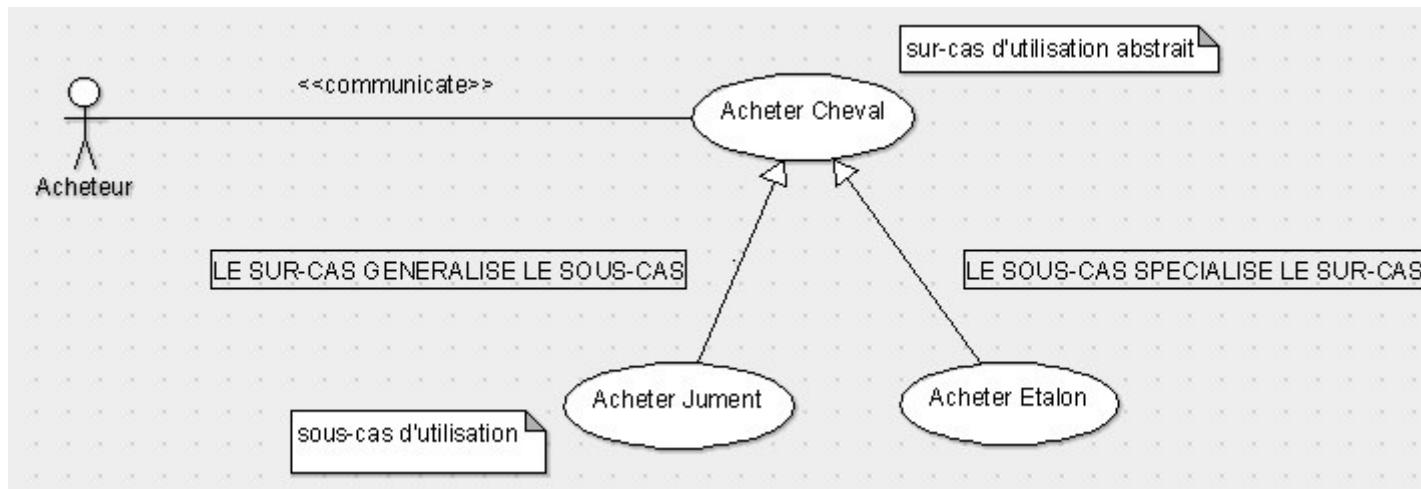
► Notation UML:



Modélisation des exigences

Spécialisation et généralisation des cas d'utilisation

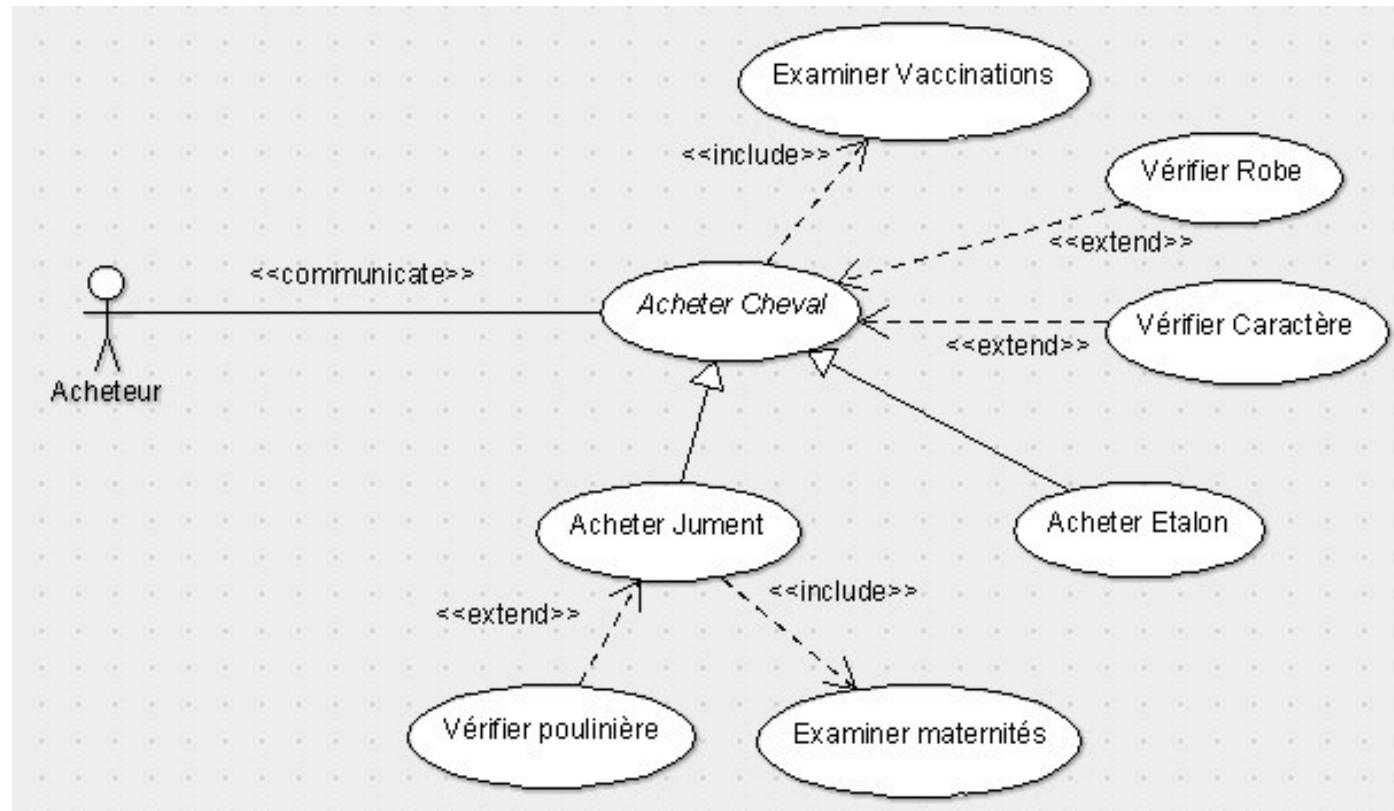
► Exemple:



Modélisation des exigences

Spécialisation et généralisation des cas d'utilisation

► Exemple:



Relations d'extensions factorisées dans le cas abstrait

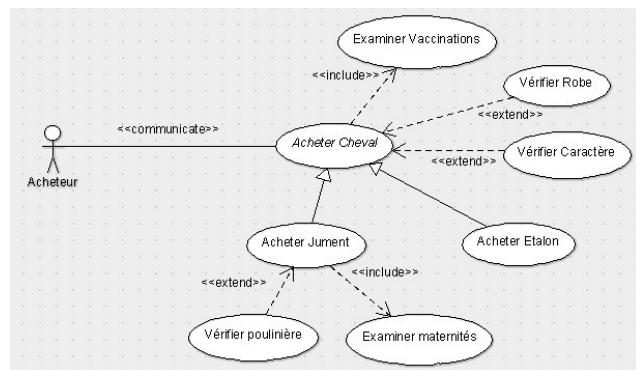


Modélisation des exigences

Cas d'utilisation: conclusion

► Consignes pour l'élaboration des diagrammes des cas d'utilisation:

- Interactions liées à un objectif de l'utilisateur
- Réalisation d'un service de bout en bout
- Verbes à l'infinitif
- Eviter la décomposition fonctionnelle



Contre-exemple d'intérêt pédagogique

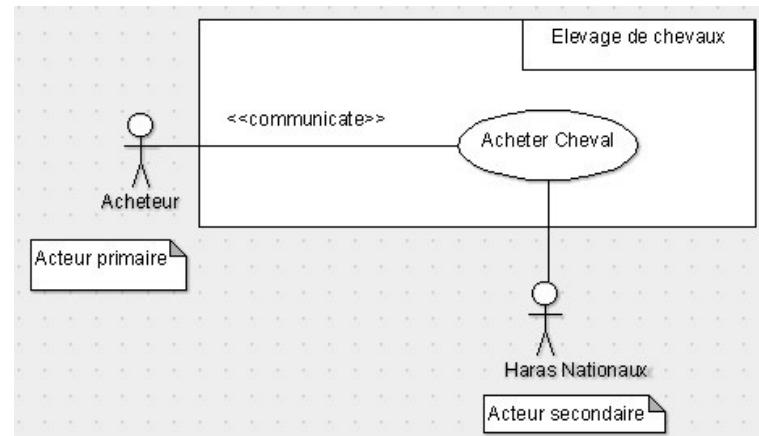


Diagramme utile



Modélisation des exigences

Cas d'utilisation: conclusion

- ▶ Objectif de la conception:
Diagramme de classe car « programmable »
 - ▶ Diagramme des cas d'utilisation:
 - Limites du système donc de l'association d'objets
 - Objectifs des utilisateurs: services rendus par les objets
- ⇒ **Limites et objectifs du système** représenté par le diagramme de classe
- ▶ Etape suivante: comment le système permet-il à l'utilisateur de réaliser son objectif?
⇒ scénario



Modélisation des exigences

Scénario

► Définition de scénario:

Instance d'un cas d'utilisation dans laquelle toutes les conditions relatives aux différents événements ont été fixées
⇒ pas d'alternative lors du déroulement

EVALUER LES CANDIDATS		
Cas d'utilisation	Acteur primaire	Responsable de casting
Système	Système informatique de gestion des castings	
Intervenants	Responsable du casting Candidats	
Niveau	Objectif utilisateur	
Préconditions	Le casting a été défini et organisé. Les candidatures des candidats ont été validées. Les candidats ont été pointés. Le responsable du casting s'est connecté au système.	
Opérations	Acteur	Système informatique de gestion des castings
1		Le système affiche la liste des castings.
2	Le responsable du casting sélectionne le casting du jour.	
3		Le système affiche la liste des personnes présentes devant passer le casting.
4	Le responsable du casting sélectionne une personne.	
5		Le système affiche la fiche de la personne choisie.
6	Le responsable du casting saisit ses commentaires et sa conclusion pour la personne.	
7		Le système affiche la liste des personnes présentes devant passer le casting.
8	Le responsable du casting indique que c'est la fin du casting.	
9		Le système affiche un récapitulatif des commentaires et des conclusions.
10	Le responsable du casting valide ces résultats.	
11		Le système envoie un message à chaque candidat pour lui indiquer son résultat.
12		Le système imprime la liste des candidats retenus.
Extensions		
10.A	Modification des résultats.	
10.A.1	Le responsable du casting demande une modification des résultats.	
10.A.2		Le système affiche la liste des personnes présentes au casting.
10.A.3	Le responsable du casting sélectionne une personne.	
10.A.4		Le système affiche les commentaires et la conclusion pour cette personne.
10.A.5	Le responsable du casting modifie la conclusion pour cette personne.	
10.A.6		Le système affiche un récapitulatif des commentaires et des conclusions. Retour à l'étape 10.



Modélisation des exigences

Scénario

- ▶ Représentation textuelle des cas d'utilisation
- ▶ Plusieurs scénarios pour un cas d'utilisation donné
- ▶ Cas d'utilisation: description commune de l'ensemble de ces scénarios par l'utilisation de branchements conditionnels



Modélisation des exigences

Scénario

- ▶ Non spécifiée dans UML
- ▶ Description pour les cas d'utilisation de leurs
 - Comportements
 - Actions
 - Réactions
- ▶ Différentes descriptions:
 - Séquence nominale: déroulement normal
 - Séquence alternative: variante de la séquence nominale
 - Séquence d'exception: cas d'une erreur



Modélisation des exigences

Scénario

- ▶ Scénarios non normés mais recommandation:

Scénarios: descriptions des interactions entre l'utilisateur et le système sans présumer de la forme de l'interface

- ▶ Intérêt de spécifier séparément les interactions et l'interface:
 - Séparation des 2 problématiques donc simplification
 - Séparation de la logique de fonctionnement (interaction) et de la couche visuelle (interface): architecture Modèle-Vue-Contrôleur
 - Séparation des rôles parmi les développeurs:
 - > informaticiens (interactions)
 - > ergonomes (interface)
- ▶ Elaboration de l'interface: étape de maquettage



Modélisation des exigences

Scénario

► Consignes d'élaboration des scénarios:

- Descriptions d'une **action de l'utilisateur** sous la forme
« L'utilisateur demande à ... »
Ex: s'authentifier, imprimer, consulter...
« L'utilisateur sélectionne ... »
Ex: un nom dans la liste...
- Descriptions d'un **retour du système** sous la forme
« Le système demande ... »
Ex: l'identifiant, le mot de passe, le numéro de compte...
« Le système affiche ... »
Ex: le solde du compte, une liste d'étudiants...



Modélisation des exigences

Scénario

Même nom pour les scénarios: Importance de cette cohérence pour la lisibilité de l'analyse

Cas d'utilisation	EVALUER LES CANDIDATS	
Acteur primaire	Responsable de casting	
Système	Système informatique de gestion des castings	
Intervenants	Responsable du casting Candidats	
Niveau	Objectif utilisateur	
Préconditions	Le casting a été défini et organisé. Les candidatures des candidats ont été validées. Les candidats ont été pointés. Le responsable du casting s'est connecté au système.	
Opérations	Acteur	Système informatique de gestion des castings
1		Le système affiche la liste des castings.
2	Le responsable du casting sélectionne le casting du jour.	
3		Le système affiche la liste des personnes présentes devant passer le casting.
4	Le responsable du casting sélectionne une personne.	
5		Le système affiche la fiche de la personne choisie.
6	Le responsable du casting saisit ses commentaires et sa conclusion pour la personne.	



Modélisation des exigences

Scénario

Cas d'utilisation	EVALUER LES CANDIDATS	
Acteur primaire	Responsable de casting	
Système	Système informatique de gestion des castings	
Intervenants	Responsable du casting Candidats	
Niveau	Objectif utilisateur	
Préconditions	Le casting a été défini et organisé. Les candidatures des candidats ont été validées. Les candidats ont été pointés. Le responsable du casting s'est connecté au système.	
Opérations	Acteur	Système informatique de gestion des castings
1		Le système affiche la liste des castings.
2	Le responsable du casting sélectionne le casting du jour.	
3		Le système affiche la liste des personnes présentes devant passer le casting.
4	Le responsable du casting sélectionne une personne.	
5		Le système affiche la fiche de la personne choisie.
6	Le responsable du casting saisit ses commentaires et sa conclusion pour la personne.	
7		Le système affiche la liste des personnes présentes devant passer le casting.
8	Le responsable du casting indique que c'est la fin du casting.	
9		Le système affiche un récapitulatif des commentaires et des conclusions.
10	Le responsable du casting valide ces résultats.	
11		Le système envoie un message à chaque candidat pour lui indiquer son résultat.
12		Le système imprime la liste des candidats retenus.
Extensions	Modification des résultats.	
10.1	Le responsable du casting demande une modification des résultats.	
10.2		Le système affiche la liste des personnes présentes au casting.
10.3	Le responsable du casting sélectionne une personne.	
10.4		Le système affiche les commentaires et la conclusion pour cette personne.
10.5	Le responsable du casting modifie la conclusion pour cette personne.	
10.6		Le système affiche un récapitulatif des commentaires et des conclusions. Retour à l'étape 10.

Nom de l'extension

n° ligne de l'opération à laquelle l'extension s'applique

lettre distinguant les extensions d'une même ligne

n° de l'opération dans l'extension

n° ligne de retour de l'extension



Modélisation des exigences

Scénario

Scénario
nominal

Nouvelle
version

Cas d'utilisation	EVALUER LES CANDIDATS	
Acteur primaire	Responsable de casting	
Système	Système informatique de gestion des castings	
Intervenants	Responsable du casting Candidats	
Niveau	Objectif utilisateur	
Préconditions	Le casting a été défini et organisé. Les candidatures des candidats ont été validées. Les candidats ont été pointés. Le responsable du casting s'est connecté au système.	
Opérations	Acteur	Système informatique de gestion des castings
1		Le système affiche la liste des castings.
2	Le responsable du casting sélectionne le casting du jour.	
...		
9		Le système affiche un récapitulatif des commentaires et des conclusions.
10.A.1	Le responsable du casting demande une modification des résultats.	
10.A.2		Le système affiche la liste des personnes présentes au casting.
10.A.3	Le responsable du casting sélectionne une personne.	
10.A.4		Le système affiche les commentaires et la conclusion pour cette personne.
10.A.5	Le responsable du casting modifie la conclusion pour cette personne.	
10.A.6		Le système affiche un récapitulatif des commentaires et des conclusions. Retour à l'étape 10.

10. Le responsable du casting valide les résultats

10.A.1 Le responsable du casting demande une modification des résultats



Modélisation des exigences

Scénario

Cas d'utilisation	EVALUER LES CANDIDATS	
Acteur primaire	Responsable de casting	
Système	Système informatique de gestion des castings	
Intervenants	Responsable du casting Candidats	
Niveau	Objectif utilisateur	
Préconditions	Le casting a été défini et organisé. Les candidatures des candidats ont été validées. Les candidats ont été pointés. Le responsable du casting s'est connecté au système.	
Opérations	Acteur	Système informatique de gestion des castings
1		Le système affiche la liste des castings.
2	Le responsable du casting sélectionne le casting du jour.	
3		Le système affiche la liste des personnes présentes devant passer le casting.
4	Le responsable du casting sélectionne une personne.	
5		Le système affiche la fiche de la personne choisie.
6	Le responsable du casting saisit ses commentaires et sa conclusion pour la personne.	
7		Le système affiche la liste des personnes présentes devant passer le casting.
8	Le responsable du casting indique que c'est la fin du casting.	
9		Le système affiche un récapitulatif des commentaires et des conclusions.
10	Le responsable du casting valide ces résultats.	
11		Le système envoie un message à chaque candidat pour lui indiquer son résultat.
12		Le système imprime la liste des candidats retenus.
Extensions		
10.A	Modification des résultats.	
10.A.1	Le responsable du casting demande une modification des résultats.	
10.A.2		Le système affiche la liste des personnes présentes au casting.
10.A.3	Le responsable du casting sélectionne une personne.	
10.A.4		Le système affiche les commentaires et la conclusion pour cette personne.
10.A.5	Le responsable du casting modifie la conclusion pour cette personne.	
10.A.6		Le système affiche un récapitulatif des commentaires et des conclusions. Retour à l'étape 10.



Modélisation des exigences

Scénario: conclusion

- ▶ Objectif de la conception:
Diagramme de classe car « programmable »
- ▶ Diagramme des cas d'utilisation:
 - Limites du système donc de l'association d'objets
 - Objectifs des utilisateurs: services rendus par les objets
- ▶ Scénario: Par quelle suite d'interaction le système permet-il à l'utilisateur de réaliser son objectif?
- ▶ Etapes suivantes:
 1. Maquettage pour compléter les scénarios
 2. Quels éléments du système permettent-ils la réalisation de ces interactions?
⇒ Diagramme de classe dans les cas simples



Atelier de Génie Logiciel

Présentation

► Modelio

- Logiciel libre <https://www.modelio.org/>
- Aide à la conception orientée objet
- Création de diagrammes UML2
 - > Cas d'utilisation
 - > Classes
 - > Séquence
 - > État
 - > Communication
 - > Activité
 - > Déploiement.
- Génération de code et reverse engineering grâce à des extensions
 - > Java
 - > C++
 - > C#



Atelier de Génie Logiciel

Interface Modelio

The screenshot displays the Modelio 3.0 interface with several windows open:

- JavaBrowser - Modelio 3.0**: Main window title.
- Model**: Left-hand navigation pane showing the current project's class hierarchy. A red box labeled "1" highlights the "JavaBrowser" node under "com.modeliosoft.examples.browser".
- avaBrowser Class diagram**: The central workspace showing a UML Class Diagram. A red box labeled "2" highlights the "Browser" class. The diagram includes classes like Thread, JFrame, PageLoader, AddressListener, and Main, with various associations and stereotypes.
- Java**: Bottom-left pane showing Java module settings. A red box labeled "3" highlights the "Module" section.
- Element**: Bottom-middle-left pane showing element properties. A red box labeled "4" highlights the "Properties" table for a selected "JavaClass".
- Symbol**: Bottom-right pane showing diagram settings. A red box labeled "5" highlights the "Properties" table for the current diagram.

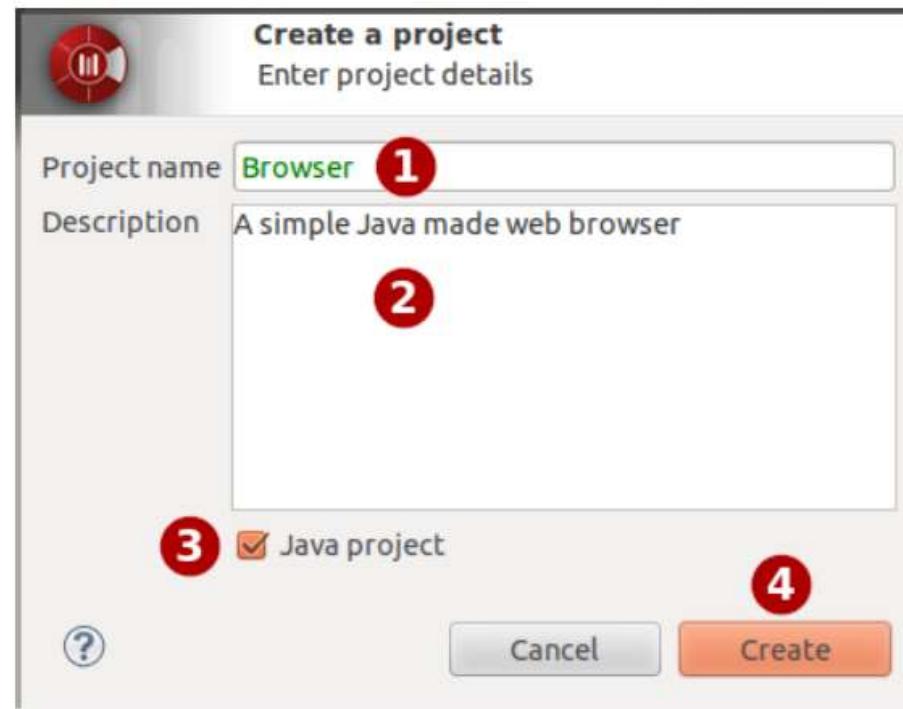
Overlaid text and numbers provide additional context:

- Hiérarchie du modèle (projet) courant** (Number 1)
- Diagramme sélectionné dans le modèle** (Number 2)
- Module** (Number 3)
- Propriétés de l'élément** (Number 4)
- Propriétés du diagramme** (Number 5)

Atelier de Génie Logiciel

Démarrage Modelio

►Création d'un projet



Exercice d'application

Diagramme des cas d'utilisation : covoiturage

Application informatique gérant le covoiturage au sein d'une entreprise.

- ▶ Sur le site, les employés peuvent rechercher et consulter les trajets proposés.
- ▶ Pour un trajet donné, ils peuvent proposer au conducteur une étape supplémentaire.
- ▶ Ils peuvent s'inscrire à un trajet et deviennent alors participants.
- ▶ Un participant à un trajet peut visualiser les étapes qu'il a demandées et l'acceptation ou le refus de ces étapes.
- ▶ Il peut confirmer ou annuler sa participation à un trajet ou à une étape.



Exercice d'application

Diagramme des cas d'utilisation : covoiturage

- ▶ Un employé peut se déclarer conducteur d'une ou de plusieurs voitures.
- ▶ Un conducteur peut proposer un trajet.
- ▶ Un conducteur peut supprimer ou modifier ses trajets.
- ▶ Il peut ajouter des étapes à ses trajets, les supprimer, accepter ou refuser des demandes d'étape.



Exercice d'application

Diagramme des cas d'utilisation : covoiturage

1. Souligner dans le texte les concepts importants puis surligner parmi eux les interactions.
2. Identifier les acteurs potentiels
3. Pour chaque acteur potentiel, définissez son cas d'utilisation principal.
4. Tracer le diagramme des cas d'utilisation



Exercice d'application

Diagramme des cas d'utilisation : covoiturage

1. Souligner dans le texte les concepts importants puis surligner parmi eux les interactions.
 - ▶ Sur le site, les employés peuvent rechercher et consulter les trajets proposés.
 - ▶ Pour un trajet donné, ils peuvent proposer au conducteur une étape supplémentaire.
 - ▶ Ils peuvent s'inscrire à un trajet et deviennent alors participants.
 - ▶ Un participant à un trajet peut visualiser les étapes qu'il a demandées et l'acceptation ou le refus de ces étapes.
 - ▶ Il peut confirmer ou annuler sa participation à un trajet ou à une étape.
 - ▶ Un employé peut se déclarer conducteur d'une ou de plusieurs voitures.
 - ▶ Un conducteur peut proposer un trajet.
 - ▶ Un conducteur peut supprimer ou modifier ses trajets.
 - ▶ Il peut ajouter des étapes à ses trajets, les supprimer, accepter ou refuser des demandes d'étape.



Exercice d'application

Diagramme des cas d'utilisation : covoiturage

2. Identifier les acteurs potentiels

- ▶ Sur le site, les employés peuvent rechercher et consulter les trajets proposés.
- ▶ Pour un trajet donné, ils peuvent proposer au conducteur une étape supplémentaire.
- ▶ Ils peuvent s'inscrire à un trajet et deviennent alors participants.
- ▶ Un participant à un trajet peut visualiser les étapes qu'il a demandées et l'acceptation ou le refus de ces étapes.
- ▶ Il peut confirmer ou annuler sa participation à un trajet ou à une étape.
- ▶ Un employé peut se déclarer conducteur d'une ou de plusieurs voitures.
- ▶ Un conducteur peut proposer un trajet.
- ▶ Un conducteur peut supprimer ou modifier ses trajets.
- ▶ Il peut ajouter des étapes à ses trajets, les supprimer, accepter ou refuser des demandes d'étape.



Exercice d'application

Diagramme des cas d'utilisation : covoiturage



FIN DU PREMIER COURS



Diagrammes UML

- ▶ Diagramme des cas d'utilisation: premier diagramme UML de la conception
- ▶ 2 grands types de diagramme UML:
 - Structure
 - Comportement
- ▶ Différentes méthodes de conception objet des systèmes basées sur ces diagrammes



Diagrammes UML

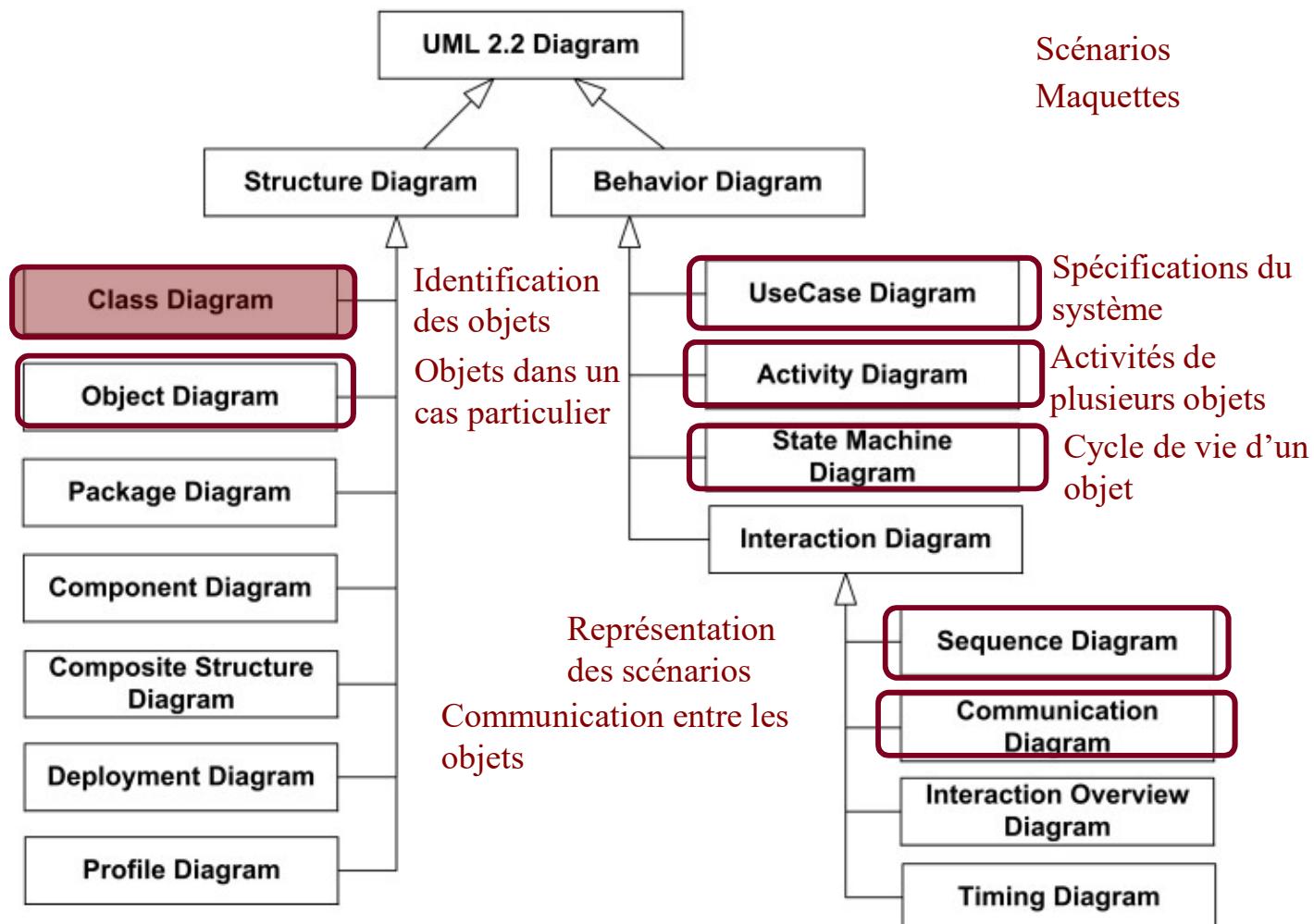
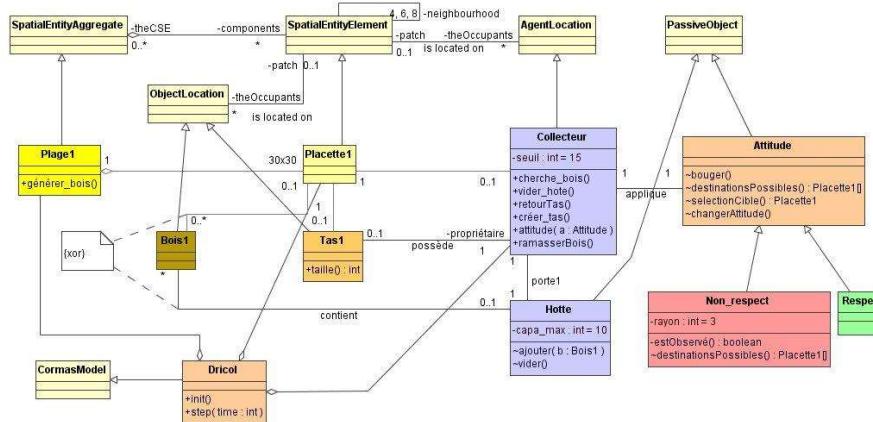


Diagramme de classe

Objet et classe

- Seul diagramme « utile » pour la programmation



- Système composé d'objets :
 - interagissant entre eux
 - effectuant des opérations propres à leur comportement
- Comportement global d'un système UML: réparti entre les différents objets



Diagramme de classe

Objet et classe

► Exemple: commerce électronique

The screenshot shows a shopping cart interface with the following details:

DÉTAILS DE VOTRE PANIER

4 ARTICLES DANS VOTRE PANIER **5,10€** < Retourner au rayon VALIDER LE PANIER

EPICERIE SUCRÉE

Produit	Prix	Quantité	Total
HARIBO - Tagada 300g	0,97€ Total 0,97€	- 1 +	1,61€ Total 1,61€
HARIBO - Dragibus 250g	1,61€ Total 1,61€	- 1 +	1,09€ Total 1,09€
HARIBO - Schtroumpf 300g	1,09€ Total 1,09€	- 1 +	1,43€ Total 1,43€
HARIBO - Chamallows 300g	1,43€ Total 1,43€	- 1 +	

4 ARTICLES DANS VOTRE PANIER **5,10€** VALIDER LE PANIER

- Objet: Panier virtuel
 - > Identifiant (numéro)
 - > Structure (nombre d'articles, montant total, date...)
 - > Comportement (annuler, valider, payer...)

► Toujours dynamique:

- un livre est un objet capable de s'ouvrir à la nième page
- voiture: démarrer, s'arrêter, rouler, allumer voyant...



Diagramme de classe

Objet et classe

► Composition d'un objet dans la méthode étudiée:

- **attributs:**
 - > structure de l'objet
 - > variable destinée à recevoir une valeur
- **méthodes:**
 - > comportement de l'objet
 - > ensemble d'instructions
 - prenant des valeurs en entrée
 - modifiant les valeurs des attributs ou produisant un résultat



Diagramme de classe

Objet et classe

► Exemples:

- **Attributs:**

- > Carte de crédit:

- Numéro
- Code de sécurité (CVV2/CV2)
- Date d'expiration



- > Panier virtuel:

- Numéro
- Nombre d'articles
- Montant total
- Date
- Nom du propriétaire

DÉTAILS DE VOTRE PANIER	
4 ARTICLES DANS VOTRE PANIER	5,10€
EPICERIE SUCRÉE	
HARIBO - Tagada 300g	0,97€ Total 0,97€
HARIBO - Dragebus 250g	1,61€ Total 1,61€
HARIBO - Schtroumpf 300g	1,09€ Total 1,09€
HARIBO - Chamallows 300g	1,43€ Total 1,43€
4 ARTICLES DANS VOTRE PANIER	
VALIDER LE PANIER	



Diagramme de classe

Objet et classe

► Exemples:

- Méthodes:

- > Carte de Crédit:

- Activer
 - Mettre en opposition



- > Panier virtuel:

- Annuler
 - Valider
 - Payer

A screenshot of a virtual shopping cart interface. The header shows "DÉTAILS DE VOTRE PANIER" and "5,10€". Below it, there's a section for "EPICERIE SUCRÉE" showing four items: HARIBO - Tagada 300g, HARIBO - Drageus 250g, HARIBO - Schtroumpf 300g, and HARIBO - Chamallows 300g. Each item has a price (0,97€, 1,61€, 1,09€, 1,43€), quantity (1), and a "Valider le panier" button.



