INFO 307
Module 3

MODELISATION ET CONCEPTION ORIENTEES OBJET(OMT)

Source: Object Modeling Techniques
by James Rumbaugh et al

Introduction

Qu'est-ce que l'orienté objet ?

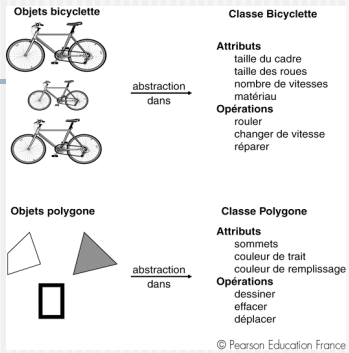
- Logiciel formé d'une collection d'objets indépendants qui incorporent à la fois une structure de données et un comportement.
- Notions caractéristiques
  - Identité
  - Classification
  - Héritage
  - Polymorphisme

Identité

- Données organisées en entités discrètes et distinguables nommées **objets** : *le premier transparent de cette présentation*, *mon ordinateur portable* …
- Objet
  - Concret : *le fichier ~baudon/toto*
  - Abstrait : *une stratégie d'ordonnancement de processus*
- Chaque objet possède une identité intrinsèque. Deux objets sont distincts même si toutes les valeurs de leurs attributs sont identiques.

Classification

- Les objets possédant la même structure de données (attributs) et le même comportement (opérations) sont les représentants d'une même **classe**.
- Une classe est une abstraction qui décrit les propriétés pertinentes pour une application.
- Chaque classe décrit un ensemble d'objets individuels potentiellement infini.
- Chaque objet est une **instance** d'une classe. Un objet possède une référence implicite à sa classe : il sait ce qu'il est.



Héritage

- Partage des attributs et des opérations (propriétés) entre classes sur la base d'une relation hiérarchique.
- Une **sur-classe** possède des informations générales que les **sous-classes** spécialisent.
- Permet de réduire considérablement les répétitions.

Polymorphisme

- Signifie que la même opération peut se comporter différemment dans des classes différentes. Par exemple, l'opération *déplacer* dans un jeu d'échec : le déplacement dépend du type de pièce.
- Une **opération** est une action qu'un objet exécute ou une transformation qu'il subit.
- L'implémentation d'une opération par une classe se nomme une **méthode**. Plusieurs méthodes peuvent implémenter la même opération (une par classe concernée).
- Un langage de programmation objet sélectionne la bonne méthode pour réaliser une opération en fonction de son nom et de l'objet sur lequel elle opère.

Méthodologie orientée objet

- Processus
  - Spécification initiale du système
  - Analyse
  - Conception du système
  - Conception des classes
  - Implémentation

Spécification initiale

- Spécification de l'application entre analystes métier et utilisateurs.

Analyse

- Précise l'objectif du système et non la façon dont il sera construit.
- Une classe décrite le sera en terme d'attributs et d'opérations visibles.
- Deux modèles
  - Modèle du domaine : description des objets du monde réel manipulés par le système.
  - Modèle de l'application : décrit les parties du système visibles par l'utilisateur.

Conception du système

- Architecture du système
- Politiques par défaut : caractéristiques de performances, communications …

Conception des classes

- Elaboration des objets du domaine et de ceux de l'application avec la même notation.
- Choix des structures de données et algorithmes.

Implémentation

- Transposition des classes et de leurs relations en concepts propres à un langage, une base de données ou une plateforme.

Modèles de description d'un système

- 3 modèles :
  - Modèle de classes
  - Modèle d'états
  - Modèle d'interaction
- Parties distinctes de la description d'un système, mais interdépendants.

Modèle de classes

- Décrit la structure statique des objets et leurs relations.
- Contient des **diagrammes de classes**, où les nœuds sont des classes et les arcs des relations entre ces classes.

Modèle d'états

- Décrit les états successifs d'un objet au cours du temps.
- Un **diagramme d'état** est un graphe dont les sommets sont des états et les arcs des transitions entre états déclanchées par des évènements.

Modèle d'interactions

- Décrit la façon dont les objets coopèrent pour obtenir un résultat.
- Commence par les **cas d'utilisation**, qui sont détaillés grâce à des **diagrammes de séquence** et des **diagrammes d'activités**.
- Un cas d'utilisation est axé sur une fonctionnalité.
- Un diagramme de séquence représente les objets qui interagissent et l'ordonnancement de leurs interactions.
- Un diagramme d'activité détaille les étapes importantes du traitement.

Thèmes de l'orienté objet

- Abstraction
- Encapsulation
- Regroupement des données et du comportement
- Partage
- Mise en évidence de la nature intrinsèque des objets
- Synergie

Abstraction

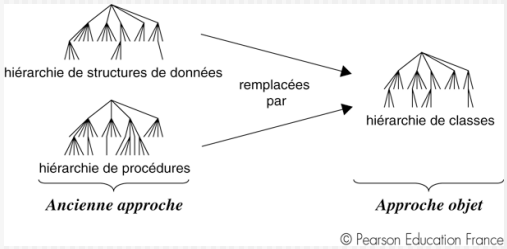
- Permet de s'attacher aux aspects essentiels sans entrer dans les détails, autrement dit de se concentrer sur ce que représente l'objet et sur son comportement avant de décider de la façon de l'implémenter.

Encapsulation (masquage de l'information)

- Sépare les aspects externes d'un objet, accessibles aux autres objets, des détails d'implémentation internes, qui leur sont cachés.
- Empêche l'aggravation de l'interdépendance de portions de programme telle qu'une modification même minime aurait des répercussions massives.

Regroupement des données et du comportement

- Le polymorphisme transfère la charge de décider quelle implémentation utiliser à la hiérarchie des classes.
- Il n'est pas nécessaire de modifier le code appelant lors de l'ajout d'une nouvelle classe.



Partage

- Le fait d'hériter des structures de données et du comportement permet aux sous-classes de partager des portions de code communes.
- Il apporte une clarté conceptuelle en mettant en évidence que différentes opérations ne représentent en réalité qu'un seul et même traitement.

Mise en évidence de la nature intrinsèque de objets

- L'approche objet met l'accent sur ce qu'*est* un objet et non sur la façon dont il est *utilisé*.

Synergie

- Identité, classification, héritage et polymorphisme peuvent être appliqués isolément. Mais ils sont complémentaires et forment une synergie quand ils sont combinés.

