Modélisation orientée objets avec UML

Guillaume Laurent

ENSMM

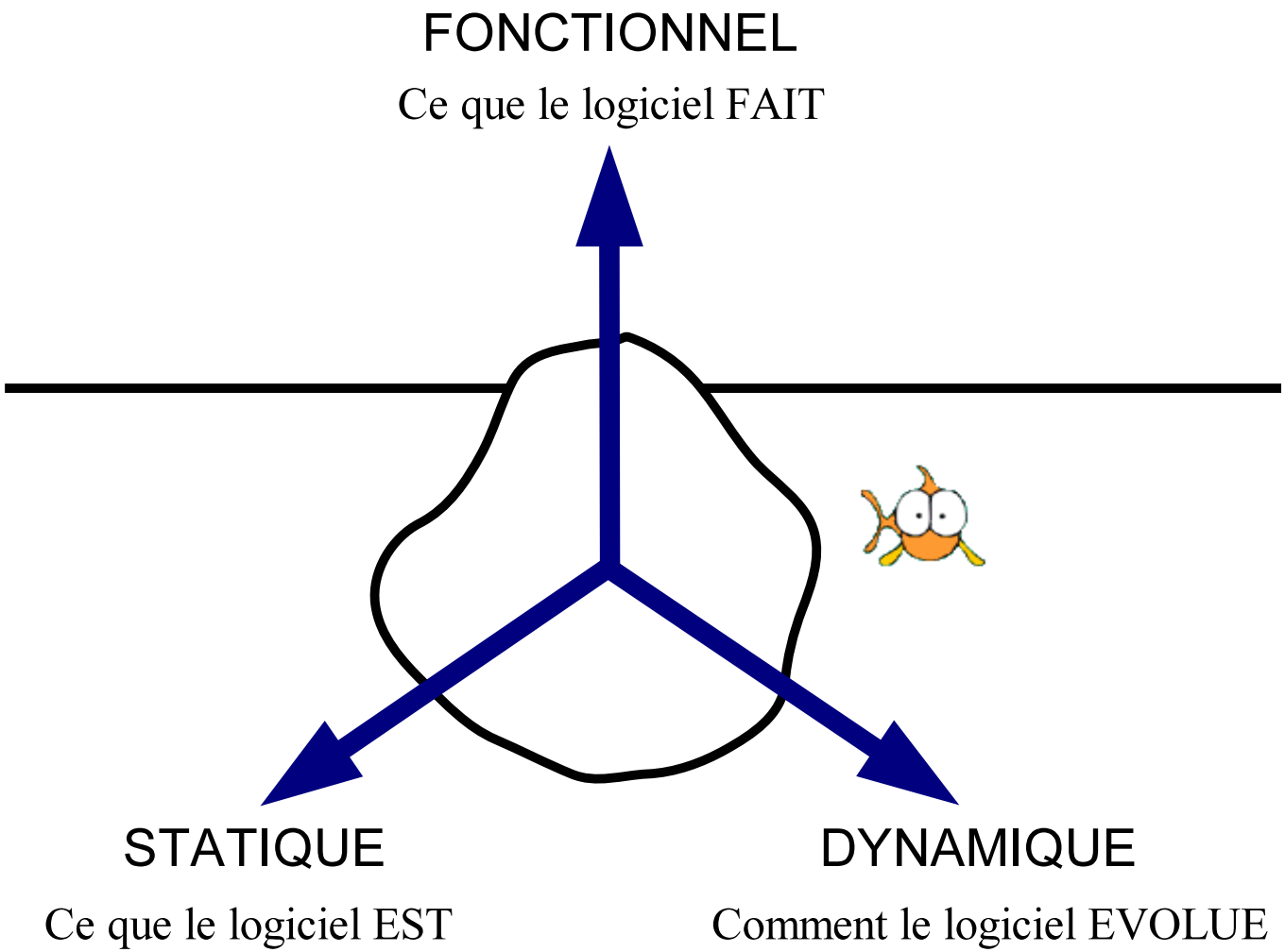
2016

Modéliser, pour quoi faire ?

- Objectifs
  - Identifier les caractéristiques pertinentes d'un système dans le but de pouvoir l'étudier
  - Faciliter la compréhension du système, synthétiser son fonctionnement (par l'abstraction et la simplification)
  - Normaliser
  - Simuler le comportement du système futur
  - Gérer le risque (état d'avancement, découverte de problèmes, etc.)
  - Communiquer

- Remarques
  - Le choix du modèle initial a une grande influence
  - Les meilleurs modèles sont ceux qui sont connectés à la réalité
  - Un modèle peut être exprimé à divers niveaux de précision
  - On peut utiliser plusieurs modèles (fonctionnels, structurels, etc.) pour décrire tous les différents aspects d'un système complexe

Axes de modélisation d'un logiciel



Modélisation formelle

- Principe : toutes les activités du logiciel sont validées par des preuves formelles, conception obtenue par raffinements successifs de la machine abstraite au code

- Intérêts/inconvénients
  - Comportement du logiciel garanti
  - Bien adaptée aux processus de développement séquentiels
  - Coûts et délais de développement importants

Modélisation fonctionnelle et structurée

- Principe : décomposer le système selon les fonctions qu'il doit réaliser par une analyse descendante modulaire

- Intérêts/inconvénients
  - Bon outil de communication pour réaliser les spécifications fonctionnelles
  - Possibilité de vérifier la cohérence du modèle (modèle semi-formel)
  - Bien adaptée aux processus de développement séquentiels
  - Modélisation partielle du logiciel
  - Conçue initialement pour des applications de gestion mais également utilisée dans les domaines de la production et des systèmes automatisés
  - Souvent associée à d'autres outils (grafcet, statecharts, réseaux de Pétri)

Modélisation orientée objets

- Principe : décomposer le logiciel en un ensemble d'entités (objets) qui interagissent entre elles (objet = données + fonctions)

- Intérêts/inconvénients
  - Réduction des coûts de développement grâce à la modularité, à la réutilisabilité et à la compacité du code
  - Réduction des coûts de maintenance grâce à l'encapsulation (18% des coûts de maintenance sont dûs à un changement de structure de données)
  - Bien adaptée aux processus de développement itératifs
  - Passage délicat de la spécification à la conception
  - Possibilité de vérifier la cohérence du modèle avec OCL (Object Constraint Language) par exemple

Qu'est-ce que UML ?

- UML est un langage universel de modélisation graphique pour l'ingénierie logiciel
- UML est un outil de communication visuelle (14 diagrammes)
- UML est une notation normalisée maintenue par l'organisation à but non lucratif OMG (Object Management Group) et disponible gratuitement sur *www.uml.org*

- UML n'est pas un processus de développement
- UML n'est pas un langage de programmation

Les principaux diagrammes UML

- **Diagrammes de classes** : décrivent les classes d'une application et leurs relations statiques
- **Diagrammes d'objets** : montrent l'état d'une application à un instant donné
- **Diagrammes de communication** : sont une représentation spatiale des interactions entre objets
- **Diagrammes de séquence** : sont une représentation temporelle des interactions entre objets
- **Diagrammes d'activité** : représentent un processus sous la forme d'un organigramme
- **Diagrammes d'états-transitions** : représentent un processus sous la forme d'un automate déterministe
- **Diagrammes de déploiement** : modélisent l'aspect matériel de l'application
- **Diagrammes de cas d'utilisation** : décrivent les services rendus par le système du point de vue de l'utilisateur

Logiciels de modélisation UML

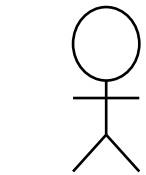
- Fonctionnalités
  - Éditeur graphique
  - Patrons de conception (design patterns)
  - Génération de code
  - Rétro-génération (reverse engineering)
  - Synchronisation code/UML

- Logiciels libres
  - ArgoUml
  - Modelio
  - Papyrus
  - UMLet (très léger)

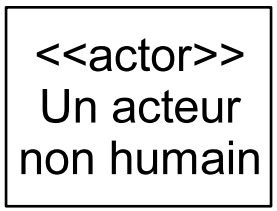
Diagrammes de cas d'utilisation

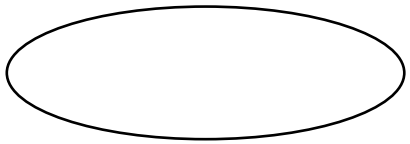
Diagrammes de cas d'utilisation

- Identifier les acteurs
  - Un acteur représente un rôle joué par une entité extérieure au système (humain ou autre système) et qui interagit directement avec le système étudié
  - Un utilisateur peut être représenté par plusieurs acteurs
  - Plusieurs utilisateurs peuvent être représentés par le même acteur



Un acteur



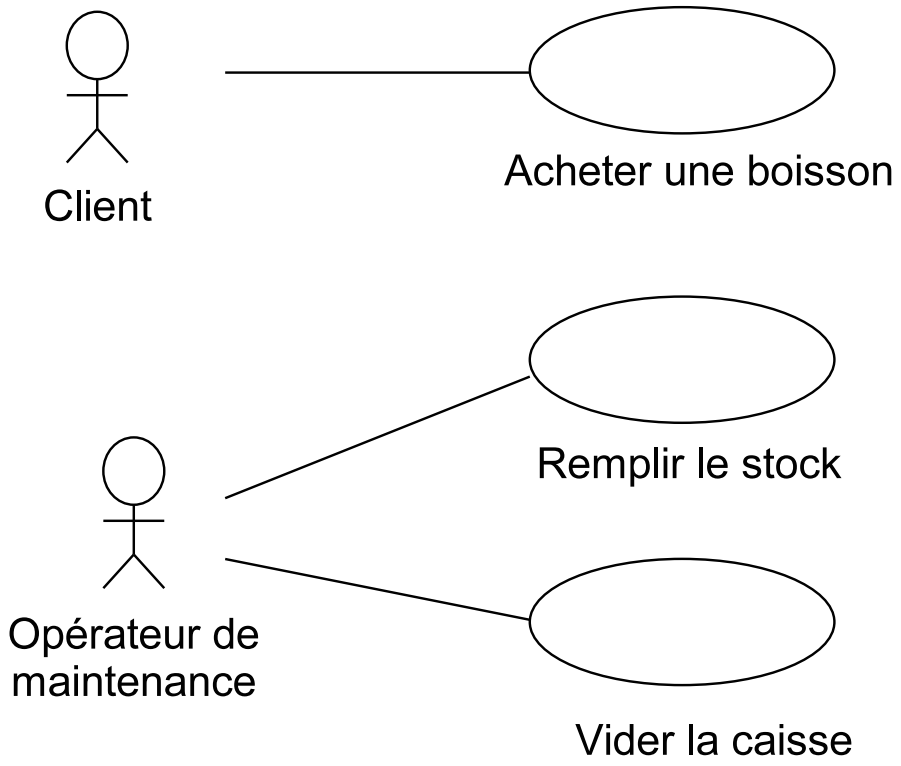


Un cas d'utilisation

- Identifier les cas d'utilisation
  - Un cas d'utilisation représente un service complet attendu du système
  - Un cas d'utilisation a un début et une fin clairement identifiés
  - Un cas d'utilisation est décrit par une forme verbale

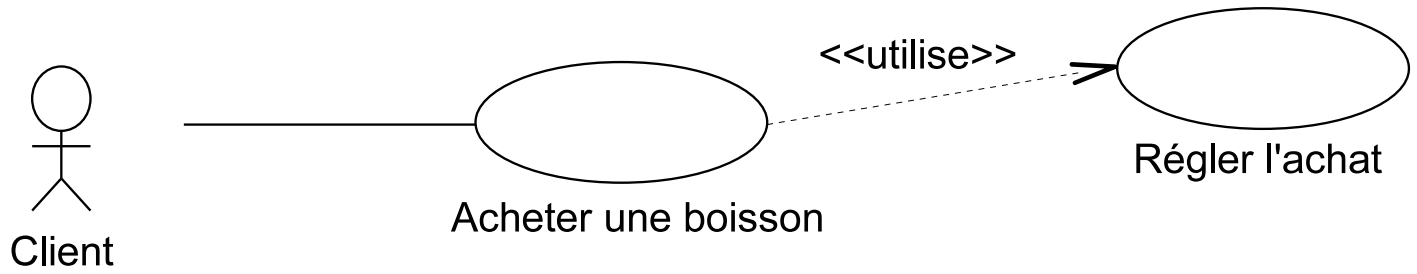
Diagrammes de cas d'utilisation

- Exemple des cas d'utilisation d'un distributeur de boissons

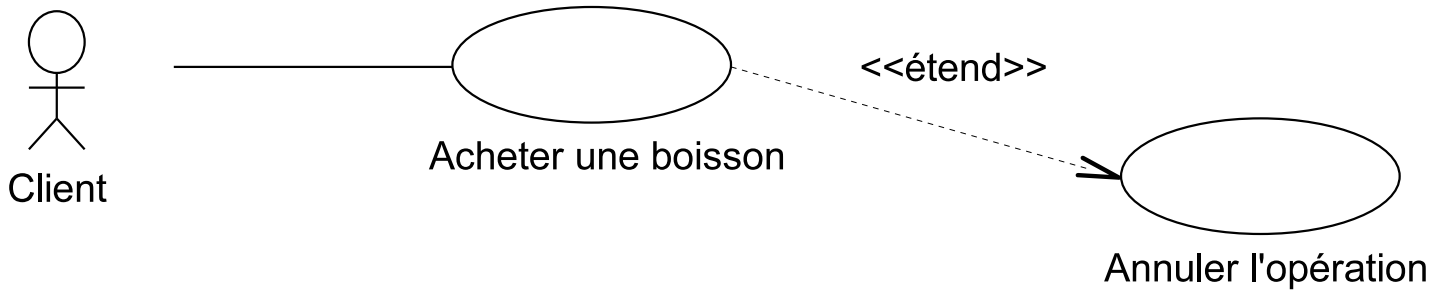


Diagrammes de cas d'utilisation

- Relation d'utilisation (include)

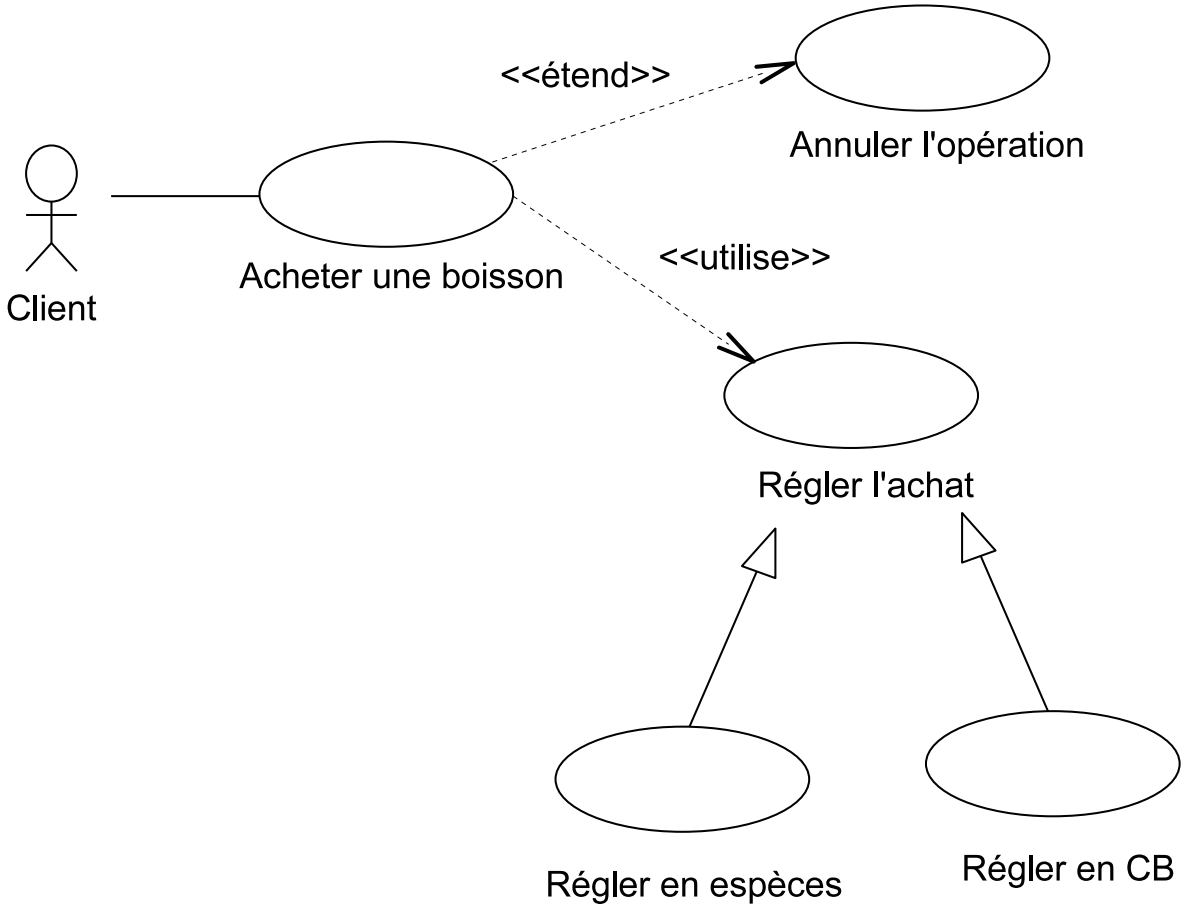


- Relation d'extension (extend)