Diagramme de classe

Objet et classe

► Définition d'une **classe d'objets**:

- Ensemble **d'objets similaires**
- **Description formelle** d'un ensemble d'objets ayant une sémantique et des caractéristiques communes

► Objets de la classe = instance de classe

(instance en anglais: cas, exemple)

- Identiques:
 - > Structure: mêmes attributs
 - > Comportement: mêmes méthodes
- Différents:
 - > Identité propre
 - > Valeurs spécifiques de ses attributs



Diagramme de classe

Objet et classe

► Représentation:

- Classe

NomClasse

NomClasse

- Objet

NomObjet:

:NomClasse

NomObjet:NomClasse



Diagramme de classe

Objet et classe

► Nomenclature des classes:

- Nom commun au singulier complété éventuellement d'adjectifs qualifiant le nom
- Nom significatif de l'ensemble des objets de la classe

► Exemples:

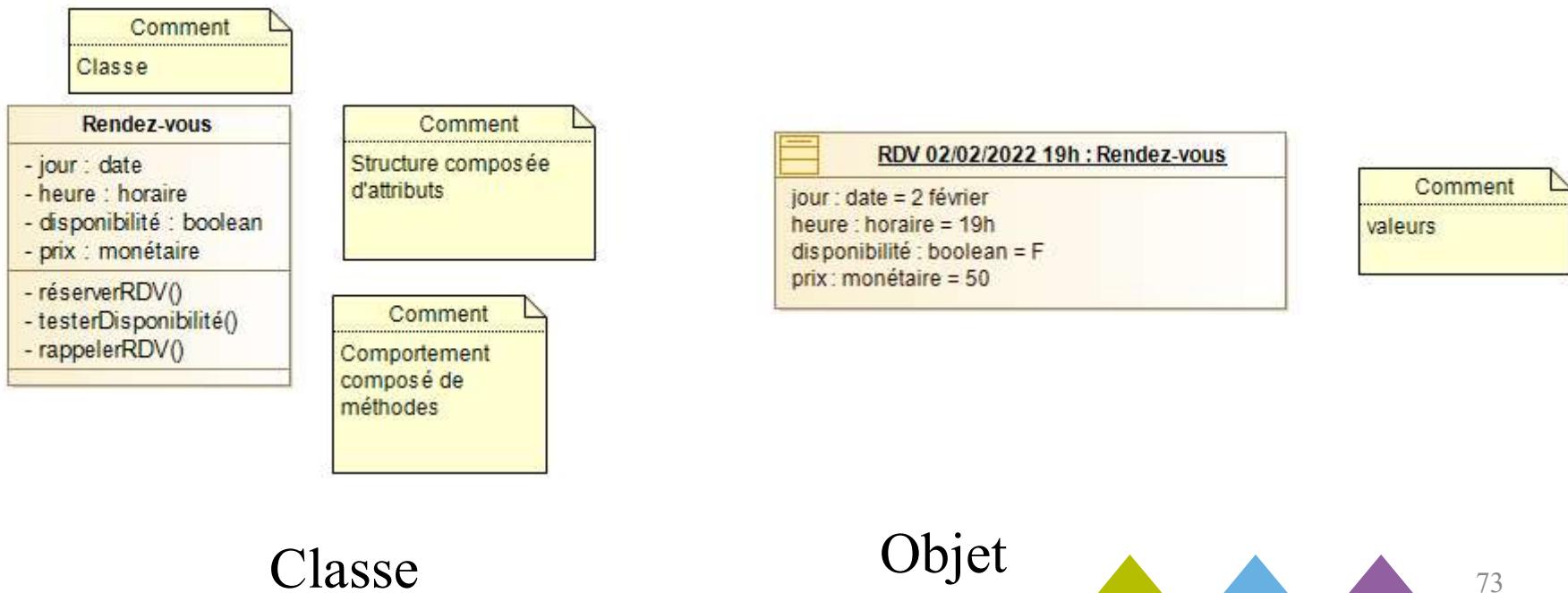


Diagramme de classe

Objet et classe

- ▶ Diagramme des classes: seul diagramme obligatoire lors d'une modélisation par objets
- ▶ Modélisation statique:
 - ni les interactions entre objets
 - ni le cycle de vie des objets
 - pas de description de l'enchaînement des méthodes



Diagramme de classe

Représentation simplifiée des classes

► 3 parties de la **forme simplifiée** d'une classe:

- **Nom:**

- > Singulier avec une majuscule
 - > Nom commun avec adjectifs si besoin

- **Attributs:**

- > Contiennent l'information portée par un objet
 - > Constituent la structure de l'objet
 - > Nom débutant par une minuscule

- **Méthodes:**

- > Services offerts par l'objet
 - > Peuvent modifier la valeur des attributs
 - > Constituent le comportement de l'objet
 - > Nom débutant par une minuscule



Diagramme de classe

Représentation simplifiée des classes

► Bonne modélisation: nombre limité

- d'attributs
- de méthodes

► Granularité des méthodes:

- Action élémentaire (\approx procédure, fonction...)
- \neq cas d'utilisation

► Forme simplifiée:

- seulement le nom des attributs et méthodes
- pas les autres caractéristiques



Diagramme de classe

Représentation simplifiée des classes

- Exemple de représentation simplifiée:

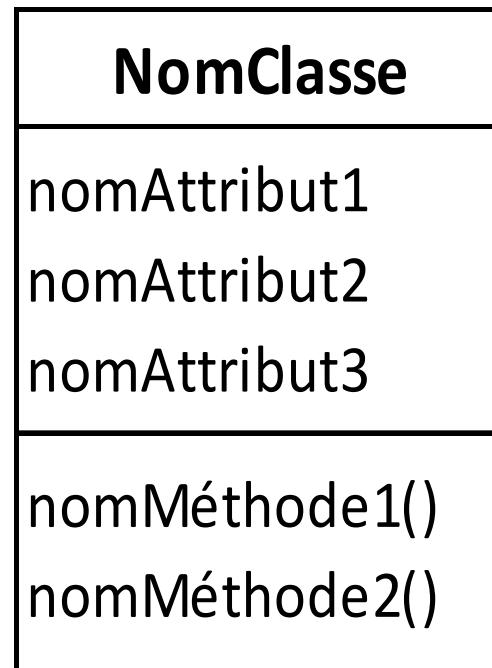


Diagramme de classe

Représentation simplifiée des classes

- ▶ Caractéristiques d'un objet:
 - Comportement: méthodes
 - Etat: vu plus tard dans le cours
 - Attributs : un nombre, une chaîne de caractères, un type simple
 - Autre caractéristique vue plus tard dans le cours: Terminaison d'association du diagramme
- ▶ Critère de choix de modélisation des propriétés:
 - Propriétés simples: attributs
 - Propriétés complexes vues plus tard dans le cours: classe reliée à la classe étudiée par une association
- ▶ Attributions de valeurs aux propriétés décrites par des attributs: lors de l'instanciation de la classe



Diagramme de classe

Représentation simplifiée des classes

► Encapsulation:

Fait de **masquer des attributs et des méthodes** de l'objet vis-à-vis des autres objets: non accessibles

► Intérêt:

- Abstraction: simplification de la représentation de l'objet vis-à-vis des objets extérieurs
- Facilitation de l'évolution d'une application
- Modification de l'implémentation
- Pas de modification de la manière dont l'objet est utilisé: interface (plus tard dans le cours)

► Encapsulation des attributs et des méthodes:

- concernant les traitements internes à l'objet
- ne devant pas être exposés aux objets extérieurs

► Appellation: attributs et méthodes privés de l'objet



Diagramme de classe

Représentation simplifiée des classes

- ▶ Trois possibilités d'encapsulation en UML des méthodes et attributs:
 - **Privé:**
 - > propriété non exposée en dehors de la classe (non exposée également dans les sous-classes (spécialisation voir diapo suivantes))
 - > différencie instances de classes et de sous-classes
 - **Protégé:** propriété exposée uniquement dans les instances de la classe et de ses sous-classes
 - **Encapsulation de paquetage:**
 - > propriété exposée uniquement dans les instances des classes de même paquetage
 - > sert essentiellement à l'écriture des diagrammes de développeurs en Java



Diagramme de classe

Représentation simplifiée des classes

► Encapsulation:

- Notation UML:

public	+	élément non encapsulé, visible par tous
protégé	#	élément encapsulé visible dans les sous-classes de la classe
privé	-	élément encapsulé visible seulement dans la classe
paquetage	~	élément encapsulé visible seulement dans les classes du même paquetage



Diagramme de classe

Représentation simplifiée des classes

► Exemple d'encapsulation:

NombreComplexPolaire
- module : float - argument : float
+ addition() + soustraction() + multiplication() + division()

NombreComplexCartésien
- partieRéelle : float - partieImaginaire : float
+ addition() + soustraction() + multiplication() + division()

Comment
attributs privés: représentation interne indifférente aux autres classes

Comment
méthodes publiques: opérations utiles aux autres classes



Diagramme de classe

Représentation simplifiée des classes

► Encapsulation:

- Résultat : objet représenté par attributs et méthodes publiques
- Définition: faite au niveau de la classe
- Intérêt:
 - > Protection des données des accès intempestifs
 - > Indépendance entre:
 - utilisation de l'abstraction
 - sa réalisation qui ne dépend que de sa spécification
- En pratique: privé par défaut



Diagramme de classe

Représentation simplifiée des classes

► Notion de type:

- Variable: tout élément pouvant prendre une valeur (attribut, paramètre/valeur de retour d'une méthode)
- Type:
 - > Contrainte appliquée à une variable
 - > Fixe l'ensemble des valeurs possibles de la variable
 - > Exemples:
 - standard:
 - Integer entier (1 5 4 ...)
 - String chaîne de caractère
 - Boolean booléen (False, True)
 - Real réels (1,54 2,3 ...)
 - Currency montant monétaire
 - Défini sous la forme d'une classe: la variable contient une référence vers une instance de cette classe (cas des classes de bibliothèques externes ou des interfaces)



Diagramme de classe

Représentation simplifiée des classes

► Notation UML des types:

- dans la représentation de la classe
- attribut : type

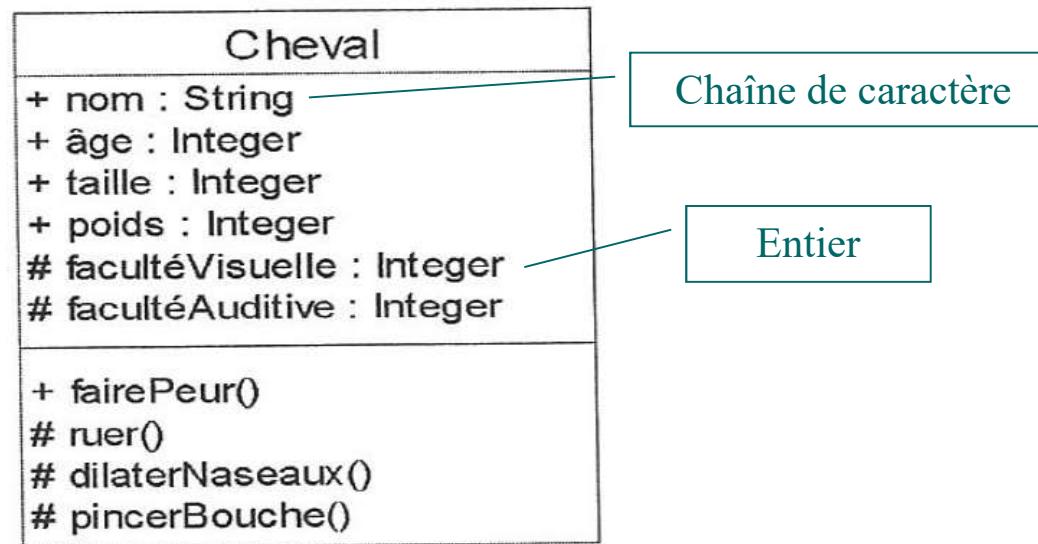


Diagramme de classe

Représentation simplifiée des classes

► Signature des méthodes:

- méthode:
 - > reçoit des paramètres en entrée: valeurs transmises lors de l'envoi d'un message appelant la méthode
 - > renvoie des paramètres en sortie: au retour de l'appel de méthode
- résultat: valeur transmise à l'objet appelant lors du retour de l'appel
- signature de la méthode: ensemble d'informations :
 - > nom de la méthode
 - > paramètres d'entrée avec leurs noms et leurs types (optionnel)
 - > type du résultat



Diagramme de classe

Représentation simplifiée des classes

► Signature des méthodes:

- Notation:

<nomMéthode> (<direction> <nomParamètre> : <type>, ...) : <typeRésultat>

- possibilité de ne pas avoir de paramètre
- type du résultat: optionnel
- Direction du paramètre (optionnel):
 - > **in**: valeur du paramètre transmise uniquement à l'appel (direction par défaut en absence d'indication)
 - > **out**: valeur du paramètre transmise uniquement au retour de l'appel de la méthode
 - > **inout**: valeur transmise à l'appel et au retour
- appel asynchrone présenté plus loin dans le cours: paramètres in uniquement



Diagramme de classe

Représentation simplifiée des classes

► Exemple:

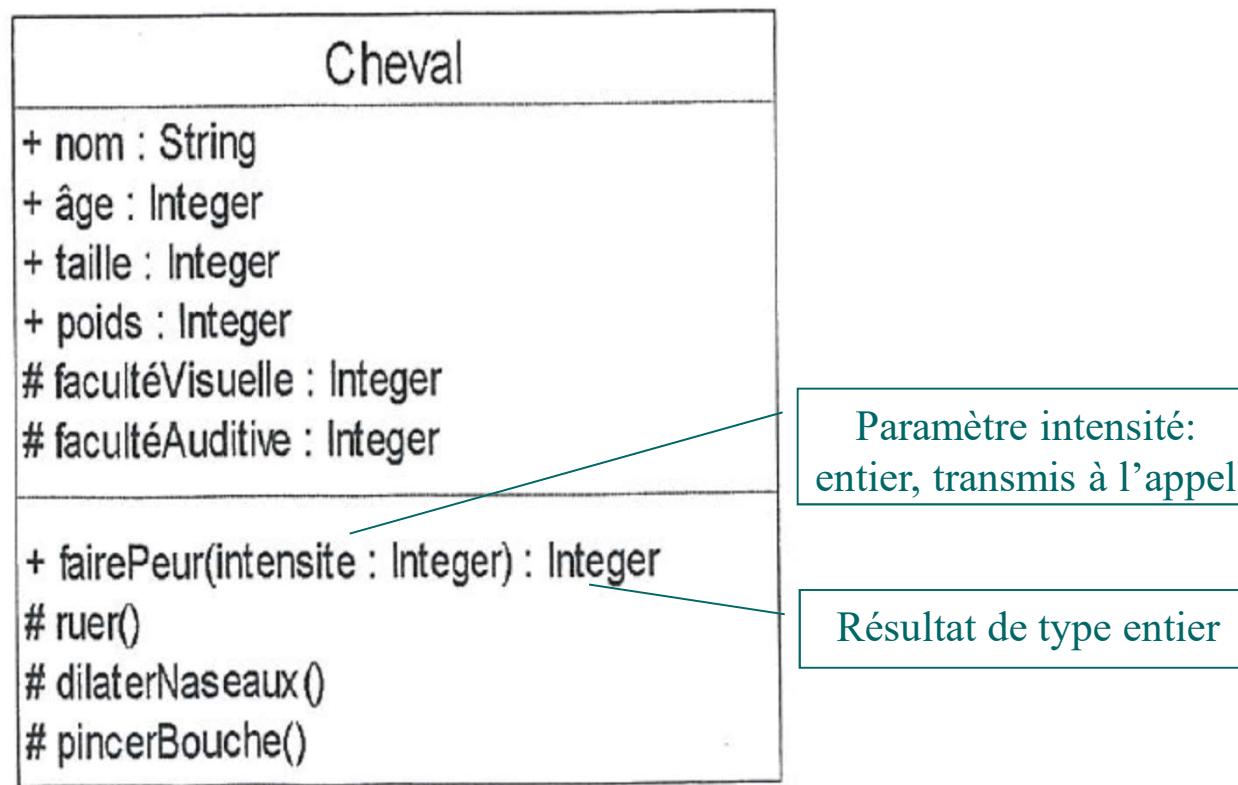


Diagramme de classe

Représentation simplifiée des classes

► Forme complète de représentation d'une classe:

- Composition de la forme complète:
 - > attributs avec:
 - caractéristiques d'encapsulation;
 - types;
 - > méthodes avec leurs signatures complètes.
- Possibilité de donner des valeurs par défaut:
 - > attributs: valeur lors de la création de l'objet;
 - > paramètres des méthodes: valeur lorsque l'appelant ne fournit pas explicitement ce paramètre.
- Intérêt: pour le développeur, description de la classe très proche de celle du langage de programmation
- Notation UML:

nomAttribut : typeAttribut = valeurDéfaut

nomMéthode (param : typeParam = valeurDéfaut) : typeRetour



Diagramme de classe

Représentation simplifiée des classes

► Exemple:

NomClasse
+nomAttribut1 : typeAttribut1 = valeurDéfaut
#nomAttribut2 : typeAttribut2 = valeurDéfaut
+nomAttribut3 : typeAttribut3 = valeurDéfaut
+nomMethode1(param1 : typeParam1 = valeurDefault, param2 : typeParam2) : typeRetour
nomMethode2(param : typeParam = valeurDefault) : typeRetour



Diagramme de classe

Représentation simplifiée des classes

- Définissez la classe Cours correspondant au cours magistral (CM) d'aujourd'hui



Diagramme de classe

Association entre objets

- ▶ Définition:
 - traduit les liens entre objets réels
 - relie des classes.
- ▶ Similitude:
 - une classe décrit un objet
 - une association décrit un lien
- ▶ Lien: occurrence d'une association
- ▶ Différents types d'association:
 - binaire: entre deux classes
 - ternaire: entre trois classes
 - n-aire: entre n classes
- ▶ Exemples:
 - parenté (père, mère...)
 - appartenance, propriété...



Diagramme de classe

Association entre objets

► Association:

- Nom:
 - > rôle de l'association ou nommage de ses terminaisons
 - > doit être significatif
- Visibilité de terminaisons: rôle
- Multiplicité/cardinalité des terminaisons
- Navigabilité: sens de lecture



Diagramme de classe

Association entre objets

► Représentation UML:

- trait continu avec le nom
- possibilité d'indiquer le sens de lecture (<>)
- possibilité de nommer les extrémités d'une association:
 - > rôles que jouent les instances de classes dans l'association
 - > un rôle:
 - même nature qu'un attribut dont le type serait la classe située à l'autre extrémité
 - public, privé, protégé ou dans paquetage
 - > présence des rôles peut rendre inutile le nom de l'association



Diagramme de classe

Association entre objets

► Exemples: est l'

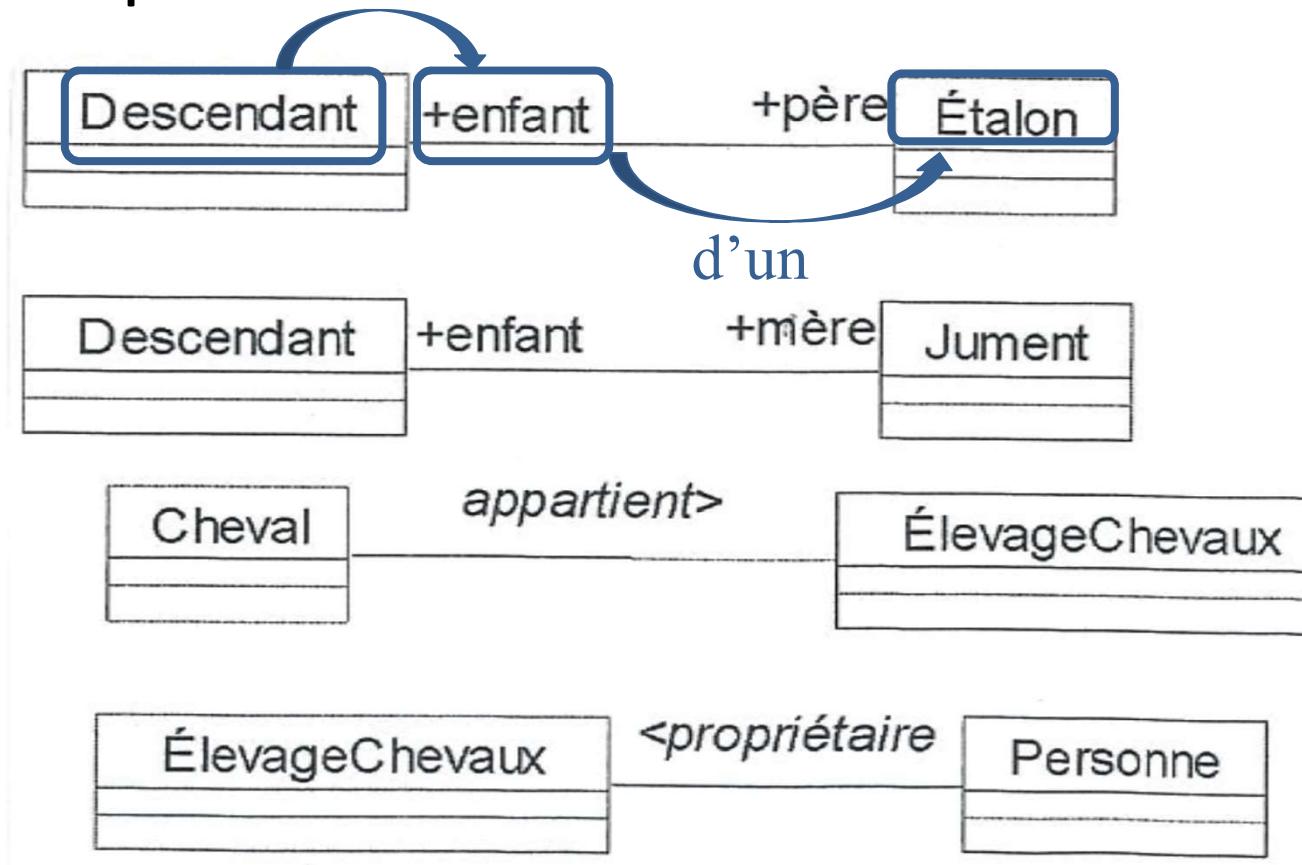
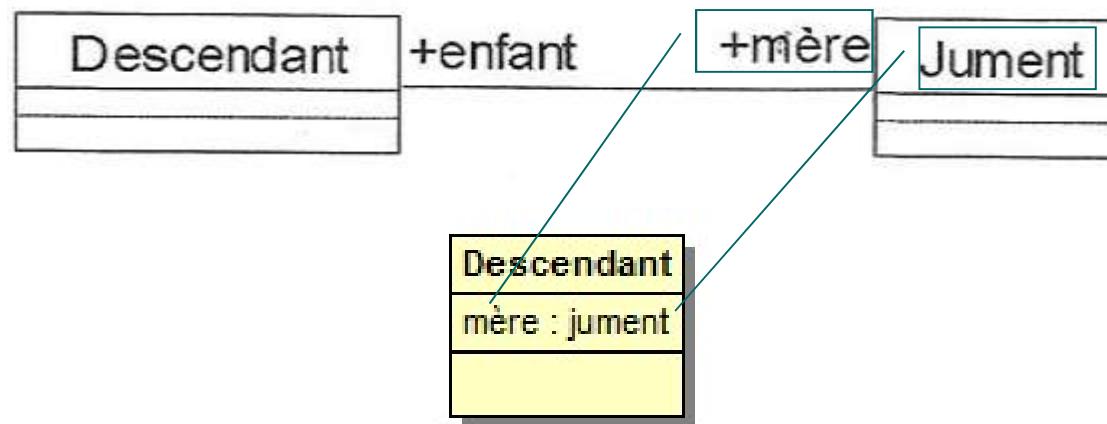


Diagramme de classe

Association entre objets

- ▶ Choix de représentation: Attribut ou classe associée?
 - Rôle d'une association : même nature qu'un attribut dont le type serait la classe située à l'autre extrémité



- Conseil:

Éviter les attributs dont le type est un concept complexe du domaine et non un type de données (réel, entier,...)



Diagramme de classe

Association entre objets

- Attribut \Leftrightarrow terminaison d'un cas particulier d'association dégénérée (suppression d'une des extrémités de l'association)

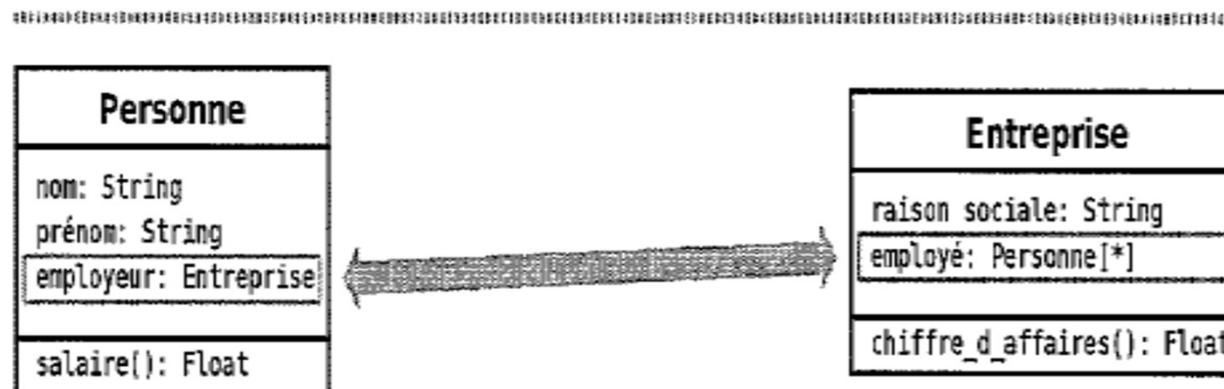
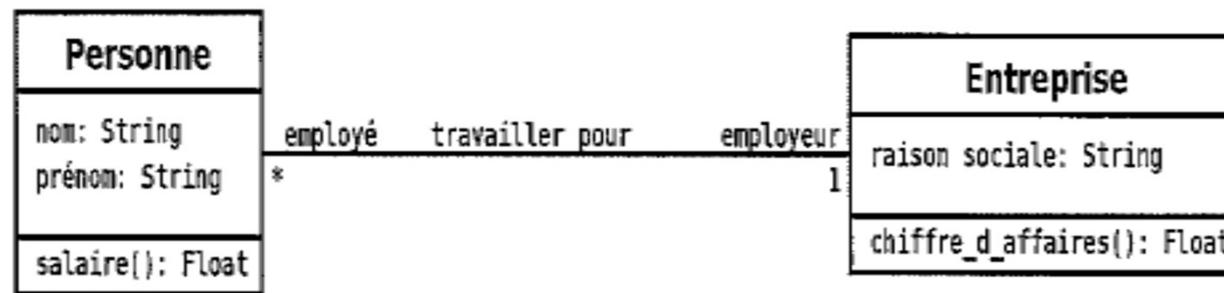
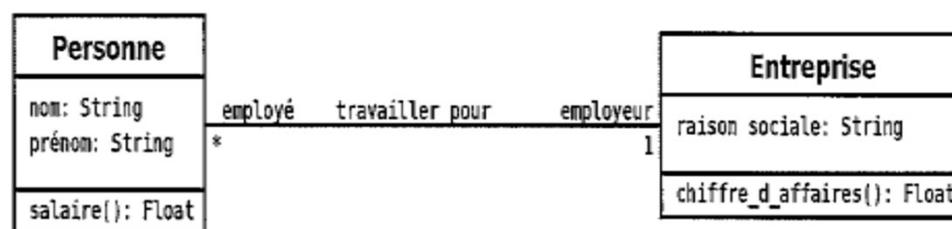


Diagramme de classe

Association entre objets

- ▶ Conséquence sur la programmation de cardinalité multiple sur Personne:
 - Dans première modélisation, Implicitement: attribut Collection de Personne dans Entreprise
 - Dans la seconde modélisation, explicitement: employé: Personne [*] est une collection



UML



Manière dont
c'est programmé



Diagramme de classe

Cardinalités des associations

- ▶ Définition de cardinalité:
indique à combien d'instances de la classe de l'extrémité, une instance de la classe située à l'autre extrémité est liée
- ▶ Possibilité d'indiquer des cardinalités minimales et maximales ⇒ intervalle de la cardinalité
- ▶ Syntaxe UML: (1 par défaut)

<i>Spécification</i>	<i>Cardinalités</i>
0..1	Zéro ou une fois
1	Une et une seule fois
*	De zéro à plusieurs fois
1..*	De une à plusieurs fois
M..N	Entre M et N fois
N	N fois



Diagramme de classe

Cardinalités des associations

► Exemples:

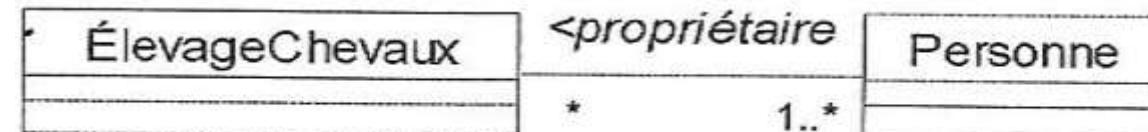
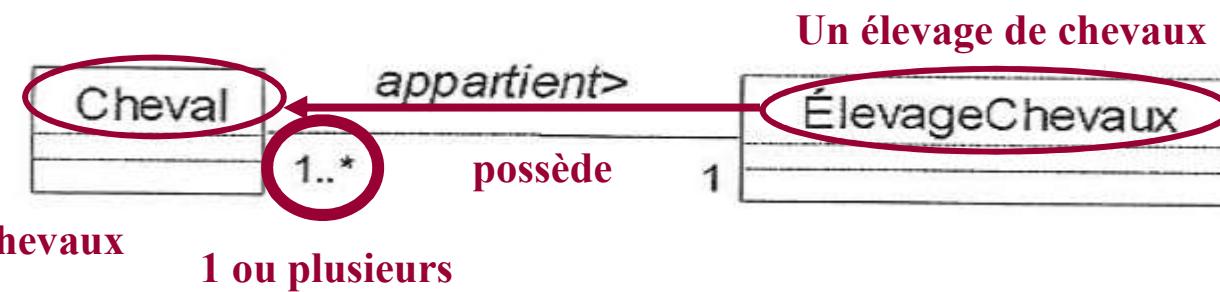
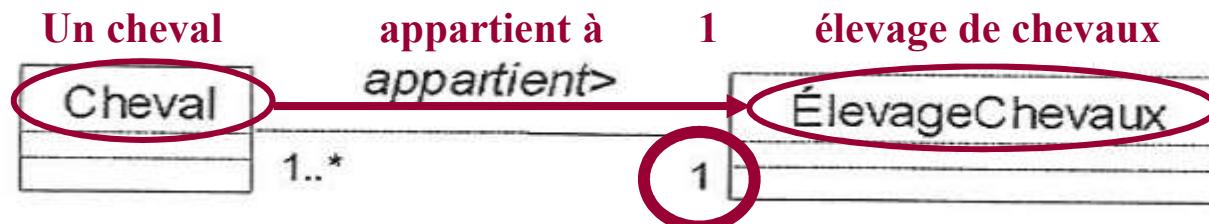


Diagramme de classe

Cardinalités des associations

► Méthode pour déterminer les cardinalités:

- Étudier les différents diagrammes d'objet possibles
- En déduire les cardinalités possibles

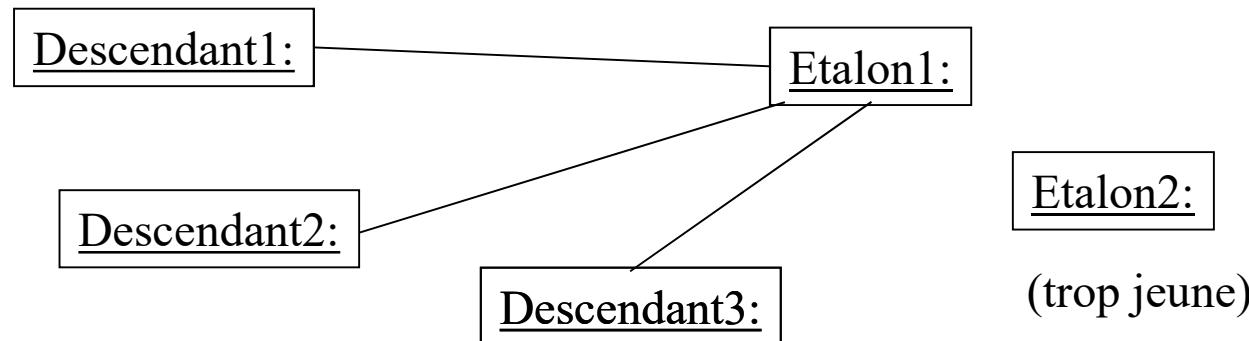


Diagramme de classe

Navigation

- ▶ Par défaut, navigation **bidirectionnelle** des associations
- ▶ Possibilité de déterminer les liens de l'association depuis une instance de chaque classe d'origine
- ▶ Problème: difficulté d'implémentation de la navigation bidirectionnelle
- ▶ Conclusion:

