

Analyse Oriente Objet VS Analyse Fonctionnelle

Approche fonctionnelle vs. approche objet

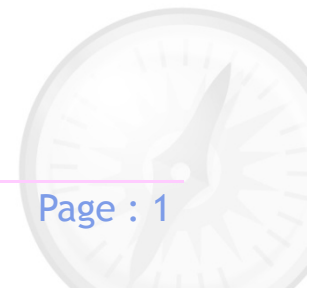
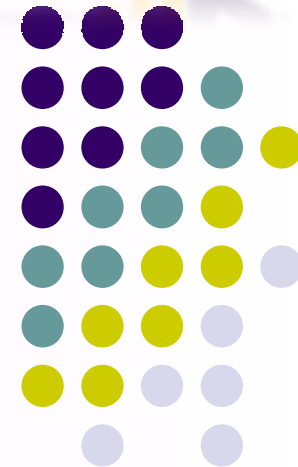
la thèse de **Church-Turing**, tout langage de programmation non trivial équivaut à une machine de Turing. Il en résulte que tout programme qu'il est possible d'écrire dans un langage pourrait également être écrit dans n'importe quel autre langage. Ainsi, la différence entre une approche fonctionnelle et une approche objet n'est donc pas d'ordre logique, mais pratique.

L'approche structurée privilégie la fonction comme moyen d'organisation du logiciel. Ce n'est pas pour cette raison que l'approche objet est une approche non fonctionnelle. En effet, les méthodes d'un objet sont des fonctions. Cependant les traitements et les données sont associés.

En approche objet, l'évolution des besoins aura le plus souvent tendance à se présenter comme un changement de l'interaction des objets. S'il faut apporter une modification aux données, seul l'objet en cause (encapsulant cette donnée) sera modifié.

l'évolution des besoins entraîne souvent une dégénérescence, ou une profonde remise en question, une modification des données entraîne généralement une modification d'un nombre important de fonctions

Ainsi la technologie objet permet une modularisation du logiciel, qui vise à maîtriser sa production et son évolution.



Approche 2O.O



L'approche orientée objet

L'approche orientée objet considère le logiciel comme une collection d'objets dissociés, identifiés et possédant des caractéristiques. Une caractéristique :

L'identité - L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état. On construit généralement cette identité grâce à un identifiant découlant naturellement du problème (par exemple un produit pourra être repéré par un code, une voiture par un numéro de chassis, etc.)

Les attributs - Il s'agit des données caractérisant l'objet. Ce sont des variables stockant des informations sur l'état de l'objet.

Les méthodes - Les méthodes d'un objet caractérisent son comportement, c'est-à-dire l'ensemble des actions (appelées opérations) que l'objet est à même de réaliser. Ces opérations permettent de faire réagir l'objet aux sollicitations extérieures (ou d'agir sur les autres objets). De plus, les opérations sont étroitement liées aux attributs, car leurs actions peuvent dépendre des valeurs des attributs, ou bien les modifier.

L'une des particularités de cette approche est qu'elle rapproche les données et leurs traitements associés au sein d'un unique objet.



Concepts O.O



L'encapsulation

consiste à masquer les détails son implémentation d'un objet, en définissant une interface. L'interface est la vue externe d'un objet, elle définit les services accessibles (offerts) aux utilisateurs de l'objet. On peut modifier l'implémentation des attributs d'un objet sans modifier son interface. L'encapsulation garantit l'intégrité des données, car elle permet d'interdire l'accès direct aux attributs des objets.

Accessibilité	Public	Private	Protected	Package freindly
A l'intérieur de la classe	Oui	Oui	Oui	Oui
Classes Package	Oui	Non	Oui	Oui
Classes dérivées	Oui	Non	Oui	Non
Classes hors packages	Oui	Non	Non	Non

L'héritage

est un mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe. Une classe peut être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques ou d'en adapter certaines. Plusieurs classes peuvent être généralisées en un classe qui les factorise, afin de regrouper les caractéristiques communes d'un ensemble de classes. L'héritage peut être simple ou multiple.