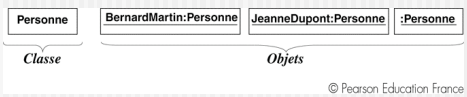
UML
Unified Modeling Language

- 1991 : première édition de *Modélisation et conception orientées objet* basée sur OMT, Object *Modeling Technique*, issue de la R&D de General Electric.
- 1994 : James Rumbaugh rejoint Rational et travaille avec Grady Booch à la fusion des notations OMT et Booch.
- 1995 : Ivar Jacobson rejoint Rational et intègre Objectory au travail d'unification.
- 1997 : l'OMG (Object Management Group) accepte UML, proposé par Rational, comme standard de modélisation objet.
- 2001 : révision par l'OMG d'UML 1.
- 2004 : adoption d'UML 2.0

Modélisation des classes

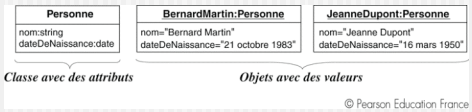
Diagrammes de classes et d'objets

- Un objet est une instance (ou occurrence) d'une classe.
- Une classe décrit un groupe d'objets qui possèdent les mêmes propriétés (attributs), le même comportement (opérations), les mêmes types de relations et la même sémantique.



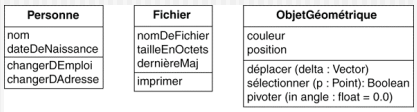
Valeurs et attributs

- Une valeur est une donnée.
- Un attribut est une propriété nommée d'une classe qui décrit le type d'une valeur contenue dans chaque objet de la classe.

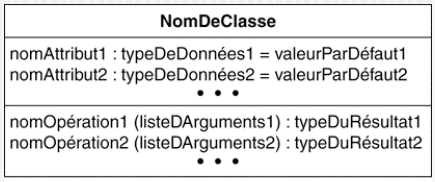


Opérations et méthodes

- Une opération est une fonction ou une procédure qui peut être appliquée aux objets ou par les objets d'une classe.
- La même opération peut s'appliquer à de nombreuses classes différentes. Dans ce cas, on dit qu'elle est *polymorphe*.
- Une méthode est l'implémentation d'une opération pour une classe donnée.



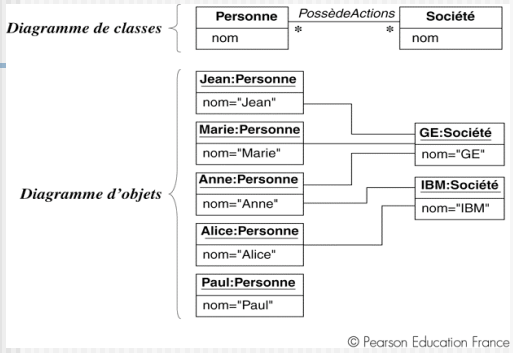
Résumé



Argument :
sensDeFlux nomArgument : type = valeurParDéfaut
sensDeFlux = *in*, *out* ou inout

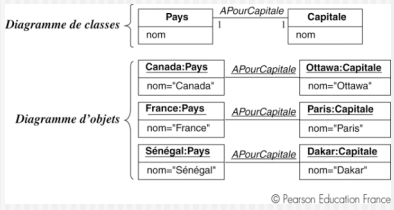
Liens et associations

- Un **lien** est une connexion physique ou conceptuelle entre des objets.
- Une **association** est une description d'un groupe de liens qui partagent une structure et une sémantique commune. Les associations sont intrinsèquement bidirectionnelles.
- La **multiplicité** représente le nombre d'instances d'une classe qui peuvent être liées à une instance d'une autre classe.
- Une **référence** est un attribut d'un objet qui se réfère à un autre objet.



Multiplicité

- La multiplicité spécifie le nombre d'instances d'une classe pouvant être liées à une seule instance d'une classe associée. Elle contraint le nombre d'objets liés.

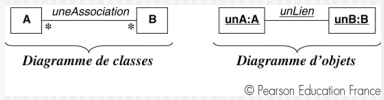


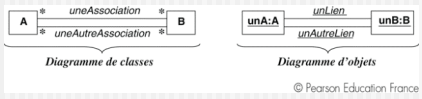
Multiplicité

- UML spécifie la multiplicité par un intervalle : *1*, *1..**, *3..5*.

Multiplicité

- La multiplicité *plusieurs* spécifie qu'un objet peut être associé à plusieurs autres objets. Mais pour chaque association, (sauf pour les bags et les séquences) il y a au plus un lien entre une paire d'objets.





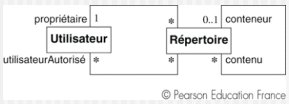
Noms d'extrémité d'association

- Chaque **extrémité d'association** peut être nommée.

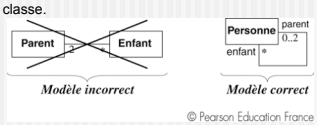
employé	employeur
Paul Lenoir	Simplex
Marie Lebrun	Simplex
Jean Leblanc	Multiplex

Noms d'extrémité d'association

- Les noms d'extrémités sont nécessaires pour les associations entre deux objets de la même classe.

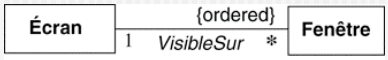


- Nommez les extrémités pour modéliser plusieurs références à la même classe.



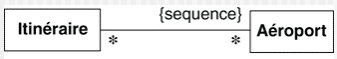
Ordonnancement

- Vous pouvez indiquer un ensemble ordonné d'objets en écrivant *{ordered}* du côté de l'extrémité appropriée.



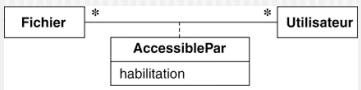
Bag et séquences

- Vous pouvez autoriser plusieurs liens (collection) en annotant une extrémité d'association avec *{bag}* (non ordonnée) ou *{sequence}* (ordonné).
- Les annotation *{ordered}* et *{sequence}* sont analogues, sauf que la première interdit les doublons alors que la seconde les autorise.



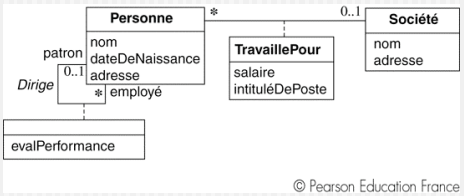
Classe-association

- Une **classe-association** est une association qui est aussi une classe.