Eviter la navigation bidirectionnelle

Spécifier le seul sens de navigation utile par < ou >



Diagramme de classe

Classes-associations

- ▶ Définition:
Association qui **décrit des liens** (entre instances de classe) portant des informations spécifiques à chaque lien
- ▶ Instances de la classe-association: occurrences de l'association
- ▶ Propriétés:
 - Attributs
 - Opérations
- ▶ Possibilité de relier à d'autres classes par des associations



Diagramme de classe

Classes-associations

► Notation UML:

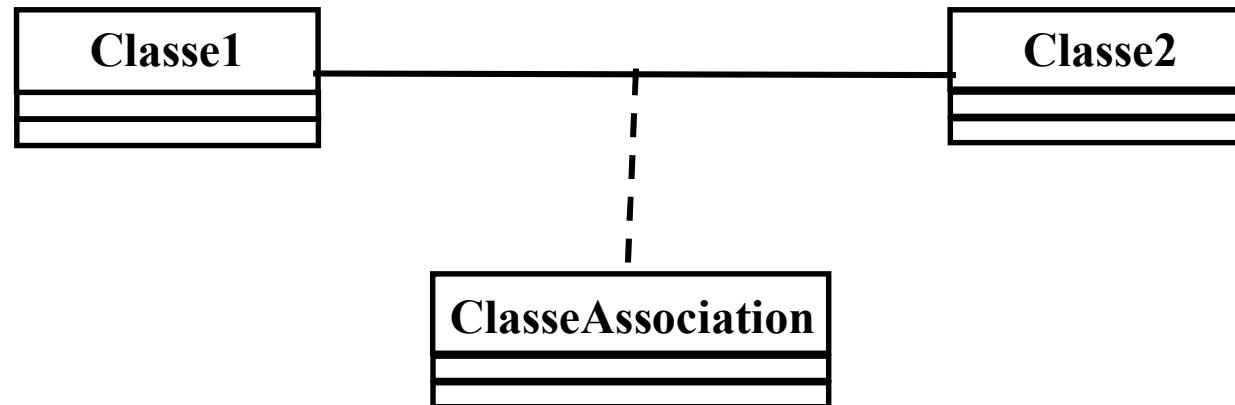


Diagramme de classe

Classes-associations

► Exemple:

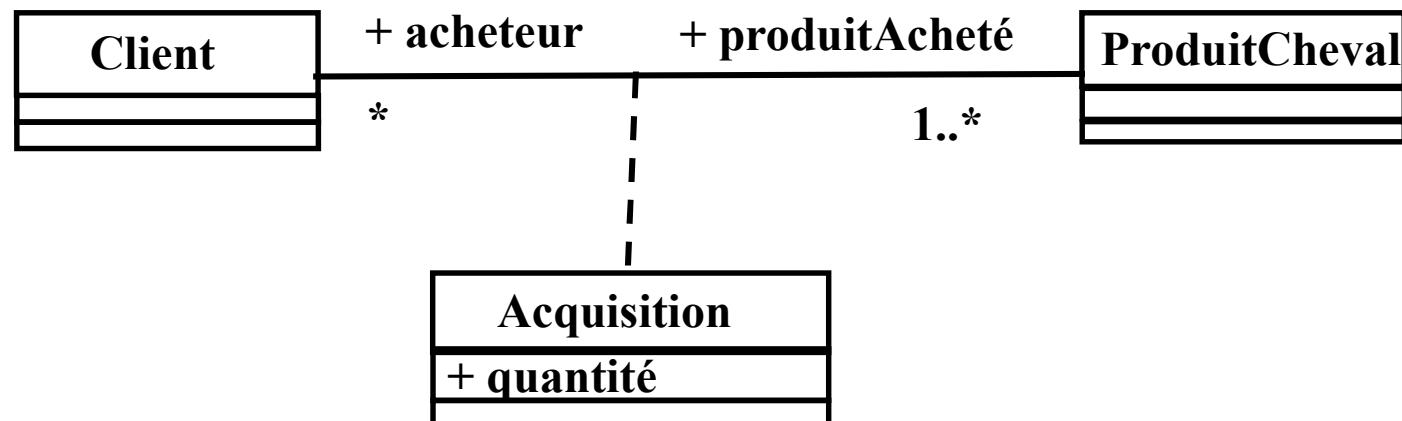


Diagramme de classe

Objets composés

- ▶ Cas d'un objet composé d'autres objets
- ▶ Définition: **association liant l'objet à ses composés**
- ▶ Définie au niveau des classes (liens entre instances de classes)
- ▶ Composants: objets formant l'objet composé
- ▶ Deux formes de composition:
 - composition faible ou **agrégation**: possibilité pour les composants d'être partagés entre plusieurs objets complexes
 - **composition forte**: la destruction de l'objet composé entraîne la destruction de ses composants
- ▶ Notation UML:
 - agrégation
 - composition forte

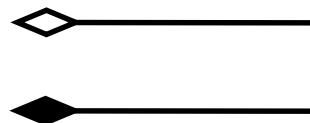


Diagramme de classe

Objets composés

► **Composition forte** ou composition:

- Les composants sont une partie de l'objet composé
- 3 conditions:
 - > Impossible que les composants soient partagés entre plusieurs objets composés
 - > Cardinalité maximale au niveau de l'objet composé: obligatoirement 1
 - > Suppression de l'objet composé \Rightarrow suppression de ces composants
- Notation UML:

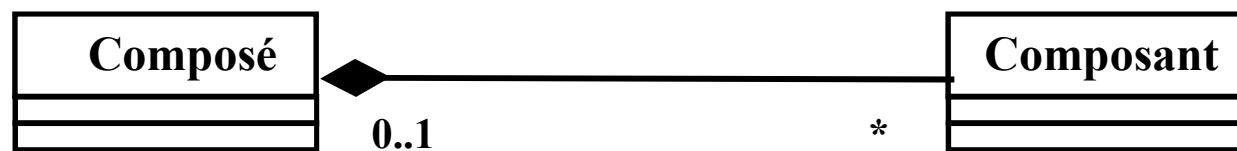


Diagramme de classe

Objets composés

- Exemple de composition forte:

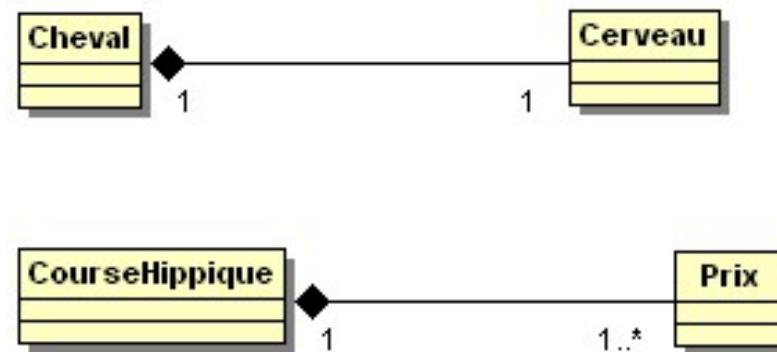


Diagramme de classe

Objets composés

► Composition faible ou agrégation :

- Possibilité que les composants soient partagés entre plusieurs objets composés:
 - > de la même association d'agrégation
 - > ou de plusieurs associations distinctes d'agrégation
- Suppression de l'objet composé ~~à~~ suppression de ces composants
- association moins contraignante que la composition: au cours de la modélisation une agrégation peut se révéler être une composition

plus de contrainte ⇒ plus de sens ⇒ modèle plus riche

- Notation UML:

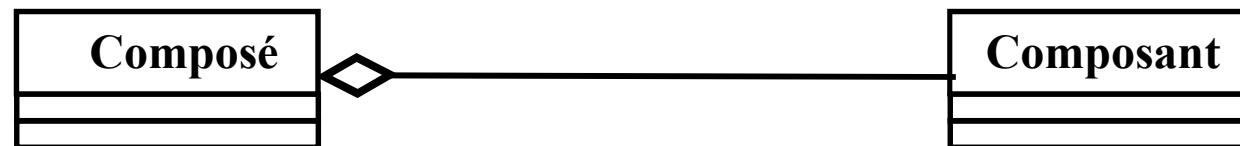


Diagramme de classe

Objets composés

► Exemple d'agrégation:

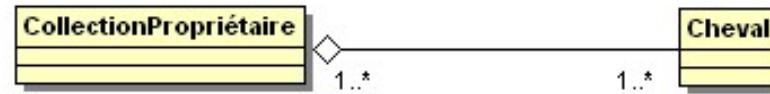
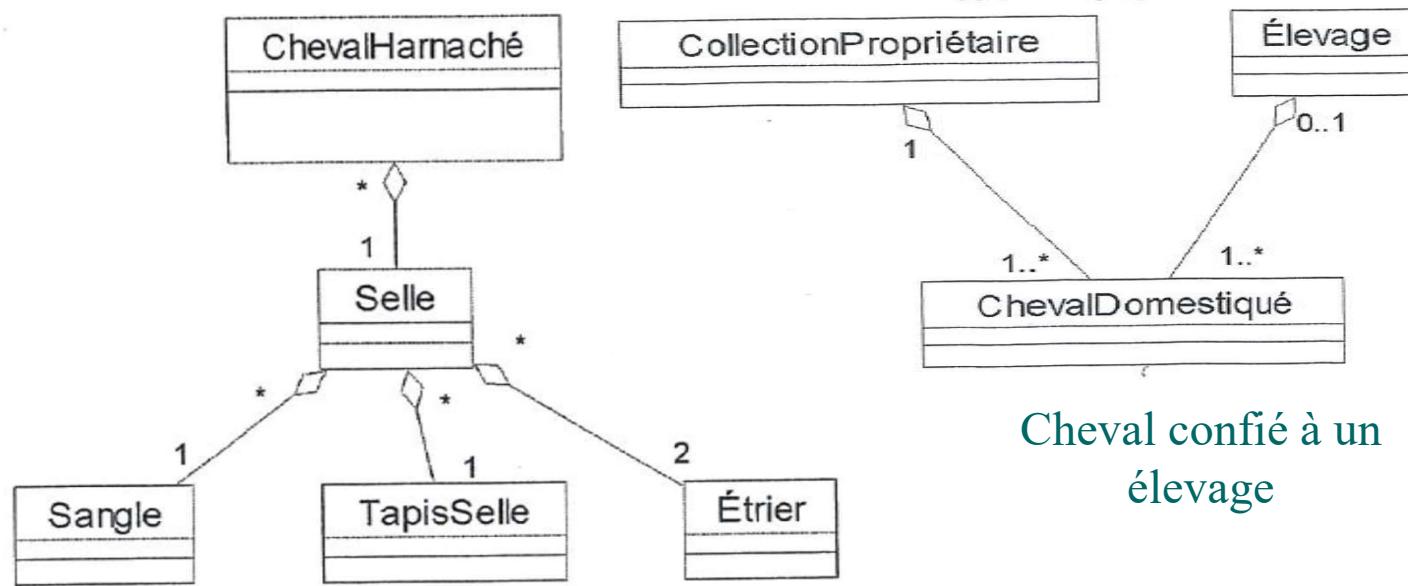


Diagramme de classe

Objets composés

► Comparaison entre composition et agrégation :

	Agrégation	Composition
Représentation		
Partage des composants par plusieurs associations	oui	non
Destruction des composants lors de la destruction du composé	non	oui
Cardinalité au niveau du composé	quelconque	0..1 ou 1



Diagramme de classe

Objets composés

► Exemples:

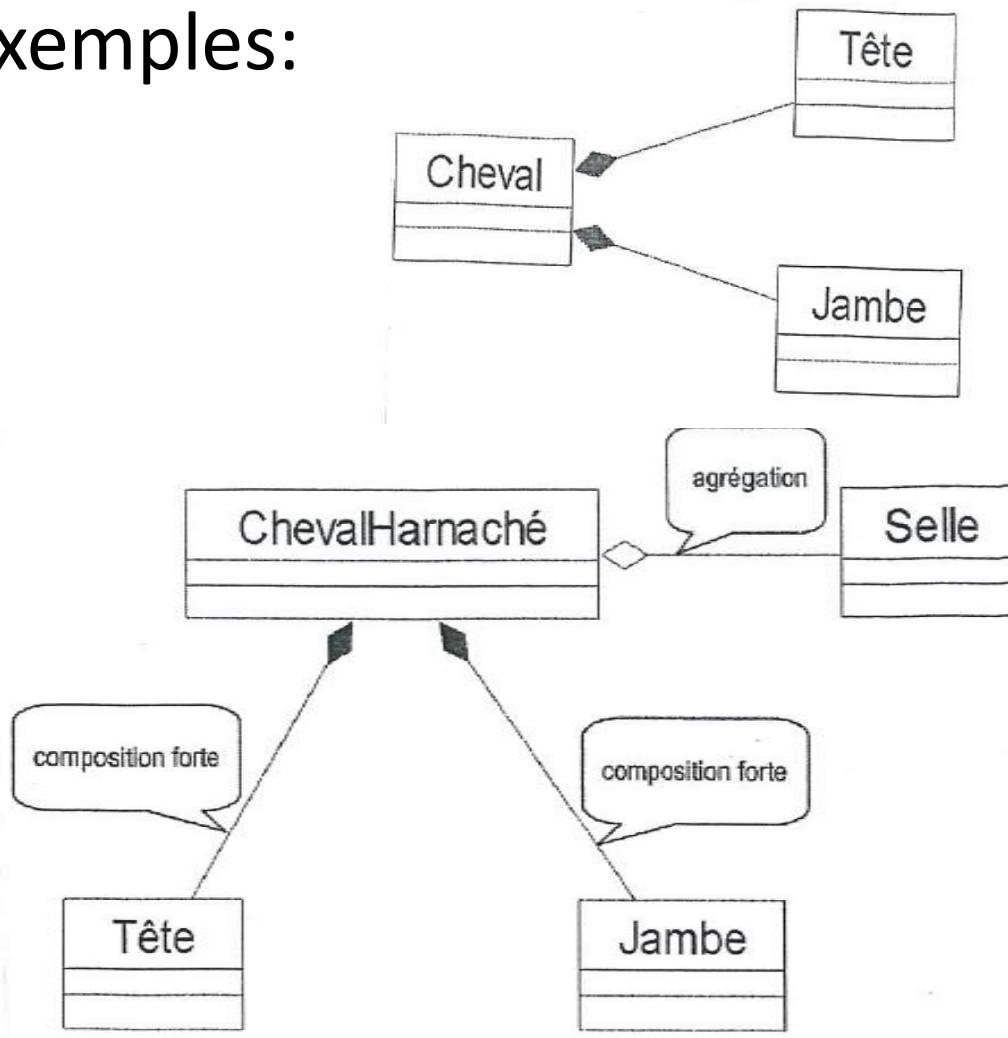


Diagramme de classe

Relation de généralisation/spécialisation entre les classes

- ▶ Spécialisation d'une classe: **sous-classe** (sous-ensemble) d'une classe
 - classe plus spécifique qu'une autre
 - toutes les instances peuvent être considérées comme également instances de cette autre classe dite sur-classe

Une sous-classe est un « cas particulier » de la sur-classe.

- ▶ Généralisation d'une classe:
 - **Surclasse** d'une classe
 - relation inverse de la spécialisation
- ▶ Possibilité de hiérarchisation des classes: plusieurs niveaux de spécialisation



Diagramme de classe

Relation de généralisation/spécialisation entre les classes

► Notation UML:

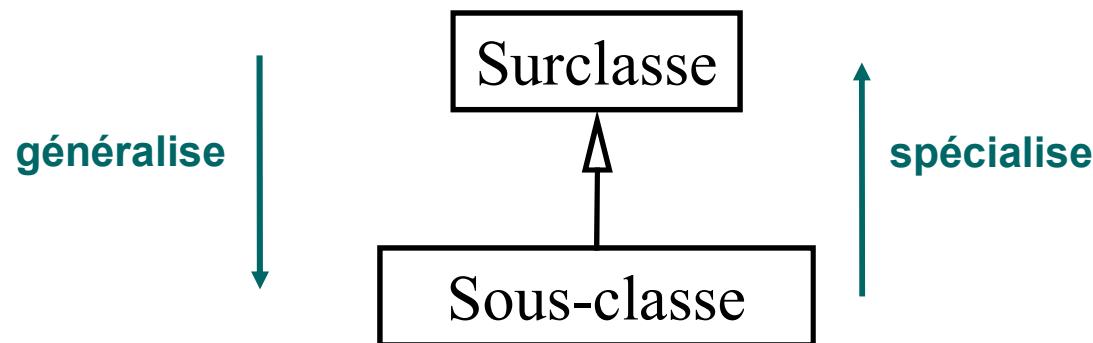


Diagramme de classe

Relation de généralisation/specialisation entre les classes

► Exemple:

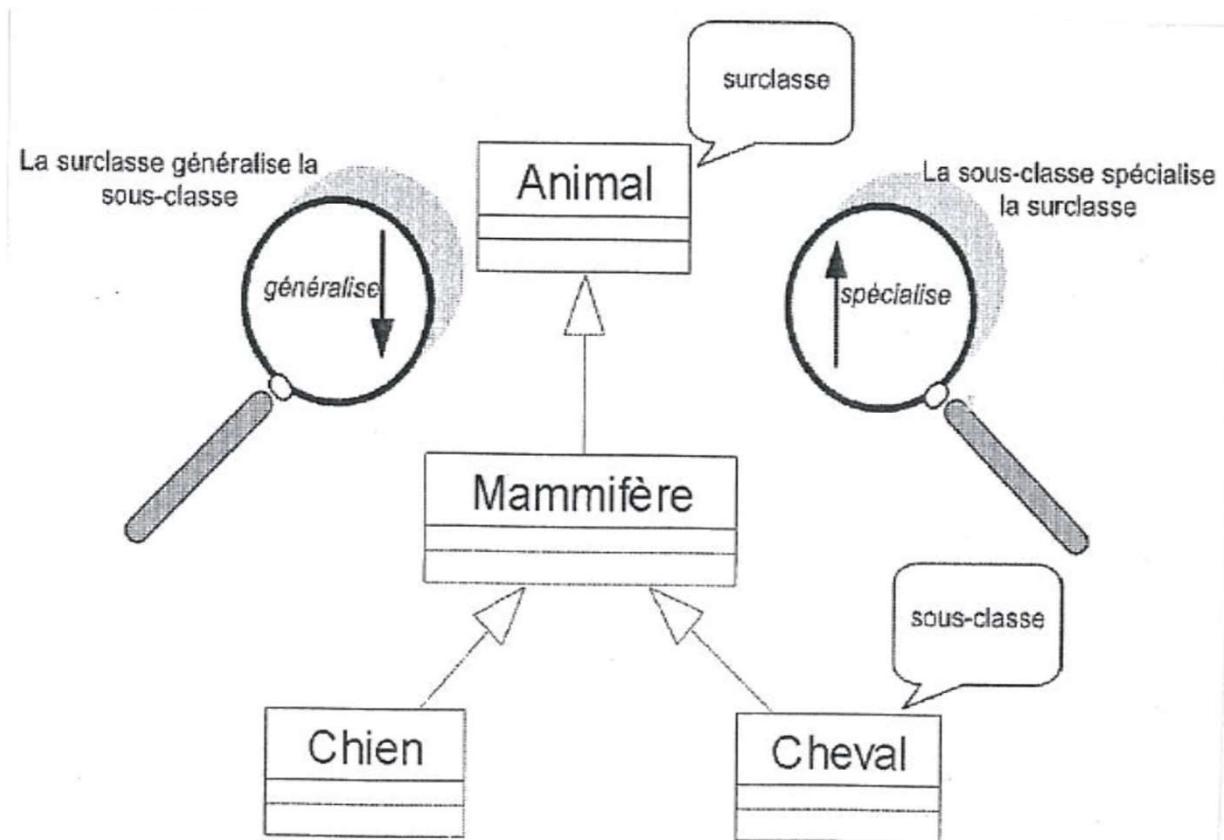


Diagramme de classe

Relation de généralisation/spécialisation entre les classes

► Exemple:

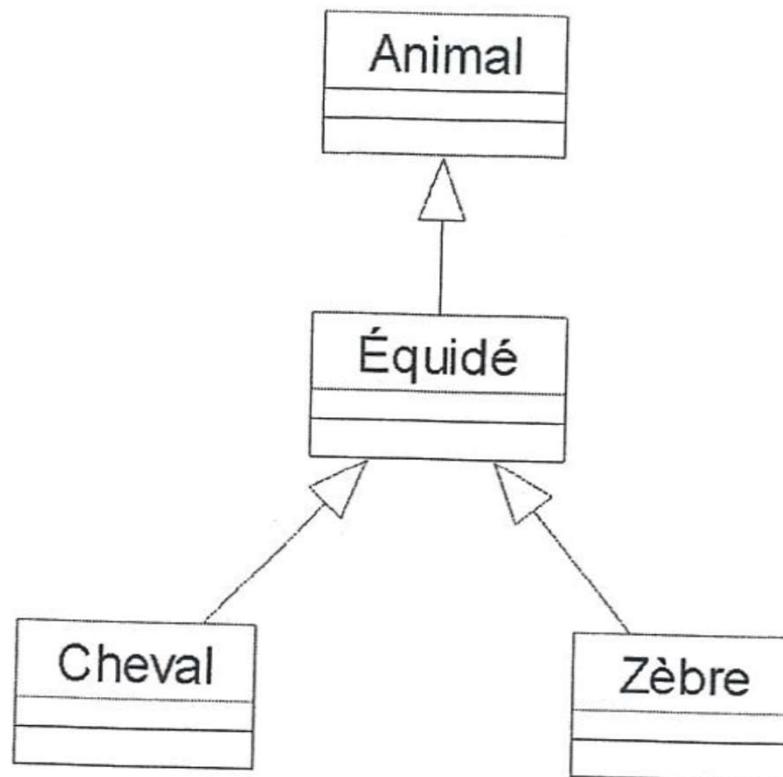


Diagramme de classe

Relation de généralisation/spécialisation entre les classes

► Héritage:

- propriété qui fait bénéficier à une **sous-classe** de la structure et du comportement de sa **surclasse**
- une classe hérite des attributs et des méthodes de ses surclasses pour en faire bénéficier ses instances
- une classe possède:
 - > les attributs et les méthodes introduits au niveau de la classe: **attributs et méthodes propres**
 - > les attributs et les méthodes définis dans la ou les surclasses: **attributs et méthodes hérités**

► Justification:

les instances d'une classe peuvent être considérées comme instances de sa ou de ses surclasses

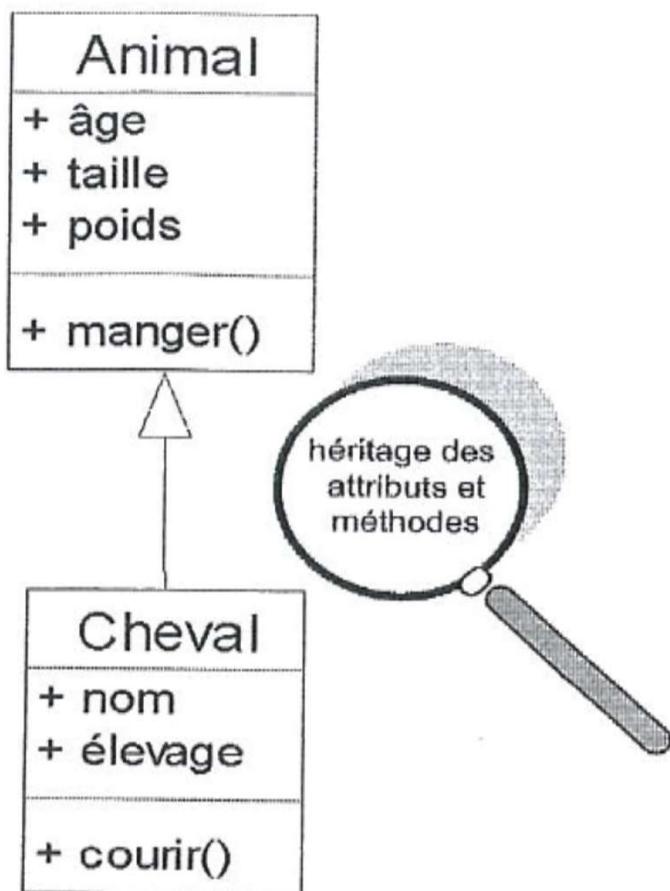
- conséquence de la spécialisation
- sous-classe = sous-ensemble de la surclasse
- instance de la sous-classe = instance de la surclasse



Diagramme de classe

Relation de généralisation/spécialisation entre les classes

► Exemple:



La classe
Cheval hérite
de la classe
Animal

Attributs de Cheval:

Nom }
Elevage } Attributs de sa
 classe

Age }
Taille } Attributs de sa
Poids } sur-classe



Diagramme de classe

Relation de généralisation/spécialisation entre les classes

► Exemple:

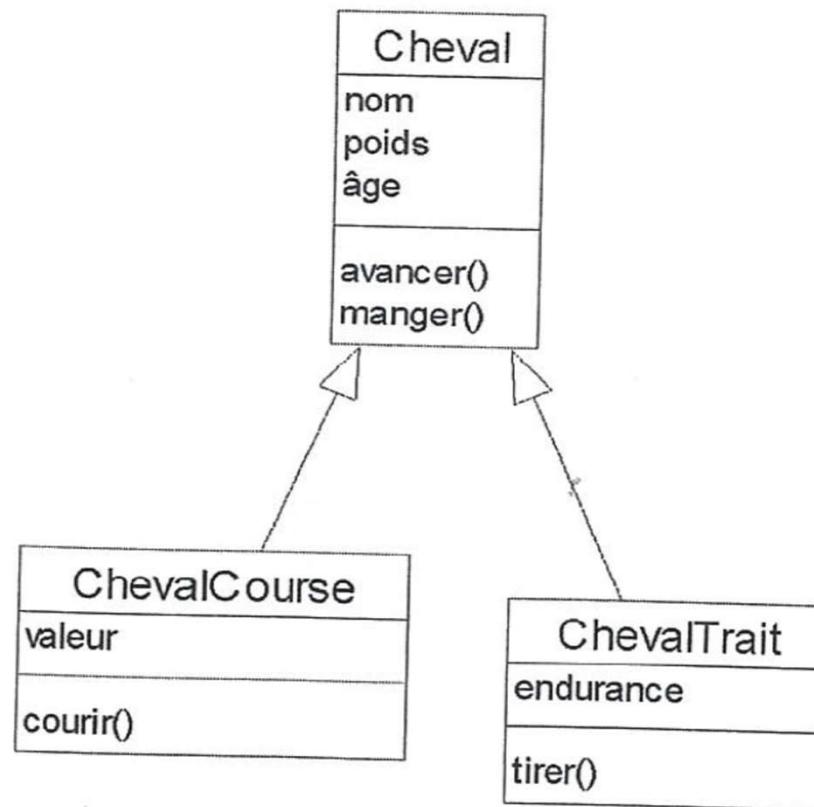


Diagramme de classe

Relation de généralisation/spécialisation entre les classes

- Héritage du point de vue de la sous-classe:

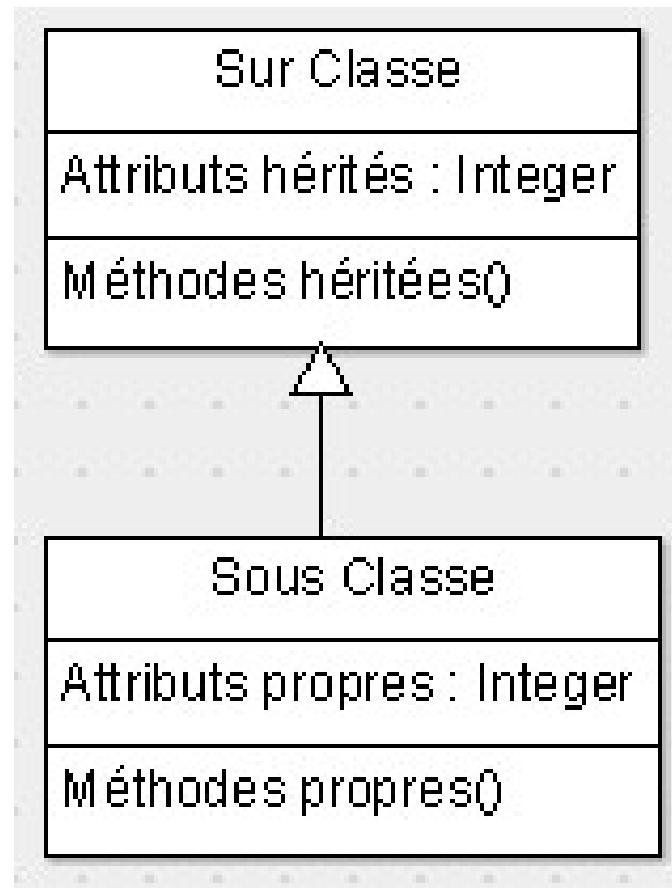


Diagramme de classe

Relation de généralisation/specialisation entre les classes

- ▶ Factorisation de l'extrémité d'une association dans une surclasse
- ▶ Intérêt: simplification du diagramme

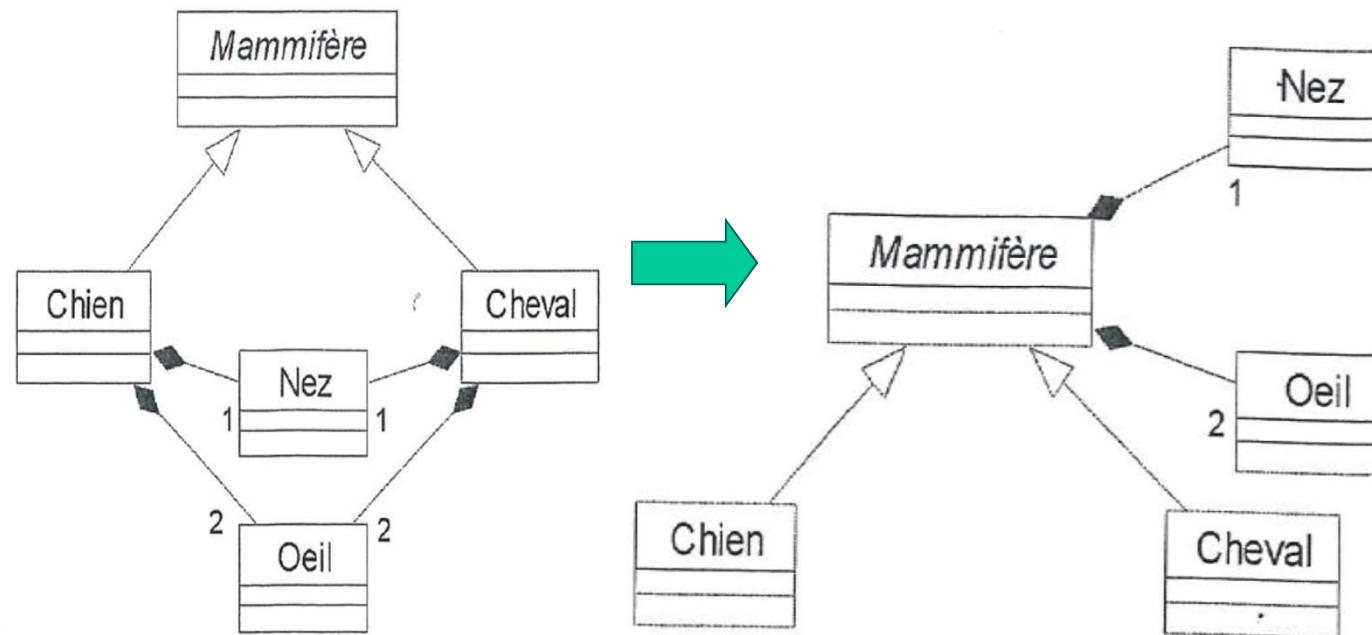


Diagramme de classe

Différents types d'association

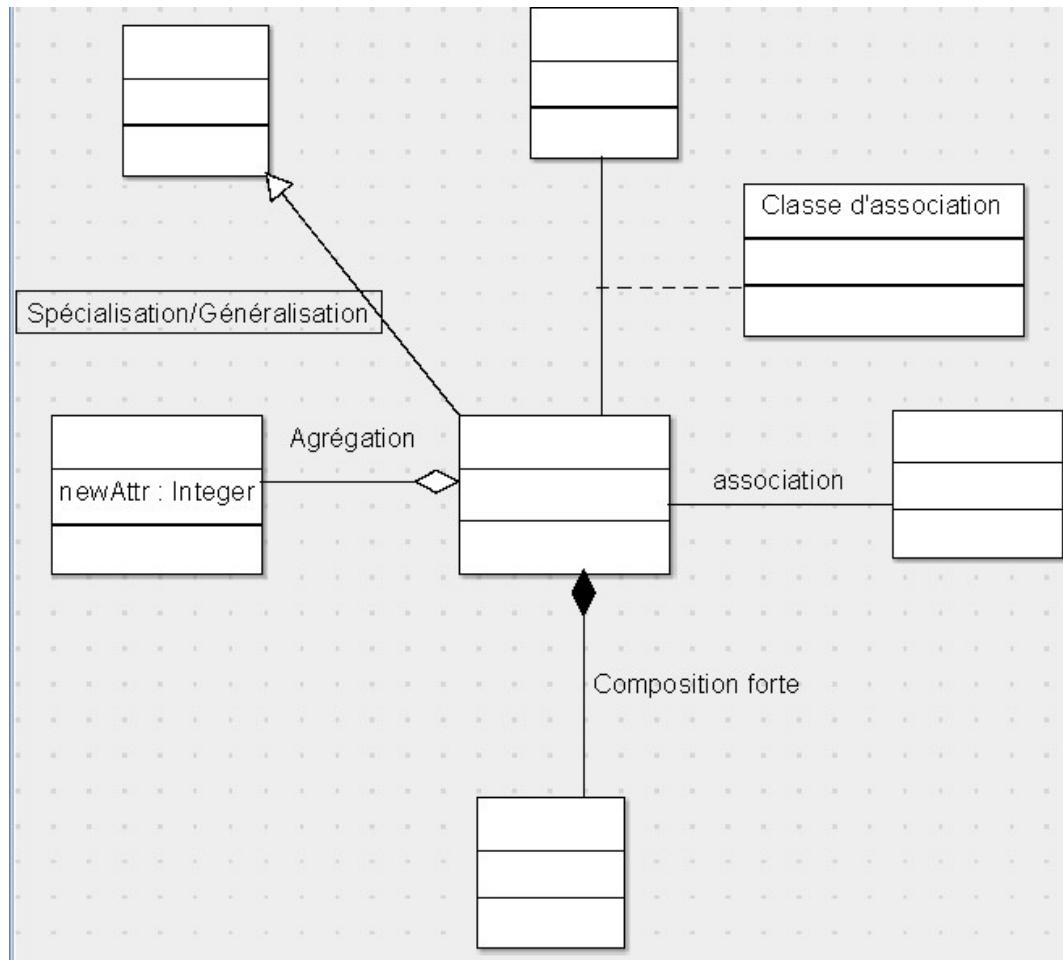


Diagramme de classe

Conclusion

- ▶ Diagramme des cas d'utilisation:
 - Limites du système
 - Objectifs des utilisateurs
- ▶ Scénarios: interactions nécessaires pour la réalisation de ces objectifs
- ▶ Selon la complexité du système:
 - simples: réalisation du diagramme de classe puis programmation
 - complexe:
 - > nécessité d'analyser le système pour réaliser le diagramme de classe
 - > première étape: identifier les objets à partir des scénarios à l'aide des diagrammes de séquences, d'objets...