Diagrammes de cas d'utilisation

- Relation de spécialisation



Exercices

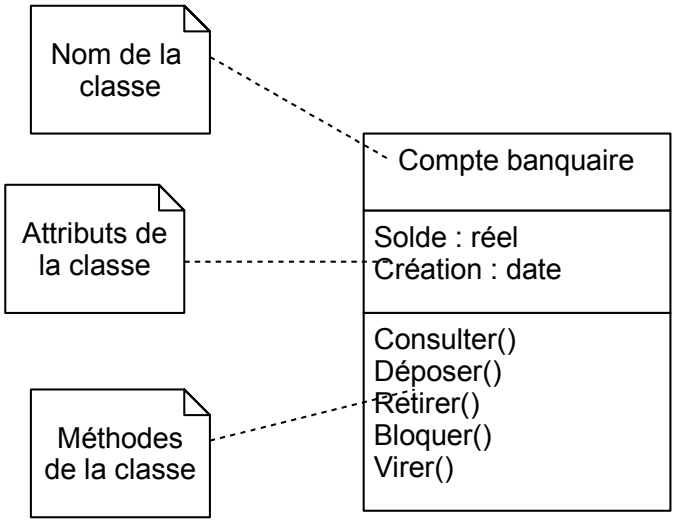
Ex.1 Modéliser les cas d'utilisation d'un logiciel de jeu d'échecs

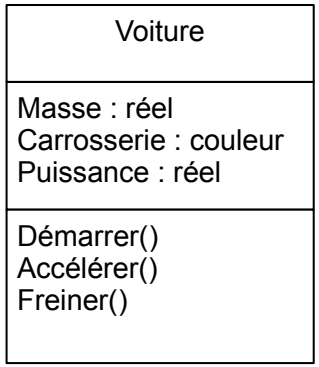
Ex.2 Modéliser les cas d'utilisation d'une caisse enregistreuse (extrait de *UML par la pratique*, Pascal Roques, Eyrolles)

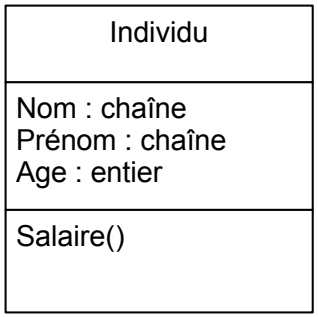
- Le déroulement normal d'utilisation de la caisse est le suivant :
  - Un client arrive à la caisse avec des articles à payer
  - Le caissier enregistre le numéro d'identification de chaque article, ainsi que la quantité si elle est supérieure à un
  - La caisse affiche le prix de chaque article et son libellé
  - Lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente
  - La caisse affiche le total des achats
  - Le client choisit son mode de paiement : liquide (le caissier encaisse l'argent reçu, la caisse indique la monnaie à rendre au client), chèque , carte de crédit (la caisse transmet une demande d'autorisation à un centre d'autorisation)
  - La caisse enregistre la vente et imprime un ticket.
  - Le caissier donne le ticket de caisse au client.
- Lorsque un paiement est terminé, la caisse transmet les informations sur le nombre d'articles vendus au système de gestion de stocks.
- Tous les matins, le responsable du magasin initialise la caisse pour la journée.

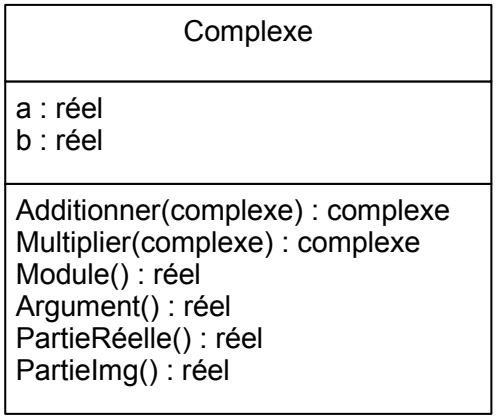
Diagrammes de classes

Représentation d'une classe

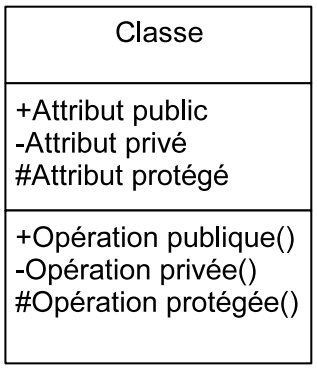


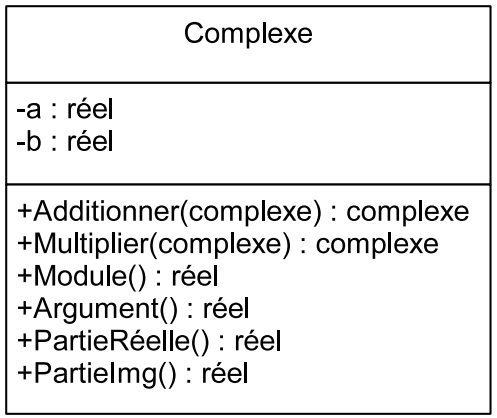


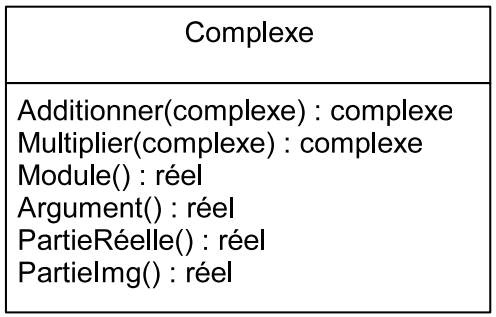




Visibilité des attributs et des méthodes

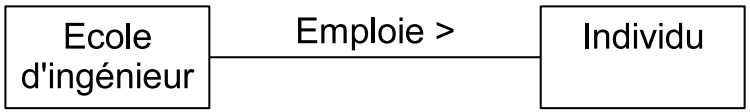


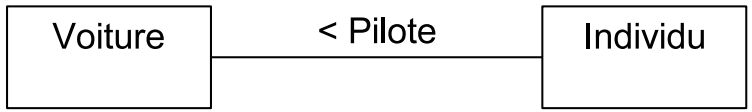




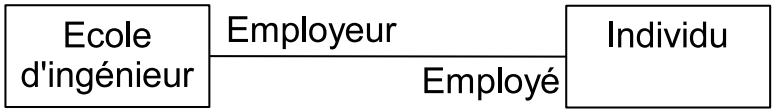
Associations de classes

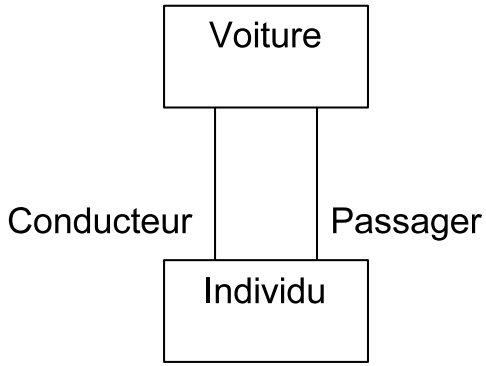
- Reflet d'une relation entre deux classes (ou plus)
- L'invocation d'une méthode (envoi de messages) est une association
- Une association peut être nommée

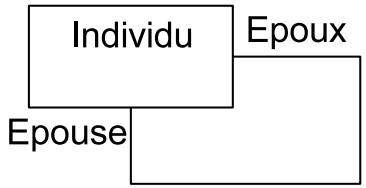




- Possibilité de décrire les rôles

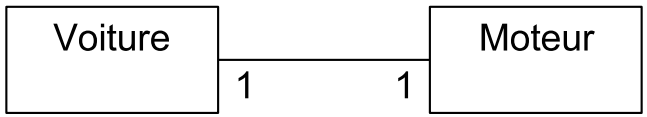


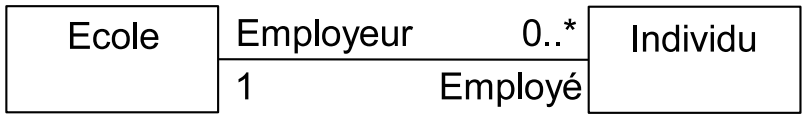


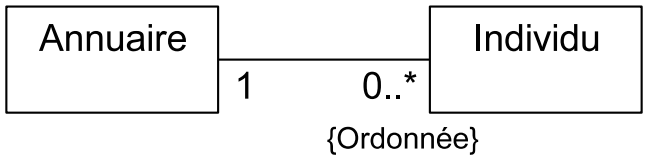


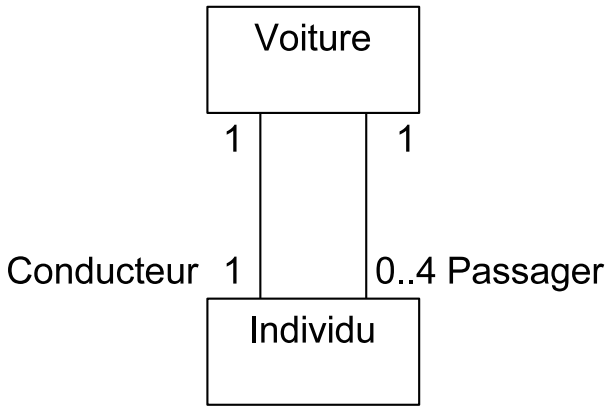
Associations de classes

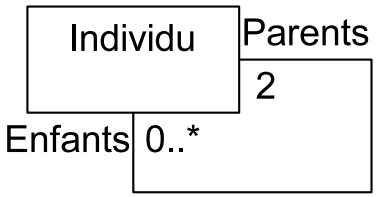
- Multiplicité des associations





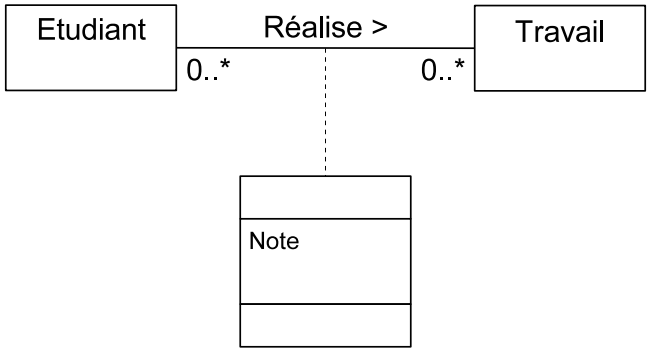


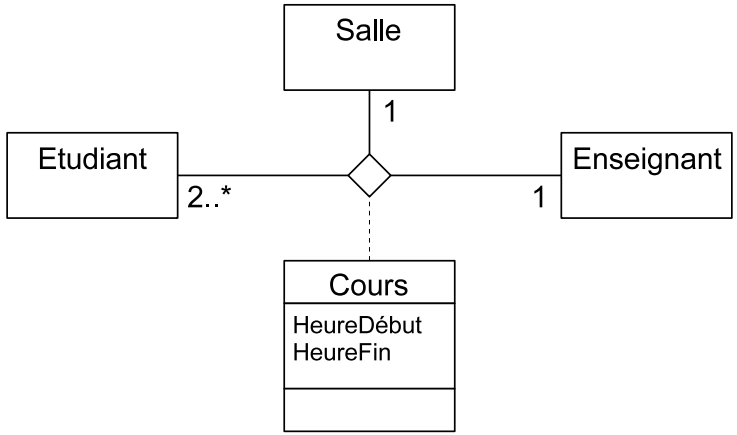




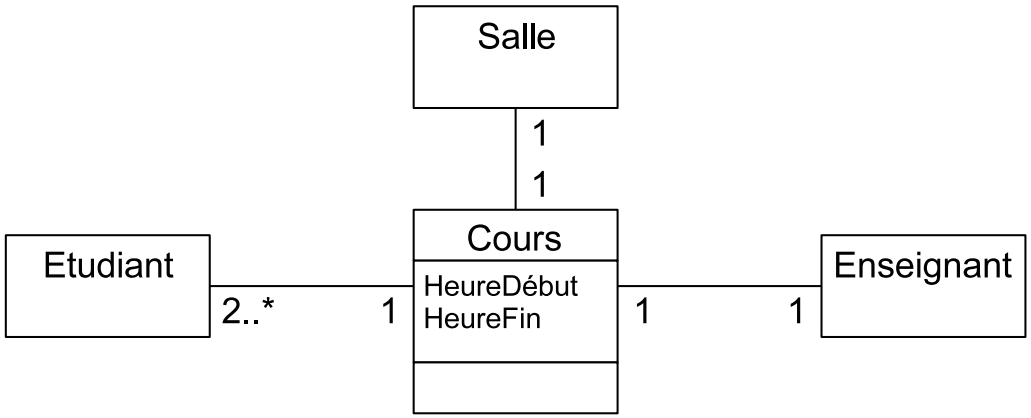
Classes d'association

- Une association peut être représentée par une classe (pour être enrichie par exemple)



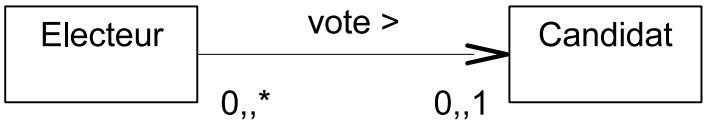


- Les associations n-aires peuvent souvent être réduites



Navigabilité

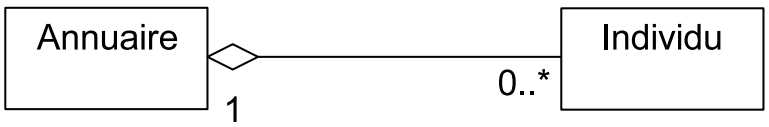
- Par défaut, les associations sont navigables dans les deux directions
- Une association peut ne nécessiter qu'un seul sens de navigation :

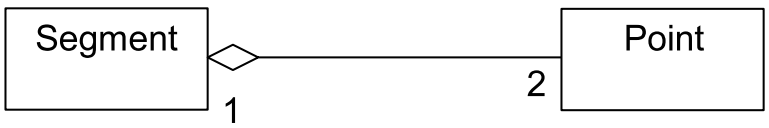


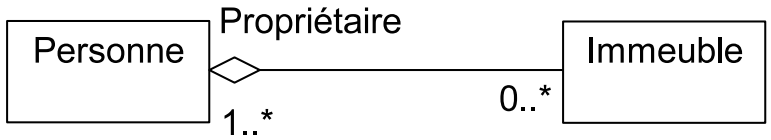
- B ne ≪ voit ≫ pas A
- B ne pas invoquer de méthode de A

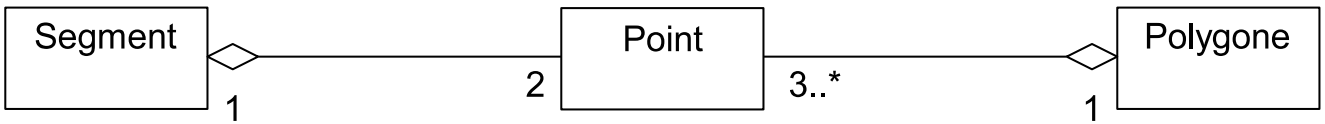
Agrégations de classes

- L'agrégation est une association non symétrique :
  - Relation de dominance et de subordination
  - Une classe fait partie d'une autre classe
  - Une action sur une classe implique une action sur une autre classe
- Une classe peut appartenir à plusieurs agrégats



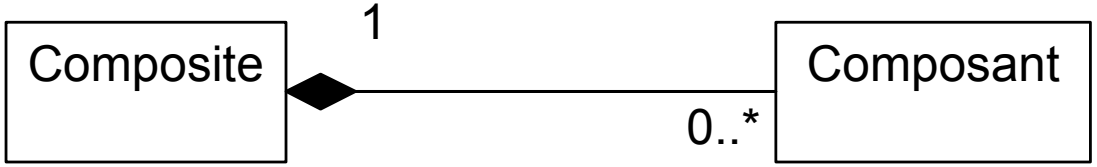


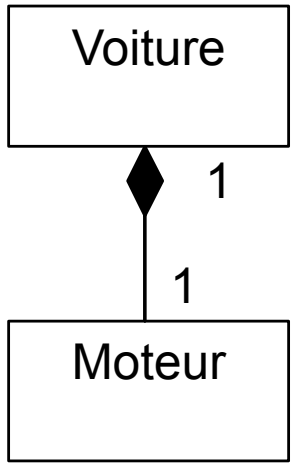




Compositions de classes

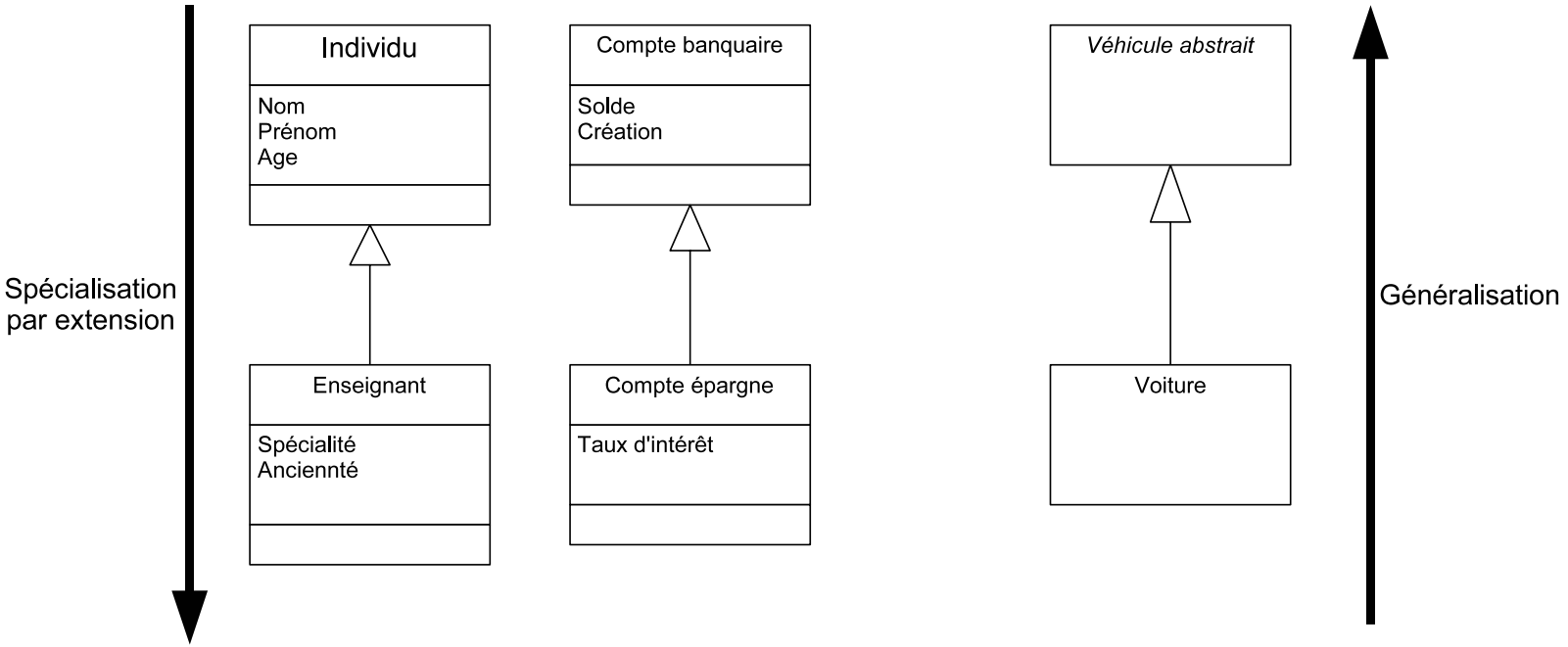
- La composition est une agrégation *forte*
- Relation d'appartenance stricte équivalente à l'attribution
- Une classe ne peut pas appartenir à plusieurs compositions
- Une classe composée ne peut exister sans son propriétaire