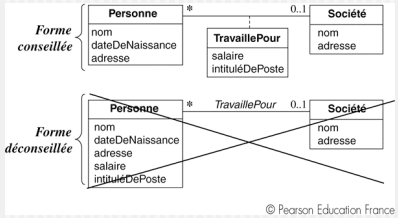
/etc/termcap	lecture	Paul Lenoir
/usr/doe/.login	lecture/écriture	Marie Lebrun
/usr/lenoir/.login	lecture/écriture	Paul Lenoir

Classe-association



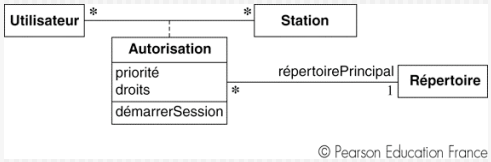
Classe-association

- Ne placez pas les attributs d'une association dans une classe.

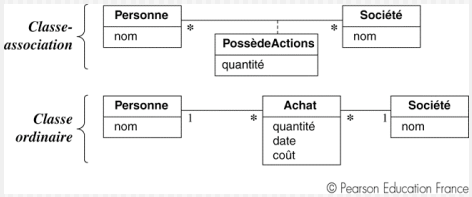


Classe-association

- Les classes-associations permettent de spécifier précisément l'identité et les chemins de navigation.

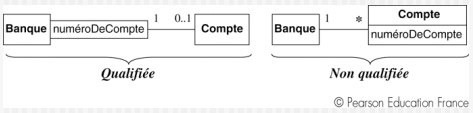


Classe ordinaire ≠ Classe-association



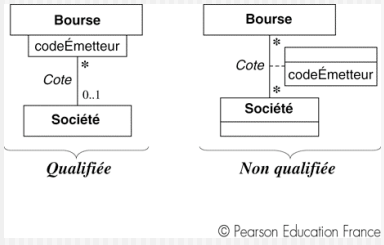
Association qualifiée

- Une **association qualifiée** est une association dans laquelle un attribut nommé **qualificateur** désambiguïse les objets situés à l'extrémité de multiplicité *plusieurs*.
- *Banque* et *Compte* sont des classes et *numéroDeCompte* le qualificateur :



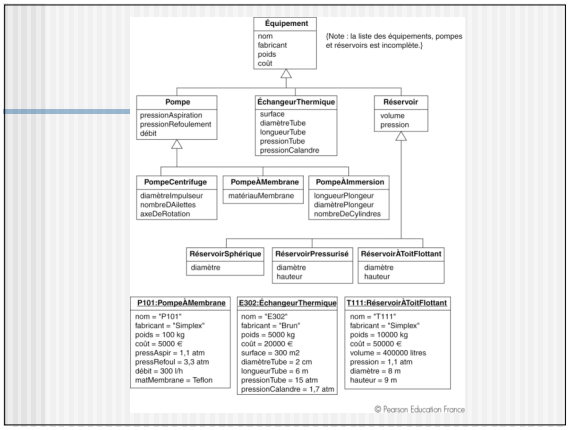
Association qualifiée

- La qualification facilite la traversée.



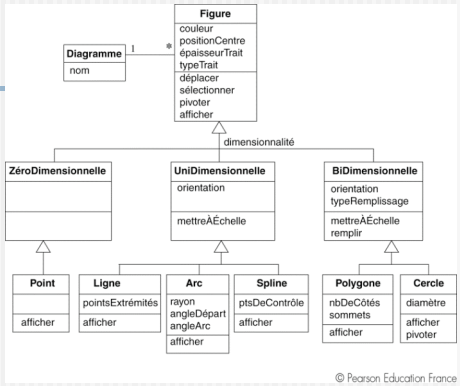
Généralisation et héritage

- La **généralisation** est une relation hiérarchique entre une classe, la **super-classe**, et une ou plusieurs variantes de cette classe, les **sous-classes**.
- On dit que chaque sous-classe **hérite** des propriétés de sa super-classe.
- Les termes **ancêtre** et **descendant** renvoient à la généralisation des classes sur plusieurs niveaux.



Nom d'ensemble de généralisation

- Un **nom d'ensemble de généralisation** est un attribut de type énuméré qui indique quel aspect d'un objet a été rendu abstrait pare une généralisation particulière.

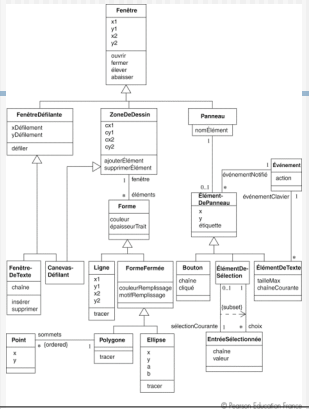


Utilisation de la généralisation

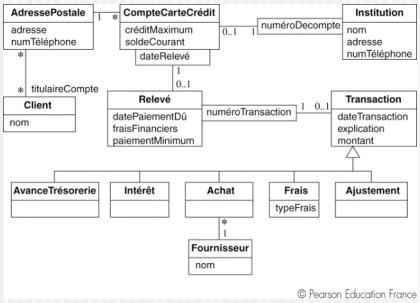
- Prendre en charge le polymorphisme
- Structurer la description des objets
- Permettre la réutilisation du code

Redéfinition

- Une sous-classe peut **redéfinir** une propriété d'une super-classe en définissant une propriété du même nom.
- Vous ne devez jamais redéfinir la **signature** d'une propriété.
- Vous ne devez jamais redéfinir une propriété de sorte qu'elle soit incohérente ave la propriété héritée d'origine.



Navigation dans les modèles de classe



Navigation dans les modèles de classe

- Quelles sont les transactions survenues sur un compte pendant une période donnée ?
- Quel volume de transactions une institution a-t-elle gérée l'année dernière ?
- Quels clients ont effectués un achat chez un fournisseur avec une carte de crédit quelconque ?
- Combien de cartes de crédit un client possède-t-il actuellement ?
- Quel est le crédit total maximum d'un client, tous ses comptes confondus ?

OCL (Object Constraint Language)

- Attribut : successivement le nom de l'objet source, une point (.), le nom de l'attribut. *unCompteCarteCrédit.créditMaximum*
- Opération : successivement le nom de l'objet source, un point (.), le nom de l'opération. Une opération est suivie d'une paire de parenthèses, même si elle n'a pas d'argument.
- Opérations spéciales, opérant sur des collections (*count*, *sum* …) : successivement le nom de la collection source, les caractères ->, le nom de l'opération.