FIN DU SECOND COURS



Diagramme de classe

Attributs de classe

► Attributs de classe:

- attribut d'un objet: chaque instance de classe contient une valeur spécifique pour chacun de ses attributs
- attribut de classe:
 - > valeur commune pour l'ensemble des objets de la classe
 - > même nom, même type, même valeur par défaut
- Accessible même en l'absence d'instance de la classe
- Valeur par défaut recommandée
- Notation UML: souligné

nomAttribut : typeAttribut = valeurDéfaut



Diagramme de classe

Attributs de classe

► Exemple d'attributs de classe:

QuartierViandeChevaline
+ poids : Integer
+ qualité : Integer
prixSansTVA : Currency
<u># tauxTVA : Integer = 55</u>
+ Calculer prixAvecTVA() : Currency

Attribut de classe



Diagramme de classe

Méthodes de classe

► Méthodes de classe:

- liées à la classe
- appel d'une méthode de classe: envoi d'un message à la classe elle-même et non à l'une de ses instances
- **ne manipule que des attributs de classe** et ses propres paramètres
- ne peut accéder aux instances de la classe ni à leurs attributs
- notation UML: souligné

+ nomMéthode (param : typeParam = valeurDéfaut) : typeRetour

► Autre vocabulaire: attributs et méthodes statiques (C++, Java)



Diagramme de classe

Méthodes de classe

► Exemple de méthodes de classe:

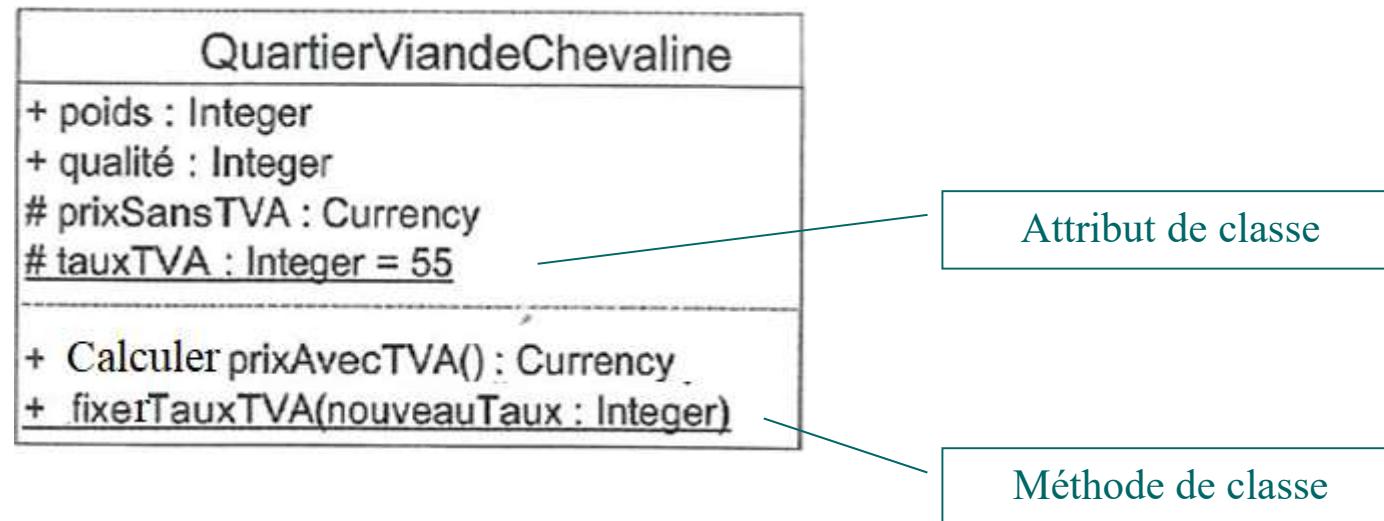


Diagramme de classe

Attributs calculés

► Attributs calculés:

- valeur donnée par une fonction basée sur la valeur d'autres attributs
- notation UML:
 - > nom précédé de /
 - > contrainte entre accolade: moyen de le calculer

/ nomAttribut : typeAttribut { nomAttribut = formule }



Diagramme de classe

Attributs calculés

► Exemple d'attribut calculé:

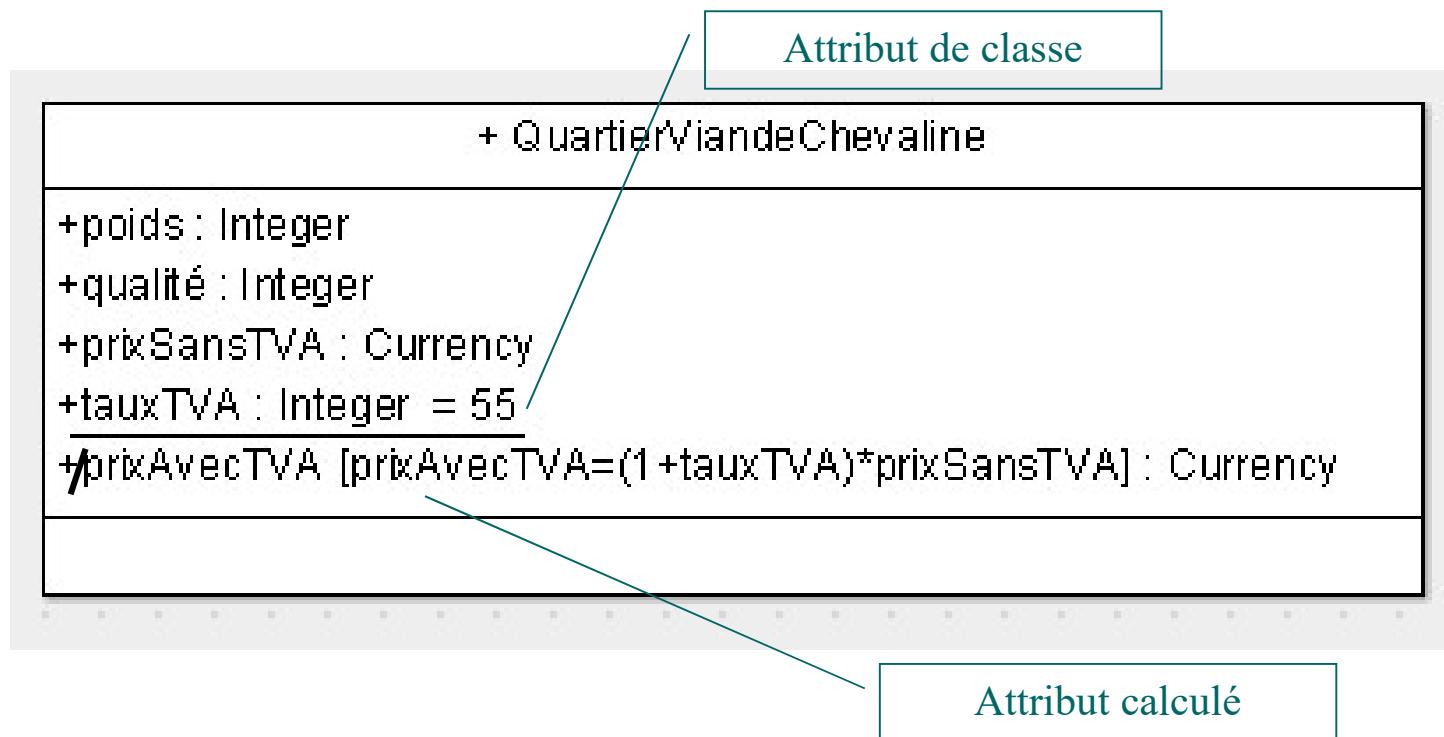


Diagramme de classe

Classes concrètes et abstraites

► Classe concrète:

- possède des instances
- constitue un modèle complet d'objet:
 - > définition de tous les attributs
 - > description complète de toutes les méthodes

► Classe abstraite:

- ne possède pas directement des instances
- ne fournit pas un modèle complet
- a pour vocation de posséder des sous-classes concrètes
- Intérêt : factoriser des attributs et méthodes communs à ses sous-classes
- classe abstraite pure: interface (cf suite du cours)



Diagramme de classe

Classes concrètes et abstraites

► Notation UML:

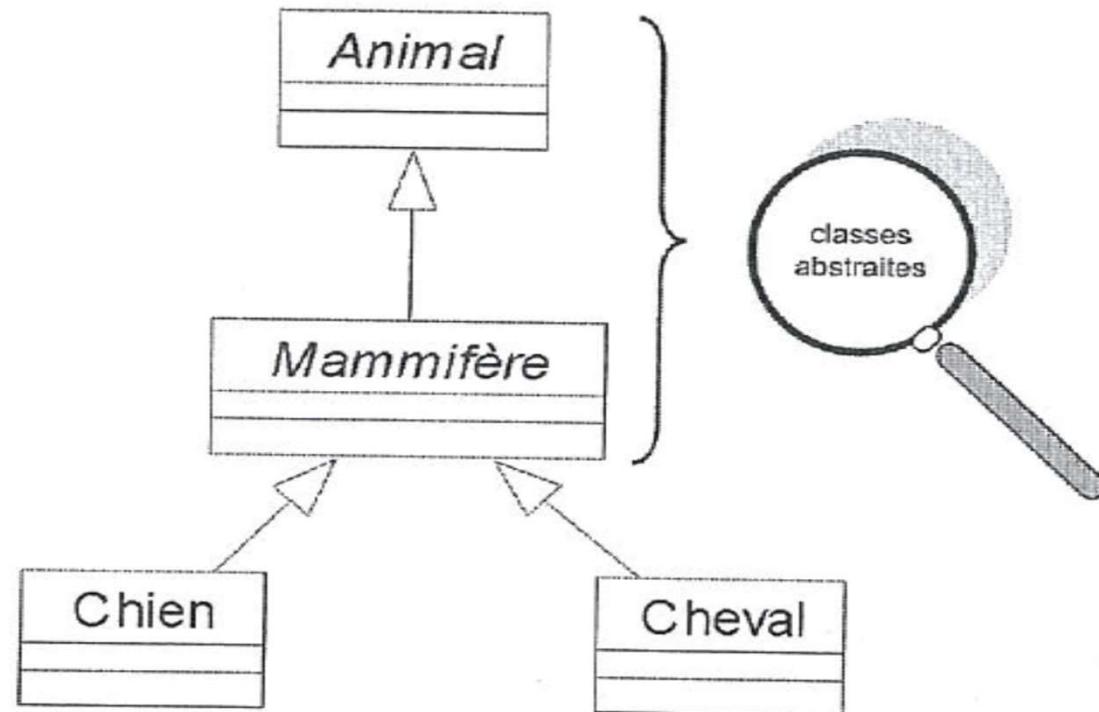
- classes concrètes caractères droits
- classes abstraites *caractères italiques*



Diagramme de classe

Classes concrètes et abstraites

► Exemple:



Tout mammifère est soit un chien, soit un cheval, soit un chat...
donc *Animal* est une classe abstraite.

On ne caractérise pas entièrement un être vivant en disant
uniquement qu'il est un mammifère: il faut préciser.



Diagramme de classe

Classes concrètes et abstraites

► Est abstraite:

- Toute classe possédant au moins une méthode abstraite (modèle incomplet: il manque au moins le code de cette méthode)
- Toute classe dont une sur-classe parent contient une méthode abstraite non redéfinie à son niveau

► Méthode abstraite:

- méthode introduite dans une classe avec sa seule signature (nom, paramètres avec nom et type, type du résultat) et sans code
- Déclaration connue mais pas la définition

► Souvent factorisation de méthodes communes aux sous-classes se traduit par la seule factorisation de la signature



Diagramme de classe

Classes concrètes et abstraites

► Exemple:

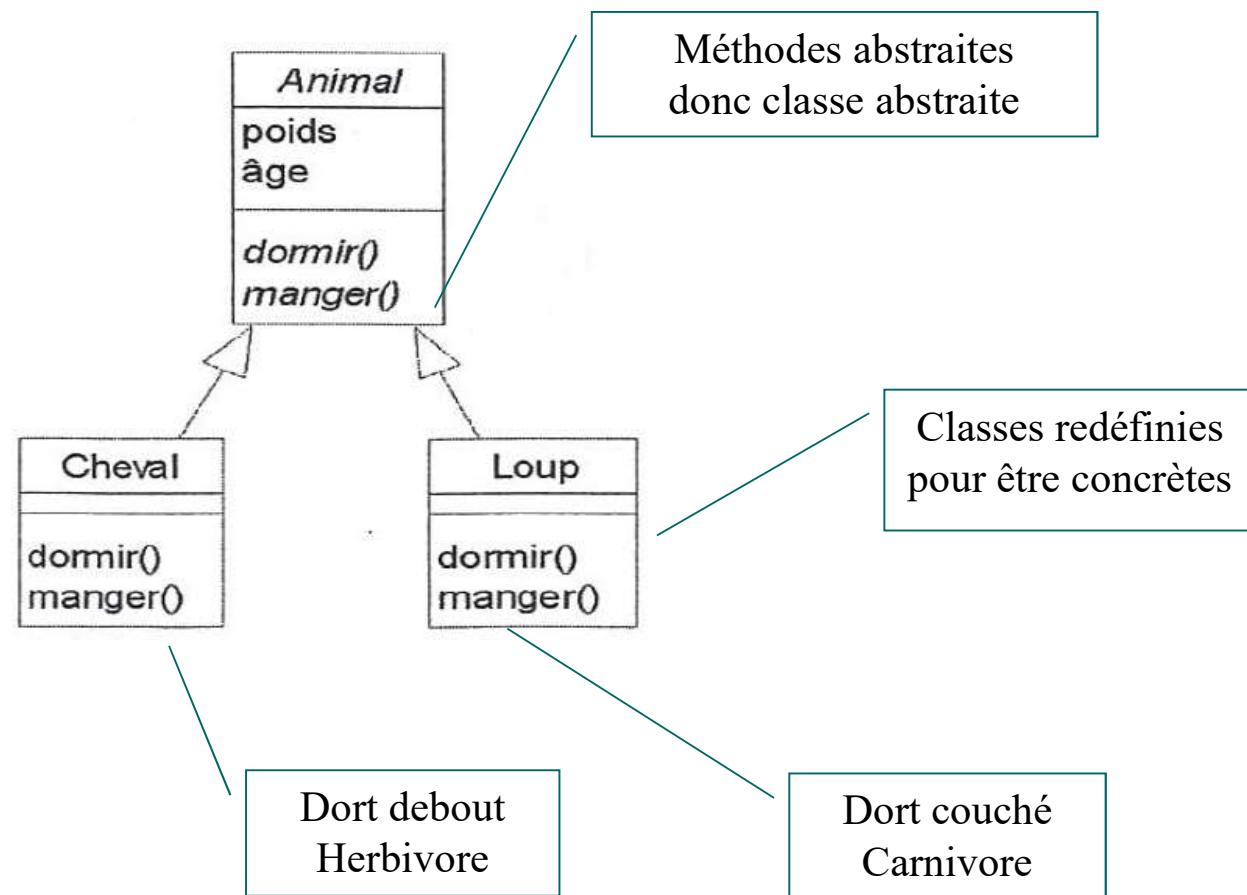


Diagramme de classe

Classes concrètes et abstraites

► *Attribut abstrait:*

- Pas dans le standard UML mais utilisé pour définir des termes (méthode Catalysis,...) et non stocker des données
- En java: uniquement au niveau de la spécification



Diagramme de classe

Polymorphisme

- ▶ Cas d'une classe (souvent abstraite) avec des sous-classes distinctes: ensemble d'objets différents car instances de sous-classes distinctes

- ▶ Définition:

lors de l'appel d'une **méthode** de la classe abstraite ayant même nom pour tous, les objets ont des **comportements différents**.

- ▶ Résumé:

- même nom de méthode
- comportement différent



Diagramme de classe

Polymorphisme

► Exemple:

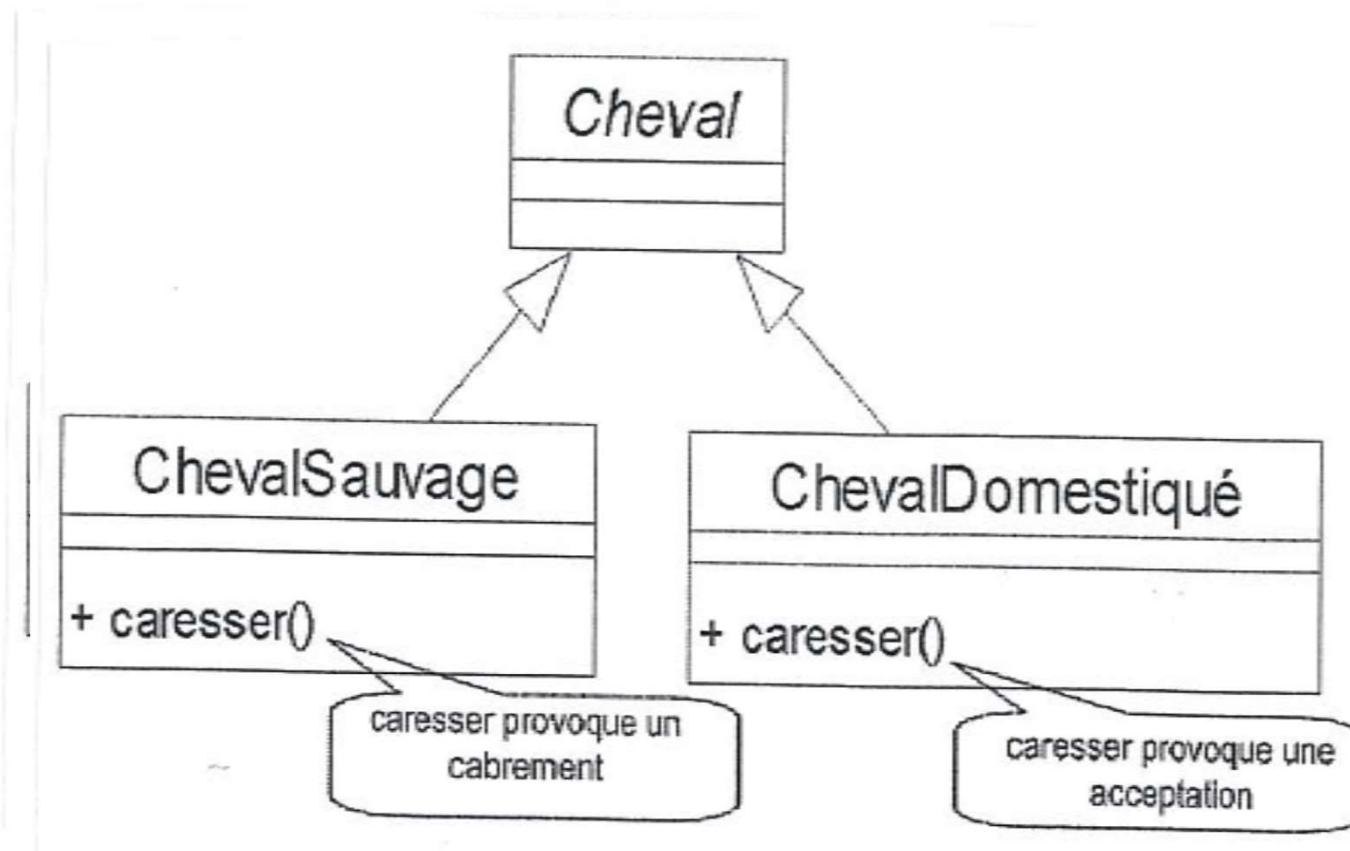


Diagramme de classe

Remarques méthodologiques

- ▶ Classes présentes dans le diagramme de classe du système:
 - celles qui constituent le système.
 - jamais de classe système
- ▶ Acteurs:
 - système ne garde pas trace de leur identité, pas d'actions subies,: pas de classe représentant les acteurs dans le diagramme
 - Système permet de gérer leur ajout, actions subies....: classe de l'acteur dans le diagramme
- ▶ Diagramme de classe de contexte (et non plus du système):
 - représentation du système et des acteurs
 - Seul cas où le système apparaît en tant que classe
 - Plutôt utiliser un diagramme des cas d'utilisation



Diagramme de classe

Remarques méthodologiques

► Classe:

- ne correspond jamais à une action d'un acteur sur le système
- jamais nommée par un verbe à l'infinitif (ou quasiment jamais)

► Méthode:

- ce qui arrive à l'objet de la classe (philosophie objet)
- et non pas ce qu'il fait sur un autre objet (procédural)

► Une classe sans attribut ni méthode:

- Inutile au modèle?
- À transformer en attribut?

► Définition des cardinalités des associations: utiliser des diagrammes des objets



Diagramme de classe

Méthodologie

► Méthode synthétique:

- Définir les classes à partir de l'identification des acteurs potentiels et des concepts clés réalisée lors de la réalisation du diagramme des cas d'utilisation.
- Représenter le diagramme de classe du système en indiquant uniquement le nom des classes et des associations et le sens de navigation des associations.
- Pour chaque classe, définir ses attributs et ses méthodes
- Pour chaque association, définir sa cardinalité si besoin grâce à un diagramme des objets.



Diagramme de classe

Méthodologie

► Méthode analytique:

- Choisir une classe parmi les concepts clés du système identifiés lors de la réalisation du diagramme des cas d'utilisation.
- Définir ses attributs et ses méthodes
- Définir les classes avec lesquelles elle est en association
- Répéter ce processus pour une des classes identifiées en choisissant un concept important pour le système

► Méthodes mixtes entre synthétiques et analytiques



Interface

► Définition d'Interface:

- **classe totalement abstraite**
- classe sans attribut et dont toutes les méthodes sont abstraites et publiques
- ne contient aucun élément d'implantation des méthodes

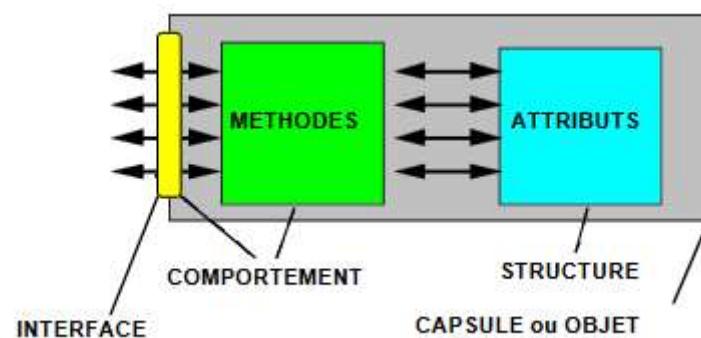
► Intérêt des interfaces:

- Définition d'un ensemble de méthodes qu'un client peut obtenir d'un objet
- Découplage entre la définition de la méthode (signature) et son implémentation



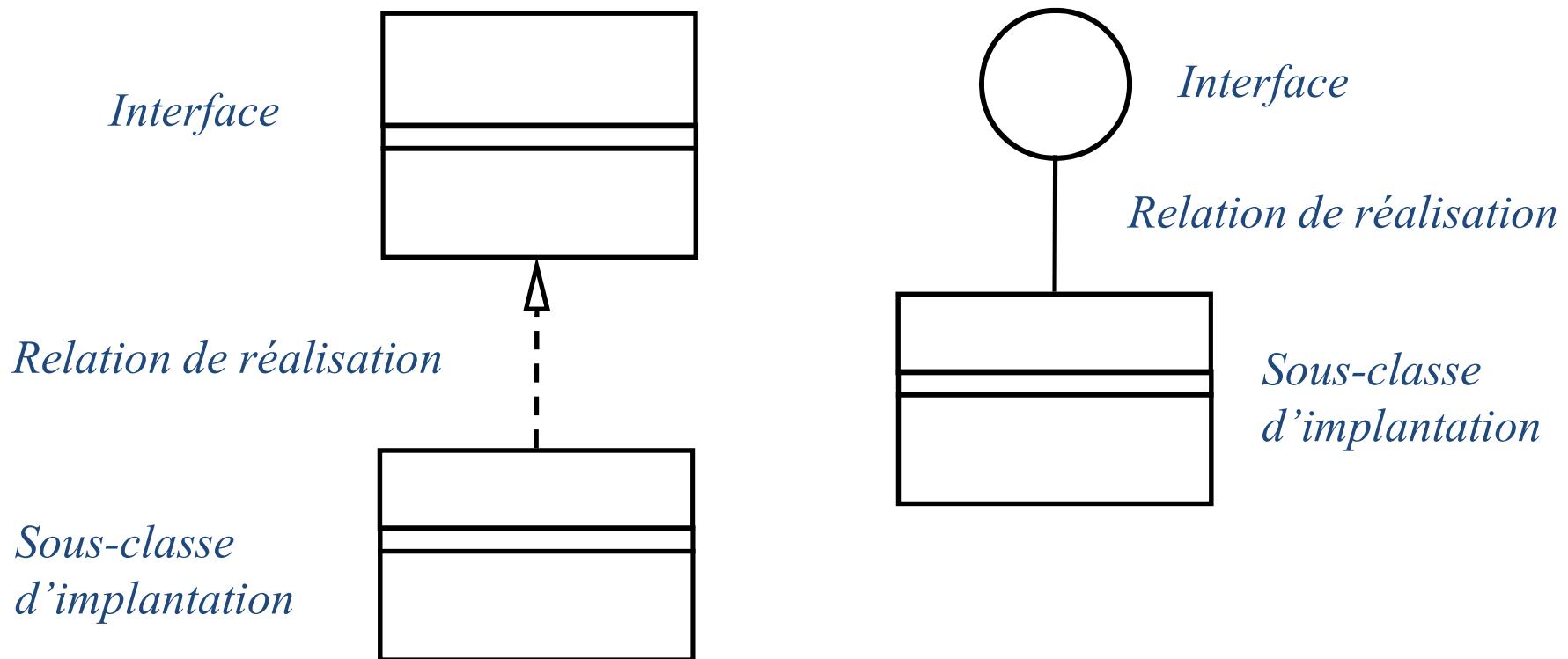
Interface

- ▶ Implantation des méthodes: réalisée par une ou plusieurs classes concrètes sous-classes de l'interface
- ▶ **relation de réalisation** = relation de spécialisation entre l'interface et une sous-classe d'implantation
- ▶ Interface: vue totale ou partielle des services offerts par une classe



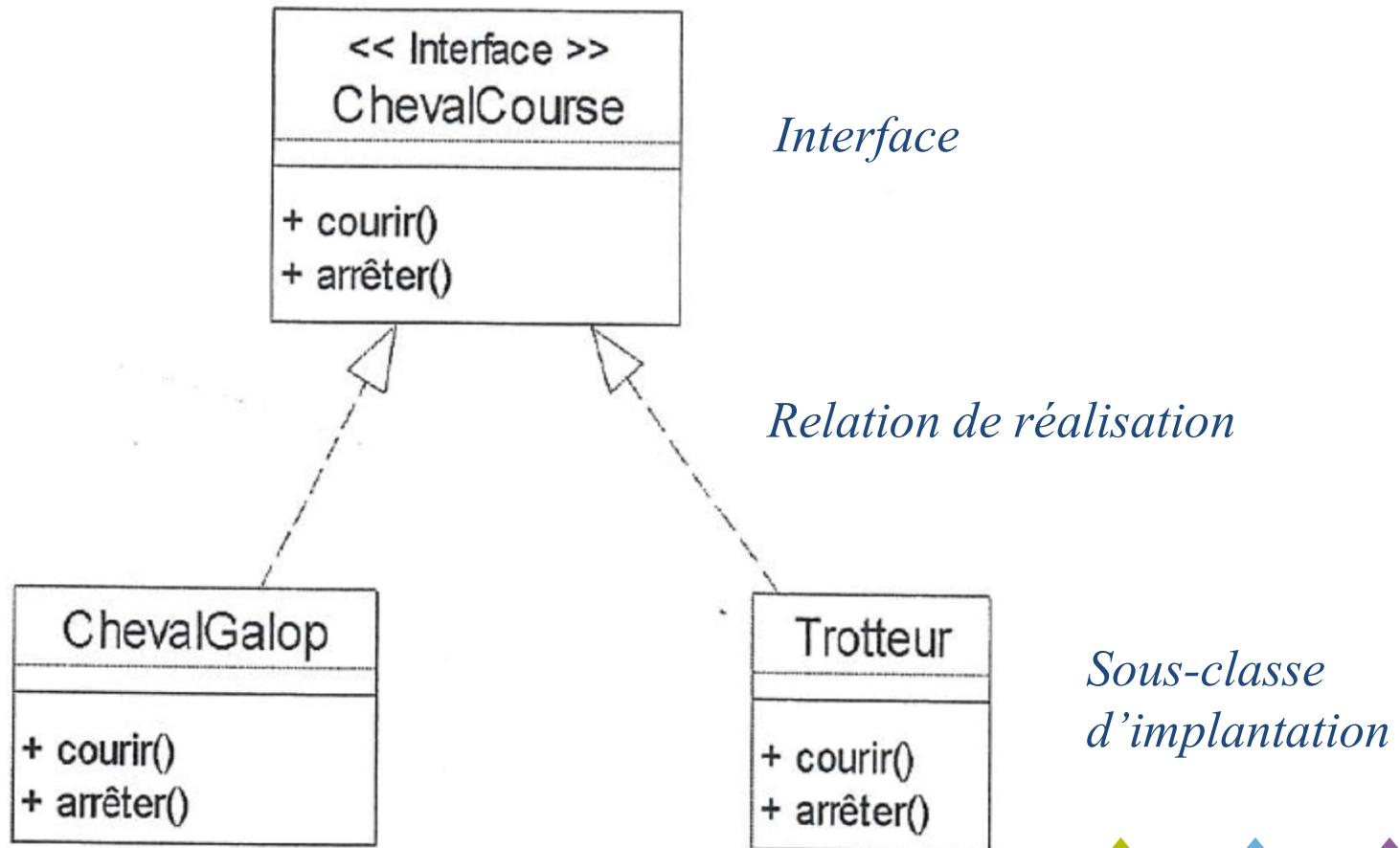
Interface

► Notation UML:



Interface

► Exemple:



Interface

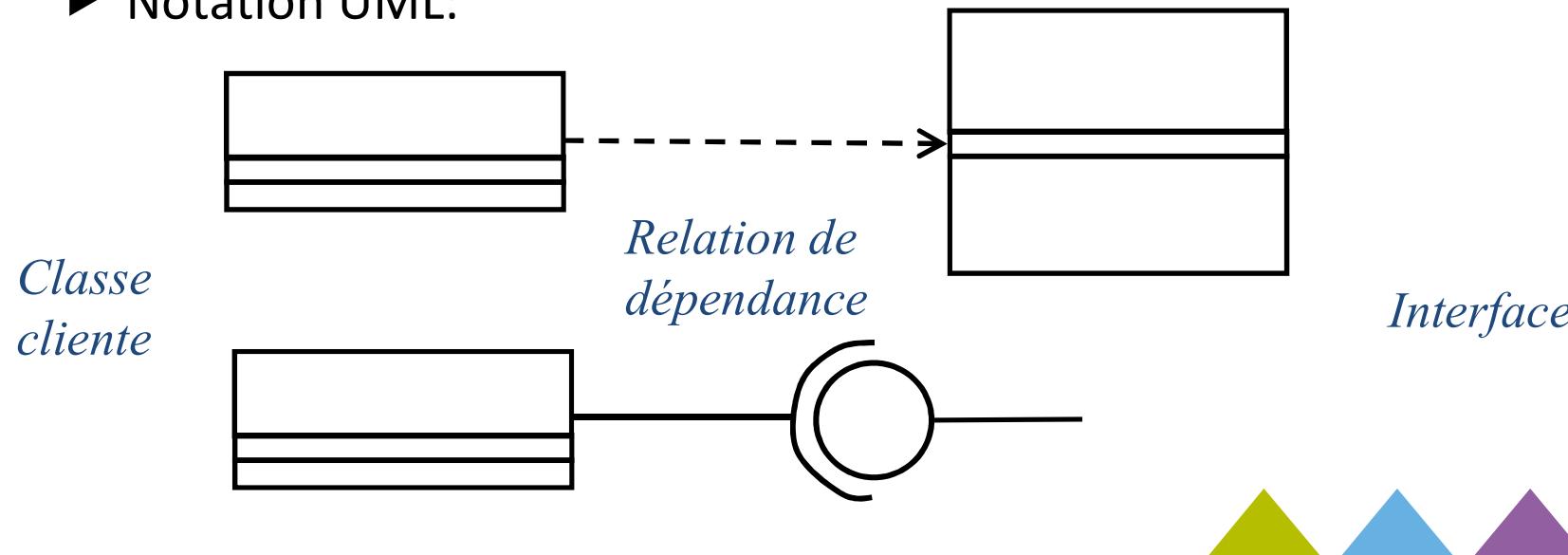
- une même classe peut réaliser plusieurs interfaces:
 - cas particulier de l'héritage multiple
 - pas de conflit:
 - > classe de réalisation (d'implantation) n'hérite que des signatures de méthodes
 - > si signatures identiques: implantée par une seule méthode dans la classe commune de réalisation



Interface

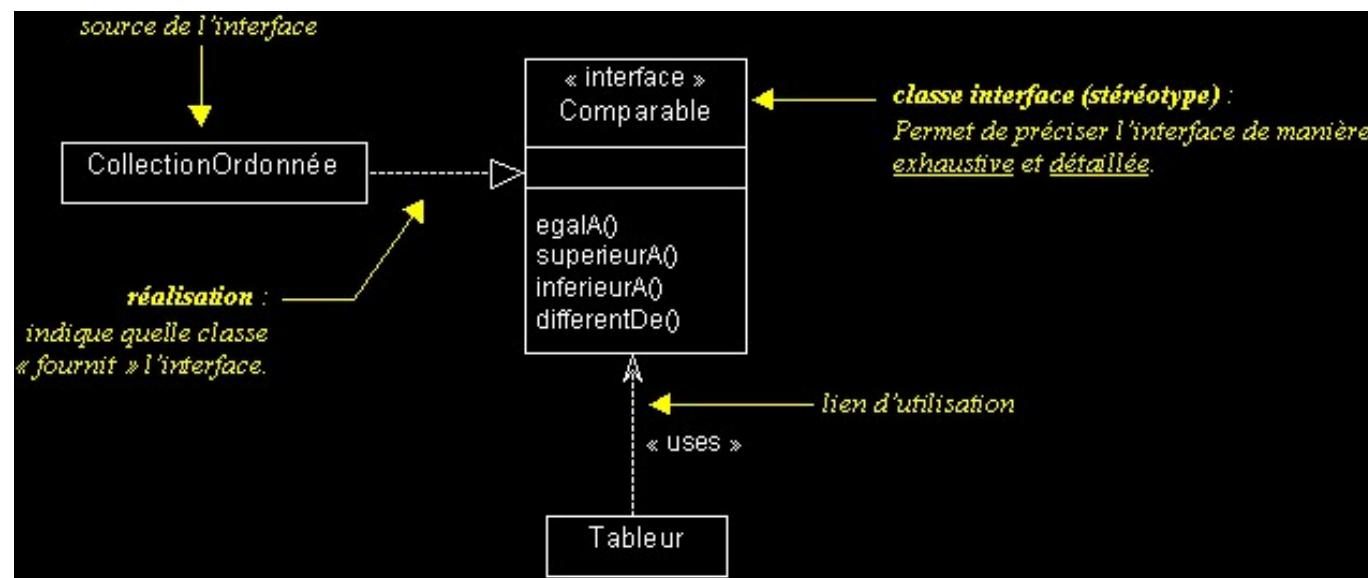
- ▶ Interface peut servir de type au sein d'une classe:
 - Interface = attribut, paramètre ou variable locale de l'une des méthodes
 - la classe est la **cliente** de l'interface: elle utilise les services fournis par l'interface
 - **relation de dépendance** entre l'interface et la classe (possible entre deux classes)

- ▶ Notation UML:



Interface

- Exemple de réalisation et relation de dépendance :



Interface

- ▶ Encapsulation: Fait de masquer des attributs et des méthodes de l'objet vis-à-vis des autres objets
- ▶ Objectif de l'encapsulation: découpler une interface (abstraite) de l'implémentation sous-jacente (concrète).
- ▶ Raison de l'encapsulation: Le modèle (abstraction) change moins vite que le code (concret).



Interface

- ▶ Nécessité d'encapsuler le comportement et pas seulement les propriétés: méthode et attributs
- ▶ Utilité: Garantir intégrité des données: données manipulables uniquement par les méthodes choisies par le concepteur
- ▶ Interface: description des méthodes et propriétés visibles
- ▶ Implantation: cachée par interface



Interface

- ▶ Méthode de conception:
 - Notion d'Interface: utilisée lors de la phase de modélisation
 - Notion d'encapsulation: utilisée lors de la phase de conception et programmation pour implanter l'interface
- ▶ Règles d'utilisation:
 - Début de la phase d'analyse:
 - > Ne pas se poser la question
 - > Tout avec encapsulation « protégé »
 - Phase de conception:
 - > Introduire des interfaces pour les objets significatifs
 - > Utiliser l'encapsulation pour les implémenter: on en revient aux règles de Yon
- ▶ Encapsulation: mécanisme de sûreté logicielle et non de sécurité



Diagramme des objets ou instances

Définition

- ▶ Diagramme des classes: représentation statique du système
- ▶ Diagramme des objets: à un moment donné, instances créées et leurs liens lorsque le système est actif
- ▶ Notation UML:
 - instance: rectangle avec son nom en style souligné et éventuellement la valeur d'un ou de plusieurs attributs
 - nom d'une instance: **nomInstance : nomClasse**
 - valeur d'un attribut: **nomAttribut : valeurAttribut**
 - lien entre instances: **traits continus**



Diagramme des objets ou instances

Exemple

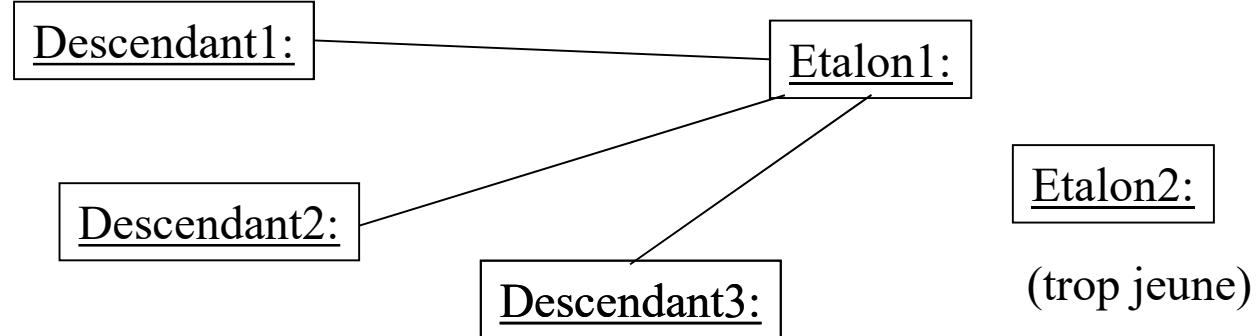


Diagramme de séquence

Modélisation de la dynamique

► Dynamique: **ensemble des interactions entre objets**

- objets d'un système interagissant pour construire la dynamique du système
- ≠ interactions entre le système et un acteur externe (décris dans les cas d'utilisation)

► Deux diagrammes en UML2:

- **diagramme de séquence** axé sur les aspects temporels
- **diagramme de communication** (UML2 et de collaboration en UML1) axé sur la représentation spatiale

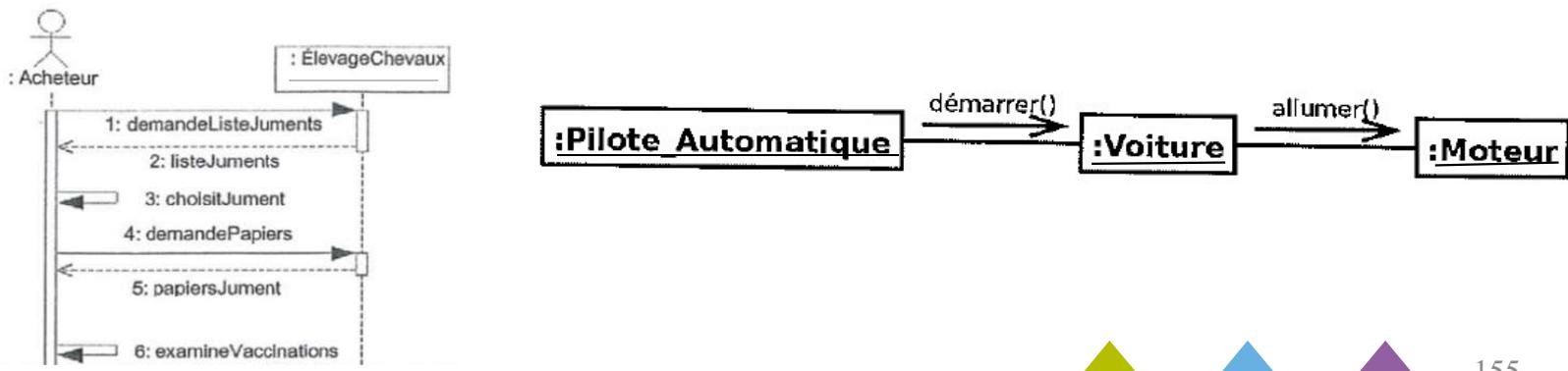


Diagramme de séquence

Modélisation de la dynamique

► Diagrammes d'interaction:

- Exploitation des diagrammes de cas d'utilisation et des scénarios pour identifier des éléments des diagrammes de classe
- Modélisent comment les objets communiquent pour réaliser un objectif de l'utilisateur ou une fonctionnalité



Diagramme de séquence

Modélisation de la dynamique

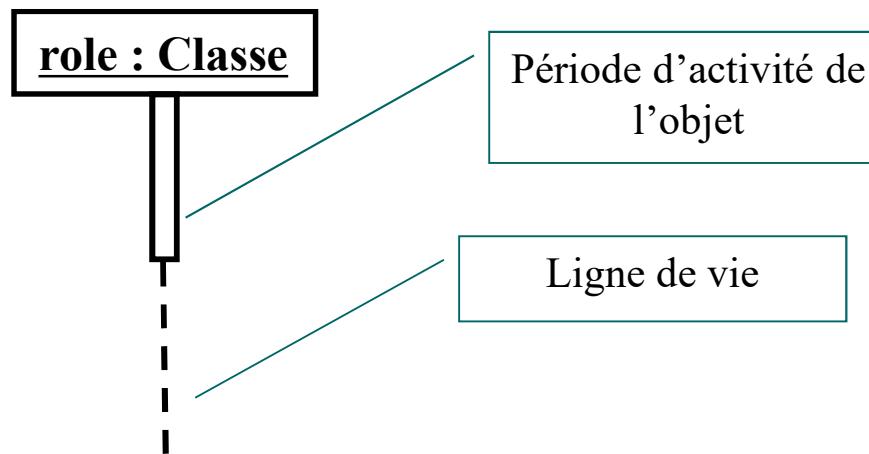
- ▶ Définition du diagramme de séquence:
 - description de la **dynamique du système**
 - description
 - > des interactions entre un groupe d'objets
 - > de la coopération entre les objets
 - **description séquentielle**
 - > des envois de messages entre les objets
 - > des flux de données échangés lors de ces envois
- ▶ Dynamique globale d'un système: ensemble de diagrammes de séquences (en général sous-fonction)
- ▶ En pratique: Représentation de la succession chronologique des opérations réalisées par un acteur
 - Objets manipulés
 - Opérations pour passer d'un objet à l'autre



Diagramme de séquence

Représentation UML

- ▶ Composition du diagramme de séquence:
 - instance des classes intervenant dans le cas d'utilisation représenté
 - **ligne de vie** associée à chaque instance:
 - > actions
 - > réactions
 - > périodes d'activité i.e. exécution de méthodes
- ▶ Notation UML des lignes de vie d'un objet:



- ▶ NB: rôle d'une instance = son nom (UML1) (en général sans accent)



Diagramme de séquence

Représentation UML

- ▶ Interaction entre objets:
 - = envoi de message
 - = appel et exécution de méthode de même nom et donc activation de l'objet récepteur
- ▶ Notation UML des envois de message:
 - numérotation (facultative) séquentielle des messages à partir de 1
 - numérotation composée si un message est envoyé alors que le traitement du précédent n'est pas terminé

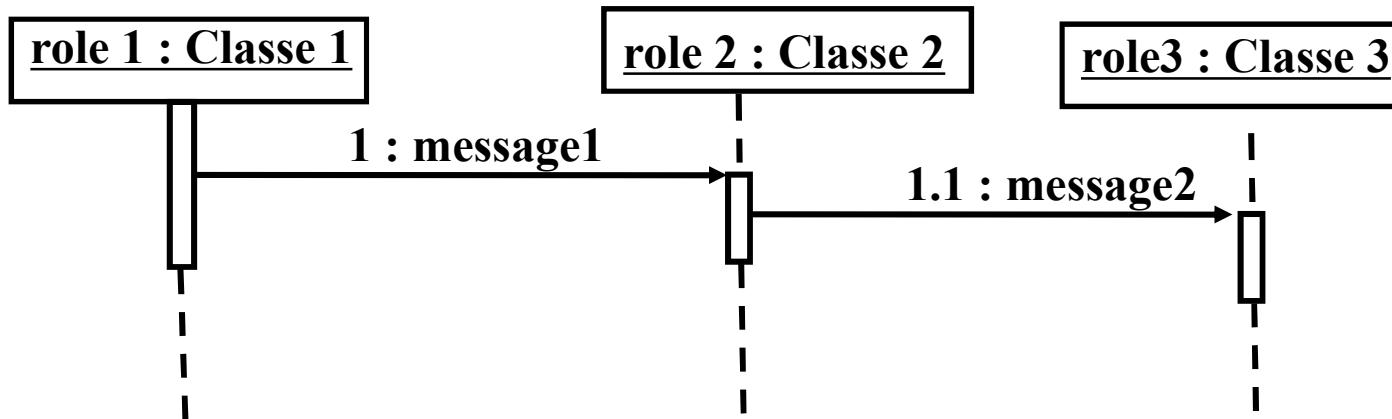


Diagramme de séquence

Messages

- Transmission d'information: paramètres transmis avec le message



Diagramme de séquence

Messages

► Différents types d'envoi de message

- message **asynchrone**:

- > pas d'attente par l'expéditeur de la fin de l'activation de la méthode invoquée chez le destinataire
 - Pas d'attente de « réponse »
 - Pas de blocage de l'expéditeur

- > cas des systèmes avec des traitements parallèles (multi-thread)

- > notation:  **asynchrone**

- > exemples:

- Interruption
 - Événement

- > par défaut: tous messages sont asynchrones



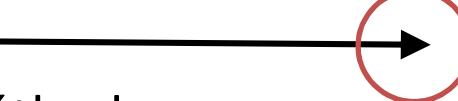
Diagramme de séquence

Messages

► Différents types d'envoi de message

- message **synchrone**:

> l'expéditeur attend que l'activation de la méthode invoquée chez le destinataire soit terminée pour continuer son activité

> notation:  **synchrone**

> Invocation de méthode:

- synchrone ou asynchrone

- en générale synchrone: blocage de l'émetteur

- message de **retour d'invocation de méthode**:

> facultatif

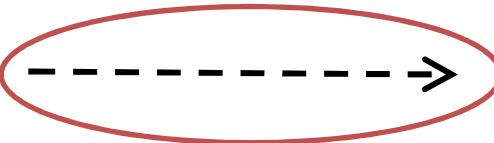
> notation:  **retour de méthode**



Diagramme de séquence

Messages

- Possibilité d'auto-envoi:

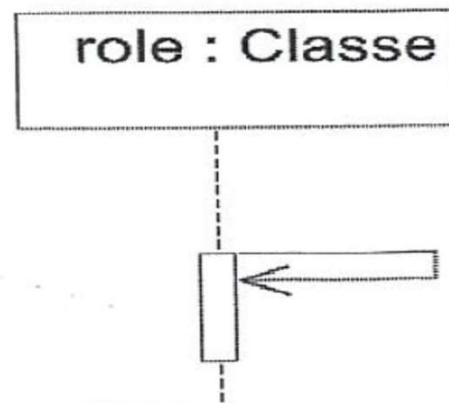
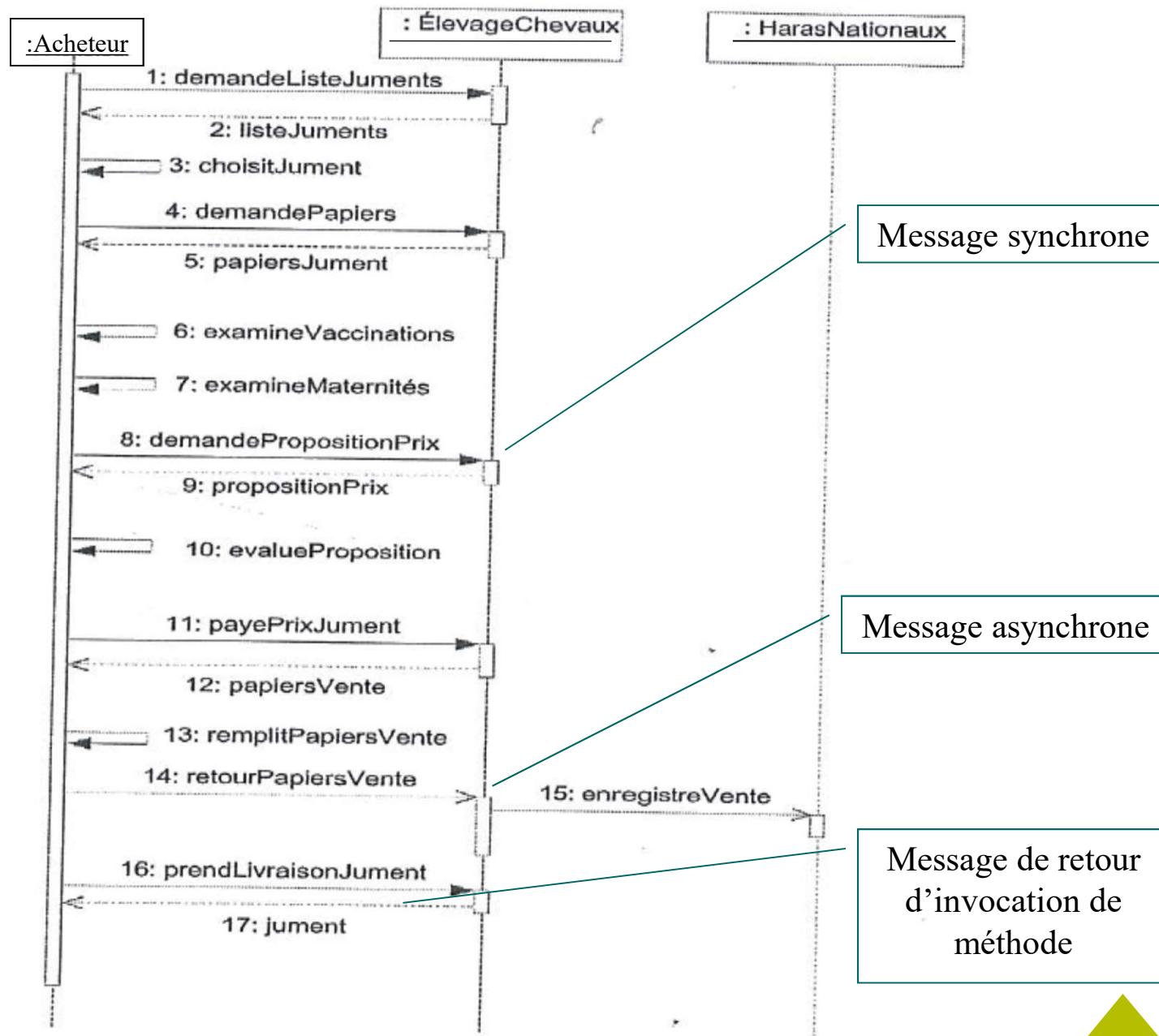


Diagramme de séquence

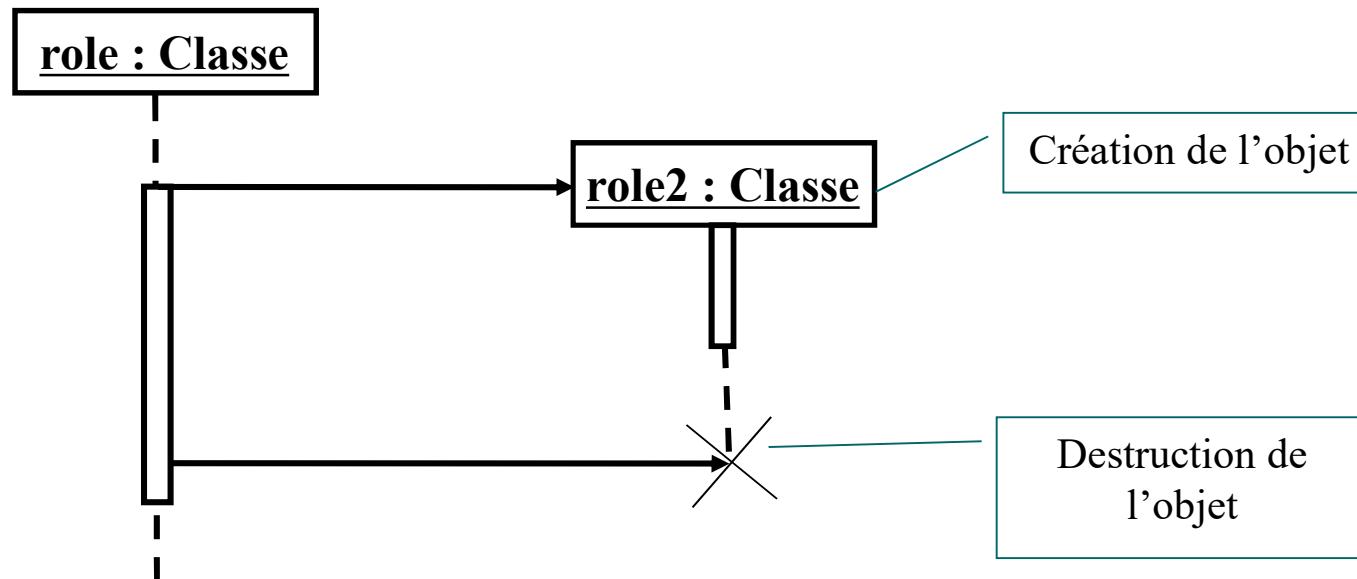


Scénario
(absence
d'alternative)

Diagramme de séquence

Messages

- ▶ Création et destruction d'objets: provoquées par des messages
- ▶ Notation UML:



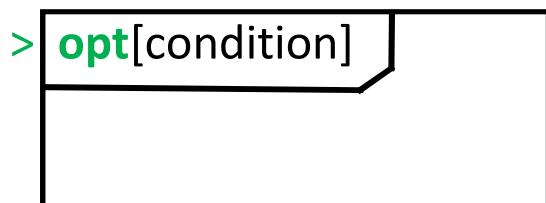
- ▶ NB: destruction d'un objet par forcément consécutive à la réception d'un message

Diagramme de séquence

Cadres d'interaction

- ▶ Définition: partie du diagramme de séquence associée à une étiquette contenant un opérateur qui en détermine la modalité d'exécution:

- **alternative:**



si la condition de test est vraie, le cadre est exécuté

- > **alt** [condition1] [condition2] [else] avec un sous cadre pour chaque condition

- **boucle:** **loop**[min,max,condition]

- > paramètres optionnels
 - > cadre exécuté min fois tant que la condition de test est vérifiée tant que le nombre d'exécution ne dépasse pas max



Diagramme de séquence

Cadres d'interaction

► Autres types:

- Choix et boucle:
Alternative, Option, Break, Loop
- Envoi parallèle de messages:
Parallel, Critical region
- Contrôle d'envoi de message:
Ignore, consider, assertion, negative
- Ordre d'envoi des messages:
Weak sequencing, strict sequencing

(à utiliser en TP si besoin après avoir recherché la signification sur le net)



Diagramme de séquence

Cadres d'interaction

► Exemple de boucle:

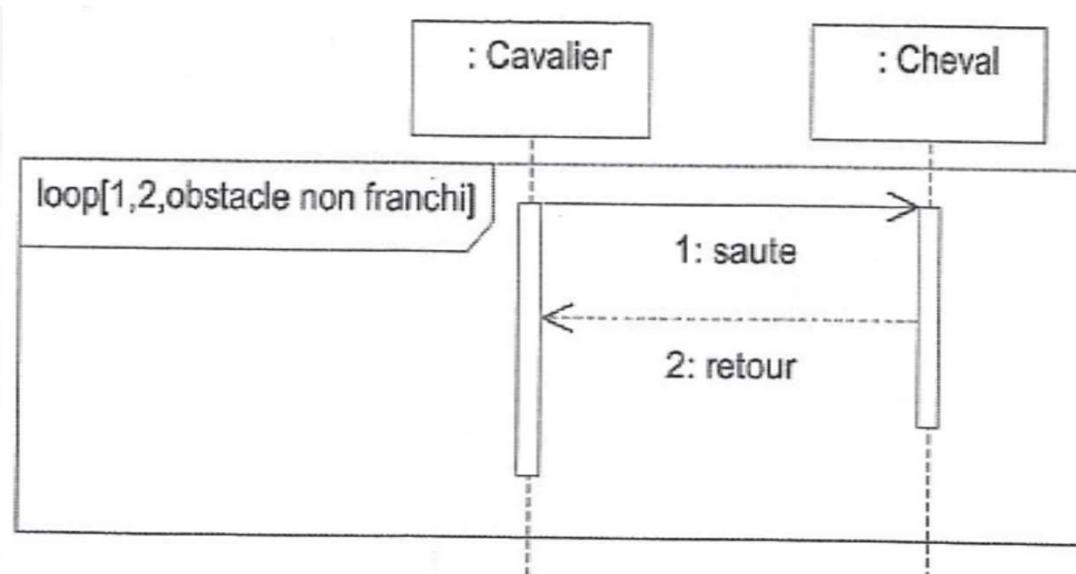


Diagramme de séquence

Cadres d'interaction

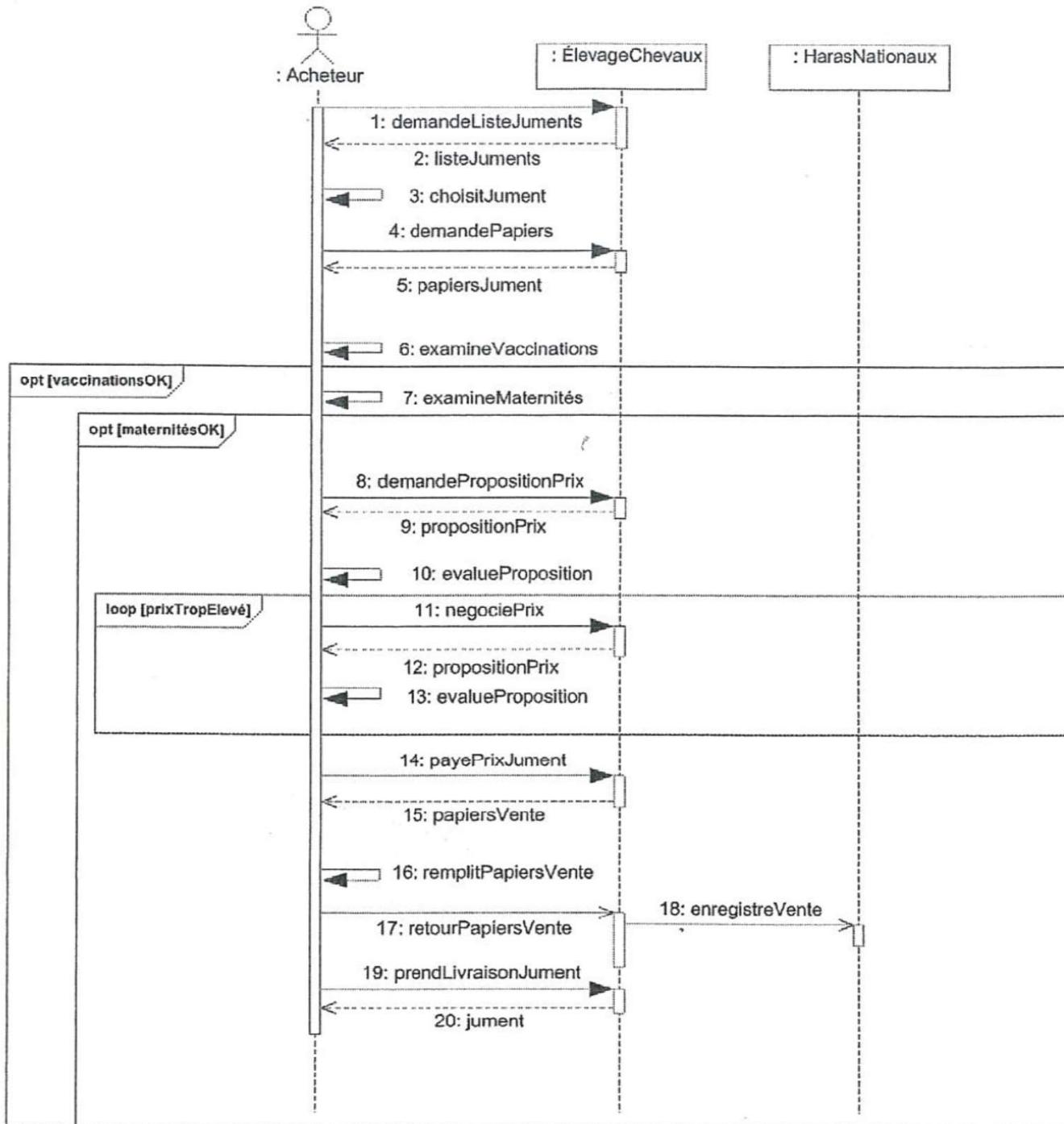


Diagramme de séquence

Exercice d'application:

- ▶ Une entreprise de mannequinat souhaite acquérir un système informatique pour gérer les castings c'est-à-dire les inscriptions et sélections des candidats.
- ▶ Lors du casting, le responsable saisit pour chaque candidat ses commentaires et sa conclusion (candidat retenu ou non retenu).
- ▶ A la fin du casting, le responsable du casting doit valider l'ensemble de ses commentaires et conclusions.
- ▶ A la validation, un message électronique est envoyé à chaque candidat pour l'informer du résultat du casting.
- ▶ La liste des candidats retenus est imprimée.



Diagramme de séquence

Exercice d'application:

Définir le diagramme de séquence du scénario de la diapo suivante:



Diagramme de séquence

Exercice d'application:

Cas d'utilisation	EVALUER LES CANDIDATS	
Acteur primaire	Responsable de casting	
Système	Système informatique de gestion des castings	
Intervenants	Responsable du casting Candidats	
Niveau	Objectif utilisateur	
Préconditions	Le casting a été défini et organisé. Les candidatures des candidats ont été validées. Les candidats ont été pointés. Le responsable du casting s'est connecté au système.	
Opérations	Acteur	Système informatique de gestion des castings
1		Le système affiche la liste des castings.
2	Le responsable du casting sélectionne le casting du jour.	
3		Le système affiche la liste des personnes présentes devant passer le casting.
4	Le responsable du casting sélectionne une personne.	
5		Le système affiche la fiche de la personne choisie.
6	Le responsable du casting saisit ses commentaires et sa conclusion pour la personne.	
7		Le système affiche la liste des personnes présentes devant passer le casting.
8	Le responsable du casting indique que c'est la fin du casting.	
9		Le système affiche un récapitulatif des commentaires et des conclusions.
10	Le responsable du casting valide ces résultats.	
11		Le système envoie un message à chaque candidat pour lui indiquer son résultat.
12		Le système imprime la liste des candidats retenus.
Extensions		
10.A	Modification des résultats.	
10.A.1	Le responsable du casting demande une modification des résultats.	
10.A.2		Le système affiche la liste des personnes présentes au casting.
10.A.3	Le responsable du casting sélectionne une personne.	
10.A.4		Le système affiche les commentaires et la conclusion pour cette personne.
10.A.5	Le responsable du casting modifie la conclusion pour cette personne.	
10.A.6		Le système affiche un récapitulatif des commentaires et des conclusions. Retour à l'étape 10.