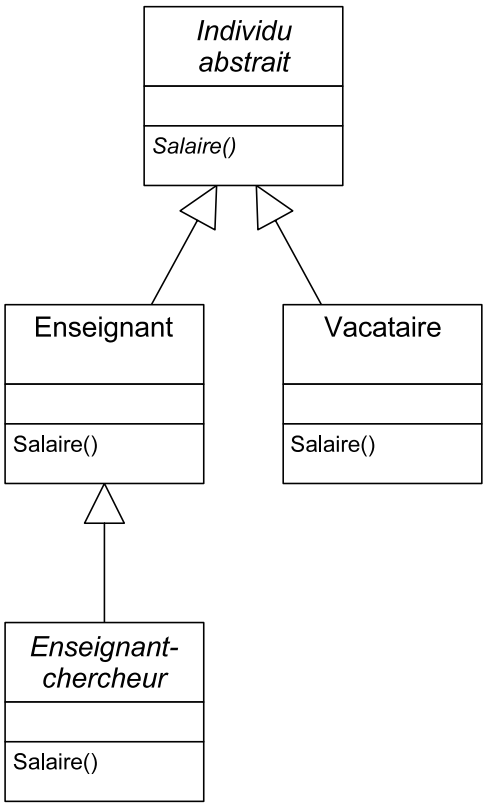
Généralisation/spécialisation

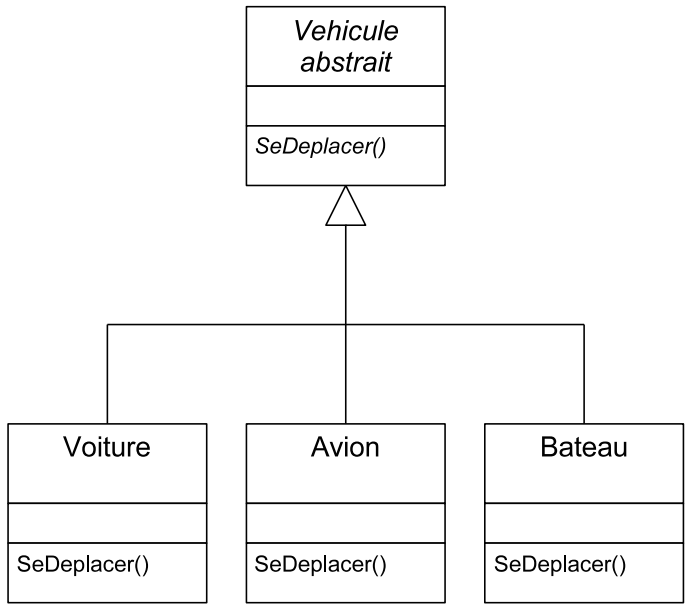
- Spécialisation de la classe existante par ajout d'attributs et/ou ajout/redéfinition de méthodes
- Relation « est un »
- La classe d'origine est plus générale que la classe dérivée



Polymorphisme

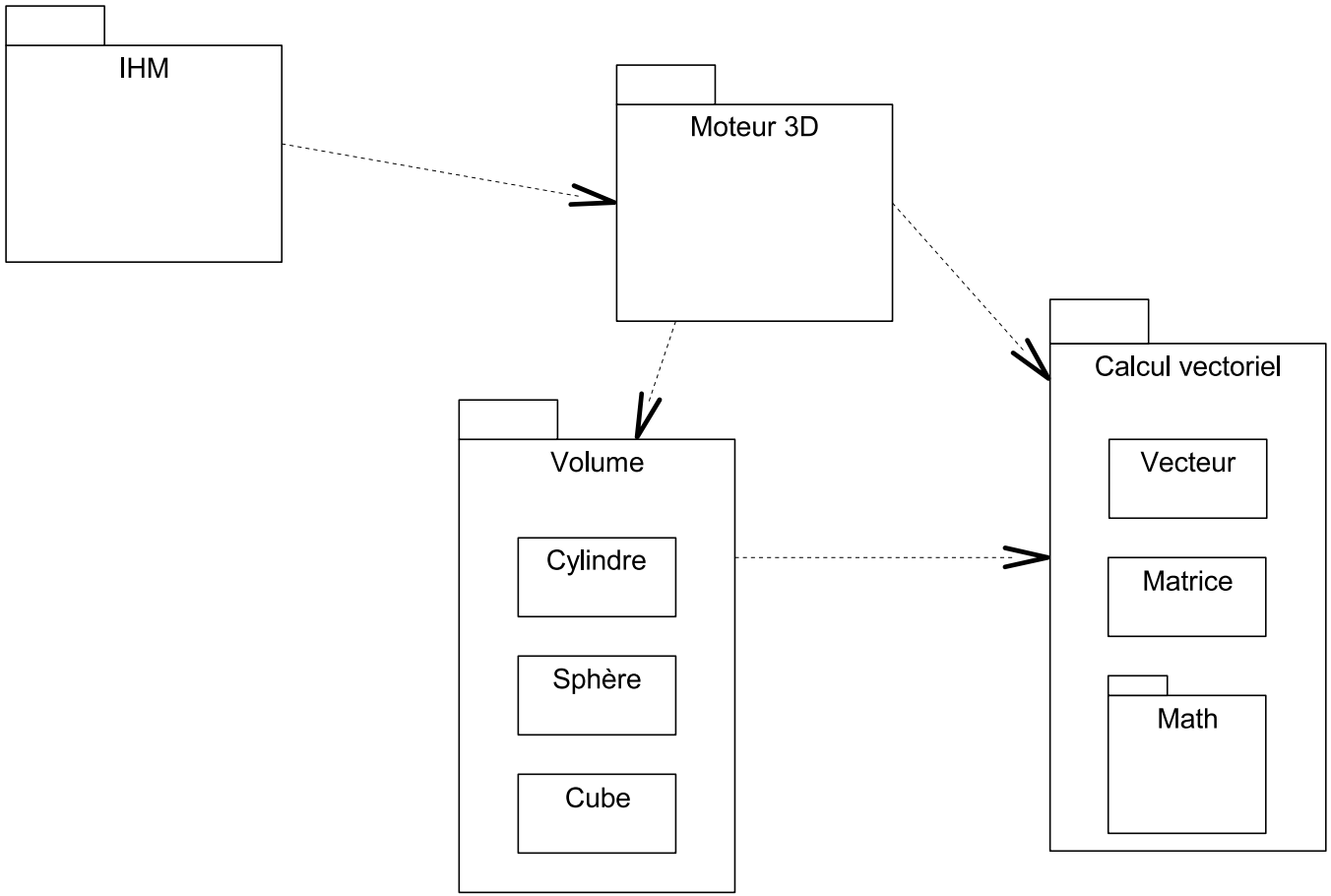
- Permet d'utiliser un nom unique pour désigner des instances de classes différentes issues d'une même arborescence
- Chaque objet désigné par ce nom réagit de manière propre à la réception d'un même message





Notion de paquetages (package)

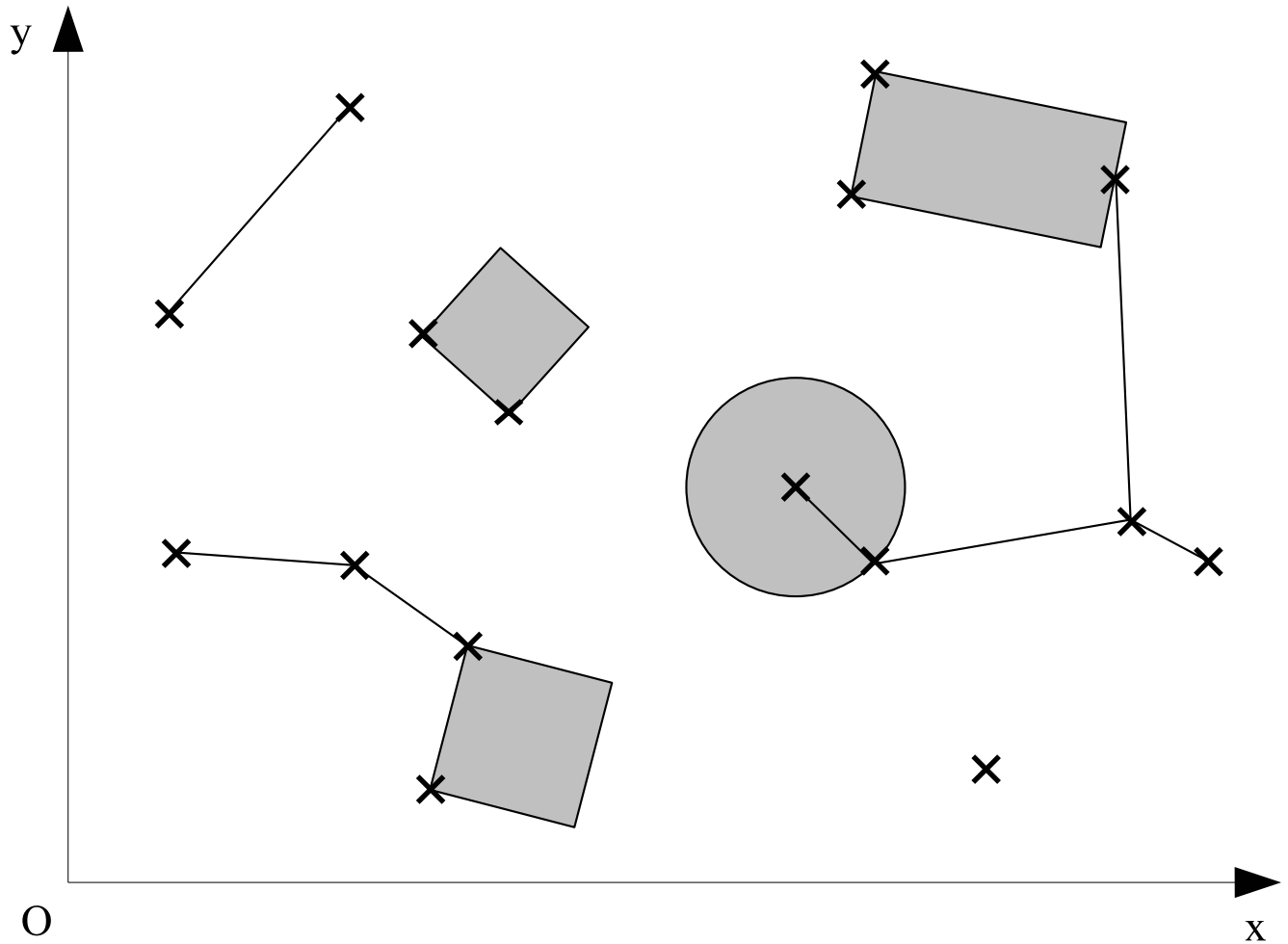
- Regrouper les classes très liées entre elles
- Faire apparaître (et minimiser !) les dépendances entre paquetages



Exercice

Ex.1 Soit le dessin géométrique ci-dessous :

- Modéliser cette situation par un diagramme d'objets
- Modéliser un dessin géométrique par un diagramme de classes



Exercice

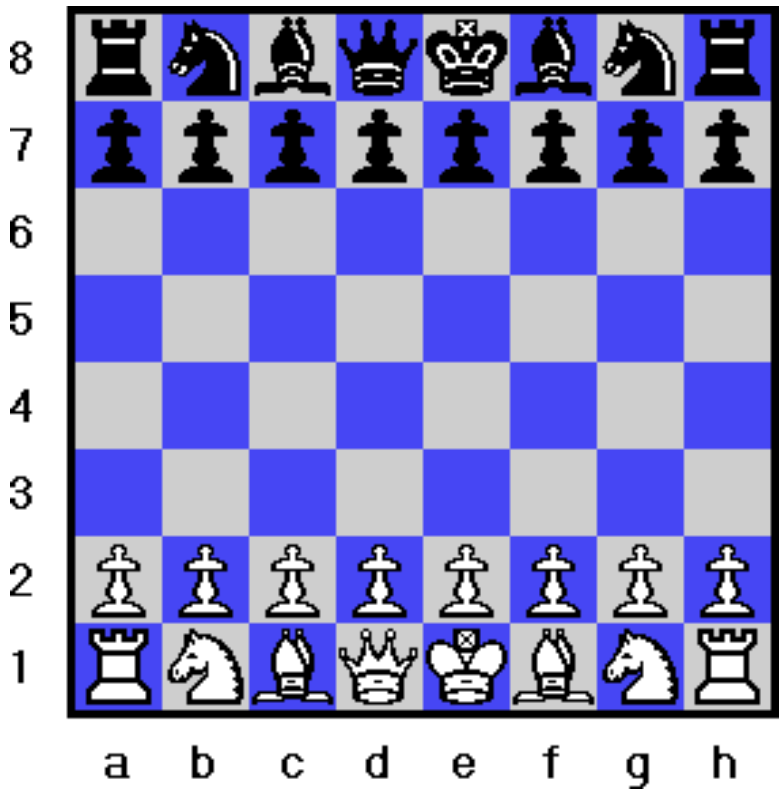
Ex.2 Modéliser les phrases suivantes par un diagramme de classes

- Un répertoire contient des fichiers
- Une pièce a des murs
- Une transaction bancaire est un achat ou une vente
- Une enveloppe a une adresse et un timbre
- Les lapins, les hirondelles, les lions, les requins, les autruches, les vautours, les truites, les pingouins sont des animaux qui se déplacent de manières différentes.
- Des poupées russes (gigognes)
- Un pays possède plusieurs villes et une seule capitale

Exercice

Ex.3 Modéliser la structure statique d'un logiciel de jeu d'échecs en se focalisant successivement sur :

- L'échiquier et les pièces
- Les joueurs et leurs pièces
- Une partie

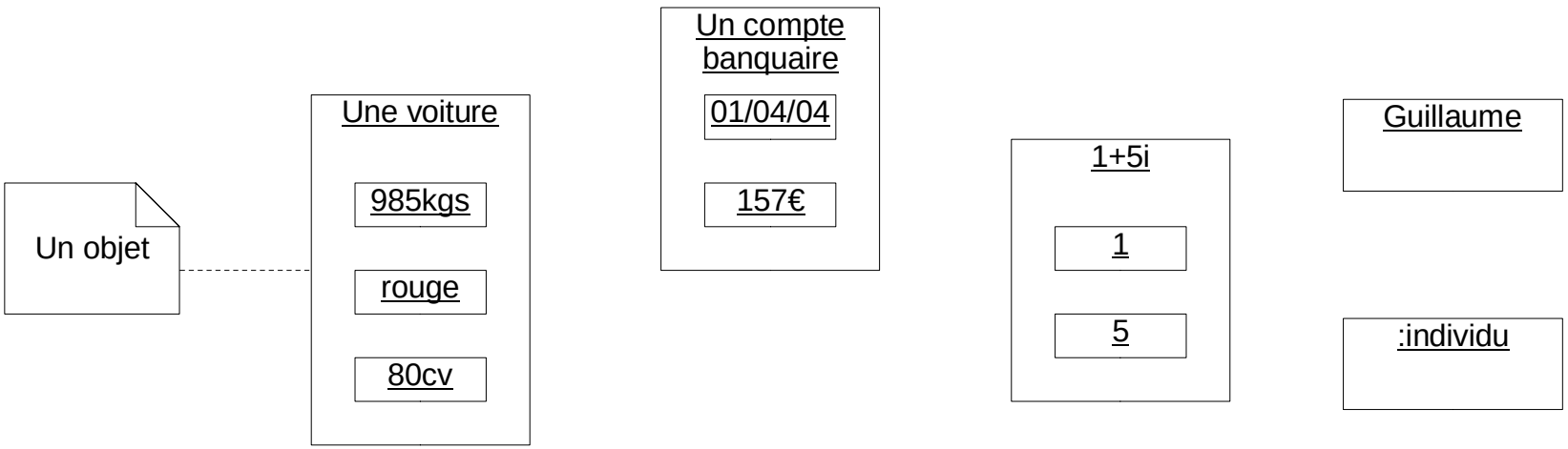


(extrait de *UML par la pratique*, Pascal Roques, Eyrolles)