OCL (Object Constraint Language)

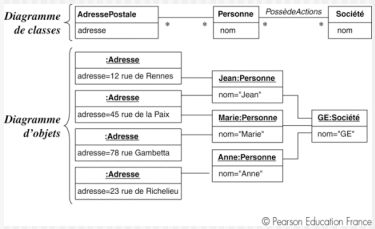
- Association simple : lorsqu'on navigue à travers une association jusqu'à une extrémité cible, cette dernière peut être indiquée par un nom d'extrémité d'association ou, en l'absence d'ambiguïté, par un nom de classe. *unClient.AdressePostale* génère un ensemble d'adresses alors que *unCompteCarteCrédit.AdressePostale* génère une seule adresse.
- Association qualifiée : un qualificateur permet une traversée plus précise. *unCompteCarteCrédit.Relevé[30 novembre 1999]* trouve pour un compte donné le relevé émis le 30 novembre 1999.
- Classe-association : étant donné un lien d'une classe-association, vous pouvez trouver les objets constituants. Inversement, étant donné un objet constituant, vous pouvez trouver les multiples liens d'une classe-association.

OCL (Object Constraint Language)

- Généralisation : la traversée d'une généralisation est implicite.
- Filtre : OCL possède plusieurs types de filtres, dont le plus courant est l'opération *select*. L'instruction *unRelevé.Transaction->select(montant > 100 €)* retourne les transactions d'un relevé dépassant cent euros.

Construction d'expressions OCL

- Une expression OCL peut générer un bag.



Exemples

- Quelles sont les transactions survenues sur un compte pendant une période donnée ?

unCompteCarteCrédit.Relevé.Transaction->select(uneDateDébut <= dateTransaction and dateTransaction <= uneDateFin)

- Quel volume de transactions une institution a-t-elle gérée l'année dernière ?

uneInstitution.CompteCarteCrédit.Relevé.Transaction->select(uneDateDébut <= dateTransaction and dateTransaction <= uneDateFin).montant->sum()

- Quels clients ont effectués un achat chez un fournisseur avec une carte de crédit quelconque ?

unFournisseur.Achat->select(uneDateDébut <= dateTransaction and dateTransaction <=uneDateFin).Releve. CompteCarteCredit.AdressePostale. Client->asSet()

Exemples

- Combien de cartes de crédit un client possède-t-il actuellement ?

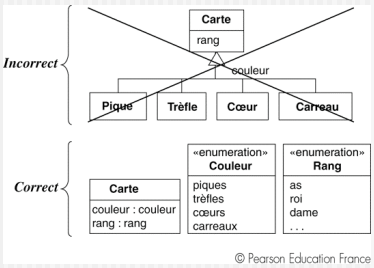
unClient.AdressePostale.CompteCarteCrédit->size()

- Quel est le crédit total maximum d'un client, tous ses comptes confondus ?

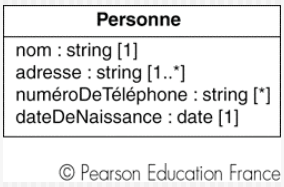
unClient.AdressePostale.CompteCarteCrédit.CréditMaximum-> sum()

Compléments sur la modélisation des classes

Énumerations



Multiplicité d'un attribut



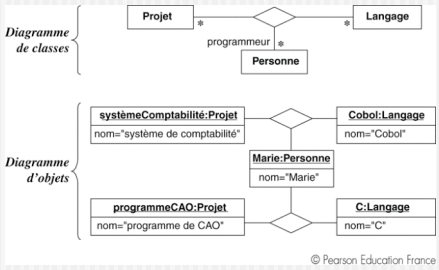
Portée

- La **portée** indique si une propriété s'applique à un objet (par défaut) ou à une classe. Les propriétés dont la portée est la classe (statiques) sont soulignés.

Visibilité

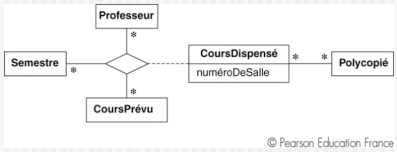
- Les valeurs possibles sont
  - *public* : +
  - *protected* : #
  - *private* : -
  - *package* : ~
- L'absence de préfixe ne révèle aucune information sur la visibilité.

Associations *n*-aires



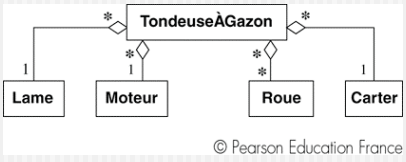
Associations *n*-aires

- Les associations *n*-aires peuvent avoir des classes-associations.
- Les langages de programmation classiques ne permettent pas d'exprimer une association *n*-aire.



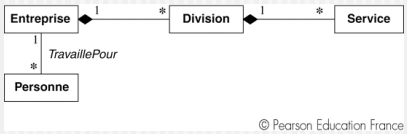
Agrégation

- L'**agrégation** est une forme d'association forte dans laquelle un objet agrégat est *constitué* de constituants *agrégés*. Elle est transitive et antisymétrique.



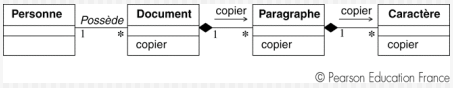
Composition

- Forme d'agrégation qui implique :
  - qu'une partie constituante ne peut appartenir à plus d'un assemblage ;
  - une fois une partie constituante affectée à un assemblage, sa durée de vie coïncide avec ce dernier.



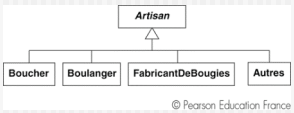
Propagation (ou déclenchement)

- Application automatique d'une opération à un réseau d'objets à partir d'un objet initial quelconque.



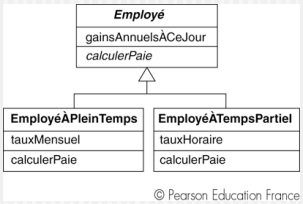
Classes abstraites

- Une classe **abstraite** est une classe qui ne peut être instanciée.



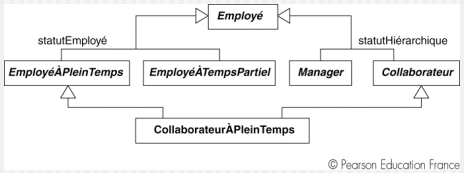
Classes abstraites

- Une classe **abstraite** peut contenir des opérations abstraites, c'est à dire définir une signature sans donner la méthode correspondante.