Concepts O.O



Polymorphisme

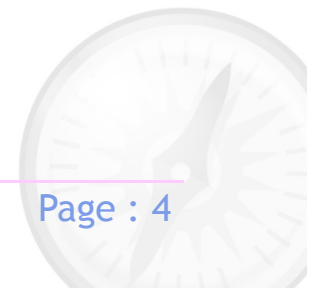
représente la faculté d'une même opération de s'exécuter différemment suivant le contexte de la classe où elle se trouve.

l'agrégation

Il s'agit d'une relation entre deux classes, spécifiant que les objets d'une classe sont des composants de l'autre classe.

La composition :

Est une agrégation forte : la classe composée et les classes composants ont la même durée de vie. Et la multiplicité du côté de la classe agrégat ou composée est 1



Modélisation pourquoi ?



Les techniques de modélisation couvrent quatre aspects:

- Elles aident à spécifier.
- Elles aident à visualiser.
- Elles aident à construire.
- Elles aident à documenter.

Les techniques de modélisation se présentent souvent comme un ensemble de concepts, de règles de construction, accompagnés d'un formalisme (langage ou graphique), qui permet de visualiser les résultats de la réflexion sur le système modélisé.

Les techniques servant à spécifier et à construire sont appelés **modèles**. En fait ce sont des méta-modèles permettant de construire les modèles de Système à modéliser.



Notion de diagramme



Le rôle d'un diagramme, s'il est suffisamment proche de la réalité est de nous donner des renseignements sur le système réel (de manière descriptive: statique ou comportementale).

Son rôle est donc de nous renseigner sur le système réel. Un diagramme de système d'information doit permettre d'expérimenter l'aspect statique et dynamique du système réel.

Ce diagramme peut représenter des éléments d'un artefact déjà construit ou à construire. Ce diagramme peut représenter l'artefact sous différents aspects (structurel, comportemental, fonctionnel) pour différentes préoccupations (conceptuel, organisationnel, logique, technique).



Méthode d'analyse et conception



Une méthode se doit de proposer 4 éléments fondamentaux :

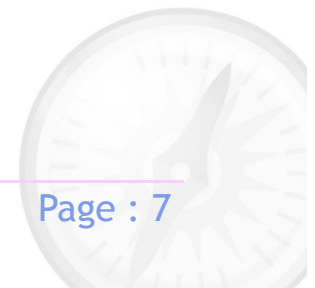
Elle doit décrire une **DEMARCHE**, qui liste les tâches à effectuer pour conduire un projet

systèmes d'information.

Elle doit fournir un **MODELE** pour décrire la sémantique des données, leur comportement.

Elle doit fournir un ensemble de **DIAGRAMMES** s'appuyant sur un **FORMALISME** de description (graphique ou langage).

Il doit être possible de trouver sur le marché des **OUTILS LOGICIELS D'AIDE** à la conception. Ces outils portent le nom d'A.G.L (Atelier de Génie Logiciel) ou C.A.S.E (Computer Aided Software Engineering)



Introduction à UML



Introduction à UML

La description de la programmation par objets a fait ressortir l'étendue du travail Conceptuel nécessaire : définition des classes, de leurs relations, des attributs et méthodes, des interfaces etc.

Pour programmer une application, il ne convient pas de se lancer tête baissée dans l'écriture du code : il faut d'abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation. C'est cette démarche antérieure à l'écriture que l'on appelle modélisation ; son produit est un modèle.