Diagrammes d'objets

Diagrammes d'objets

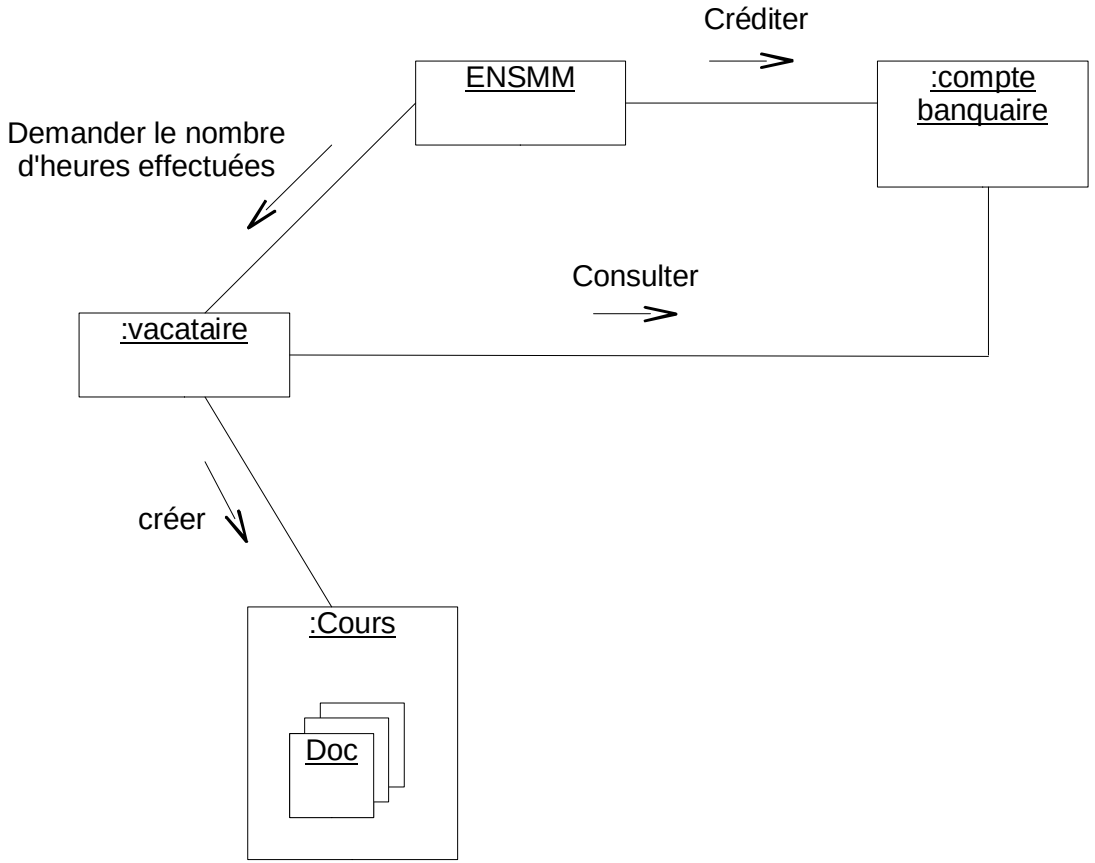
- Un objet est l'instance (la réalisation) d'une classe
- Un objet est défini par :
  - son état : valeurs des attributs à un instant donné
  - son identité : propriété qui permet de distinguer tout objet des autres (référence en java, adresse en C++)



Diagrammes de communication

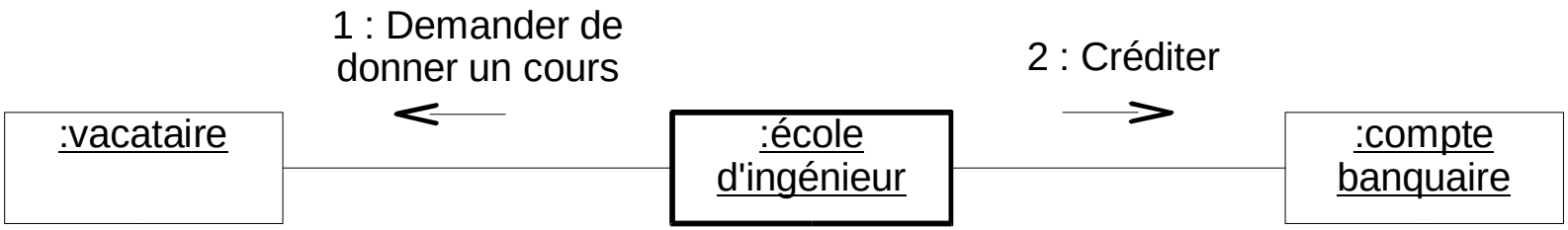
Concept de message

- Message : quand un objet (le client) demande à un autre objet (le serveur) d'effectuer une de ses opérations, on parle de message



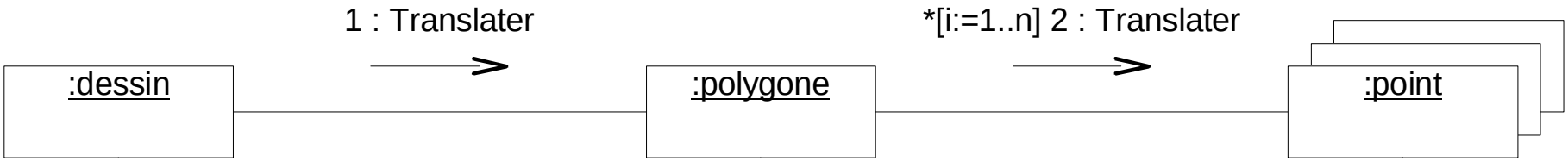
Diagrammes de communication

- Représentent les interactions (échanges de messages) entre objets selon un point de vue spatial
- Extension des diagrammes d'objets par l'ajout de la représentation des messages échangés
- Vues généralement partielles et focalisées sur une communication de quelques objets
- Les messages sont numérotés pour indiquer l'ordre des envois

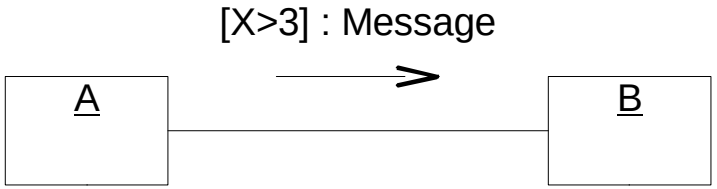


Diagrammes de communication

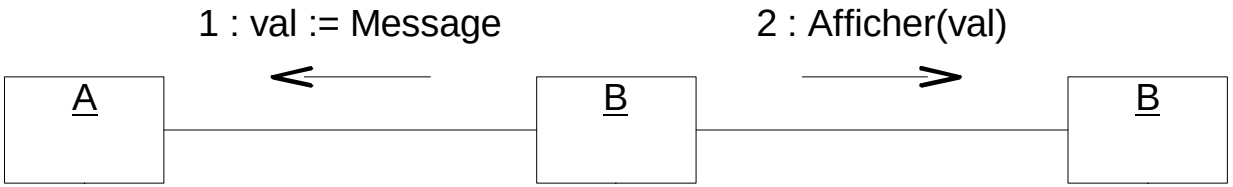
- Itérations



- Messages conditionnels

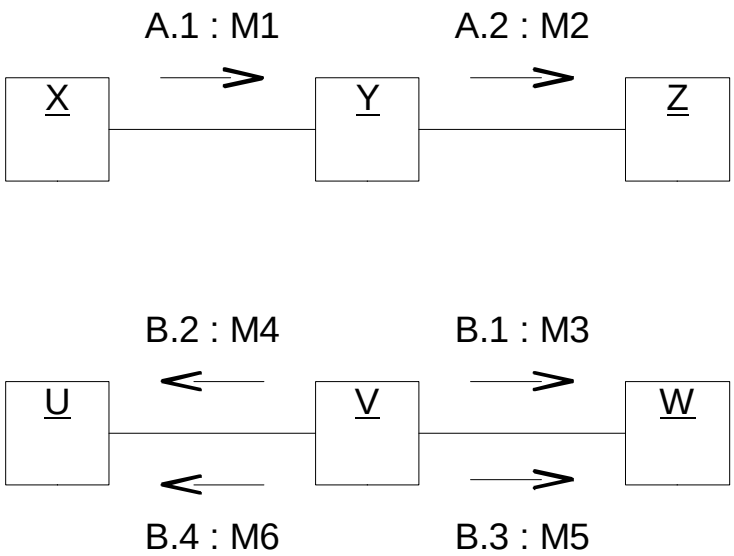


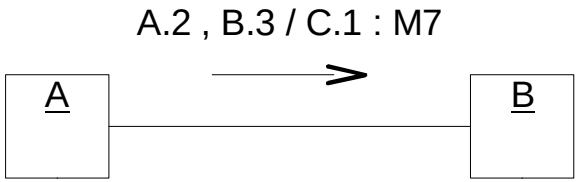
- Résultats et paramètres (ou arguments)



Diagrammes de communication

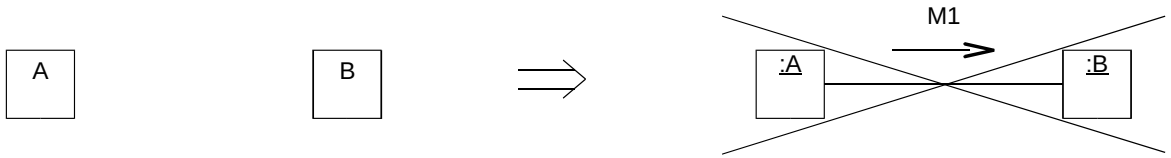
- Exécutions parallèles et synchronisation



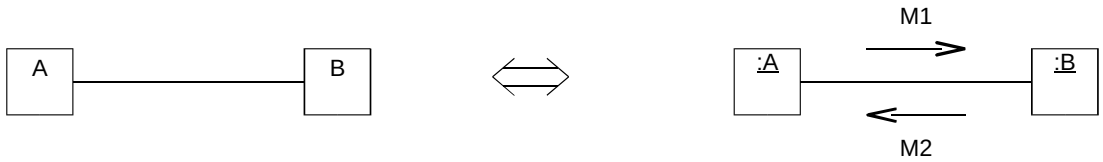


Diagrammes de communication

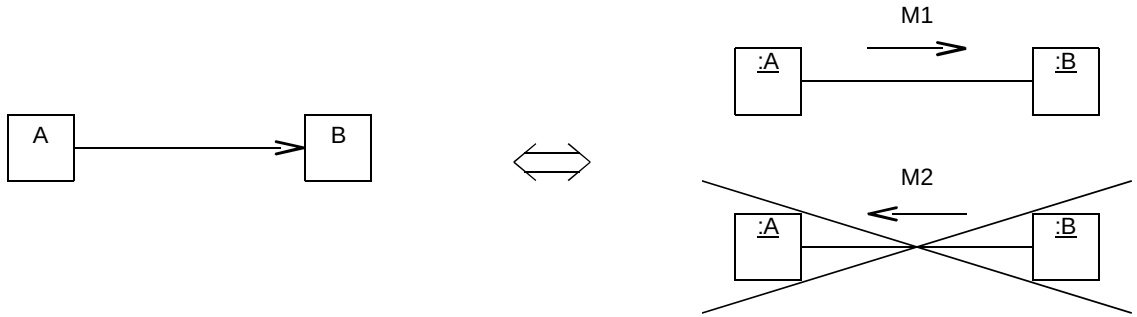
- Deux objets peuvent échanger un message ssi les classes de ces objets sont en relation (association, agrégation ou composition)



- Le sens des messages entre deux objets est bidirectionnel ssi aucun sens de navigation n'est précisé

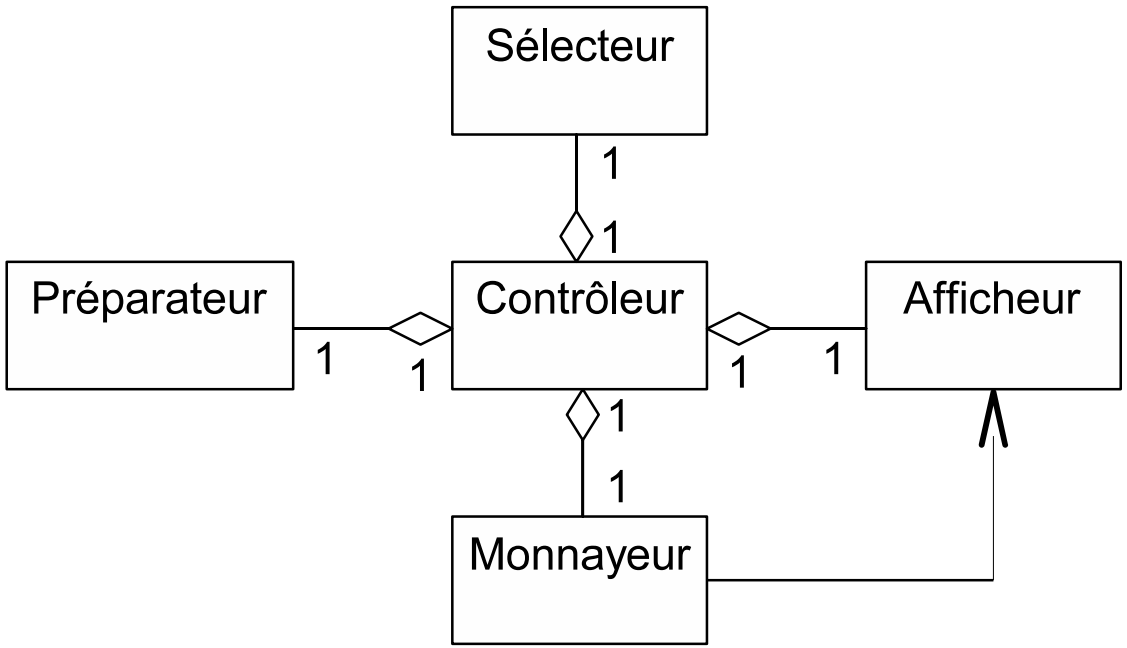


- Le sens des messages entre deux objets est unidirectionnel ssi un sens de navigation est précisé



Exercice

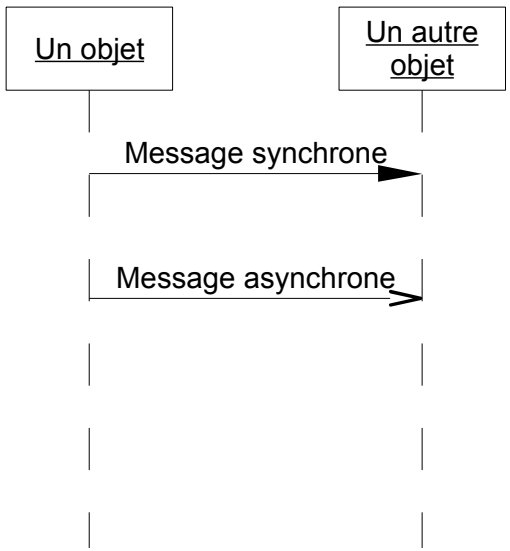
Ex.1 Décrire à l'aide d'un diagramme de communication l'enchaînement nominal de distribution d'une boisson pour un distributeur de boissons dont la structure est définie par le diagramme de classes ci-dessous

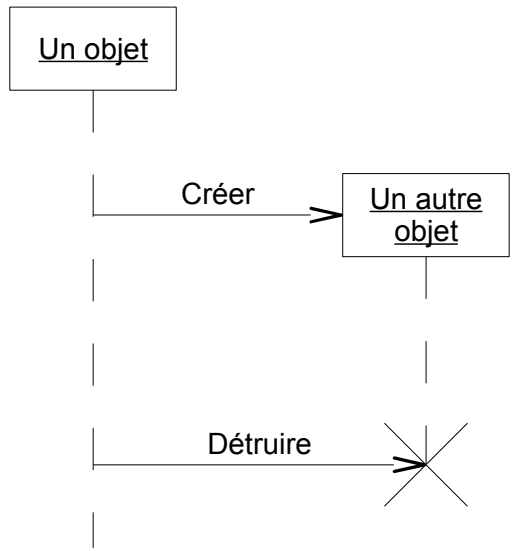


Diagrammes de séquence

Diagrammes de séquence

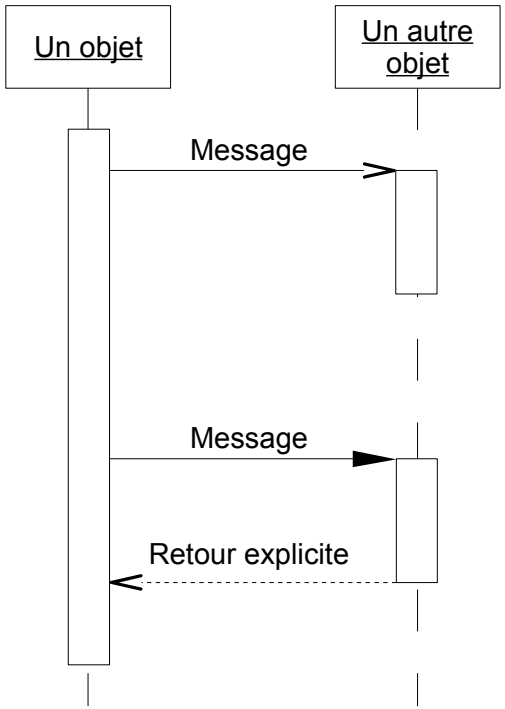
- Représentent les interactions (échanges de messages) entre objets selon un point de vue temporel
- Le contexte des objets n'est pas représenté