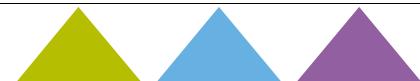




Diagramme de séquence

Conclusion

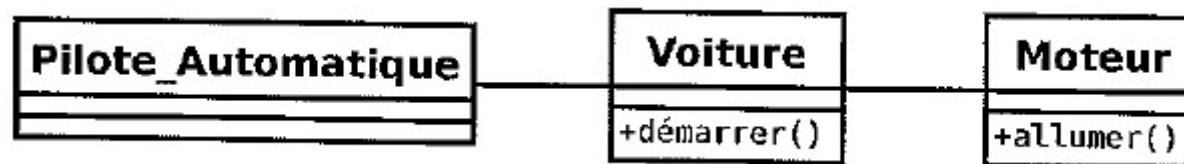
- ▶ Diagrammes de séquence : illustration et vérification du comportement d'un ensemble d'objets (système ou sous-système)
- ▶ Cadres d'interaction: description de cas d'utilisation par des diagrammes de séquence
- ▶ Autre type de représentation de la dynamique du système : diagramme de communication
- ▶ Comparaison:
 - diagramme de séquence: aspects temporels
 - diagramme de communication: liaisons entre classes
- ▶ Diagrammes de séquence :
 - découverte des objets du système
 - découverte des méthodes des objets



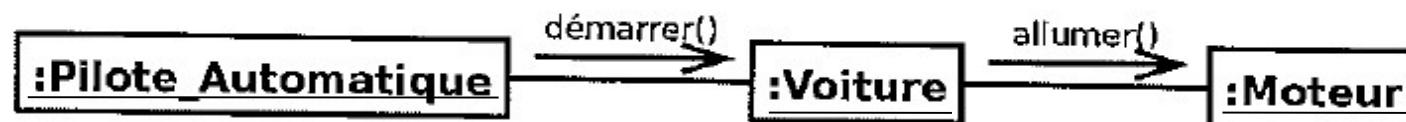
Diagramme de séquence

Conclusion

- ▶ Diagramme de classe



- ▶ Diagramme de communication



- ▶ Diagramme de séquence

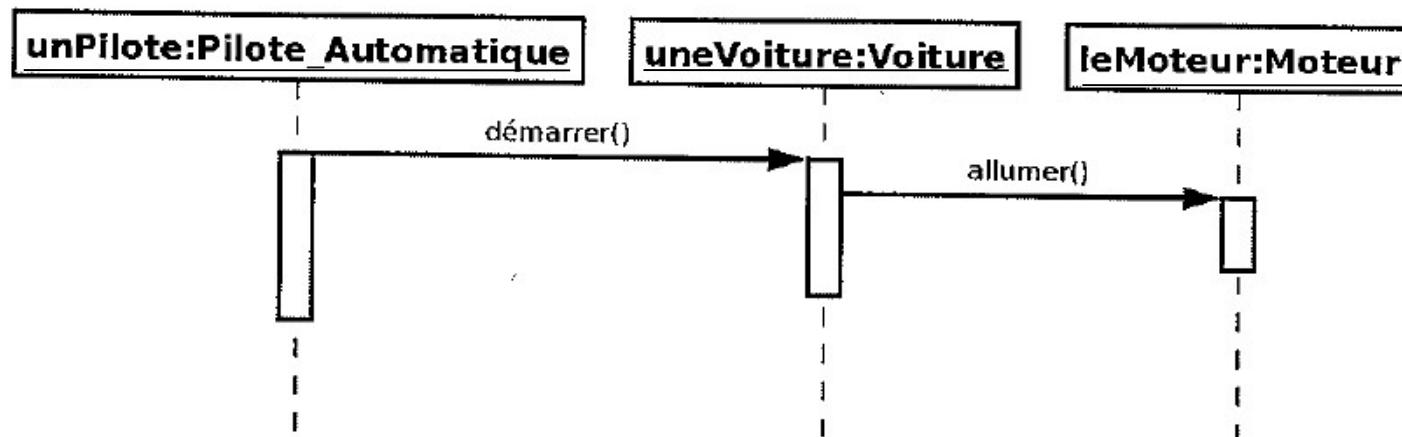


Diagramme de séquence

Conclusion

- ▶ Diagramme des cas d'utilisation:
 - Limites du système
 - Objectifs des utilisateurs
- ▶ Scénarios: interactions nécessaires pour la réalisation de ces objectifs
- ▶ Diagramme de séquence:
 - Représentation des scénarios
 - Interactions
 - > Découverte des objets du système
 - > Identification des attributs et méthodes de ces objets
- ▶ Diagramme de classe: structure et comportement du système



FIN DU TROISIEME COURS



Diagramme état-transition

Modélisation du cycle de vie des objets

► Déjà vus:

- interactions entre objets
- aspects statiques des objets

► Cycle de vie d'un objet:

différentes étapes ou états que celui-ci va suivre pour concourir, au sein du système, à la réalisation d'un objectif.

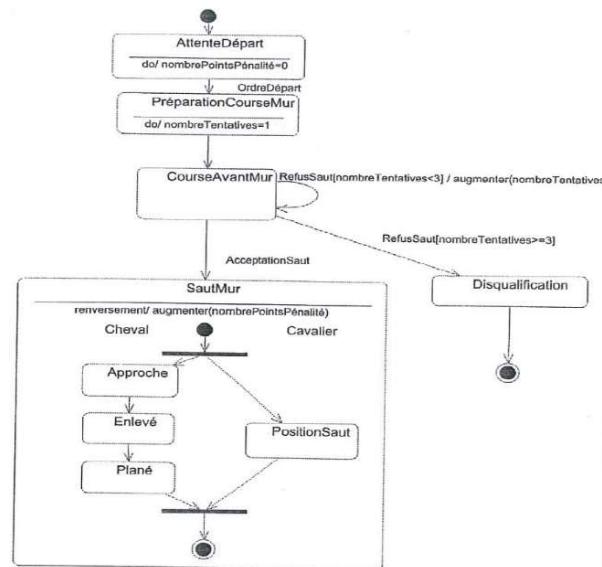


Diagramme état-transition

Notion d'état

- ▶ Etat d'un objet:
 - Correspond à un moment de son cycle de vie
 - Deux cas:
 - > Inactif: en attente d'un signal venant d'autres objets
 - > Actif: réalise une activité
- ▶ Activité:
 - exécution d'une série de méthodes et d'interactions avec d'autres objets
 - liée à un objectif
- ▶ Activation d'un objet:
 - Java, C++: activation de l'une de ses méthodes
 - UML: réception d'un signal
- ▶ Ensemble des états:
 - État initial: état de l'objet juste après sa création
 - 0, 1 ou plusieurs états finaux: destruction de l'objet



Diagramme état-transition

Changement d'état

- ▶ Evénement:
 - fait qui déclenche le changement d'état
 - lié à la réception d'un signal par l'objet = réception d'un message
- ▶ Possibilité pour un objet d'émettre un signal qu'il recevra lui-même pour changer d'état
- ▶ Description des signaux en UML:
 - classes de signaux avec le stéréotype « signal »
 - attributs des objets des classes de signaux: paramètres de message
 - hiérarchisation des signaux
- ▶ Réification:
 - ex: décrire un message par une classe
 - transformation en objet



Diagramme état-transition

Changement d'état

► Exemple:

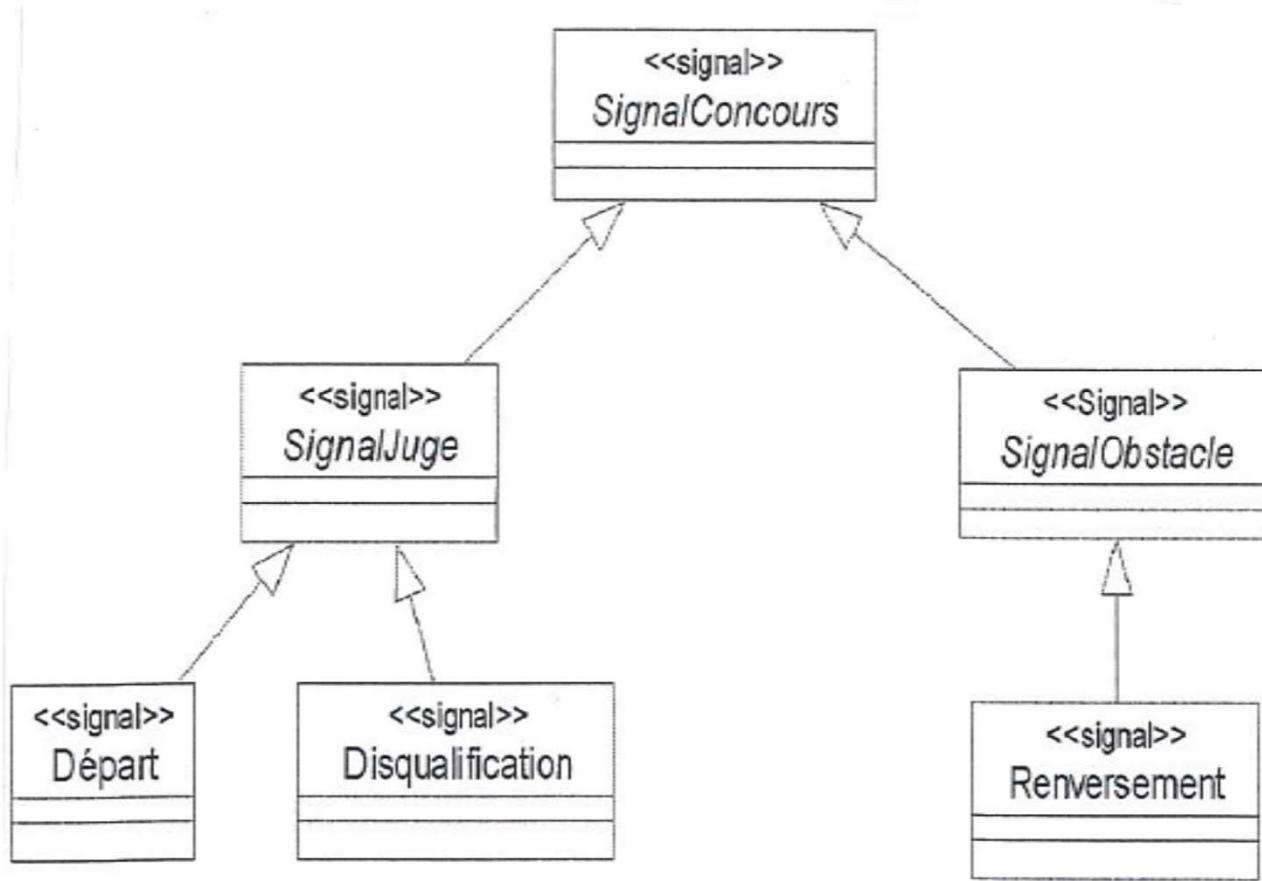


Diagramme état-transition

Changement d'état

► Transition:

- lien orienté entre deux états
- possibilité pour l'objet de passer de l'état d'origine à son état de destination
- franchie lorsque l'objet réalise ce passage
- souvent associée à un événement:
 - > transition franchie si l'objet:
 - se trouve dans l'état d'origine de la transition
 - reçoit l'événement
 - > deux cas:
 - si objet inactif: franchissement immédiat
 - si objet actif: franchissement dès que l'activité associée à l'état est terminée



Diagramme état-transition

Changement d'état

► Différentes transitions:

- **transition automatique:**

- > pas associée à un événement
- > franchie dès que l'objet a terminé l'activité liée à l'état d'origine

- **transition réflexive:**

- > même état d'origine et de destination
- > si elle est liée à un événement, la réception de celui-ci ne fait pas concrètement changer d'état (mais provoque un appel de méthode, un envoi de message...)
- > intérêt d'une transition réflexive et automatique: réalisation d'une activité en boucle



Diagramme état-transition

Modélisation du diagramme état-transition

► Diagramme états-transitions:

- représente le cycle de vie des instances d'une classe
- décrit:
 - > les états
 - > les transitions qui les lient
 - > les événements qui provoquent le franchissement des transitions
- utile uniquement pour les objets ayant un cycle de vie (plusieurs états)
- contient un unique état initial
- peut contenir un ou des états finaux: étape
 - > où l'objet n'est plus nécessaire dans le système
 - > où il est détruit
- objets permanents dans le système: n'ont pas d'état final



Diagramme état-transition

Elaboration du diagramme état-transition

► Notation UML:

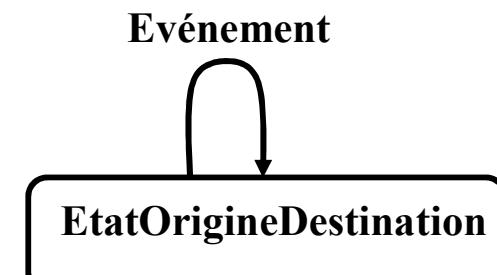
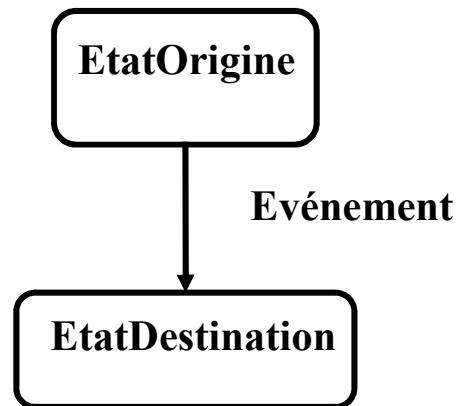
- état: 

- état initial: 

état final: 

- transition:

> transition franchie par événement:



> transition automatique: pas d'indication de l'événement



Diagramme état-transition

Elaboration du diagramme état-transition

► Notation UML:

- Choix des noms d'événement et d'état lié à la complexité des diagrammes:
 - > Diagramme simple: possibilité de noms longs
 - > Diagramme complexe:
 - noms courts
 - tableau décrivant la signification des noms
- En général, diagrammes génériques: une occurrence d'état par diagramme



Diagramme état-transition

Elaboration du diagramme état-transition

► Exemple:

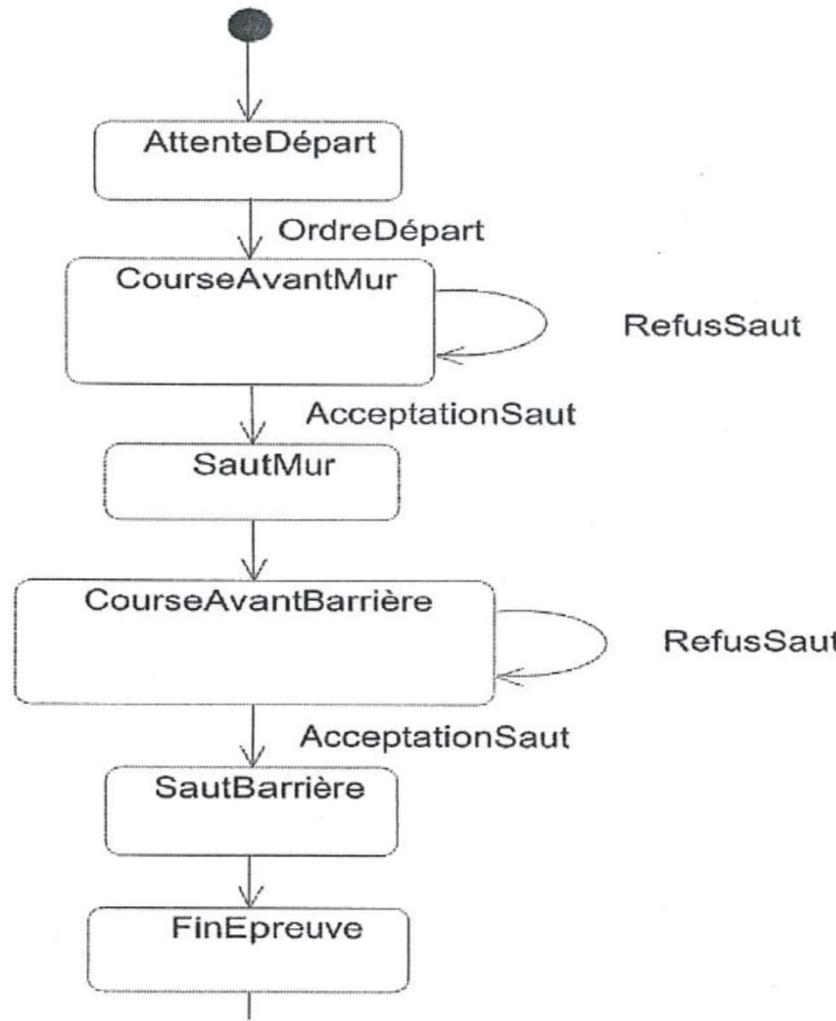


Diagramme état-transition

Elaboration du diagramme état-transition

► Conditions de garde:

- condition associée à une transition
- franchissement de la transition si:
 - > objet dans l'état d'origine
 - > objet a terminé l'activité liée à l'état d'origine
 - > réception de l'événement associé à la transition (s'il y en a un)
 - > condition de garde remplie
- exprimée entre crochets

► Notation UML:

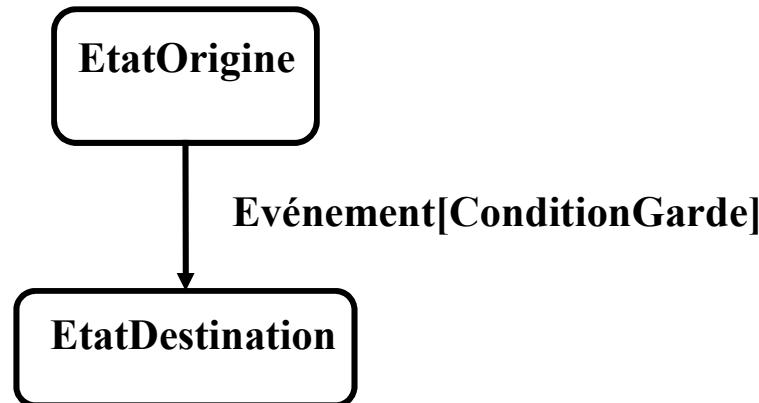


Diagramme état-transition

Elaboration du diagramme état-transition

► Exemple:

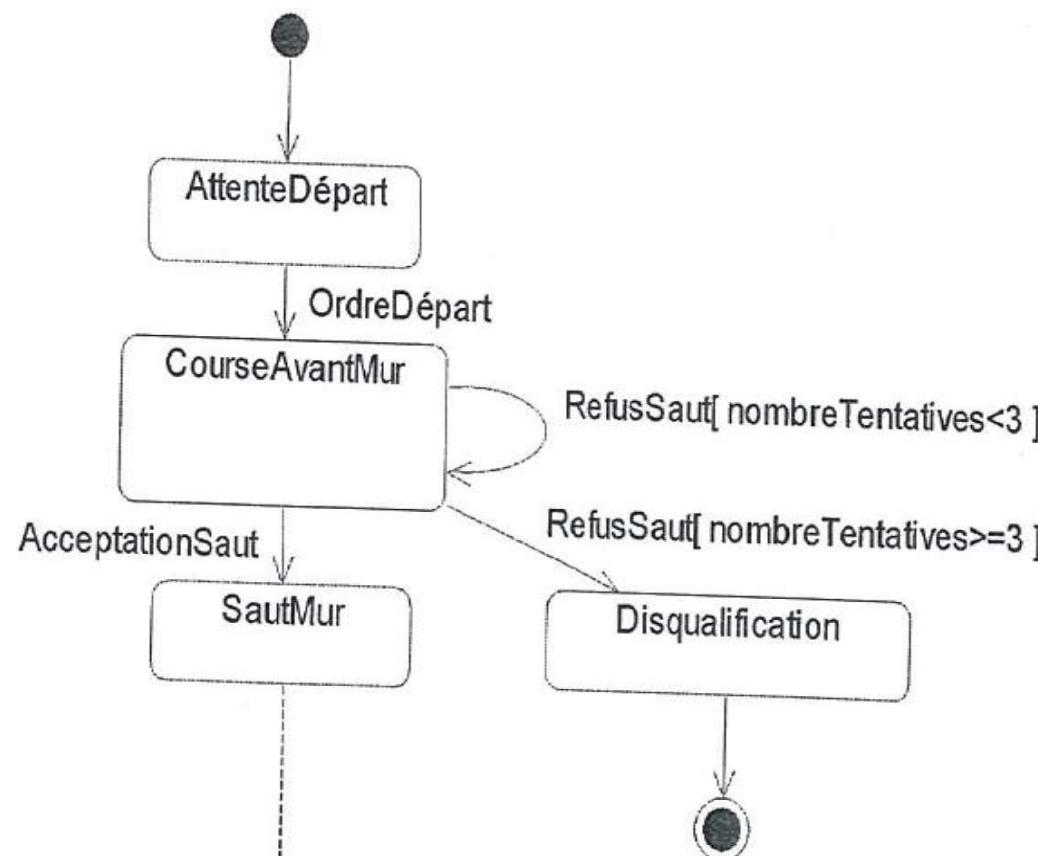


Diagramme état-transition

Elaboration du diagramme état-transition

► Possibilité d'avoir des activités liées:

- à un état: **do/ activité**
- lors du franchissement d'une transition: **événement[conditionGarde]/ activité**
- à l'entrée dans un état: **entry/ activité**
- à la sortie d'un état: **exit/ activité**
- au sein d'un état, lors de la réception d'un événement: **événement/ activité**

► Activité = série d'actions

► Exemples d'action:

- affecter une valeur à un attribut: **attribut=valeur**
- créer ou détruire un objet;
- effectuer une opération: **calculer**
- envoyer un signal à un autre objet (désigné par son nom) ou à soi-même: **^nomSignal**



Diagramme état-transition

Elaboration du diagramme état-transition

► Notation UML

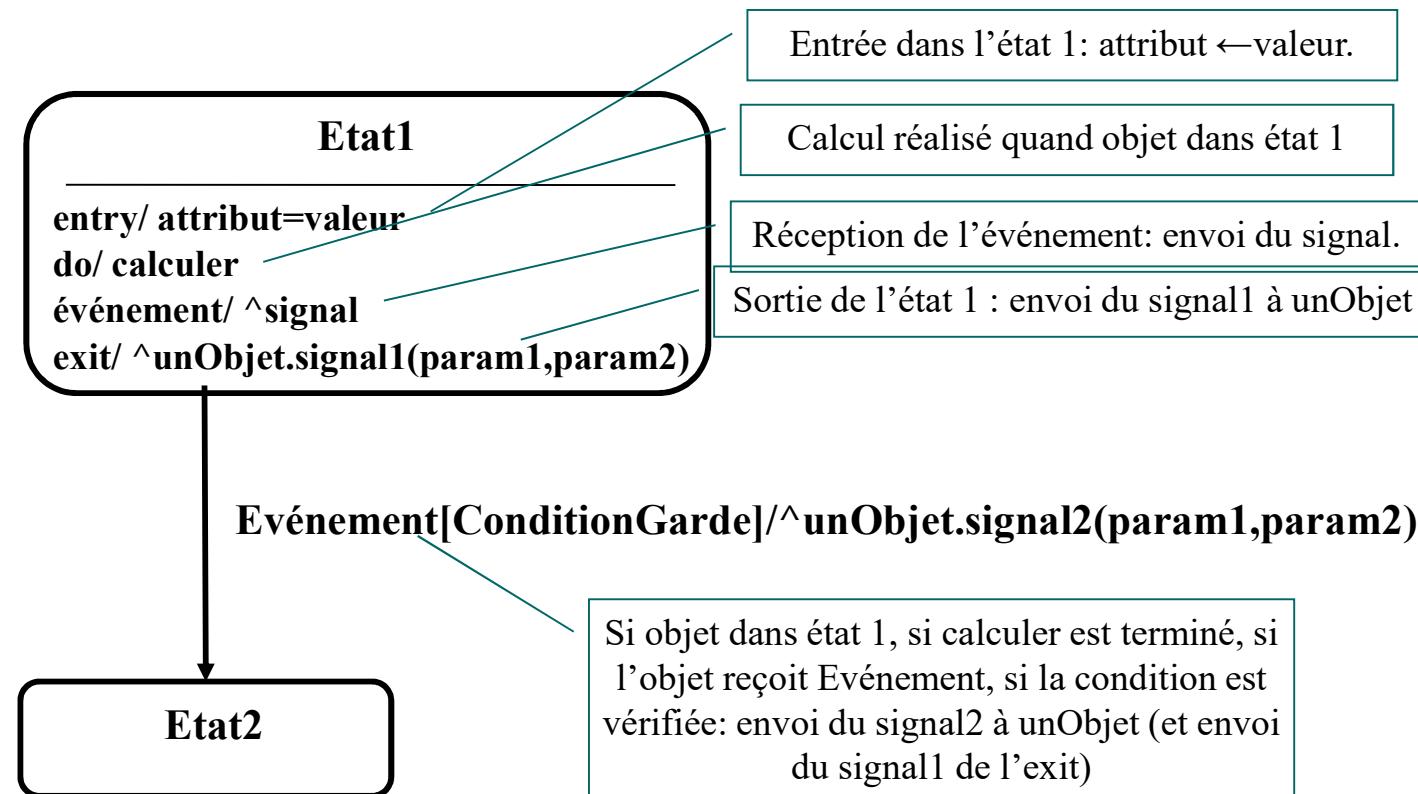
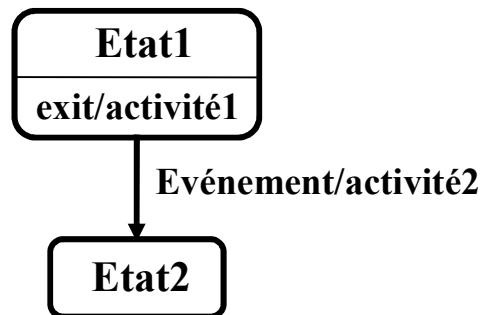


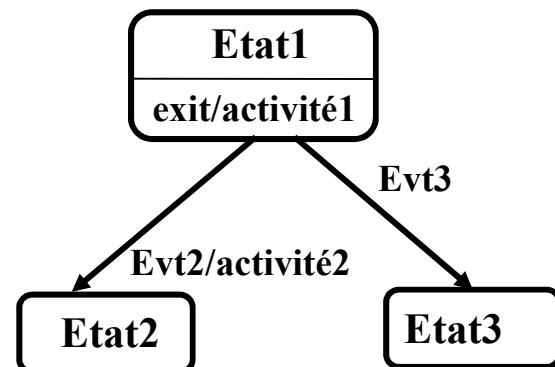
Diagramme état-transition

Elaboration du diagramme état-transition

► Différence entre les deux activités:



- **exit/activité1:** réalisé lorsqu'on sort de l'état1 quelque soit la transition franchie
- **Evènement/activité2:** réalisé lorsqu'on sort de l'état1 pour franchir la transition vers l'état2.



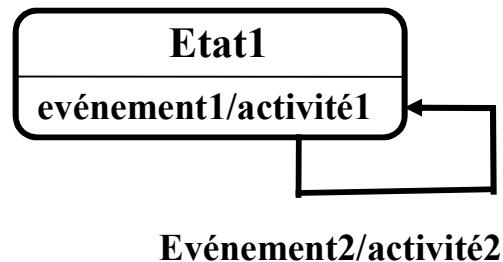
- **Etat1 actif et Evt3:** activité1 puis activation de l'état 3
- **Etat1 actif et Evt2:** activité1 puis activité2 puis activation de l'état 2



Diagramme état-transition

Elaboration du diagramme état-transition

► Différence entre les deux activités:



- **Évènement1/activité1** réalisé lorsque l'état 1 est actif et que l'événement 1 intervient: pas de changement d'état
- **Évènement2/activité2:** réalisé lorsque l'état 1 est actif et que l'événement 2 intervient: l'objet sort de l'état 1, réalise l'activité 2 et l'objet rentre à nouveau dans l'état 1

Pas de différence entre les activités 1 et 2 mais changement d'état donc:

- *réalisation des activités liées à des exit et entry*
- *et possible conséquence sur d'autres interactions si elles sont conditionnées à l'état actif.*



Diagramme état-transition

Elaboration du diagramme état-transition

► Exemple:

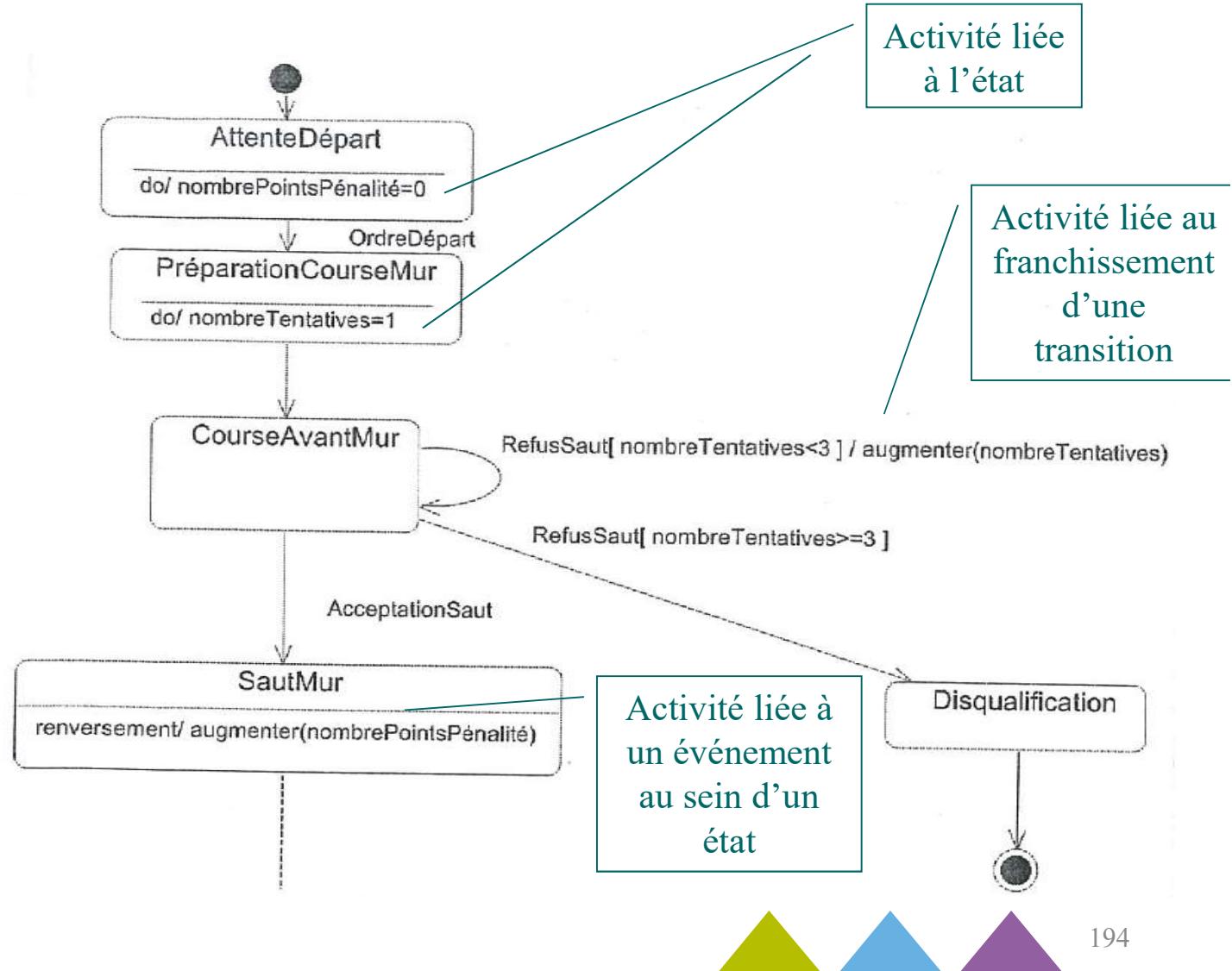


Diagramme état-transition

Elaboration du diagramme état-transition

► Etat composé:

Description par un diagramme d'états-transitions

► Notation UML: Etat 2 composé

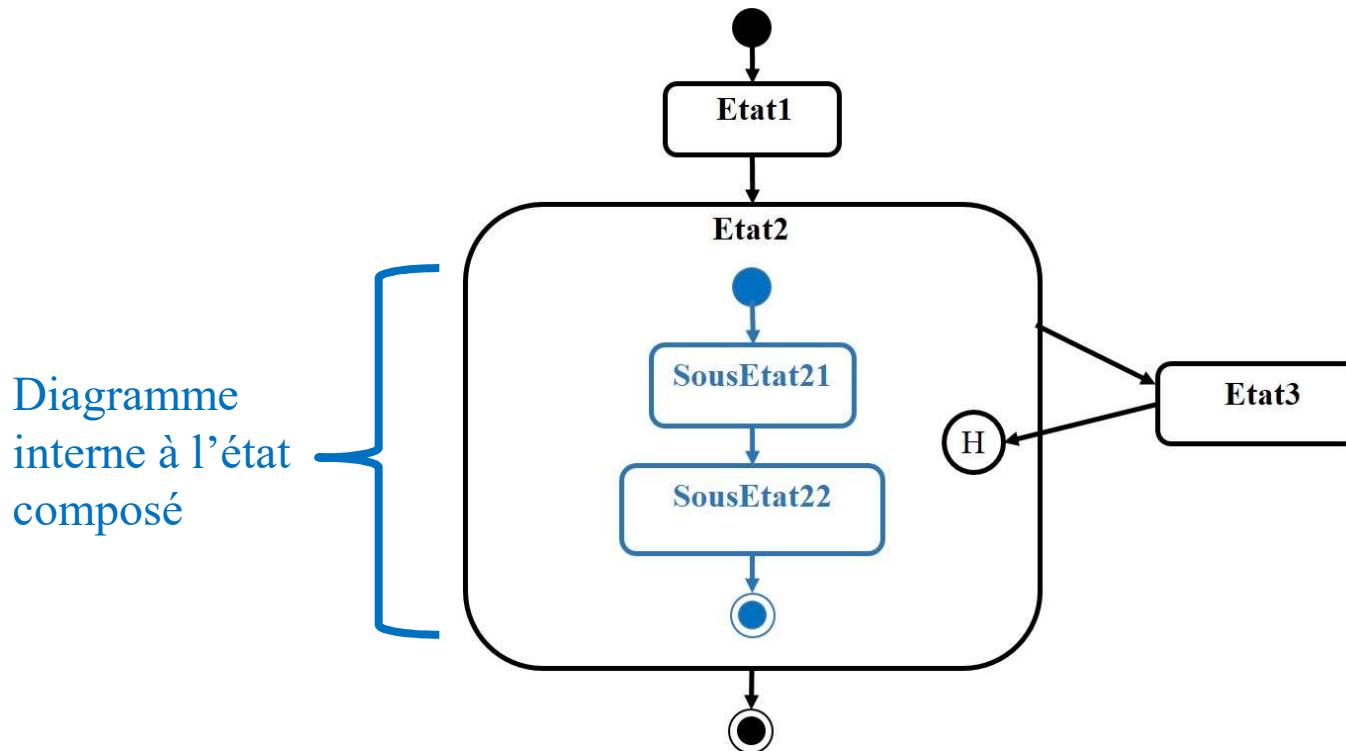
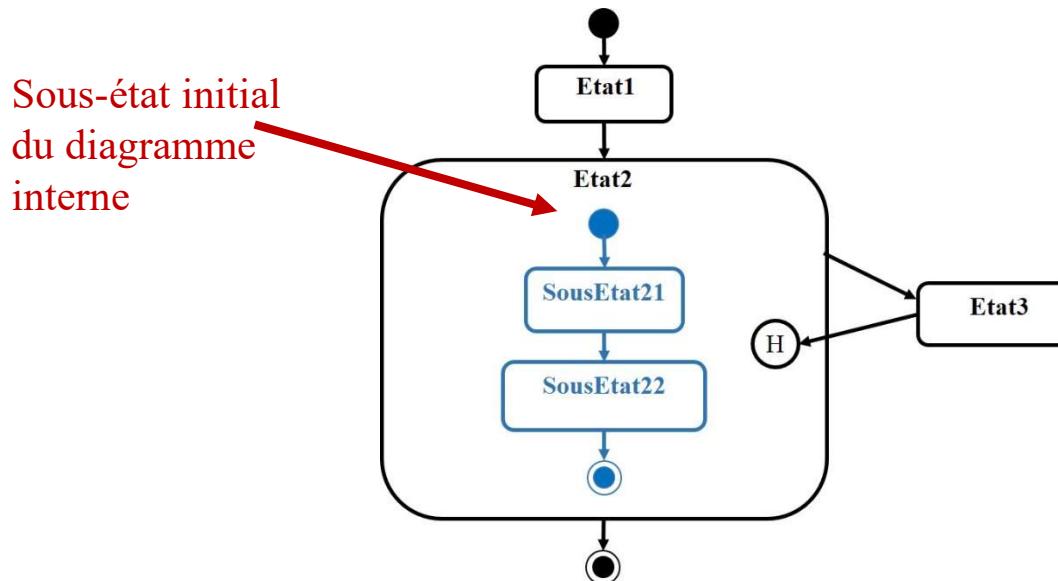


Diagramme état-transition

Elaboration du diagramme état-transition

► Etat composé:

- Entrer dans un état composé \Leftrightarrow entrer dans le sous-état initial du diagramme interne d'états-transitions



- Entrée dans le ou les sous-états finaux \Leftrightarrow sortie de l'état composé

► Diagramme interne d'état-transitions: zéro, un ou plusieurs états finaux



Diagramme état-transition

Elaboration du diagramme état-transition

- Possibilité de mémoriser le sous-état actif lorsqu'un objet quitte un état composé en utilisant le sous-état spécial de mémoire H (history):

H= dernier sous-état actif mémorisé

- Sous-état composé de sous-états: deux sous-états de mémoire:
 - H: retour au sous-état de niveau le plus élevé (toujours sous-état)
 - H*: retour au sous-état imbriqué (sous-sous-...sous-état éventuel)

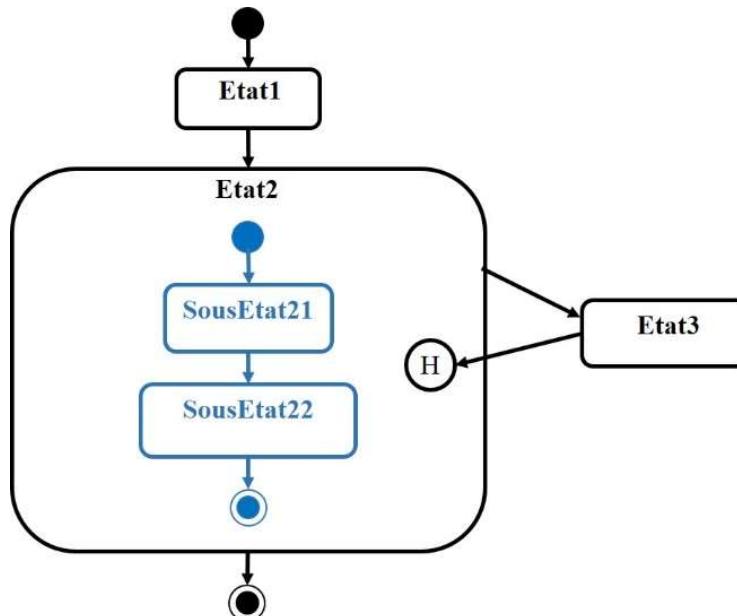


Diagramme état-transition

Elaboration du diagramme état-transition

► Exemple du lave-vaisselle:

- 3 sous-états: rinçage, lavage, séchage
- De multiples sous-sous-états dans chacun de ces 3 sous-états
- H: retour au sous-état de niveau le plus élevé: on reprend le cycle en début de rinçage, lavage ou séchage
- H*: retour au sous-état imbriqué (sous-sous-état où on s'était précisément arrêté) par exemple seconde partie du lavage

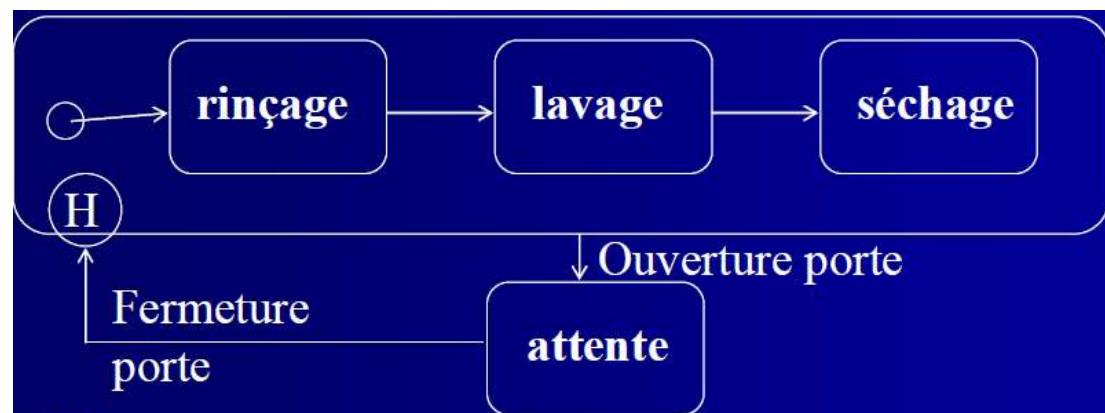
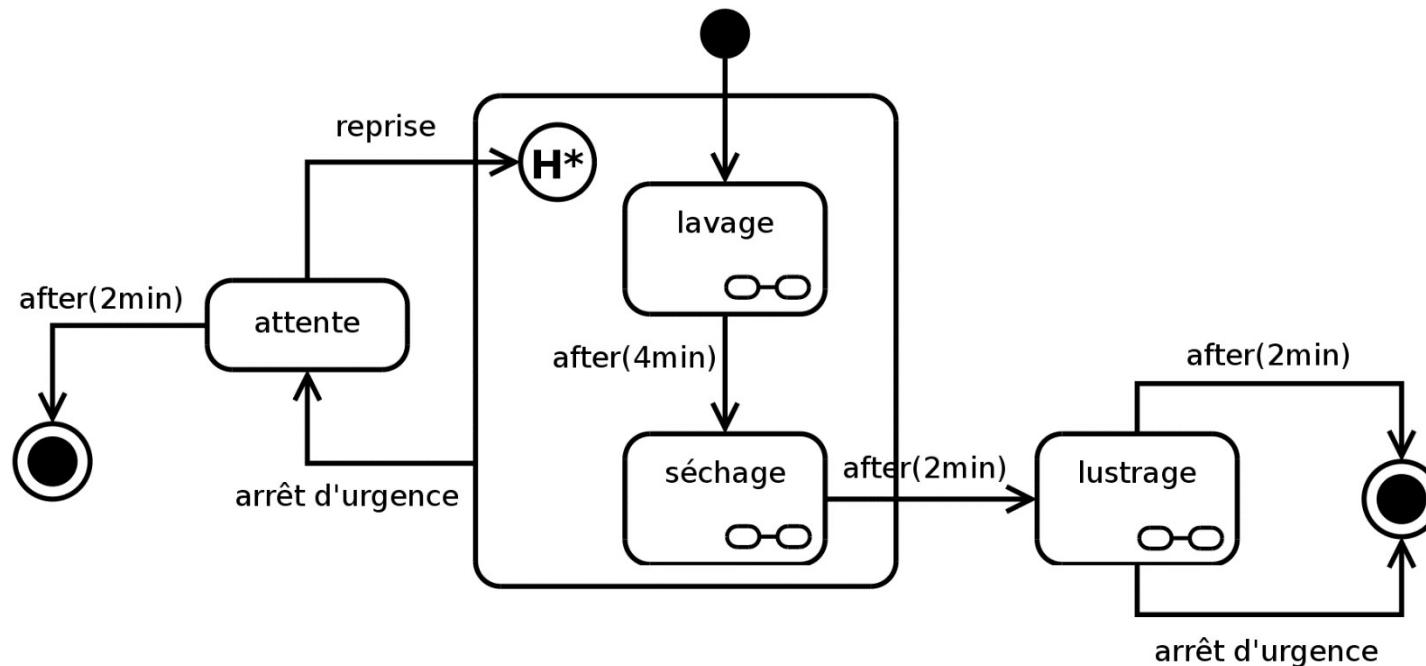


Diagramme état-transition

Elaboration du diagramme état-transition

- Exemple de reprise dans le dernier état actif : lavage de voiture



H*: reprise du lavage ou séchage exactement où le programme a été interrompu

H: reprise de toute la séquence de lavage ou de séchage



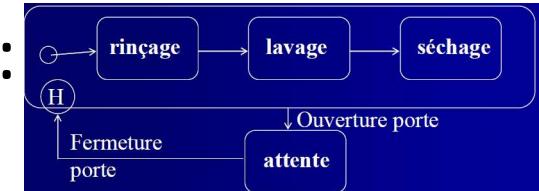
Diagramme état-transition

Elaboration du diagramme état-transition

► Remarques sur les états historiques :

- **H: état historique plat**

- > accès limité aux états de son niveau d'imbrication.
 - H du niveau de rinçage, lavage, séchage
 - retour en H: retour au début de rinçage, lavage, séchage
- > pseudo-état qui mémorise le dernier sous-état actif d'un état composite
 - transition vers H = transition vers le dernier état visité de l'état englobant
 - transition sortante de H non étiquetée = état à exécuter si la région n'a pas encore été visitée.



- **H*: état historique profond**

- > Pseudo-état qui permet d'atteindre le dernier état visité dans la région, quel que soit son niveau d'imbrication



Diagramme état-transition

Elaboration du diagramme état-transition

► Différents types de transition dans l'état composé:

- **fourche:**

- > évolution en parallèle de sous-états
- > possède plusieurs sous-états de destination
- > après franchissement de la fourche, l'objet se trouve dans tous les sous-états de destination

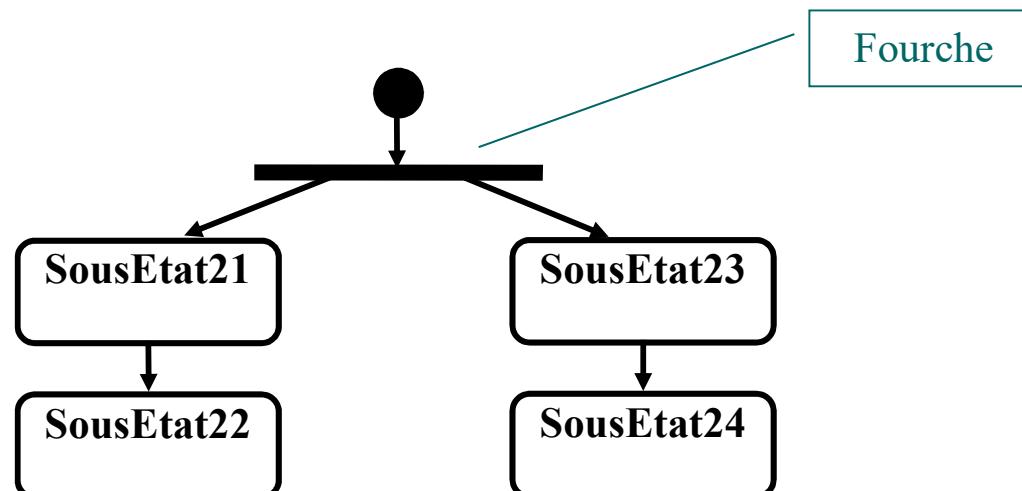


Diagramme état-transition

Elaboration du diagramme état-transition

► Différents types de transition dans l'état composé:

- **synchronisation:**

- > possède plusieurs sous-états d'origine
- > possède un seul sous-état de destination
- > franchissable uniquement si l'objet se trouve dans tous les sous-états d'origine

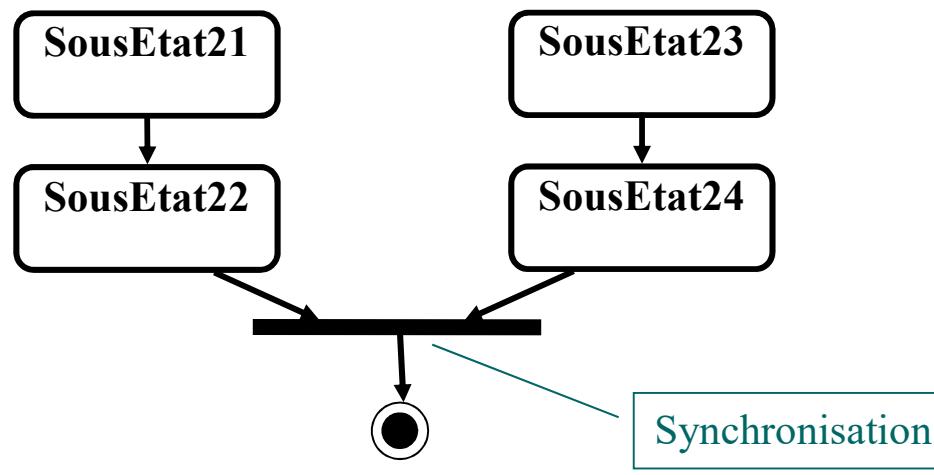


Diagramme état-transition

Elaboration du diagramme état-transition

► Notation UML :

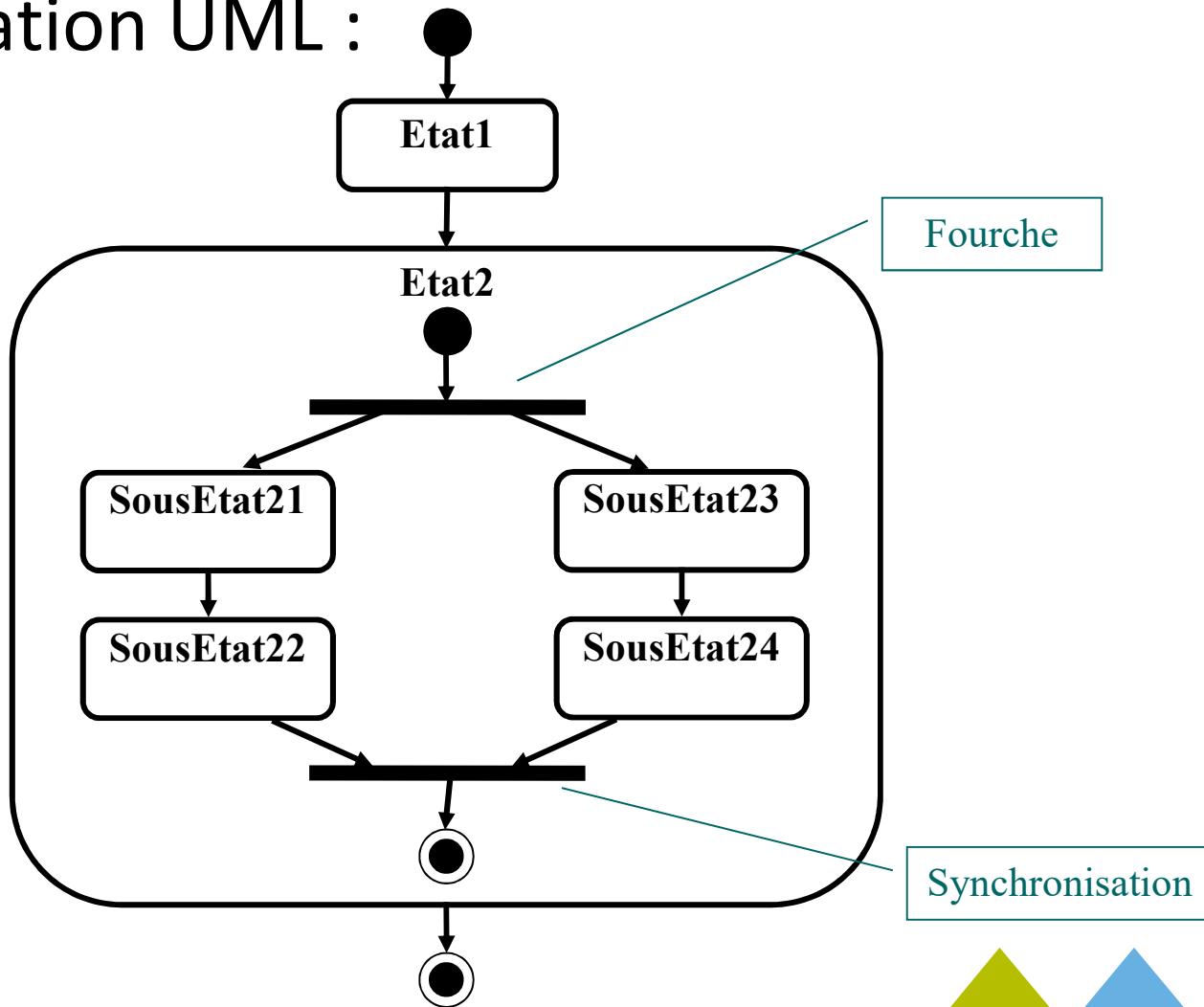
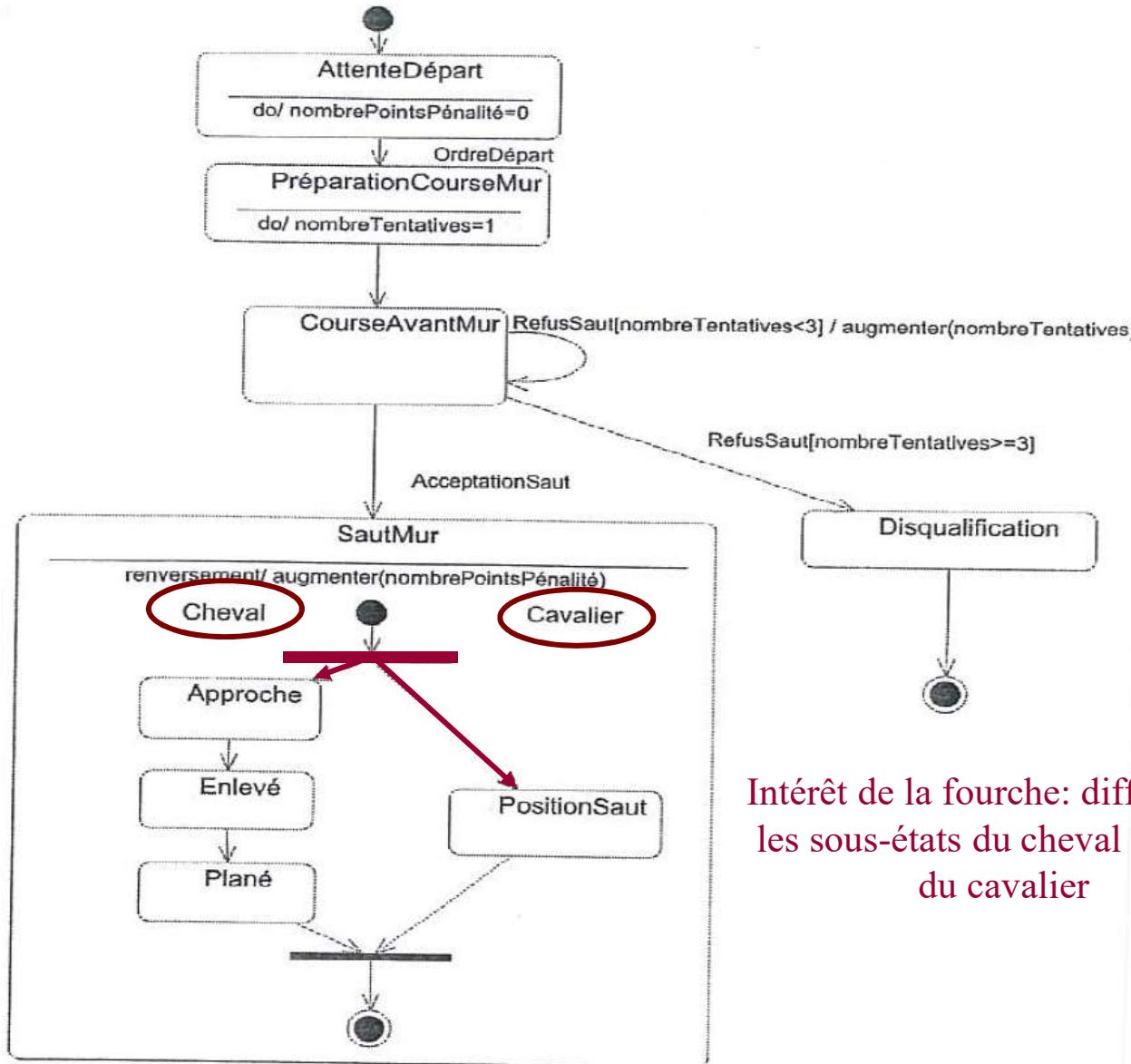


Diagramme état-transition

Elaboration du diagramme état-transition



Intérêt de la fourche: différencier les sous-états du cheval de ceux du cavalier



Diagramme état-transition

Conclusion

- ▶ Diagramme d'états-transitions:
 - décrit le cycle de vie des objets chargés d'assurer la dynamique du système
 - utilisé séparément pour chacun des objets
- ▶ Enjeu: s'assurer que les objets puissent répondre aux interactions décrites dans les diagrammes de séquences et de communication



Diagramme état-transition

Conclusion méthodologique

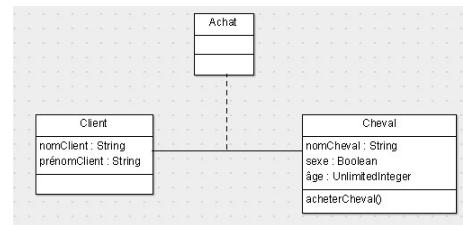
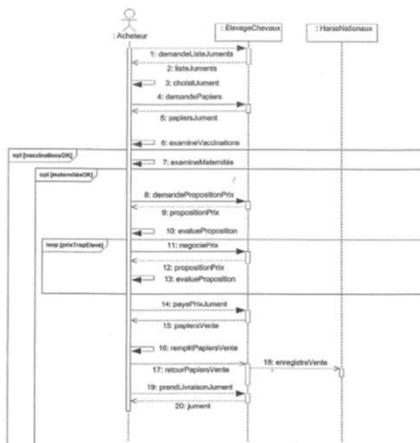
► Diagramme des cas d'utilisation:

- Limites du système
- Objectifs des utilisateurs



► Scénarios et Diagramme de séquence système:

- Interactions nécessaires pour réaliser les objectifs utilisateurs



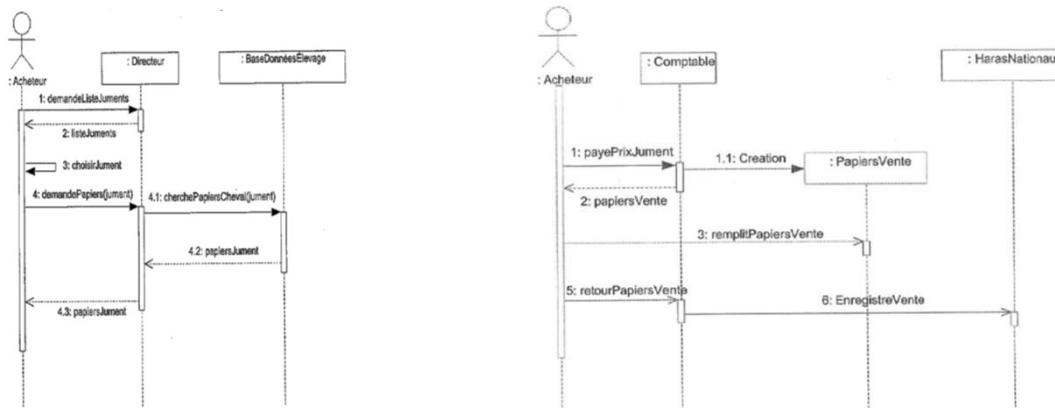
⇒ Premier diagramme de classe



Diagramme état-transition

Conclusion méthodologique

- ▶ Diagrammes de séquence détaillés:
⇒ Découverte des objets du système



- ⇒ Identification des attributs et méthodes de ces objets
- ⇒ Seconde version du diagramme de classe: complétée

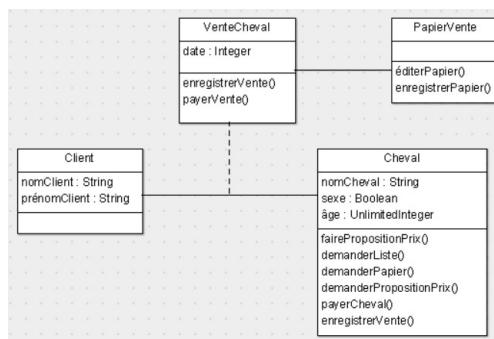
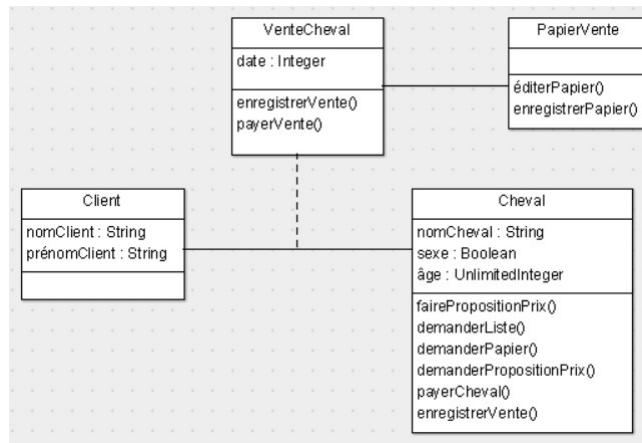


Diagramme état-transition

Conclusion méthodologique

- ▶ Diagramme état-transition: préciser et vérifier les attributs et les méthodes des objets dans le diagramme de classe

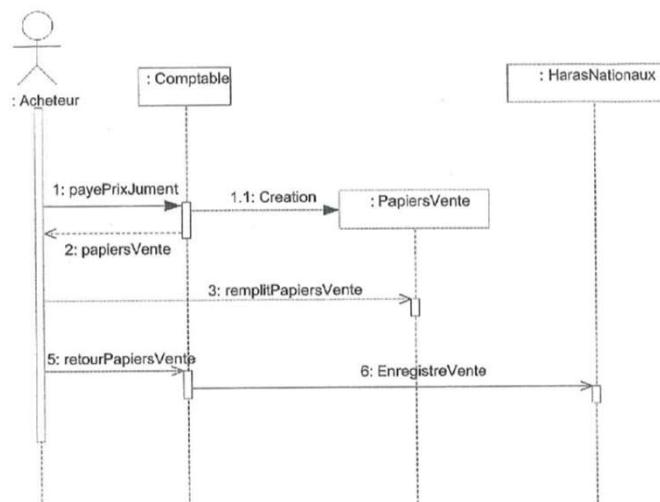


- ▶ Exemple d'objet: Papier Vente
 - Diagramme état-transition: cycle de vie de ces documents

Diagramme état-transition

Conclusion méthodologique

► Objet: Papier Vente



PapierVente
dateCréation : Integer dateEnvoi : Integer dateEnregistrement : Integer
éditerPapier() enregistrerPapier() remplirPapier() envoyerPapier()

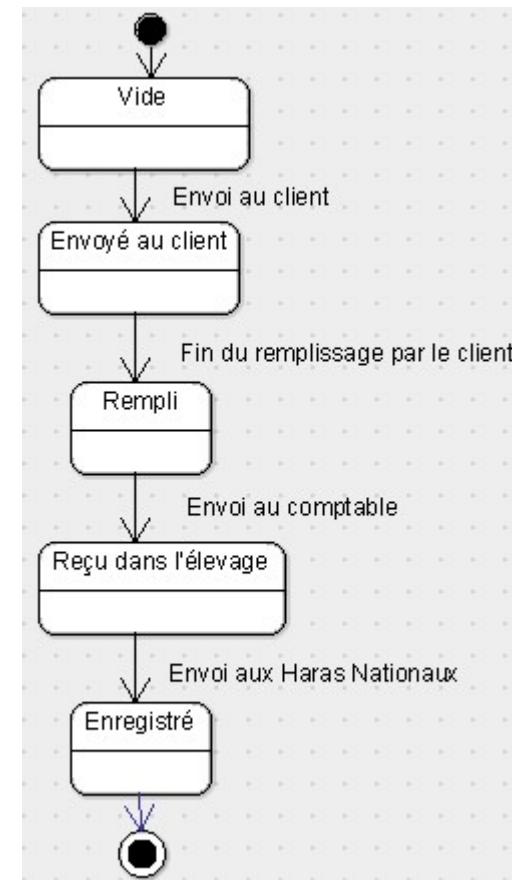


Diagramme état-transition

Exercice d'application

► Décrire le cycle de vie de la fiche d'un candidat lors d'un casting par **un diagramme état transition** sachant que:

- La fiche est créée par le responsable administratif avec le nom et les coordonnées du candidat.
- La fiche est envoyée au candidat après le casting.
- La fiche est archivée en fin de casting.



Diagramme état-transition

Exercice d'application



Diagramme d'activité

Définition

- ▶ Forme spécifique du diagramme d'états-transitions:
 - chaque état est associé à une activité
 - toutes les transitions sont automatiques
- ▶ Transitions: appelées **enchaînements**
- ▶ Etendu pour décrire les activités de plusieurs objets
- ▶ Intérêt: description des enchaînements entre les activités de différents objets ce que ne permet pas un diagramme états-transitions
- ▶ Utile pour décrire tous les points à préciser dans le modèle:
 - les cas d'utilisation: activités vues par les acteurs
 - les méthodes
 - ...



Diagramme d'activité

Activités et enchaînements d'activités

- ▶ Activité: série d'action
- ▶ Action:
 - plus petit traitement exprimable en UML
↔ instruction élémentaire d'un langage de programmation
- ▶ Exemples d'actions:
 - affecter une valeur à un attribut;
 - créer ou détruire un objet;
 - effectuer une opération;
 - envoyer un signal à un autre objet ou à soi-même.