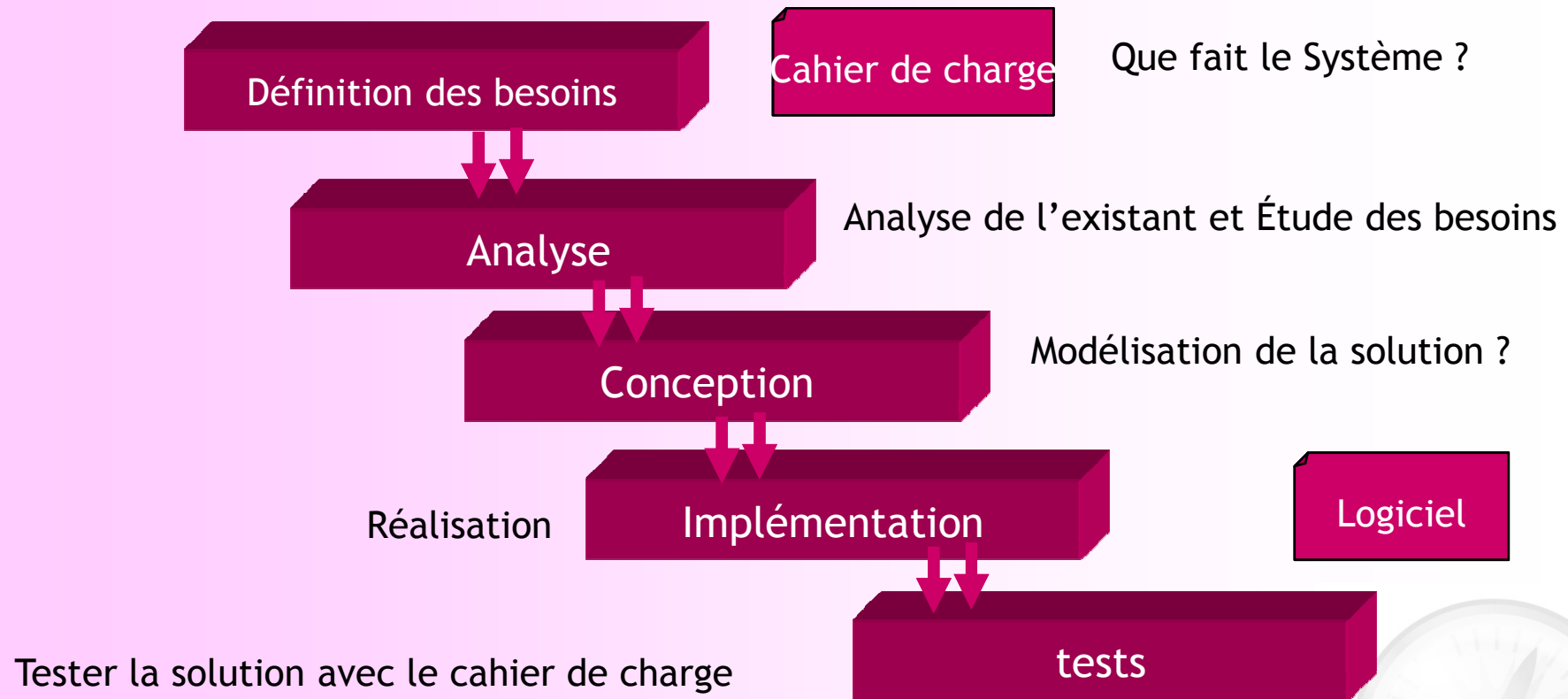
L'approche objet permet en principe à la maîtrise d'ouvrage de s'exprimer de façon précise selon un vocabulaire qui, tout en transcrivant les besoins du métier, pourra être immédiatement compris par les informaticiens.



Introduction à UML



Les phases de Modélisation



Historique



Los Amigos

Apparition de + ieurs Méthodes :
(Booch, Classe-Relation, Fusion, HOOD,
OMT, OOA, OOD, OOM, OOSE, etc.)

1979-1980

Merise

1990

1995

Booch et
Rumbaugh
construisent une
méthode unifiée,
Unified Method 0.8

1996

Jacobson
produit
UML 0.9

1997

L'OMG
adopte
UML 1.1

2008

UML 2.1.1
La version
d'UML en
cours en




Description d'UML

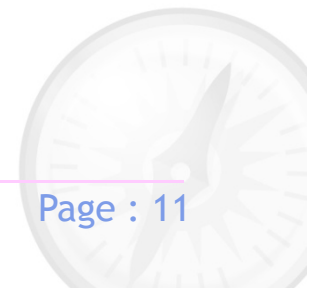


U.M.L (UNIFIED MODELING LANGUAGE).

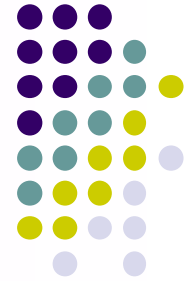
U.M.L est un langage qui contient les éléments constitutifs de tout langage, à savoir : des concepts, une syntaxe et une sémantique. Il est destiné aux phases amont de la réalisation d'un logiciel. U.M.L est une Technique de modélisation UNIFIEE issue de méthodes plus anciennes comme O.M.T, de O.O.S.E et de O.O.D.

De plus, UML a choisi une notation supplémentaire : il s'agit d'une forme visuelle Fondée sur des diagrammes.