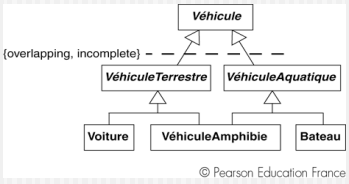
Héritage multiple

- Héritage multiple entre classes disjointes.



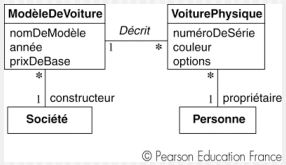
Héritage multiple

- Héritage multiple avec recouvrement. *overlapping* signifie qu'une instance peut appartenir à plusieurs sous-classes, *incomplete* que toutes les sous-classes n'ont pas été indiquées.



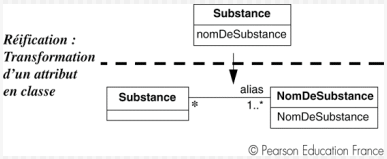
Métadonnée

- Une métadonnée est une donnée qui en décrit une autre.

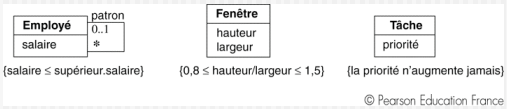


Réification

- La réification consiste à transformer une entité qui n'est pas un objet en objet. Elle peut être utile dans les méta-applications.



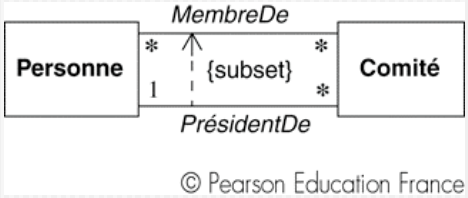
Contraintes



Contraintes sur les ensembles de généralisation

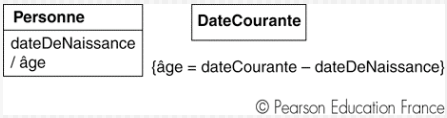
- *Disjoint* : les sous-classes sont mutuellement exclusives.
- *Overlappling* : les sous-classes peuvent partager des objets.
- *Complete* : la généralisation énumère toutes les sous-classes possibles.
- *Incomplete* : certaines sous-classes peuvent être absentes.

Contraintes sur les liens

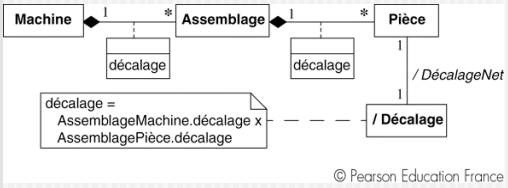


Données dérivées

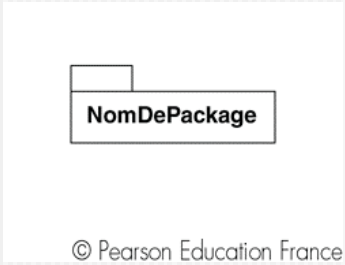
- Un élément dérivé est une fonction (au sens mathématique) d'un ou plusieurs éléments, qui peuvent à leur tour être dérivés. L'élément dérivé est redondant parce que les autres le déterminent entièrement. La notation d'un élément dérivé est une barre oblique (/).



Données dérivées



Paquetages



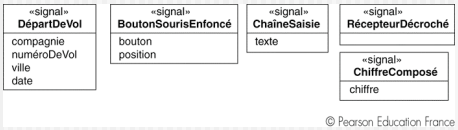
Modélisation des états

Événements

- Un **événement** est une occurrence ou un fait qui a lieu à un moment donné. Par exemple
  - *L'utilisateur appuie sur le bouton de gauche*
  - *Le vol 123 part de Chicago*
  - *Électricité allumée*
  - *La température passe en dessous de zéro.*

Événements de signal

- Événement qui consiste à émettre ou recevoir un signal
- Un **signal** est un message entre objets, alors que **l'événement de signal** en est une occurrence dans le temps.



Événements de changement

- Un **événement de changement** est causé par la satisfaction d'une expression booléenne. La notation UML de ce type d'événement est le mot-clé *when.* Par exemple :
  when (charge de la batterie < limité inférieure)

Événement temporel

- Un **événement temporel** est causé par l'occurrence d'un temps absolu ou par l'écoulement d'une durée.
- Utilise le mot-clé *when*
  when(date = 01/01/2006)
  ou *after*
  after(10 secondes)

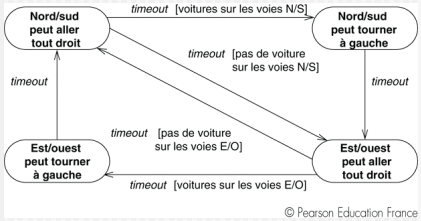
État

- Un **état** est une abstraction des valeurs ou des liens d'un objet. Par exemple, l'état d'une banque est *solvable* si ses actifs excèdent ses dettes.

Transition et condition

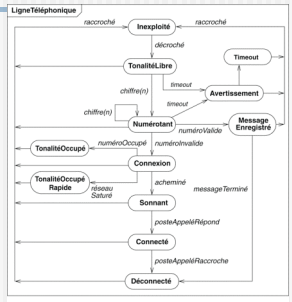
- Une **transition** est le passage instantané d'un état à un autre. On dit que la transition est **franchie** lors du passage de l'état source à l'état cible.
- Une **condition de franchissement** est une expression booléenne qui doit être vraie pour qu'une transition soit franchie.
- Une condition de franchissement n'est évaluée qu'une seule fois, alors qu'un événement de changement est évalué en permanence.
- Le nom d'un événement est en italique et une condition entre crochets.

Transition et condition



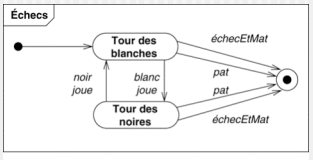
Les feux pour aller à gauche sur les voies NS et EO ne passent au vert que s'il y a présence de voitures. Sinon, c'est le feu opposé qui passe directement au vert.

Diagrammes d'états



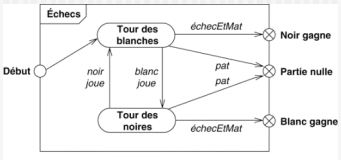
Diagrammes d'états irréversibles

- Les diagrammes d'états irréversibles représentent des objets dont la durée de vie est finie et qui possèdent un état initial et un état final. L'objet entre dans l'état initial à sa création. L'entrée dans l'état final implique la destruction de l'objet.