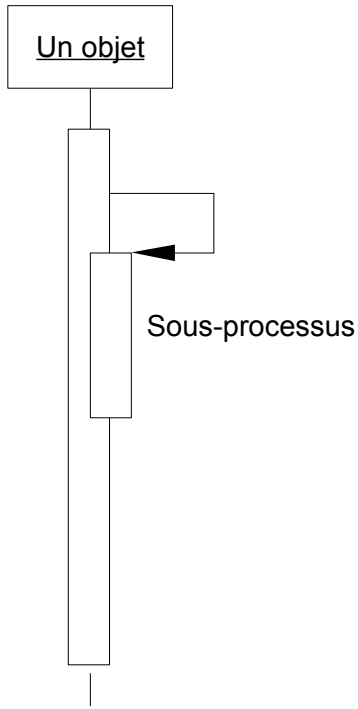




Diagrammes de séquence

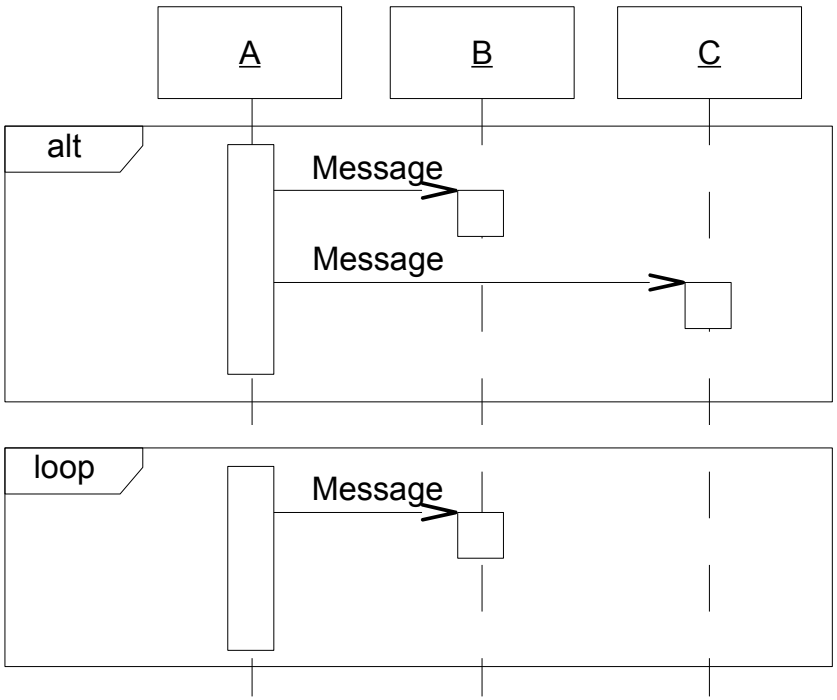
- Activation

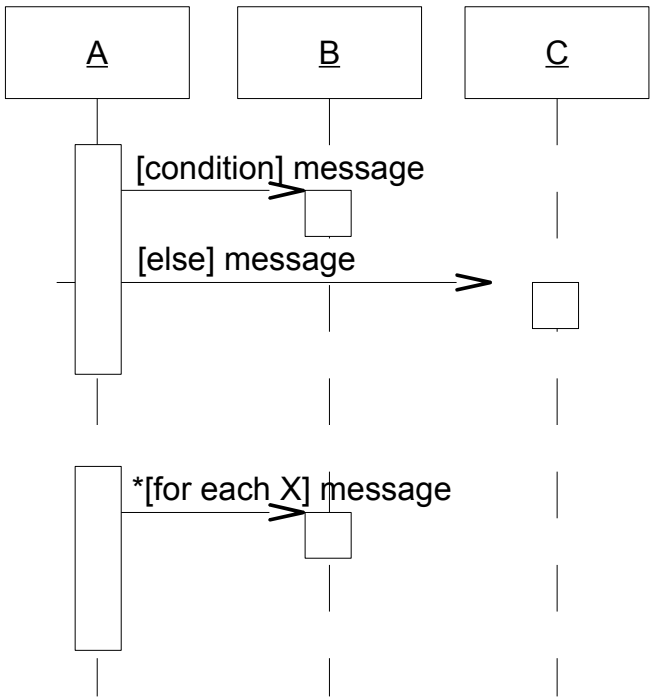




Diagrammes de séquence

- Cadre d'interaction (structures de contrôle)

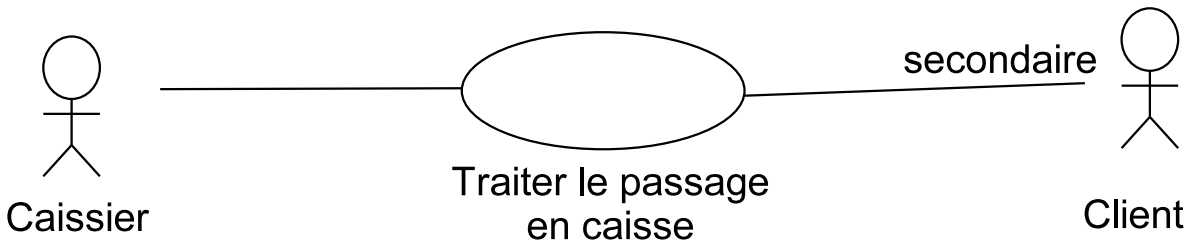




Exercices

Ex.1 Réaliser le diagramme de séquence correspondant au diagramme de collaboration du distributeur de boissons

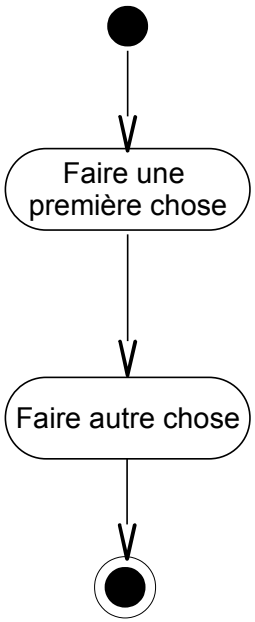
Ex.2 Réaliser le diagramme de séquence système du cas d'utilisation « traiter le passage en caisse » de l'exercice sur la caisse enregistreuse

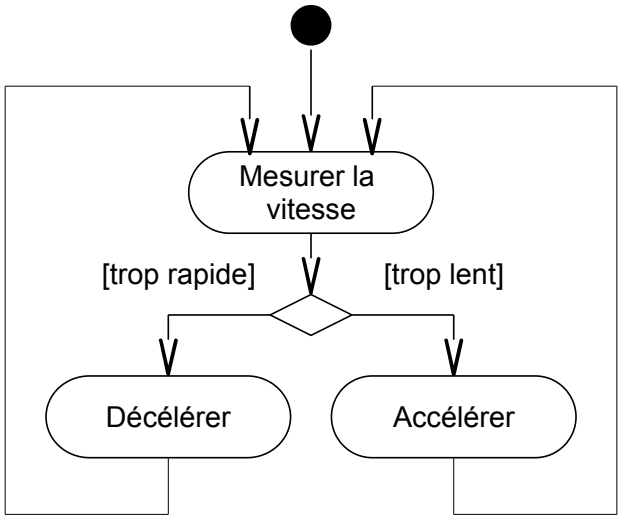


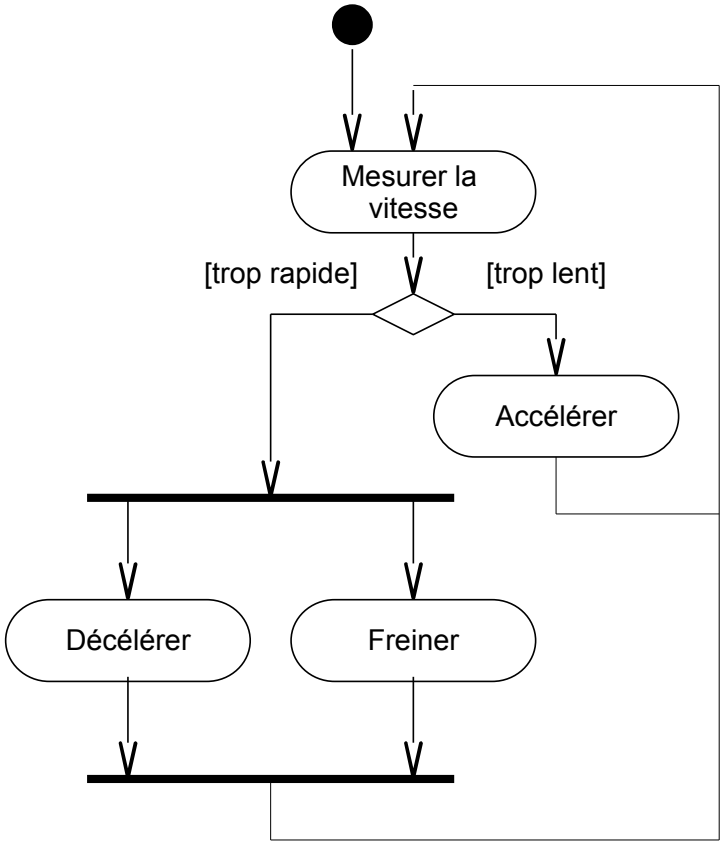
Diagrammes d'activités

Diagrammes d'activités

- Représentations proche de l'organigramme







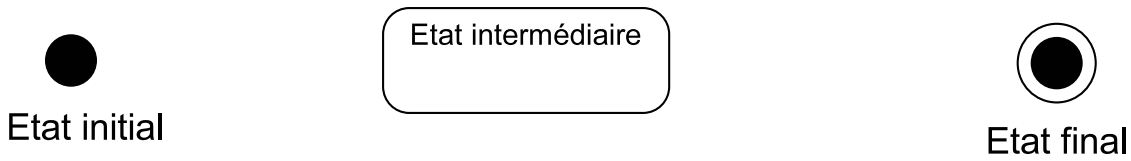
Exercice

Ex.1 Modéliser le comportement de l'opération « PréparerBoisson() » de l'objet « Préparateur » d'un distributeur de boissons

Diagrammes d'états-transitions

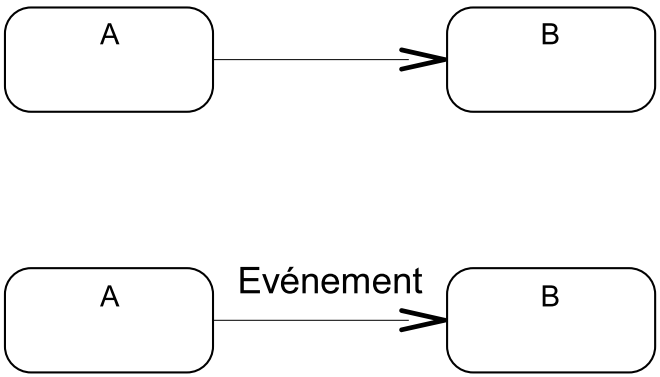
Diagrammes d'états-transitions

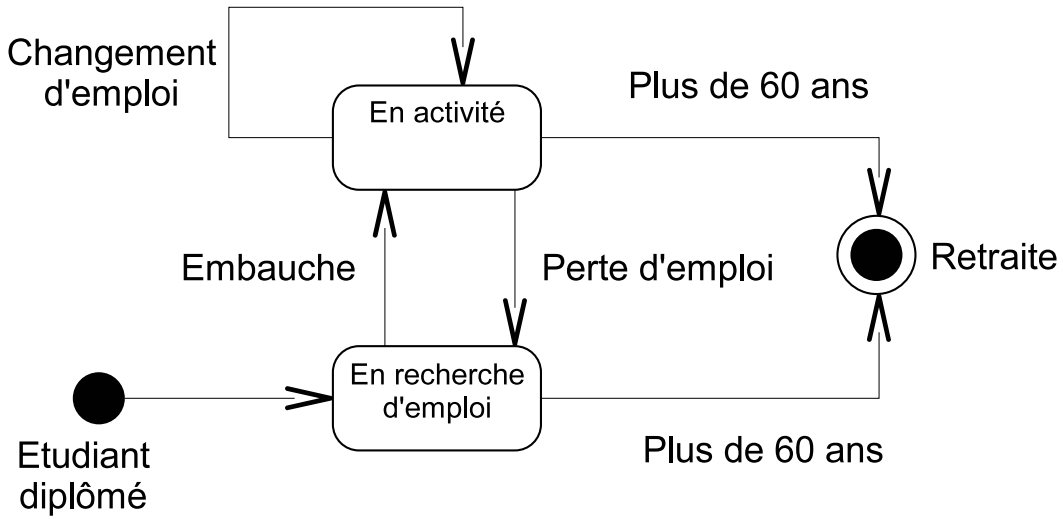
- Permettent de modéliser des automates à états déterministes
- Représentation des états :



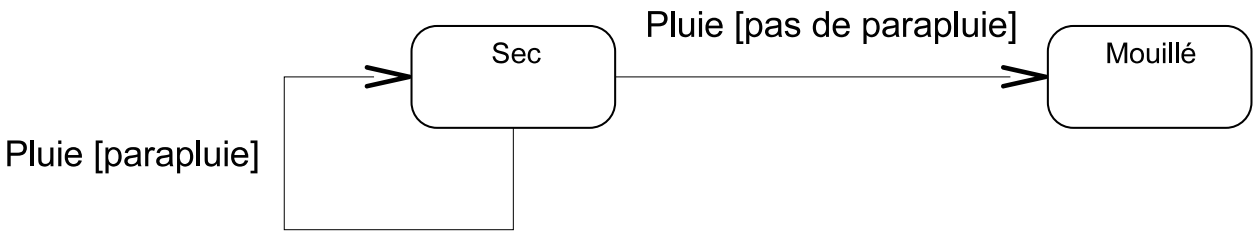
Diagrammes d'états-transitions

- Transitions et événements



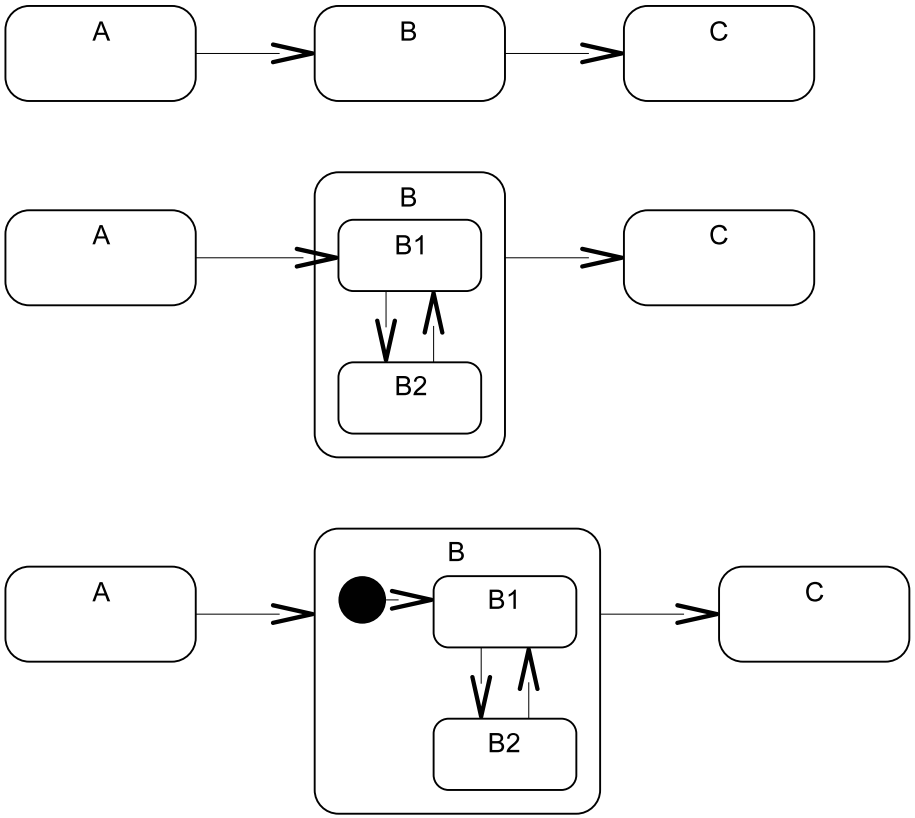


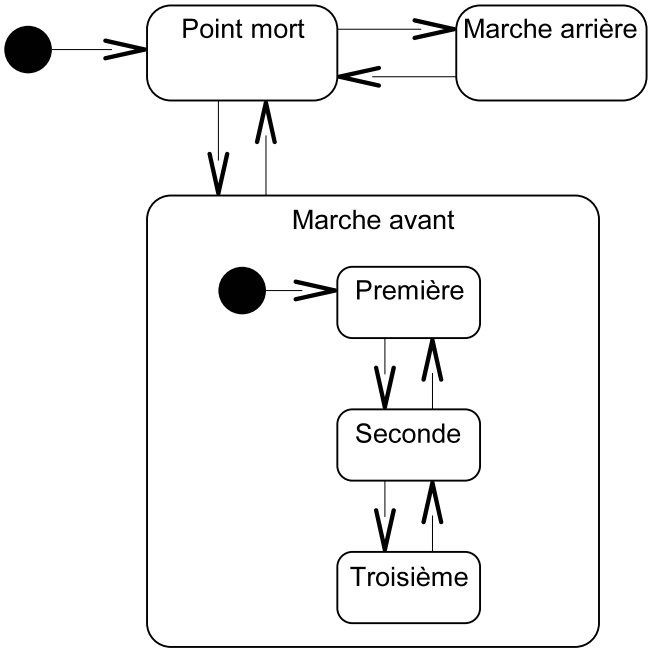
- Gardes (événements conditionnels)