- **UML n'est pas une méthode** (i.e. une description normative des étapes de la modélisation) : un langage graphique qui permet de représenter et de communiquer les divers aspects d'un système d'information. 
- Aux graphiques sont bien sûr associés des textes qui expliquent leur contenu. UML est donc un métalangage qui fournit les éléments permettant de construire le modèle Du langage de du projet.
- Les auteurs d'UML préconisent d'utiliser une démarche :
 - ✓ guidée par les besoins des utilisateurs du système,
 - ✓ centrée sur l'architecture logicielle,
 - ✓ itérative et incrémentale.



Diagrammes UML (13)



UML 2.0 comporte ainsi treize types de diagrammes. Ils se répartissent en deux grands groupes :

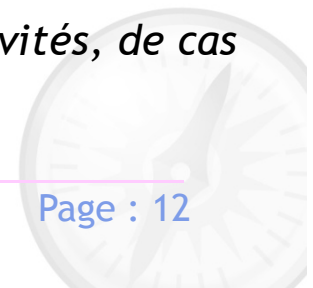
Diagrammes structurels ou diagrammes statiques (UML Structure)

- diagramme de classes (Class diagram)
- diagramme d'objets (Object diagram)
- diagramme de composants (Component diagram)
- diagramme de déploiement (Deployment diagram)
- diagramme de paquetages (Package diagram)
- diagramme de structures composites (Composite structure diagram)

Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior)

- diagramme de cas d'utilisation (Use case diagram)
- diagramme d'activités (Activity diagram)
- diagramme d'états-transitions (State machine diagram)
- **Diagrammes d'interaction (Interaction diagram)**
- diagramme de séquence (Sequence diagram)
- diagramme de communication (Communication diagram)
- diagramme global d'interaction (Interaction overview diagram)
- diagramme de temps (Timing diagram)

- *Les plus utiles pour la maîtrise d'ouvrage sont les diagrammes : d'activités, de cas d'utilisation, de classes, d'objets, de séquence et d'états-transitions.*



Introduction à UML



Les phases de Modélisation

Dans la suite, nous allons présenter une méthode simple et générique qui se situe à mi-chemin entre UP (Unified Process), qui constitue un cadre général très complet de processus de développement, et XP (eXtreme Programming) qui est une approche minimaliste à la mode centrée sur le code.

Définition des besoins

- 1-Diagramme d'acteurs
- 2- Diagramme de contexte statique
- 3 - Diagramme Uses cases
- 4-Description textuelle de haut niveau

Analyse

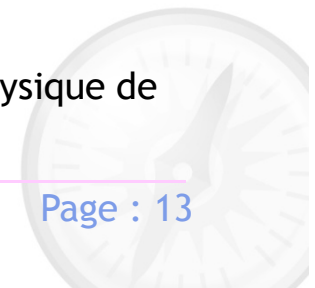
- 1- Réaliser le diagramme de séquence "boîte noire" par scénario de use case détaillé
- 2- Réaliser le diagramme d'activités
- 3- Réaliser le diagramme global d'interaction (Interaction overview diagram)
- 4- Réaliser le diagramme de classes d'analyse (Dialogues, contrôles et métier)
- 5- Réaliser le diagramme de séquence "boîte blanche " (Système => Σ Objets en collaboration)

Conception

- 1- Réaliser le diagramme de collaboration
- 2- Réaliser le diagramme de classe de conception ou de métier
- 3- Réaliser le diagramme d'objets
- 4- Réaliser le diagramme d'états de transitions
- 5- Utiliser quelques des Designs Pattern
- 6- diagramme de composants (Component diagram)
- 7- diagramme de déploiement (Deployment diagram)

Implémentation

- 1- Implémenter en Java le diagramme de classe
- 2- translation du Diagramme de classe au Modèle physique de données



Étapes de la modélisation UML



Définir les besoins :

Déduire le diagramme de contexte statique.

Regrouper les exigences et réaliser le diagramme des uses cases.

Analyse :

1- Réaliser le diagramme de séquence "boîte noire" par scénario de use case détaillé :
(les interactions entre l'acteur et le système informatique : événements et opérations ;
agrémenter le diagramme de séquences de notes et de commentaires)

2- Réaliser le diagramme de classe d'analyse :

(recenser les groupes nominaux par use case : les classes et les objets ; réaliser les associations entre les classes et préciser les cardinalités ; enrichir le diagramme de classe en insérant les attributs.