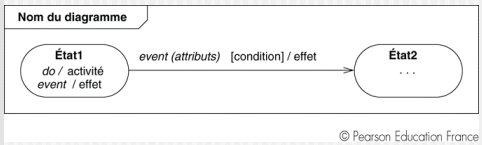


Diagrammes d'états irréversibles

- Il est possible d'indiquer des états initiaux et finaux via des points d'entrée et de sortie, qui sont positionnés sur le périmètre du diagramme. Ces états peuvent être nommés.

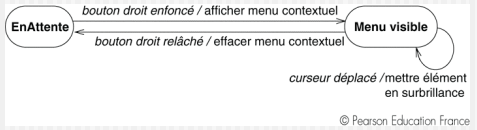


Résumé de la notation de base des diagrammes d'états



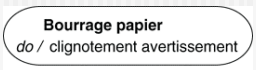
Effets et activités

- Un **effet** est une référence à un comportement exécuté en réponse à un événement.
- Une **activité** est le comportement réel qui peut être invoqué par un nombre quelconque d'effets.
- La notation d'une activité est une barre oblique (*/*) suivie du nom de l'activité, le tout placé après le nom de l'événement qui provoque l'activité.



Activité-do

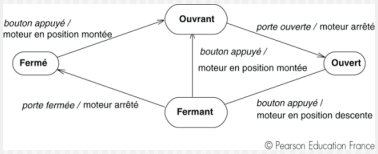
- Une activité associée au mot-clé do (**activité-do**) a lieu pendant une durée significative.
- Par définition, une telle activité ne peut se produire que dans un état et ne peut être attachée à un transition.



Activités d'entrée et de sortie

- Au lieu de représenter des activités sur des transitions, vous pouvez les lier à l'entrée ou à la sortie d'un état. Les deux notations ont le même pouvoir expressif.

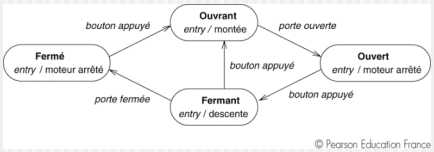
Activités sur les transitions



Activités d'entrée et de sortie

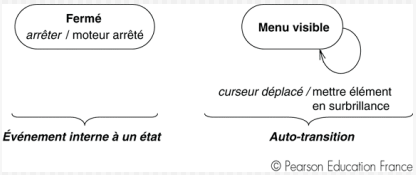
- Une activité d'entrée (resp. sortie) est représentée dans l'état précédée par *entry/* (resp. *exit/*).

Activités liées à l'entrée dans un état



Événement interne vs. Auto-transition

- Seule l'auto-transition provoque l'exécution des activités d'entrée et de sortie



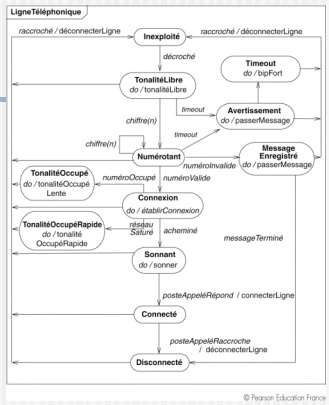
Transition d'achèvement

- Une flèche sans nom d'événement indique une transition automatique, franchie quand l'activité associée à l'état source est achevée

Envoi de signaux

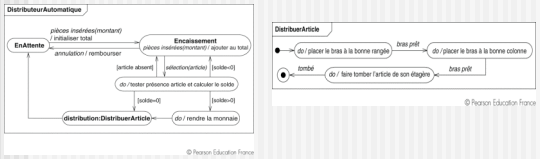
- Un objet peut exécuter une activité consistant à envoyer un signal à un autre objet.
- L'activité *send.cible.S(attributs)* envoie le signal S avec le ou les attributs indiqués à l'objet (ou aux objets) cible(s).
- Si un objet peut recevoir des signaux de plusieurs objets, l'ordre dans lequel les signaux concurrents sont reçus peut affecter l'état final. C'est ce que l'on nomme *condition de concurrence critique*.

Exemple



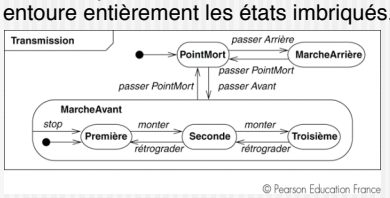
États développés

- Une possibilité pour structurer un modèle consiste à tracer un diagramme de haut niveau avec des sous-diagrammes qui développent certains états



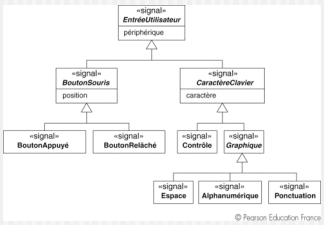
États imbriqués

- Il est possible d'imbriquer les schémas pour montrer leurs points communs et les comportements qu'ils partagent. Le nom de l'état *composite* est associé au contour qui entoure entièrement les états imbriqués.



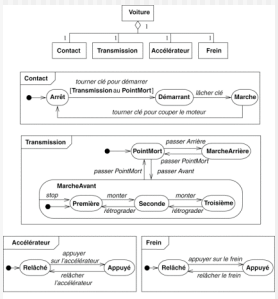
Généralisation de signaux

- Les signaux peuvent être organisés en une hiérarchie de généralisation avec héritage des attributs



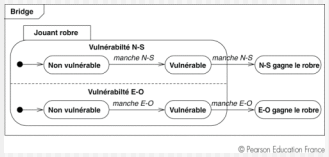
Concurrence d'agrégation

- Le diagramme d'états d'un assemblage est une collection de diagrammes d'états, un diagramme par sous-partie.



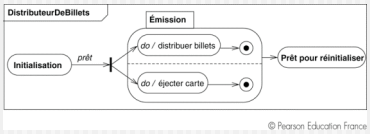
Concurrence à l'intérieur d'un objet

- Il est possible de partitionner des objets en sous-ensembles d'attributs ou de liens, chacun d'eux ayant son propre sous-diagramme.



Synchronisation des activités concurrentes

- Parfois un objet doit exécuter plusieurs activités concurremment. Les étapes internes de ces activités ne sont pas synchronisés, mais les activités doivent être achevées avant que l'objet ne puisse passer dans l'état suivant.