Diagramme d'activité

Activités et enchaînements d'activités

► Notation UML:

- activité:



- activité initiale = première exécutée ●
- activité finale = fin de l'exécution de l'ensemble des activités d'un diagramme (pas obligatoire et pas forcément unique) ○



Diagramme d'activité

Activités et enchaînements d'activités

► Enchaînement d'activités:

- lien orienté entre deux activités
- franchi dès que l'activité d'origine est terminée
- son franchissement provoque l'enclenchement de l'activité de destination

► Notation UML:

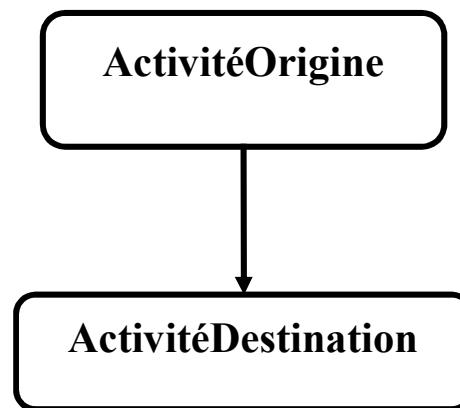


Diagramme d'activité

Activités et enchaînements d'activités

► Différents types d'enchaînement d'activités:

- **alternative:**
 - > chaque branche est dotée d'une condition de garde
 - > exclusivité des conditions de garde: un seul choix
- **fourche:**
 - > plusieurs activités de destination
 - > dès qu'elle est franchie, toutes les activités de destination sont enclenchées en parallèle
- **synchronisation:**
 - > plusieurs activités d'origine et une seule de destination
 - > franchie quand toutes les activités d'origine sont terminées

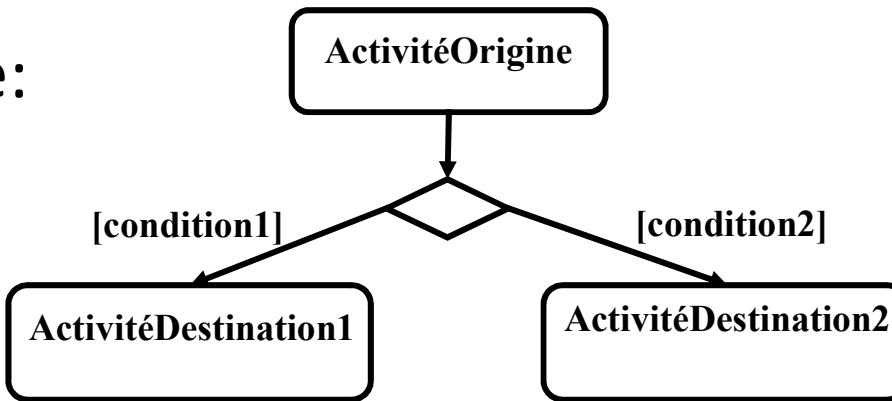


Diagramme d'activité

Activités et enchaînements d'activités

► Notation UML:

- alternative:



- fourche:

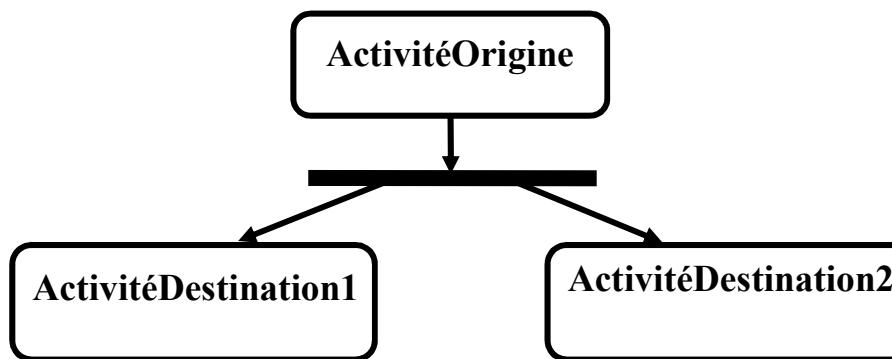


Diagramme d'activité

Activités et enchaînements d'activités

► Notation UML:

- synchronisation:

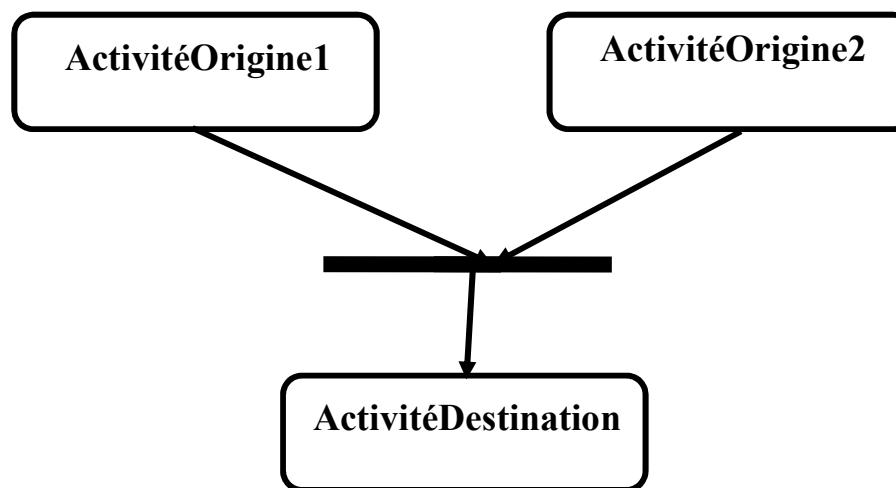


Diagramme d'activité

Activités et enchaînements d'activités

► Notation UML:

- Exemple

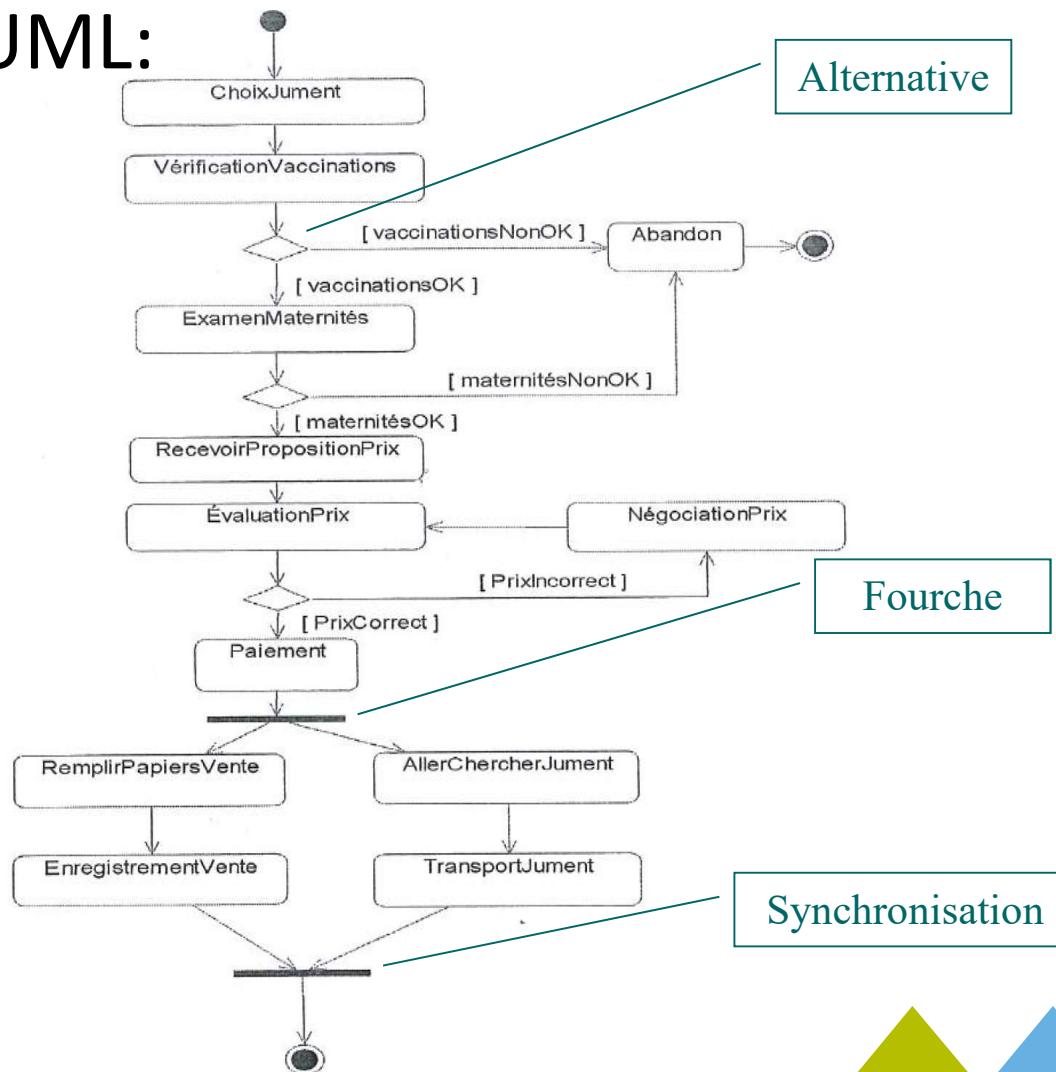
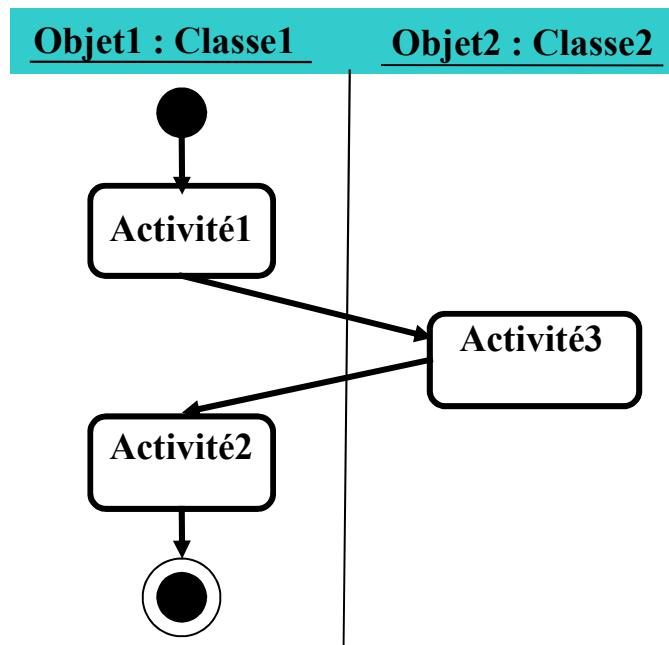


Diagramme d'activité

Activités et enchaînements d'activités

- ▶ **Travées ou couloirs:** divisions du diagramme d'activités regroupant toutes les activités dont un unique objet assure la réalisation
- ▶ enchaînement coupant la ligne de séparation de deux travées ⇒ changement d'objet entre l'activité d'origine et activité de destination
- ▶ Notation UML:



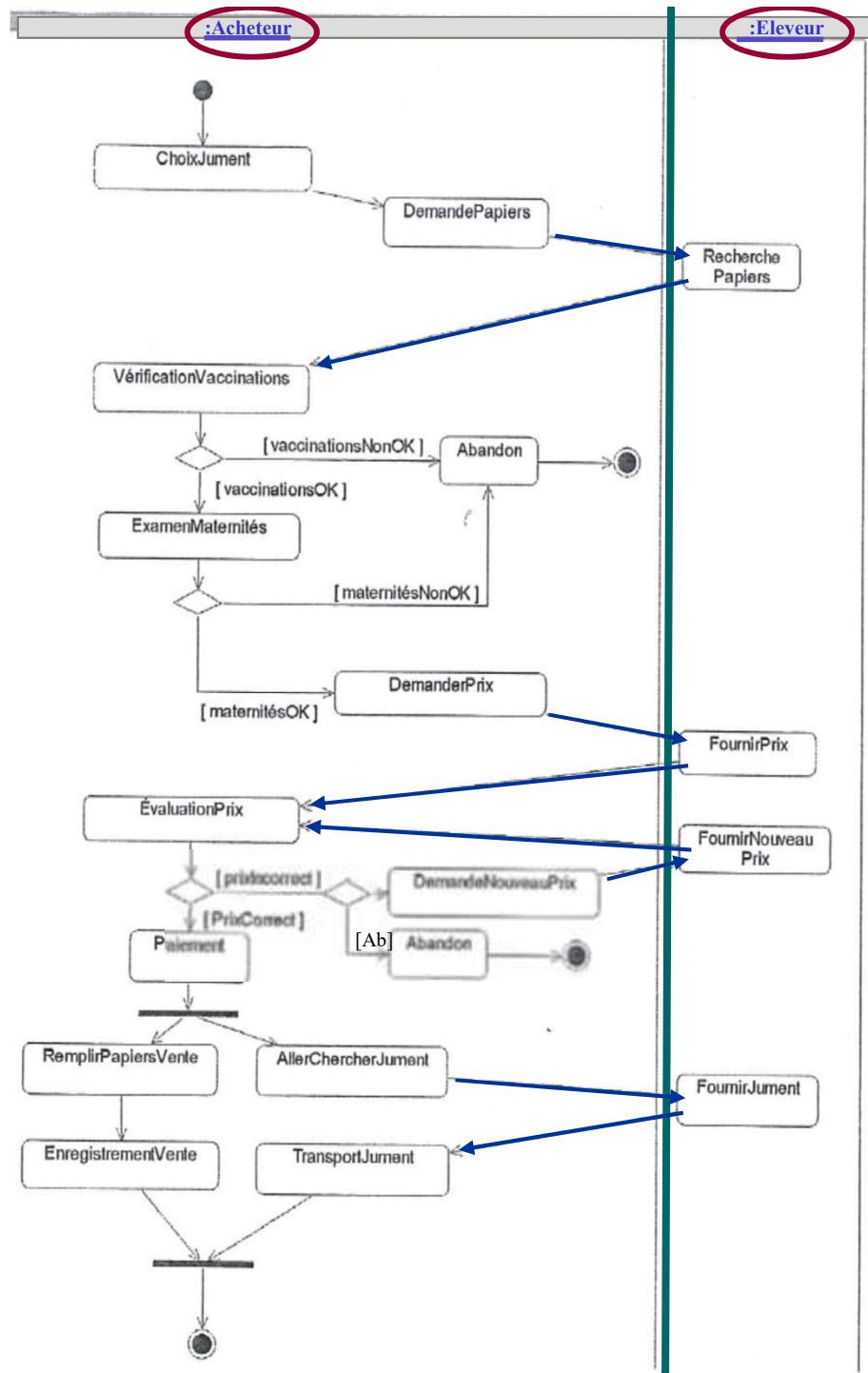


Diagramme d'activité

Activités et enchaînements d'activités

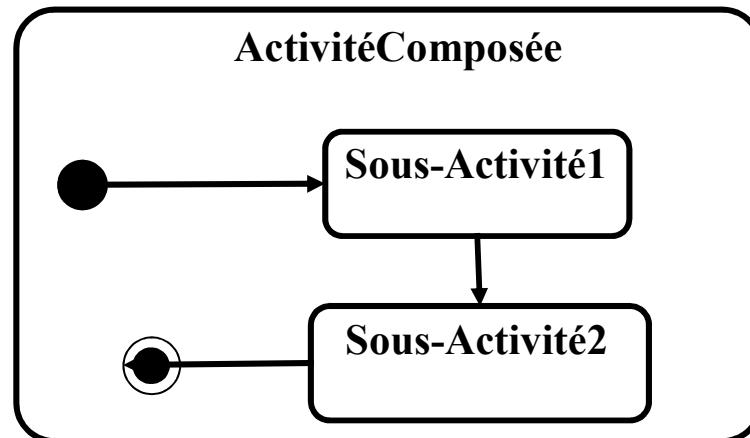
► Exemple



Diagramme d'activité

Activités composées

- ▶ Activité composée:
 - d'autres activités: sous-activités
 - décrite par un diagramme d'activité spécifique
- ▶ Notation UML:
 - composition d'une activité



- activité composée



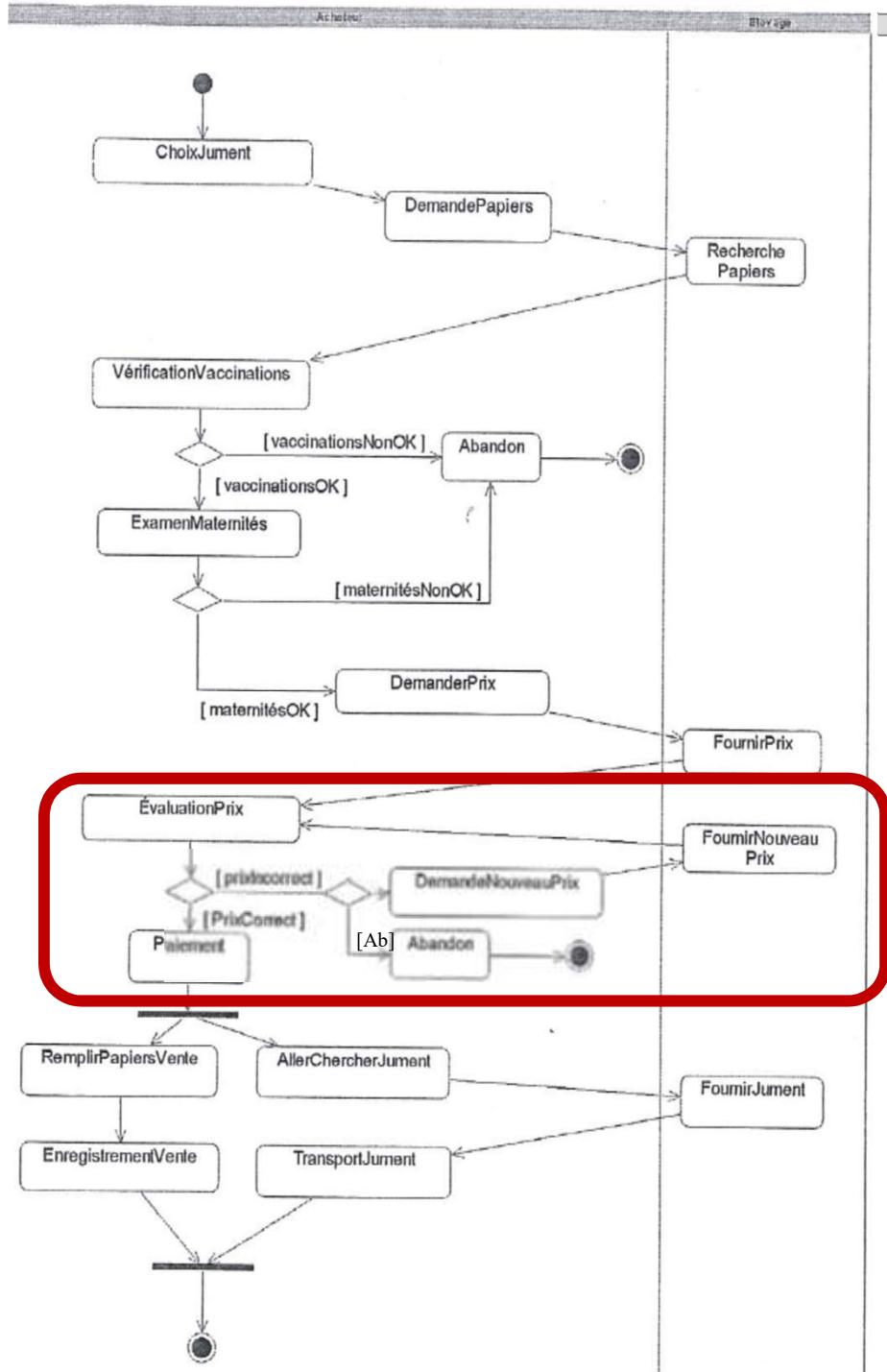


Diagramme d'activité

Activités composées

► Exemple:



Diagramme d'activité

Activités composées

► Exemple: définition de l'activité composée

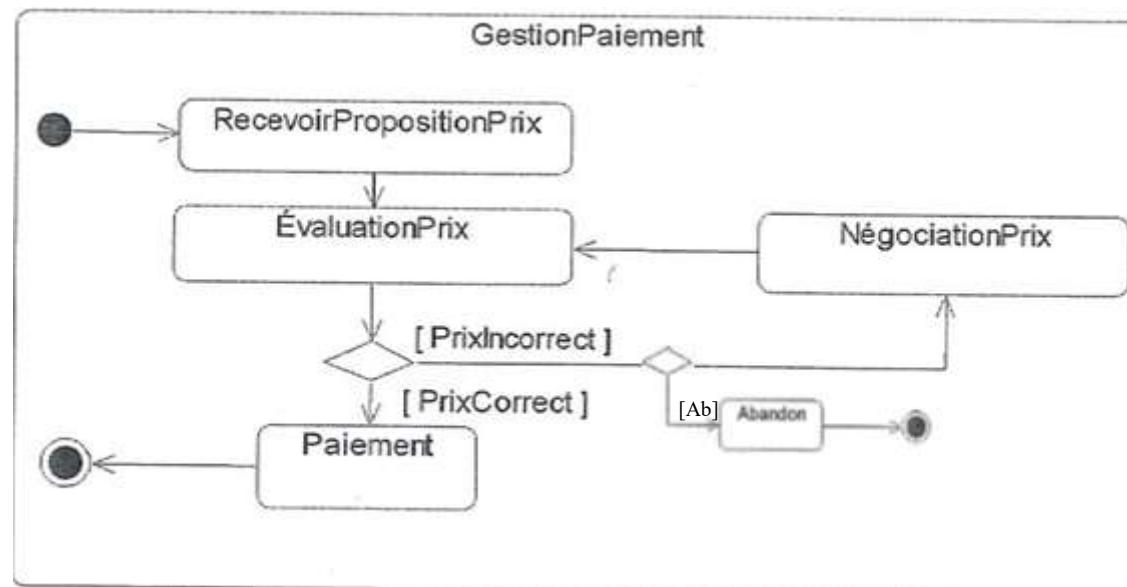


Diagramme d'activité

Activités composées

- Exemple: inclusion de l'activité composée

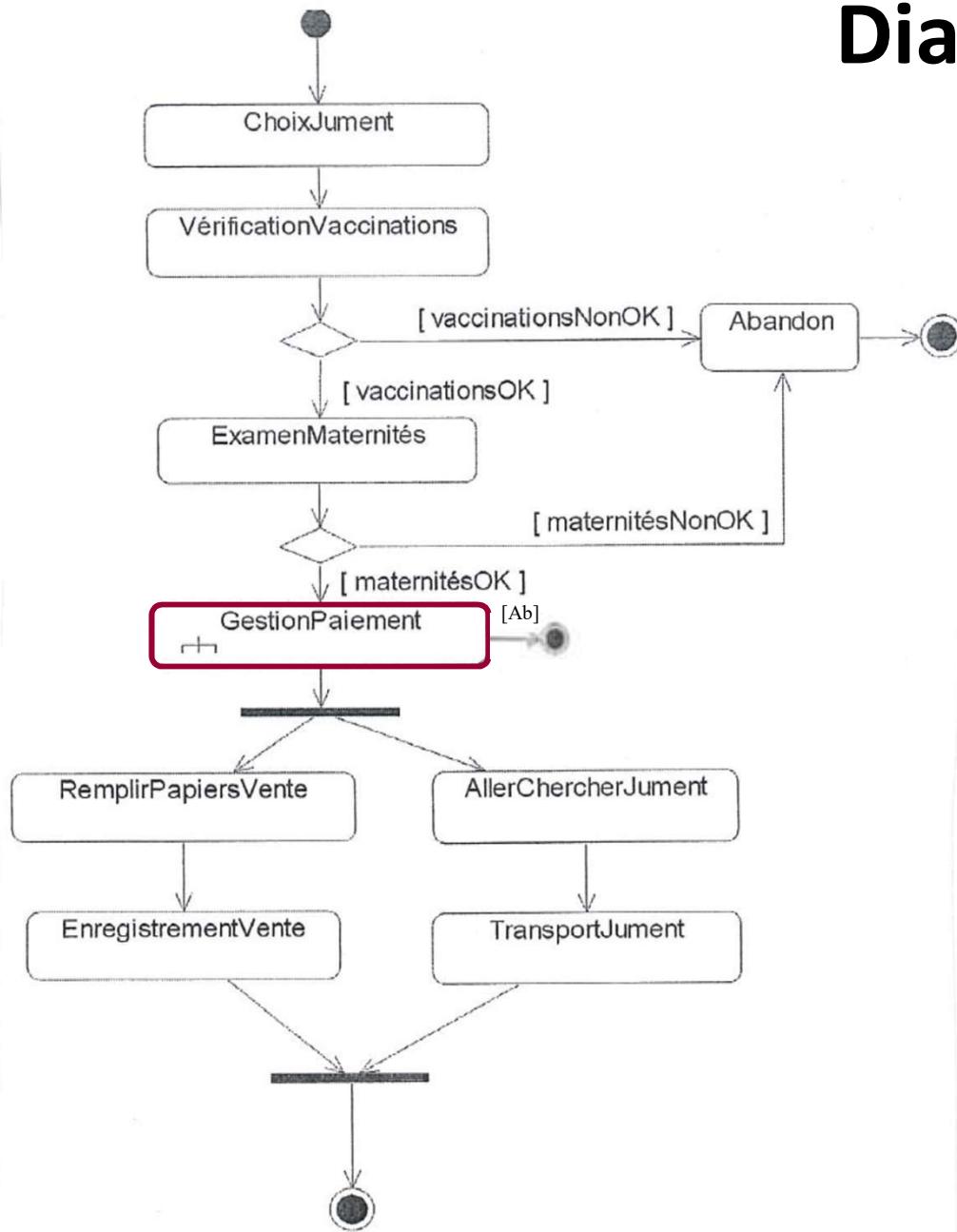


Diagramme d'activité

Diagramme de vue d'ensemble des interactions

- Définition: diagramme d'activités où chaque activité peut être décrite par un diagramme de séquence

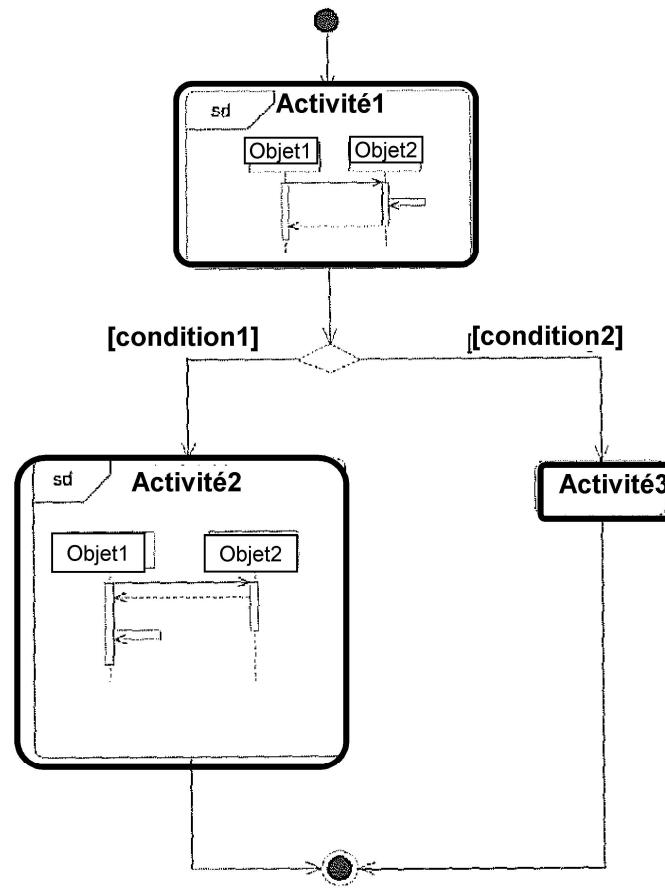


Diagramme d'activité

Conclusion

► Diagramme d'activités:

- représentation des activités que réalisent un ou plusieurs objets
- utile pour la description:
 - > d'une activité du diagramme états-transitions (un objet)
 - > d'une méthode
 - > de l'activité du système ou d'un sous-système en assignant les responsabilités à chaque acteur
 - > d'un cas d'utilisation



Diagramme d'activité

Conclusion: comparaison entre diagramme

	Diagramme d'état transition	Diagramme d'activité
Objectif	Cycle de vie	Interactions
Nombre d'objet représenté	1	1 ou plusieurs
Noeud	Etat	Activité
Activités	État/transition	Oui
Changement d'état/d'activité	Transition	Enchainement
	Événement/ condition de garde	Toujours automatique
Composé	État	Activité
Mémorisation	État historique	-



Diagramme d'activité

Exercice d'application

- Panier virtuel:



- Construire le diagramme d'état-transition du panier
- Construire le diagramme d'activité du cas d'utilisation Acheter un panier

Diagramme d'activité

Exercice d'application

- Diagramme d'état-transition du panier



Diagramme d'activité

Exercice d'application

- Diagramme d'état-transition du panier: état composé



Diagramme d'activité

Exercice d'application

- Diagramme d'état-transition du panier: état composé



Diagramme d'activité

Exercice d'application

- ▶ Diagramme d'activité du cas d'utilisation Acheter un panier



Méthodologie de conception

Diagrammes et outils utilisés en TD TP

- 1 ► Diagramme des cas d'utilisation → Exigences
- 2 ► Scénarios (textuel) → Enchaînement des interactions décrivant un cas d'utilisation
- 3 ► Diagramme de séquence (idem mais graphique)
 - Système → interactions entre système et acteurs
 - Détailé → découverte des objets du système
- 4 ► Diagramme de classe → description statique du système
(diagramme d'objet)
- 5 ► Diagramme d'état transition → cycle de vie d'un objet (vérification des méthodes des classes)
- 6 ► Diagramme d'activité → vérification des méthodes définition des cas d'utilisation, des méthodes, des activités...

2 bis

4 bis

Exemple d'enchaînement de développement



Méthodologie de conception

Introduction

► Trois phases du projet:

- Analyse orientée objet:

Recherche et description des objets du domaine du système

- Conception orientée objet:

Définition des objets logiciels et de leur collaboration

- Programmation orientée objet:

Implémentation des objets conceptuels



Méthodologie de conception

Introduction

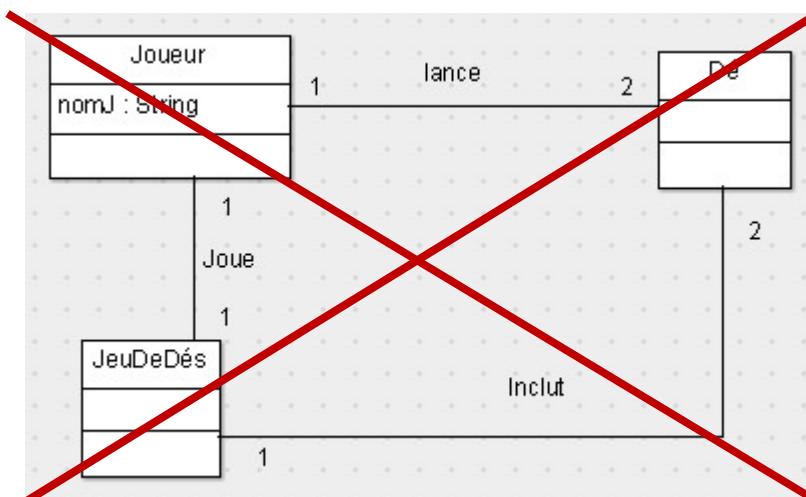
► Attention:

la conception et programmation objet s'inspire du monde réel mais n'en est ni une simulation ni un modèle direct

Exemple du Jeu de dés:

Monde réel: le joueur lance les dés

Conception objet: l'objet Système JeuDeDés jette les dés ie envoie des messages aux objets Dé



Méthodologie de conception

Capture des besoins : Etude préliminaire

- ▶ Définissez en priorité la frontière fonctionnelle du système
- ▶ Pensez que les acteurs candidats sont systématiquement:
 - les utilisateurs humains directs:
 - > différents profils d'utilisateurs
 - > administrateur
 - > opérateur de maintenance
 - > ...
 - les autres systèmes connexes qui interagissent directement avec le système:
 - > BD à distance
 - > Fournisseur d'énergie
 - > ...



Méthodologie de conception

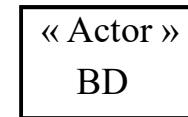
Capture des besoins : Etude préliminaire

- Pensez à stéréotyper les acteurs pour fournir des icônes plus évocatrices au lecteur.

Acteur humain



Acteur système informatique



(classeur)

- Utilisez un diagramme de séquence pour représentez le contexte dynamique:
- système = objet central
 - un objet par acteur
 - messages:
 - > liens entre système et acteurs
 - > décrits textuellement à part
- Ne pas oubliez de recueillir les besoins fonctionnels des futurs utilisateurs



Méthodologie de conception

Capture des besoins : Etude préliminaire

► Ne recensez pas comme acteur:

- des entités externes qui interagissent avec le système par le biais d'un des vrais acteurs
- des entités internes au système (voire des futures classes)

► Ne confondez pas rôle et entité concrète:

- une entité peut jouer successivement différents rôles et être modélisée par différents acteurs
- un même rôle peut être tenu simultanément par plusieurs entités concrètes modélisées par le même acteur



Méthodologie de conception

Capture des besoins : Besoins fonctionnels

- ▶ Déterminez les besoins fonctionnels en défendant le point de vue des acteurs
- ▶ Distinguez l'acteur principal des acteurs secondaires
- ▶ Chaque cas d'utilisation doit être prédéfini:
 - intention de l'acteur
 - quelques séquences d'actions

⇒ identification des cas d'utilisation (et non définition)
- ▶ Détaillez les rôles (acteur principal ou secondaire) et le sens des associations dans le diagramme des cas d'utilisation
- ▶ Limitez le nombre de cas d'utilisation à 20 (et même moins...)
- ▶ Utilisez la possibilité de définir un acteur abstrait pour factoriser un rôle commun