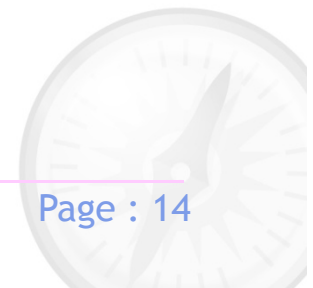
3- Réaliser le diagramme de séquences "boîte blanche".

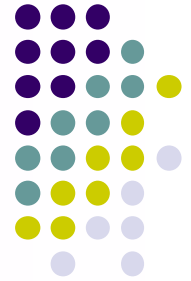
Conception

1- Réaliser le diagramme de collaboration à partir des diagrammes de classe d'analyse et du diagramme de séquence "boîte blanche" :

2- Réaliser en parallèle les diagrammes d'état des objets les plus complexes afin de détecter les méthodes internes à ces objets.

3- Réaliser le diagramme de classe de conception.





Définition des besoins



Ph 1 : Définition de besoins

Diagramme de cas d'utilisation



Définition

Il permet de montrer les différentes possibilités d'utilisation du système. Les concepts utilisés pour les cas d'utilisation sont utilisés pour définir le comportement du système sans spécifier sa structure interne.

Un diagramme de cas d'utilisation représente le comportement d'un système, d'un sous système, d'une classe ou d'un composant avec lequel l'utilisateur interagit.

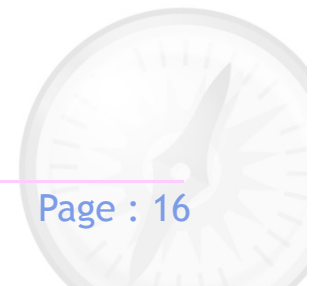
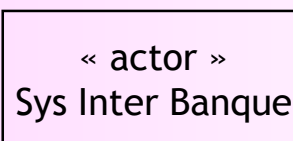
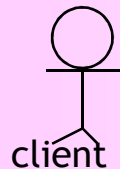
Il décortique la fonctionnalité du système en unités cohérentes, les cas d'utilisation, Ayant un sens pour les acteurs. Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique. *Cette première étape de modélisation permet de produire un logiciel conforme aux attentes des utilisateurs.* Pour élaborer les cas d'utilisation, il faut se fonder sur des entretiens avec les utilisateurs.

Éléments des diagrammes de cas d'utilisation

1- Acteur

Un acteur est l'idéalisation d'un rôle joué par une personne externe, un processus ou Une chose qui interagit avec un système.

- Il se représente par un petit bonhomme avec son nom (i.e. son rôle) inscrit dessous.
- Il est également possible de représenter un acteur sous la forme d'un classeur



Évacuation des acteurs

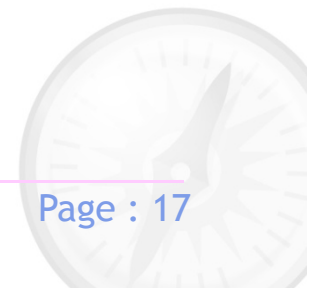
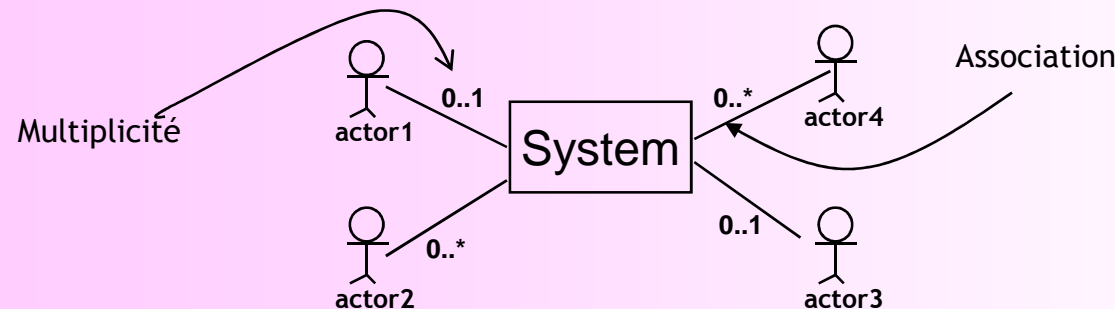


Pour Dégager les acteurs d'un Système , on peut poser les questions suivantes:

- Qui utilise le système.?
- Qui installe le système?
- Qui démarre le système?
- Qui maintient le système?
- Qui ferme le système?
- Quels autres systèmes utilisent le système?
- Qui a besoin d'information venant du système?
- Quelque chose est il produit automatiquement par le système?
- Il est possible de définir une généralisation sur les acteurs?