Exemple

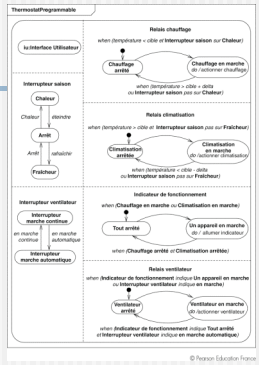
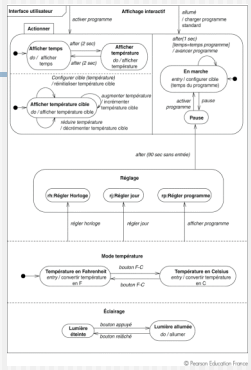


Diagramme d'états d'un thermostat programmable

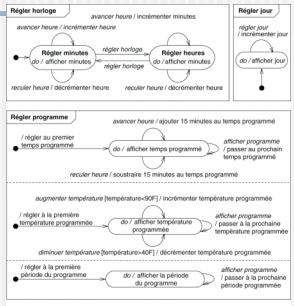
Exemple (suite)

Sous-diagramme de l'interface utilisateur du thermostat



Exemple (suite)

Sous-diagrammes de l'interface utilisateur du réglage du thermostat



Relations entre modèle de classes et modèle d'états

- Le modèle d'états spécifie les séquences possibles de modification des objets du modèle de classes. Un diagramme d'états décrit tout ou partie du comportement des objets d'une classe donnée. Les états sont des classes d'équivalence des valeurs et des liens d'un objet.

Modélisation des interactions

Modèles d'interactions

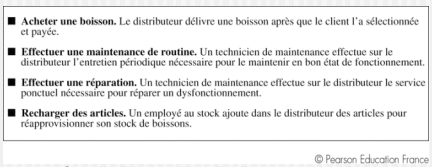
- Modèles de cas d'utilisation
- Modèles de séquences
- Modèles d'activités

Acteur

- Un acteur est un utilisateur direct du système : un objet ou un ensemble d'objets qui communique directement avec le système sans en faire partie.
- Un acteur a un objectif unique, bien défini.

Cas d'utilisation

- Un cas d'utilisation est une partie cohérente des fonctionnalités qu'un système peut fournir en interagissant avec les acteurs.
- Un système implique un ensemble de cas d'utilisation et un ensemble d'acteurs.
- L'ensemble des cas d'utilisation représente l'ensemble des fonctionnalités du système à un niveau de détail donné.



- **Acheter une boisson.** Le distributeur délivre une boisson après que le client l'a sélectionnée et payée.
- **Effectuer une maintenance de routine.** Un technicien de maintenance effectue sur le distributeur l'entretien périodique nécessaire pour le maintenir en bon état de fonctionnement.
- **Effectuer une réparation.** Un technicien de maintenance effectue sur le distributeur le service ponctuel nécessaire pour réparer un dysfonctionnement.
- **Recharger des articles.** Un employé au stock ajoute dans le distributeur des articles pour réapprovisionner son stock de boissons.

Cas d'utilisation d'un distributeur automatique

Cas d'utilisation

- Un cas d'utilisation rassemble tous les comportements significatifs pour une fonctionnalité donnée du système.
- Un cas d'utilisation doit représenter une transaction complète, qui apportent une valeur aux utilisateurs. Par exemple, *composer un numéro* n'est pas un bon cas d'utilisation ; il fait simplement partir du cas d'utilisation *passer un appel*.

Cas d'utilisation : Acheter une boisson.

Résumé : Le distributeur délivre une boisson après que le client l'a sélectionnée et payée.

Acteurs : Client.

Préconditions : Le distributeur est en attente de l'insertion de pièces.

Description : Le distributeur commence dans l'état « en attente » dans lequel il affiche le message « Insérez de la monnaie ». Un client insère des pièces dans le distributeur. Le distributeur affiche le montant total inséré et allume les boutons correspondant aux articles pouvant être achetés avec ce montant. Le client appuie sur un bouton. Le distributeur délivre l'article correspondant et rend la monnaie si le coût de la boisson est inférieur au montant inséré.

Exceptions :

Annulation : Si le client appuie sur le bouton Annulation avant de sélectionner un article, le montant qu'il a inséré lui est rendu et le distributeur se remet dans l'état « en attente ».

Épuisé : Si le client appuie sur le bouton d'un article épuisé, le message « Article momentanément indisponible » est affiché. Le distributeur continue à accepter des pièces ou une sélection.

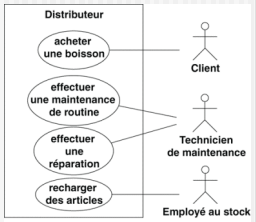
Somme insuffisante : Si le client appuie sur le bouton d'un article dont le montant est plus élevé que la somme insérée, le distributeur affiche le message « Vous devez insérer *nn,nn* de plus », où *nn,nn* est le montant manquant. Le distributeur continue à accepter des pièces ou une sélection.

Pas de monnaie : Si le client a inséré suffisamment de pièces pour acheter l'article et que la machine ne soit pas en mesure de rendre la monnaie correcte, le message « Impossible de rendre la monnaie » est affiché et le distributeur continue à accepter des pièces ou une sélection.

Postconditions : Le distributeur est en attente de l'insertion de pièces.

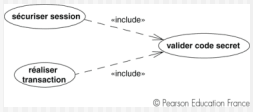
Diagramme de cas d'utilisation

- Un rectangle contient les cas d'utilisation du système.
- Un nom dans une ellipse représente un cas d'utilisation.
- L'icône d'un bonhomme représente un acteur
- Des lignes connectent les cas d'utilisation aux acteurs qui y participent.



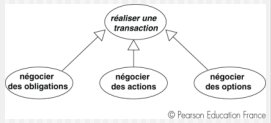
Relations entre cas d'utilisation

- La relation *include* insère un cas d'utilisation dans la séquence des comportements d'un autre cas d'utilisation.
- La relation *extend* ajoute un comportement incrémental à un cas d'utilisation.

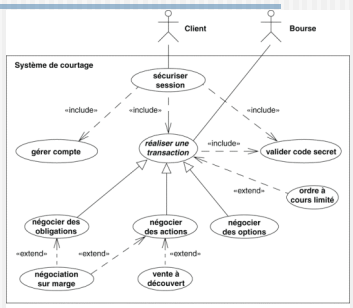


Relations entre cas d'utilisation

- La généralisation permet de représenter les variantes spécifiques d'un cas d'utilisation générale, de façon analogue à la généralisation de classes.