Diagrammes d'états-transitions

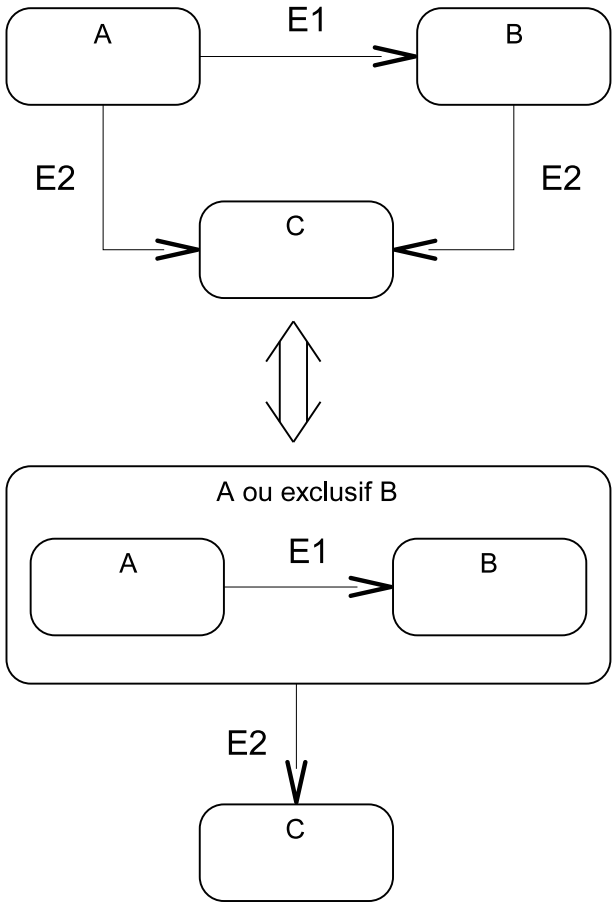
- Généralisation d'états
  - Super-états décomposés en plusieurs états disjoints (décomposition disjonctive)
  - Même démarche que la spécialisation/généralisation des classes, les sous-états héritent des caractéristiques de leur super-état

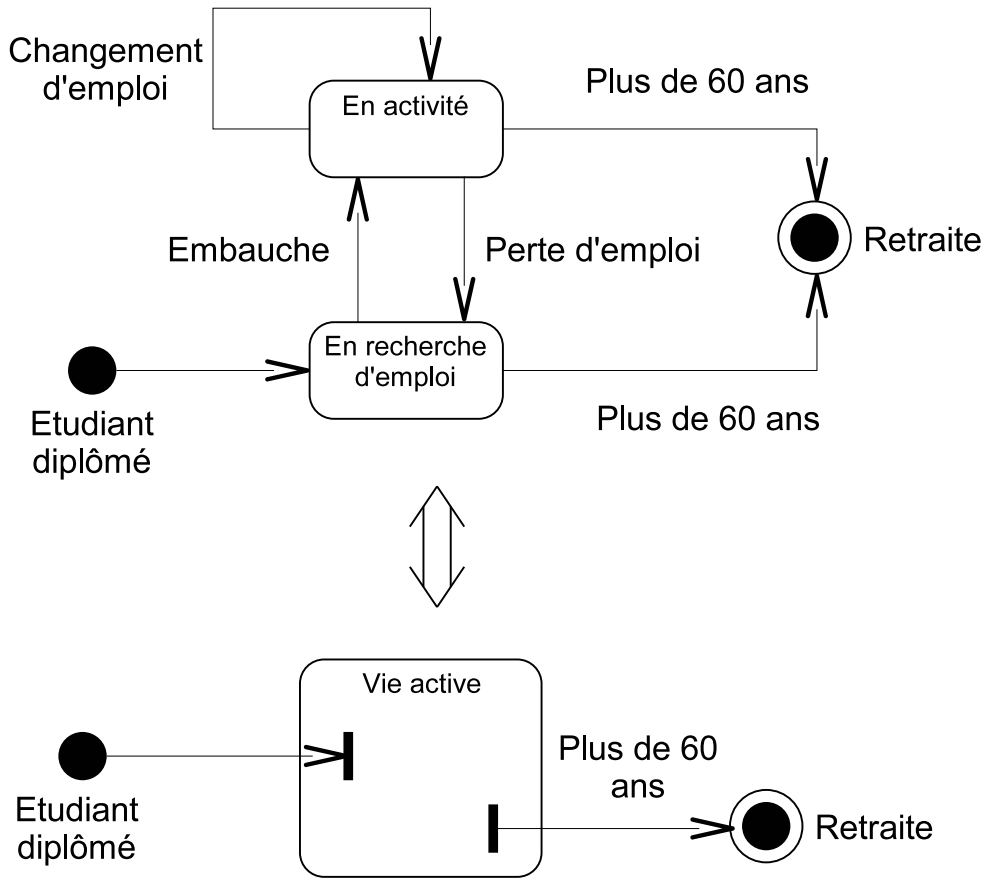




Diagrammes d'états-transitions

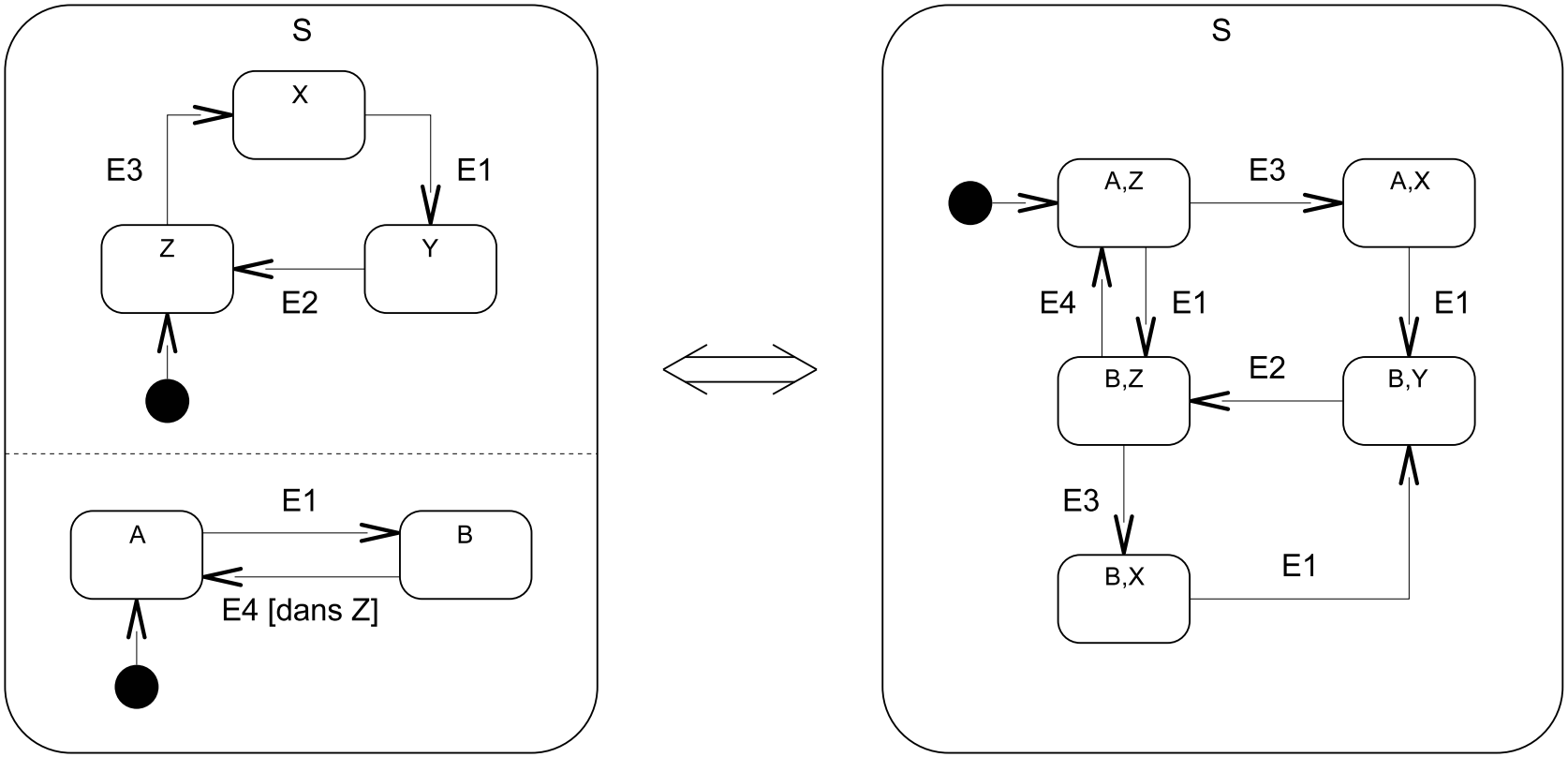
- Généralisation d'états
  - Facilite la représentation et permet d'occulter les détails





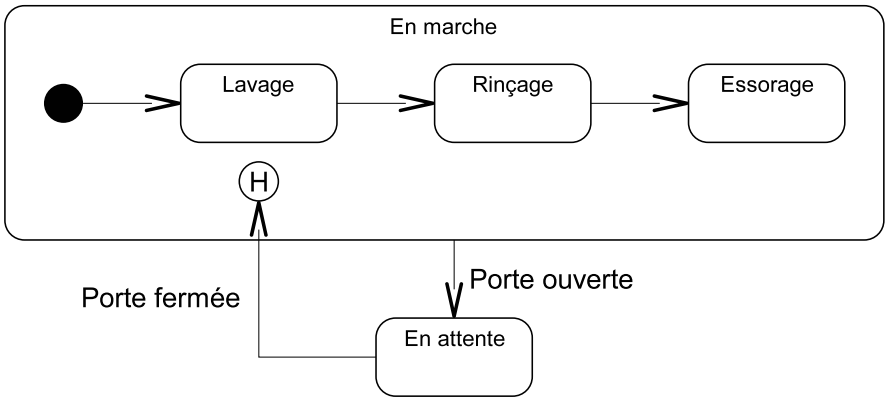
Diagrammes d'états-transitions

- Agrégation d'états
  - Composition d'un état à partir de plusieurs autres états indépendants (composition conjonctive)
  - Permet de décrire des automates parallèles

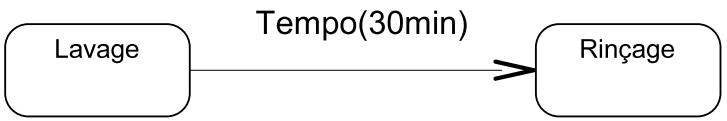


Diagrammes d'états-transitions

- Historique
  - Mécanisme pour mémoriser le dernier sous-état visité
  - H* indique une mémorisation pour tous les états imbriqués

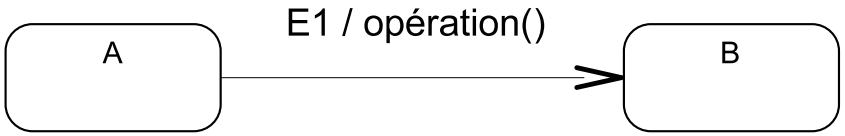


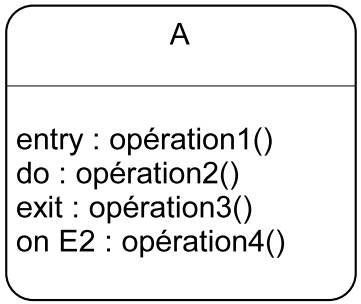
- Déclenchement d'un événement au bout d'un certain temps passé dans un état (transition temporisée)



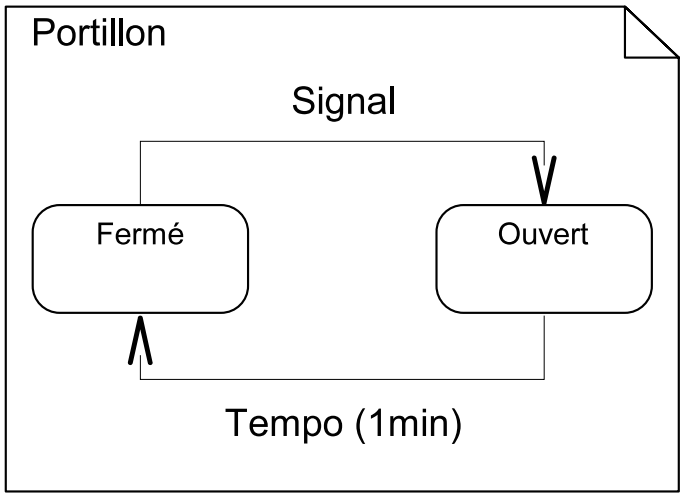
Diagrammes d'états-transitions

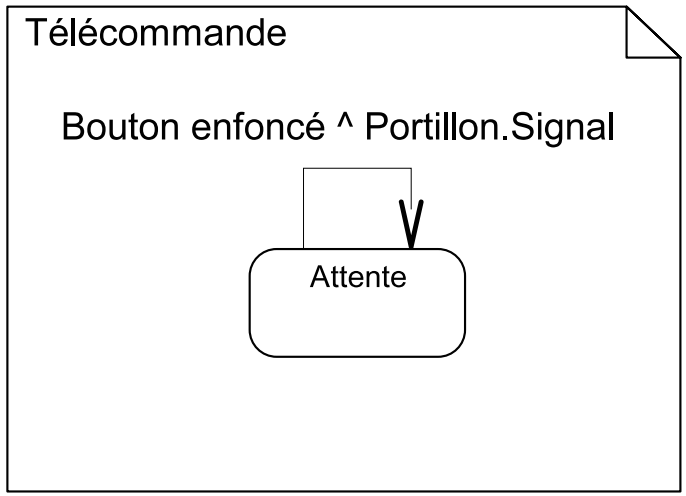
- Déclenchements d'opérations





- Déclenchements d'événements relatifs à un autre objet





Exercices

Ex.1 Modéliser le fonctionnement d'un distributeur de boissons à l'aide d'un diagramme d'états-transitions

Ex.2 Modéliser le fonctionnement d'un téléphone public à pièces à l'aide d'un diagramme d'états-transitions

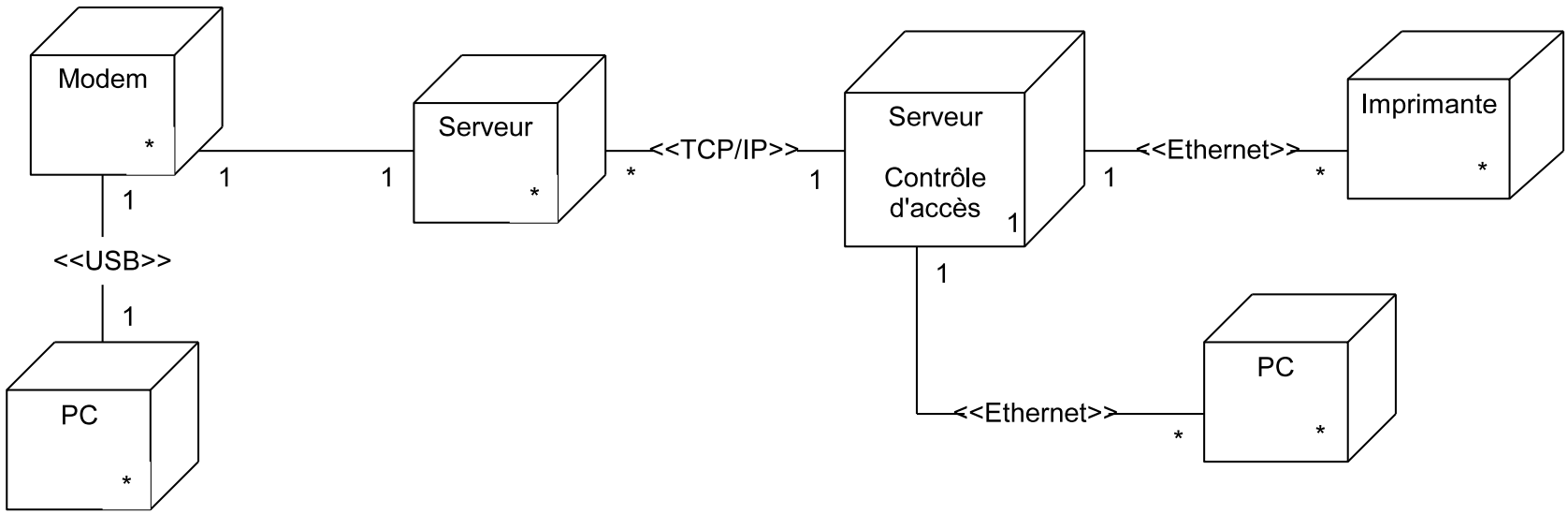
- Le prix minimal d'une communication est de 0,2 euros
- Après l'introduction de la monnaie, l'utilisateur a 2 minutes pour composer son numéro
- La ligne peut être libre ou occupée
- Le correspondant peut raccrocher le premier
- Le téléphone consomme de l'argent dès que l'appelé décroche et à chaque unité de temps
- On peut ajouter des pièces à tout moment
- Lorsque l'on raccroche, le solde de monnaie est rendu

(extrait de *UML par la pratique*, Pascal Roques, Eyrolles)