Diagramme de contexte statique :

Bien que ce diagramme ne fasse pas partie des diagrammes UML « Officiels », on l'a très souvent trouvé utile, il permet de spécifier le nombre d'instances d'acteurs connectés au système à un moment donné.



Éléments d'un Diagrammes de cas d'utilisation

2- Cas d'utilisation :

Un cas d'utilisation est une unité cohérente représentant une fonctionnalité visible de l'extérieur. Il réalise un service de bout en bout, avec un déclenchement, un déroulement et une fin. Un cas d'utilisation se représente par une ellipse

Un cas d'utilisation

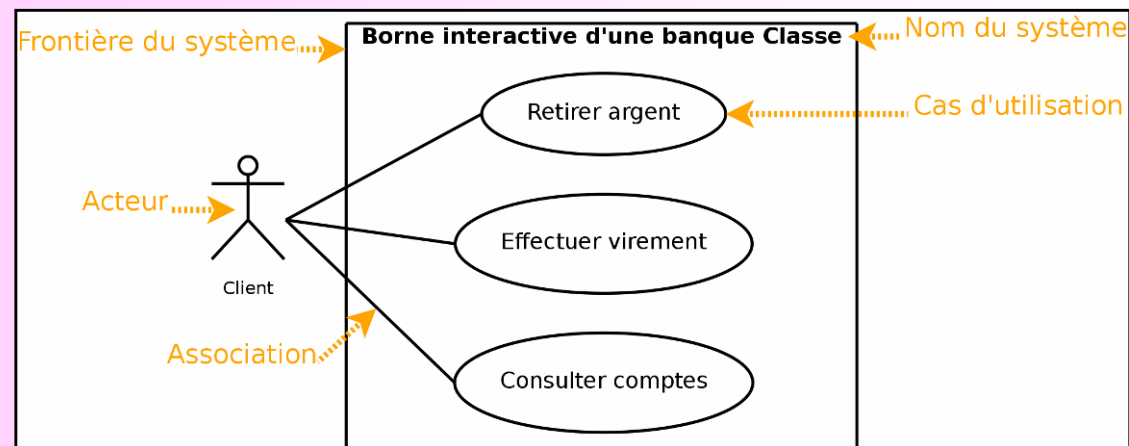
Une relation d'association

Un chemin de communication entre un acteur et un cas d'utilisation est représenté un trait continu

Représentation d'un diagramme de cas d'utilisation

- La frontière du système est représentée par un cadre. Le nom du système figure à l'intérieur du cadre, en haut. Les acteurs sont à l'extérieur et les cas d'utilisation à l'intérieur.

Exemple :
Cas d'utilisation

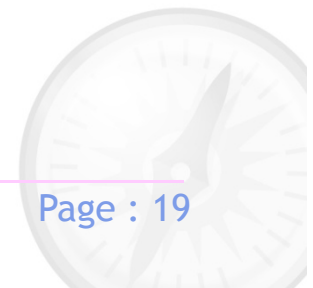
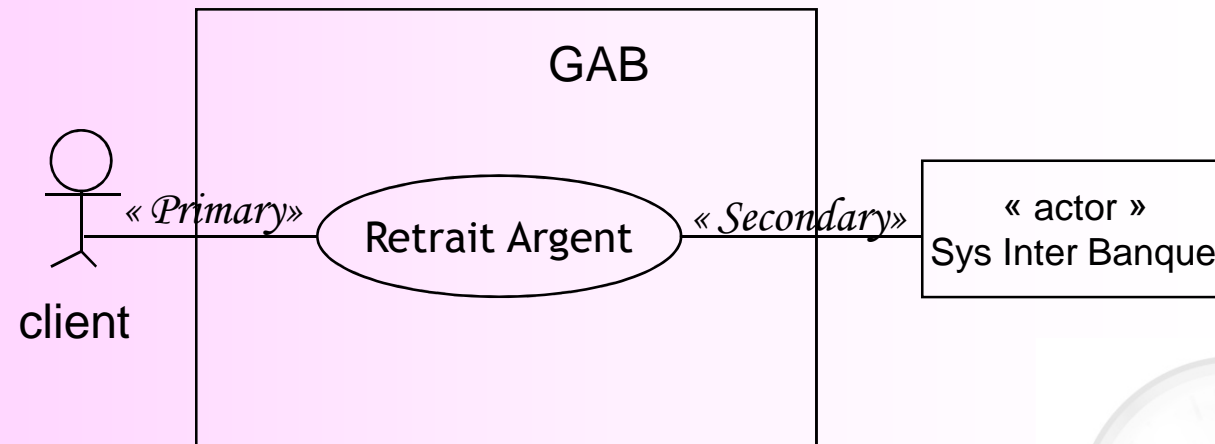