Combinaison de relations entre cas d'utilisation



Modèle de séquence

- Le modèle de séquence précise les thèmes fonctionnels introduits par les cas d'utilisation.
- Deux types de modèles existent : les scénarios et les diagrammes de séquence.

Scénario

- Un scénario est une séquence d'événements qui ont lieu lors du fonctionnement du système.

Bernard Martin se connecte.
Le système établit une connexion sécurisée.
Le système affiche les informations concernant le portefeuille.
Bernard Martin saisit un ordre d'achat pour 100 actions de GE au prix du marché.
Le système vérifie que les fonds sont suffisants pour la réalisation de la transaction.
Le système affiche un écran de confirmation avec le coût estimé.
Bernard Martin confirme l'achat.
Le système passe l'ordre à la bourse.
Le système affiche le numéro d'ordre de la transaction.
Bernard Martin se déconnecte.
Le système établit une connexion non sécurisée.
Le système affiche un écran de remerciement.
Un compte-rendu de la transaction est réalisé et transmis au système de courtage en ligne.

Diagramme de séquence

- Un diagramme de séquence représente les participants à une interaction et la séquence des messages qu'ils s'envoient.
- Les acteurs et le système sont représentés par des lignes verticales appelées *lignes de vie* et chaque message est symbolisé par une flèche horizontale allant de l'émetteur au récepteur.
- Le temps d'écoule sur l'axe vertical, du haut vers le bas, sans échelle de temps.
- Les diagrammes de séquence peuvent contenir des signaux concurrents.
- La description du comportement de chaque cas d'utilisation nécessite plusieurs cas d'utilisation.

Diagramme de séquence

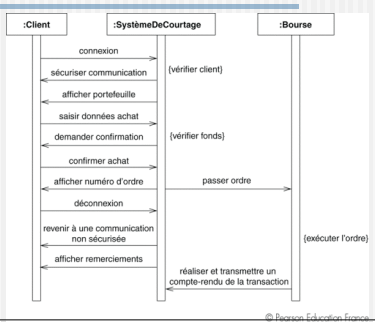
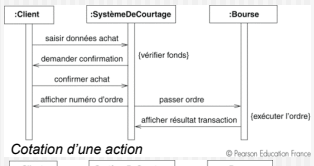


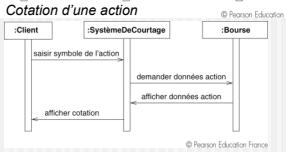
Diagramme de séquence

- Pour éviter les répétitions, vous pouvez tracer un diagramme de séquence séparé pour chaque tâche.

Achat d'une action

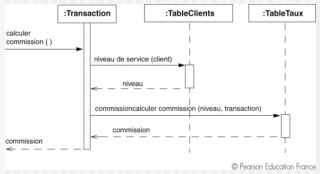


Cotation d'une action

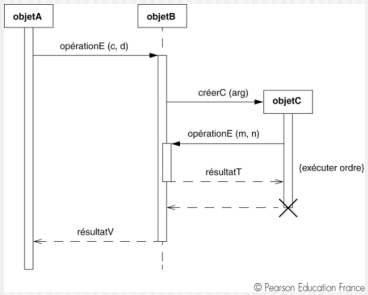


Objets passifs

- Un objet passif (sans thread de contrôle propre) n'est pas activé tant que que l'un de ses comportements n'a pas été déclanché.
- Une fois que l'opération est terminée et que le contrôle retourne à l'appelant, l'objet passif devient inactif.

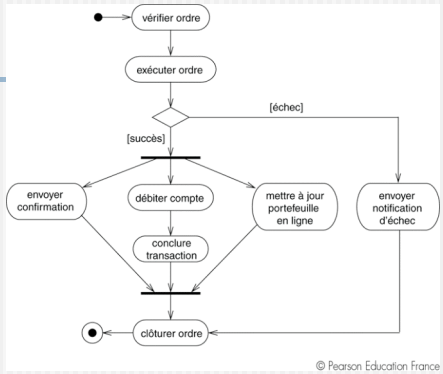


Objets temporaires



Modèle d'activités

- Un diagramme d'activités représente la suite d'étapes qui constituent un processus complexe, par exemple un algorithme. Il exprime le flux de contrôle, en se concentrant sur les opérations plutôt que sur les objets.



Activités

- Les étapes d'un diagramme d'activités sont les opérations, plus spécifiquement les activités décrites dans le modèle d'états.
- L'objectif d'un diagramme d'activités est de représenter les étapes d'un processus complexe et les contraintes de séquencement.
- Certaines activités de déroulent en continu jusqu'à ce qu'un évenement exterieur ne les interrompe, mais la plupart finissent par achever leur travail et s'interrompent d'elles-mêmes .

Activités

- Une activité peut se décomposer en activités plus fines
- Si une activité a plus d'un successeur, chaque flèche doit porter une étiquette avec une condition entre crochets
- Un cercle plein avec une flèche sortante représente le début d'une activité, un cercle plein entouré d'un cercle vide un point d'achèvement

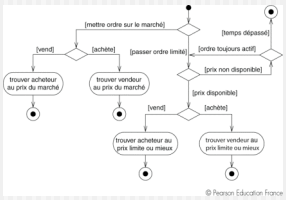
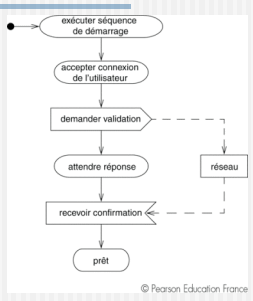


*Diagramme d'activité d'*exécuter ordre

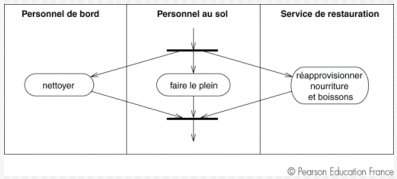
Emission et réception de signaux

- L'émission d'un signal est représenté par un pentagone convexe, la réception par un pentagone concave.



Couloir d'activité

- Dans un modèle métier, il est utile de savoir quelles entités de l'organisation humaine sont responsables d'une activité.
- Placer une activité dans un couloir d'activité particulier signifie qu'elle est effectuée par une ou plusieurs personnes de l'organisation.



Flux d'activité

- Il est parfois utile de visualiser les relations entre une opération et les objets qui sont les valeurs de ses arguments ou ses résultats.
- Souvent un même objet passe par différents états durant l'exécution d'un diagramme d'activité.