Diagrammes de déploiement

Diagrammes de déploiement

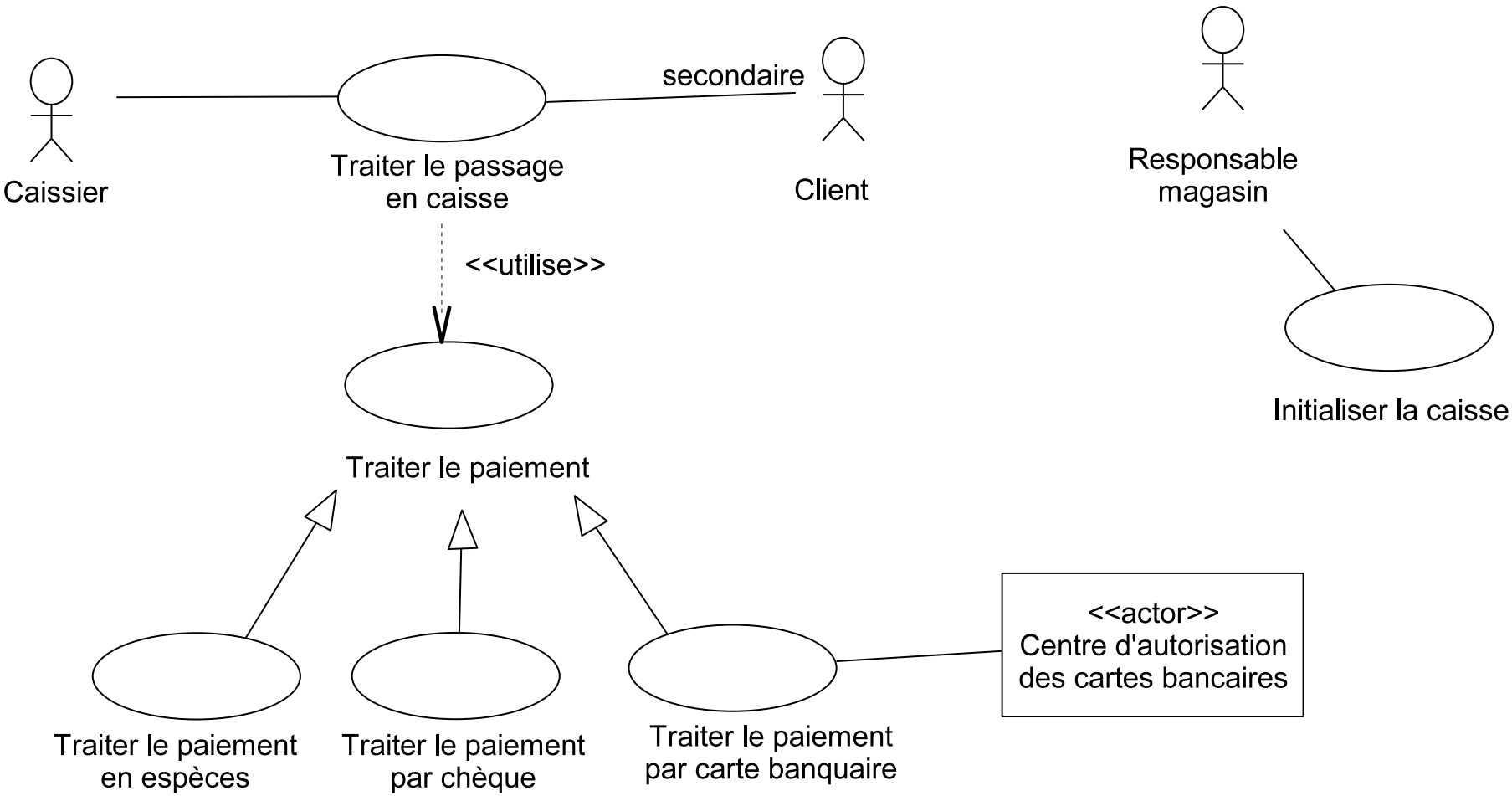
- Montre la disposition physique des différentes ressources matérielles (noeuds), leurs interconnexions et la répartition des programmes sur ces matériels
- Un système est généralement décrit par un petit nombre de diagrammes de déploiement (généralement un seul suffit)



Corrections des exercices

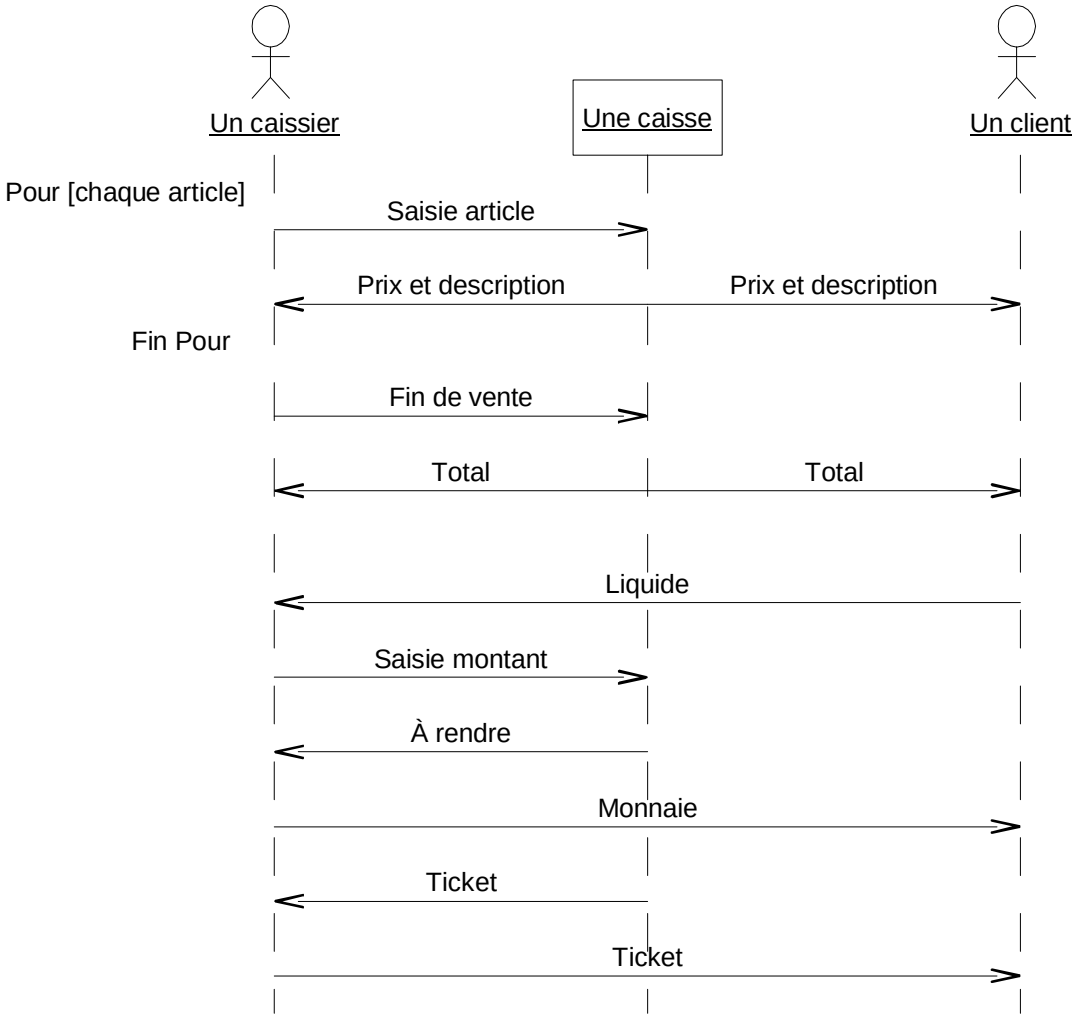
Caisse enregistreuse 1/2

- Exemple de diagrammes de cas d'utilisation pour la caisse enregistreuse



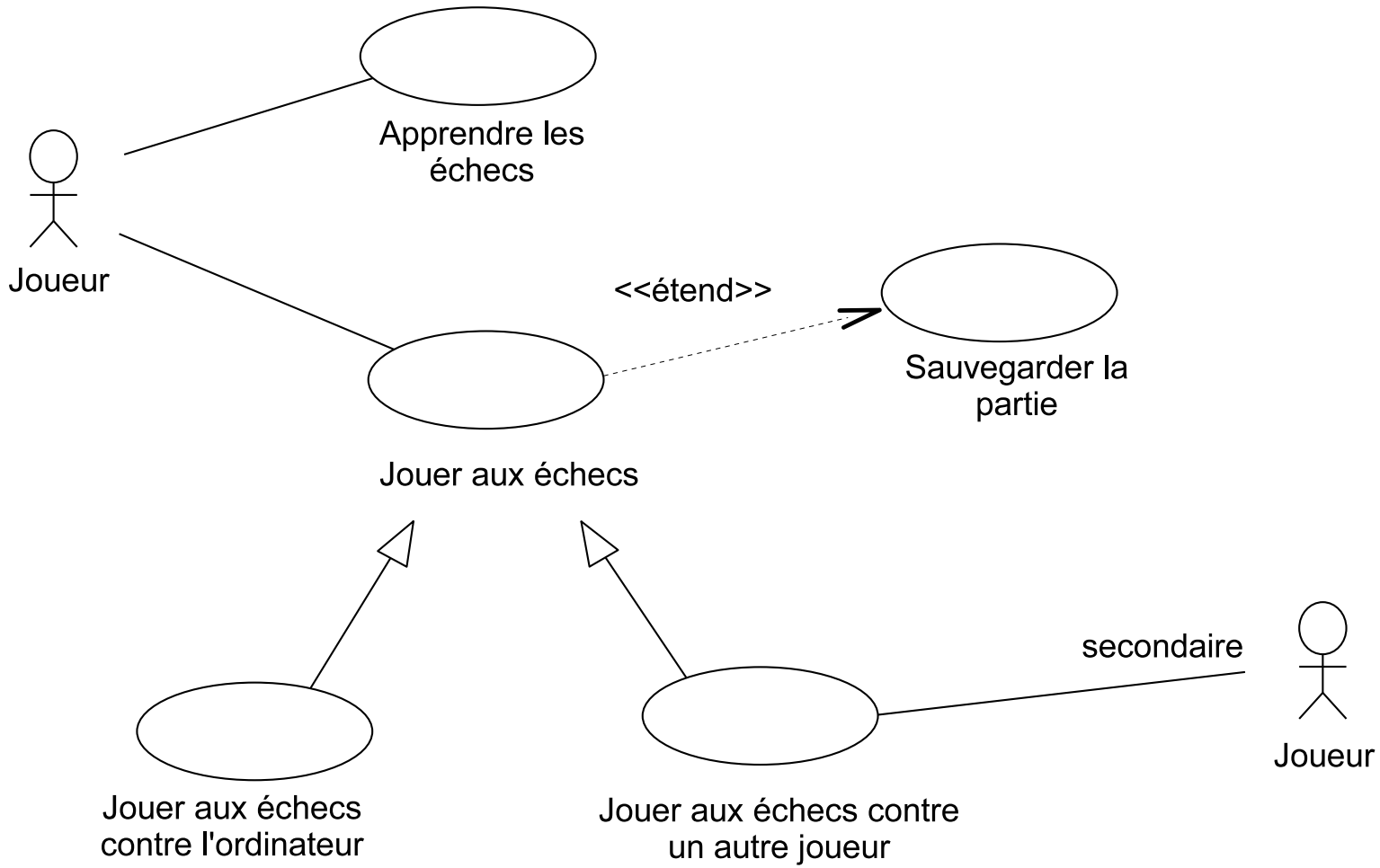
Caisse enregistreuse 2/2

- Exemple de diagramme de séquence système pour le cas *traiter le passage en caisse* (paiement en espèces)



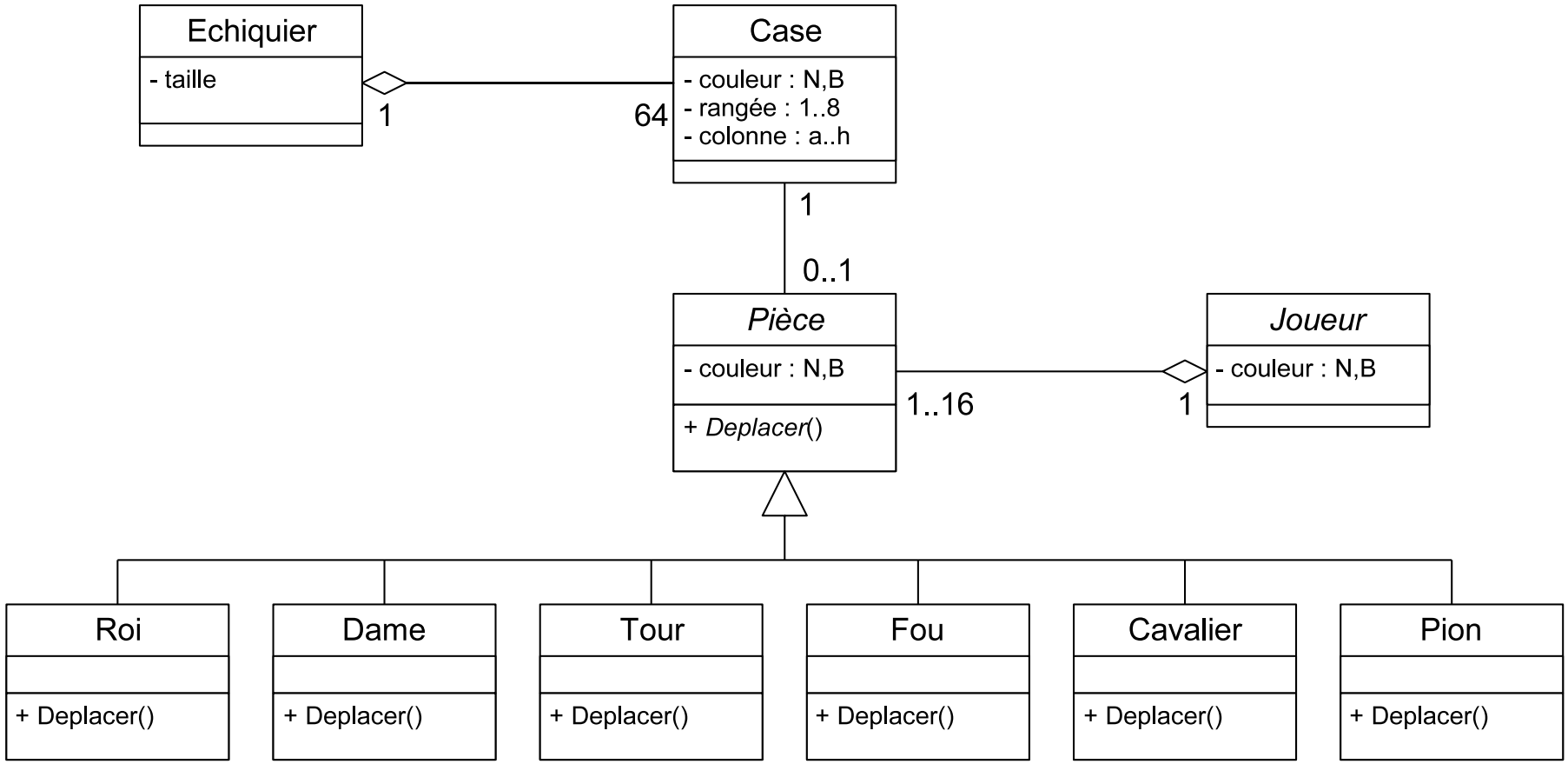
Jeu d'échecs 1/3

- Exemple de diagramme de cas d'utilisation pour le jeu d'échecs



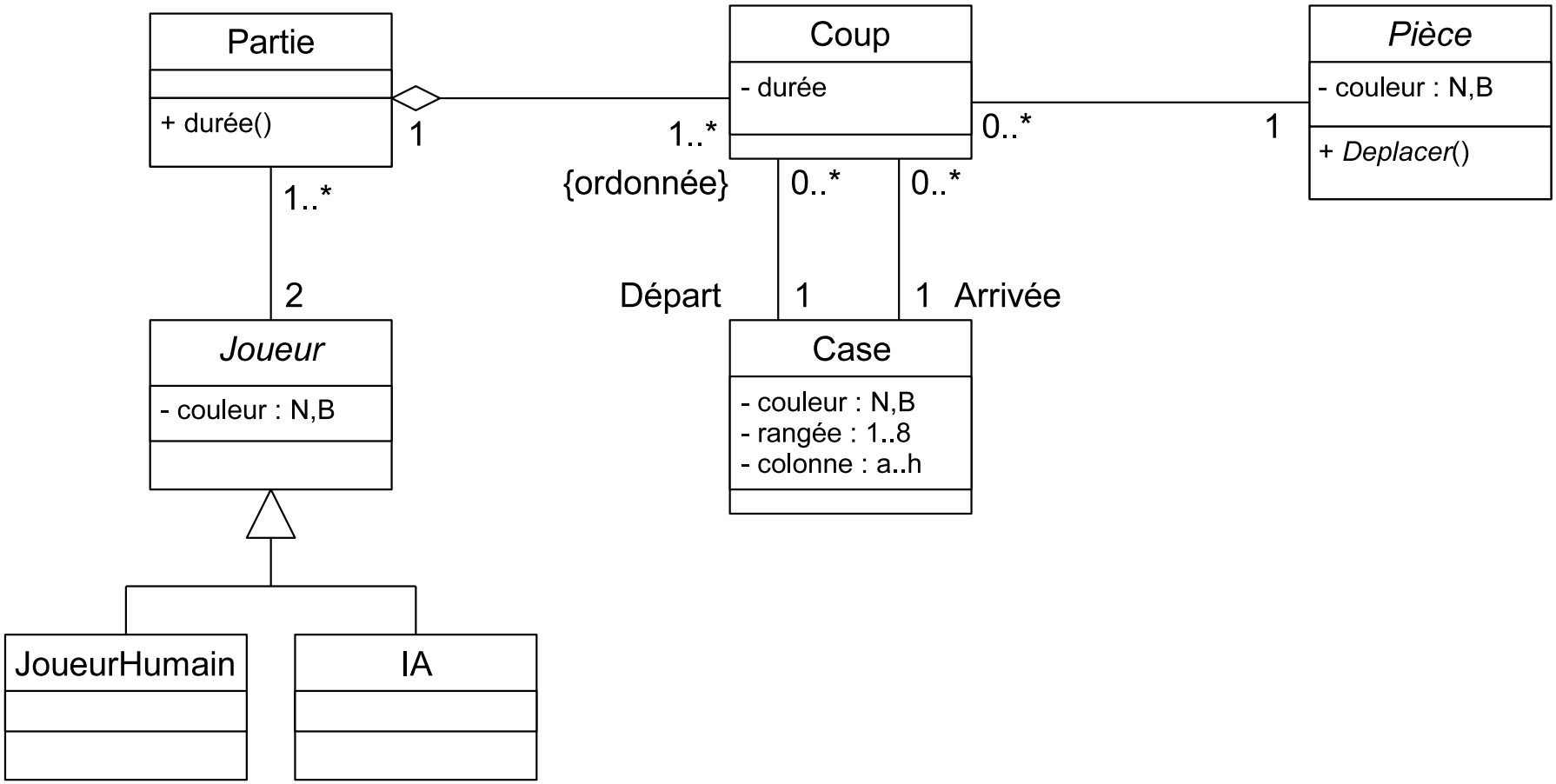
Jeu d'échecs 2/3

- Exemple de diagramme de classes pour le jeu d'échecs (échiquier)



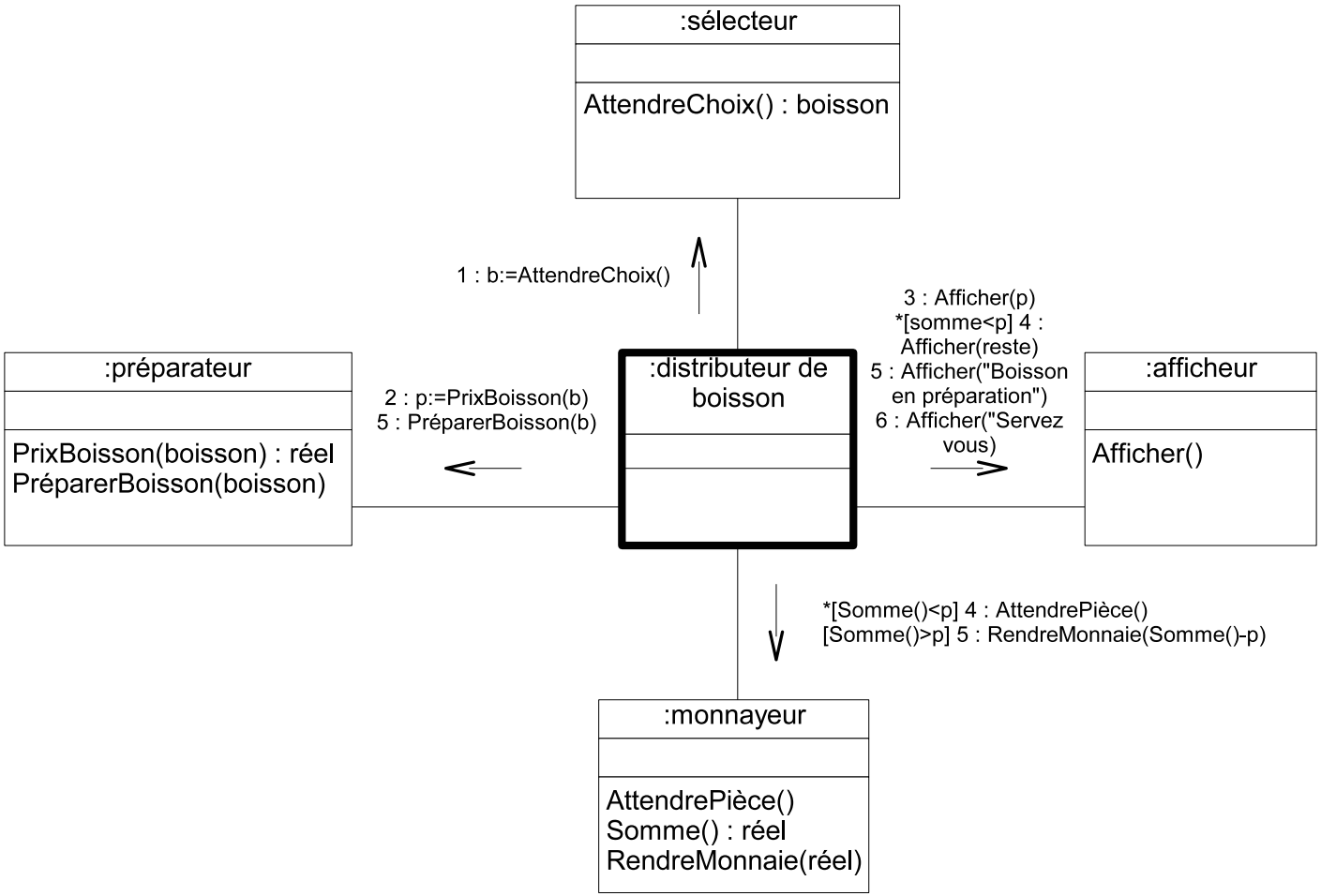
Jeu d'échecs 3/3

- Exemple de diagramme de classes pour le jeu d'échecs (partie)



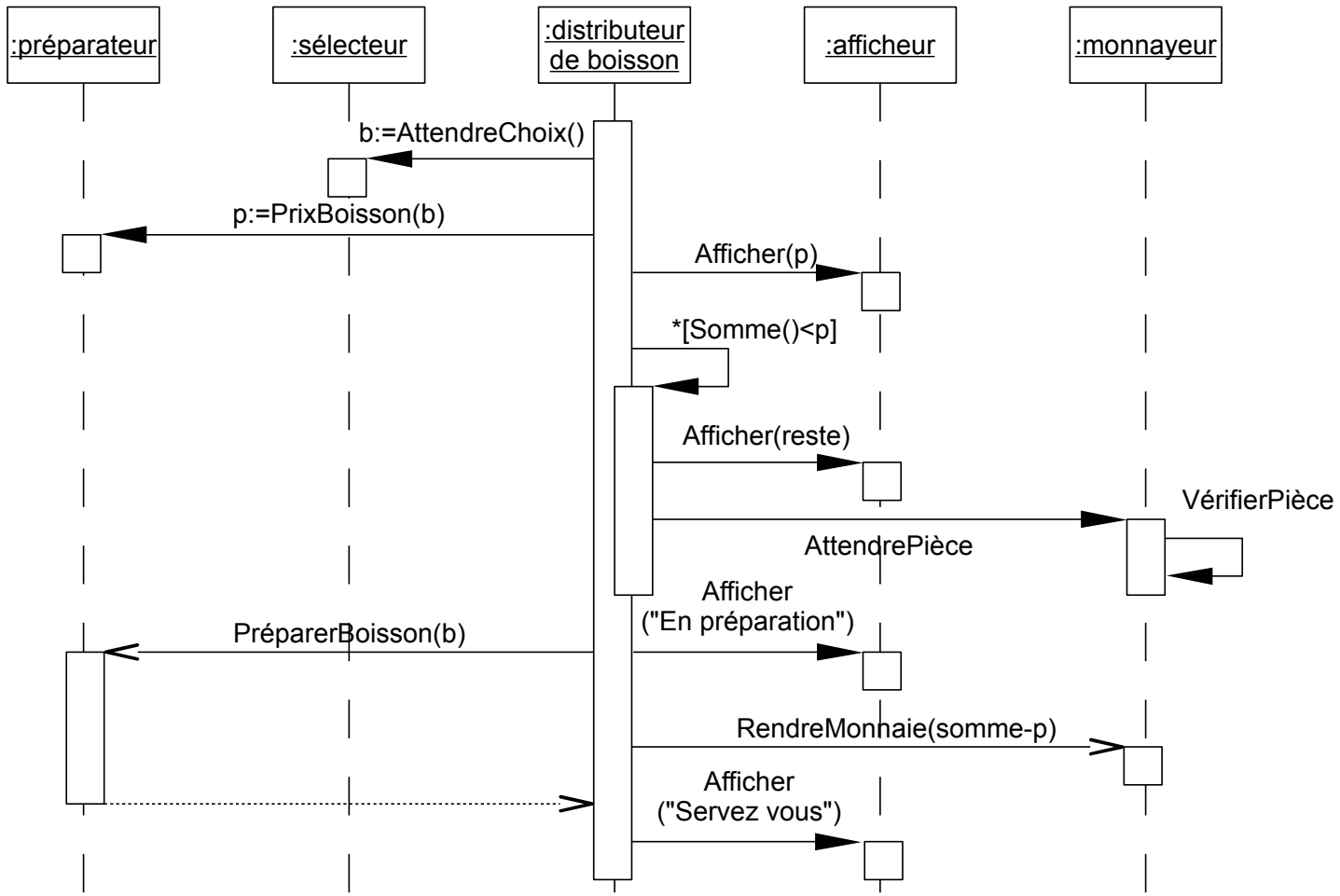
Distributeur de boissons 1/4

- Exemple de diagramme de communication décrivant l'enchaînement nominal de distribution d'une boisson



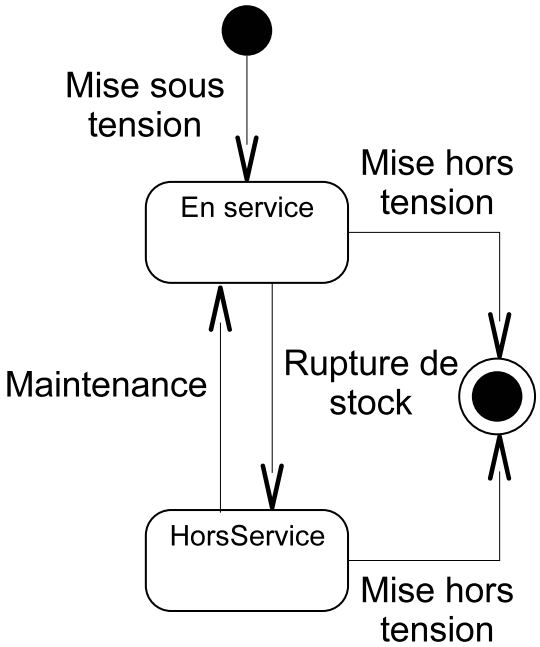
Distributeur de boissons 2/4

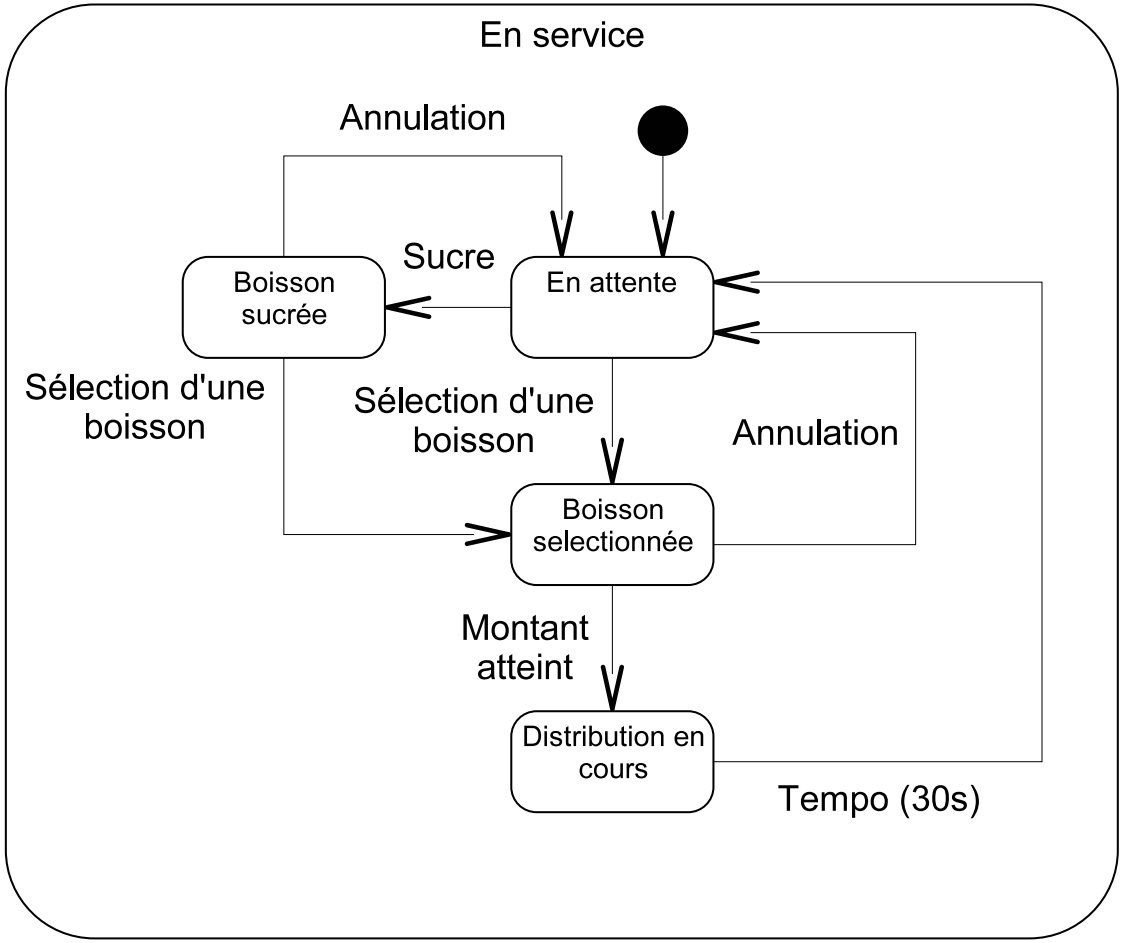
- Exemple de diagramme de séquence décrivant l'enchaînement nominal de distribution d'une boisson



Distributeur de boissons 3/4

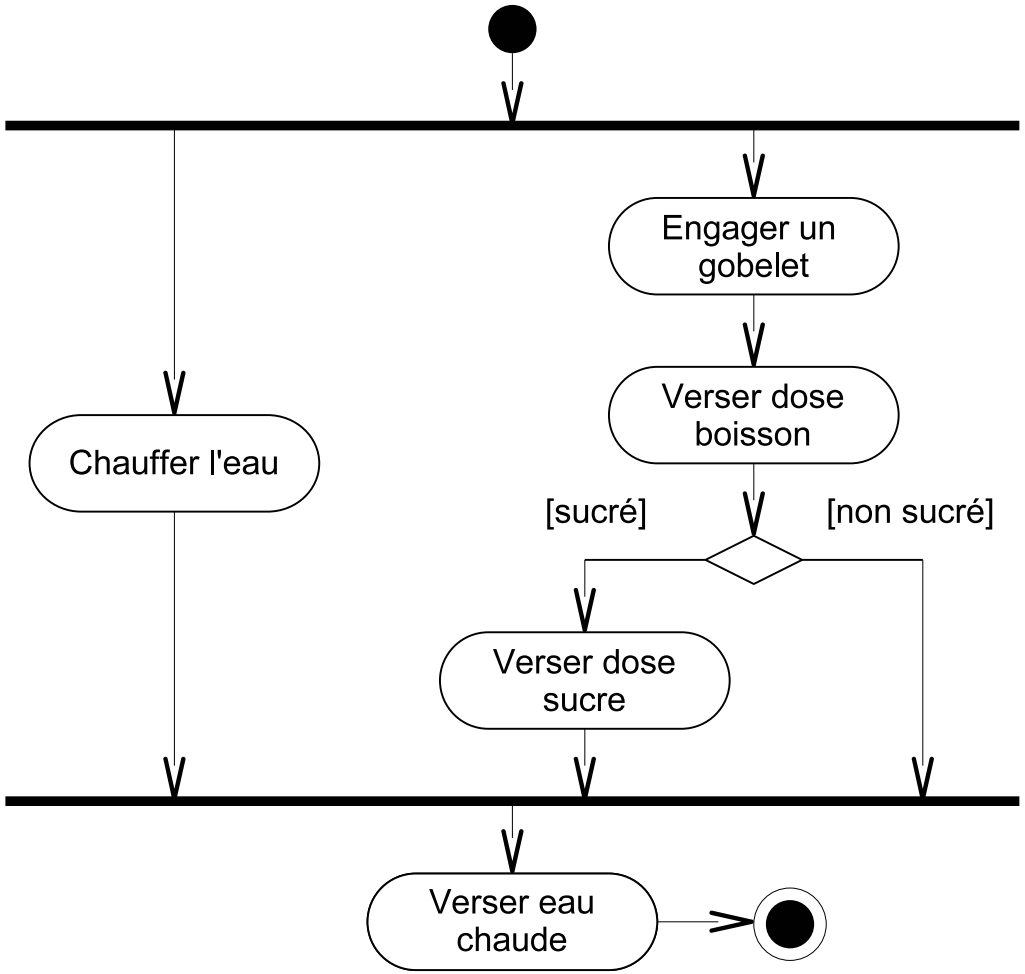
- Exemple de diagramme d'états-transitions décrivant le fonctionnement d'un distributeur de boissons





Distributeur de boissons 4/4

- Exemple de diagramme d'activités pour le processus de distribution d'une boisson



Téléphone public 1/1

- Exemple de diagramme d'états-transitions décrivant le fonctionnement d'un téléphone public à pièces