Types d'acteurs : Principal ou Secondaire



Acteurs principaux et secondaires

- Un acteur est qualifié de principal pour un cas d'utilisation lorsque ce cas rend service à cet acteur. Les autres acteurs sont alors qualifiés de secondaires. Un cas d'utilisation a au plus un acteur principal. Un acteur principal obtient un résultat observable du système.
- Un acteur secondaire est sollicité pour des informations complémentaires. En général, l'acteur principal initie le cas d'utilisation par ses sollicitations. Le stéréotype « primary » vient orner l'association reliant un cas d'utilisation à son acteur principal, le stéréotype « secondary » est utilisé pour les acteurs secondaires.



Types d'associations entre cas d'utilisations



Types d'associations et représentations :

Il existe principalement deux types de relations :

- l'inclusion et l'extension
- la généralisation/spécialisation.

Association « include » :

Une relation d'inclusion définit que le cas d'utilisation contient le comportement définit dans un autre cas d'utilisation.

Association « Extend »:

Une relation d'extension définit que l'instance d'un cas d'utilisation peut être augmentée avec un comportement quelconque défini dans un cas d'utilisation étendu. Il faut spécifier le point d'extension sur le cas d'origine de destination.

Association « Généralisation/Spécialisation »:

généralisation entre deux cas d'utilisation, signifie que le cas d'utilisation spécialisé est plus spécifique que le cas d'utilisation général. Le spécialisé hérite de toutes les propriétés et les associations du cas d'utilisation.

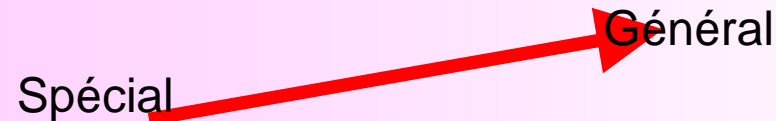
- ✓ *Une dépendance se représente par une flèche avec un trait pointillé. Si le cas A inclut ou étend le cas B, la flèche est dirigée de A vers B.*
- ✓ *Le symbole utilisé pour la généralisation est une flèche avec un trait plein dont la pointe est un triangle fermé désignant le cas le plus général.*



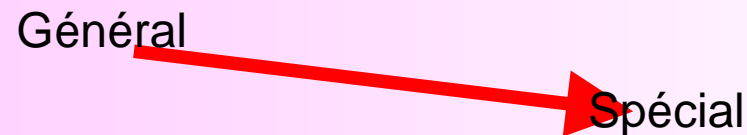
Héritage (Généralisation/Spécialisation)



La généralisation est une association ascendante du spécial au général



La spécialisation est une association descendante du général au spécial
(inverse)

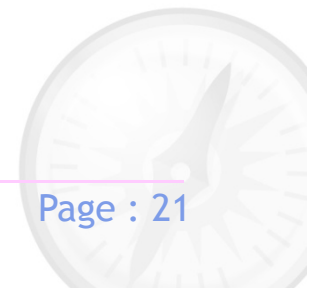


Exemple :

Le Versement bancaire peut se faire par dépôt d'argent par chèque ou dépôt d'argent en espèce. (Spécialisation)

Le dépôt d'argent par chèque ou le dépôt d'argent en espèce se sont des Versements Bancaires. (Généralisation)

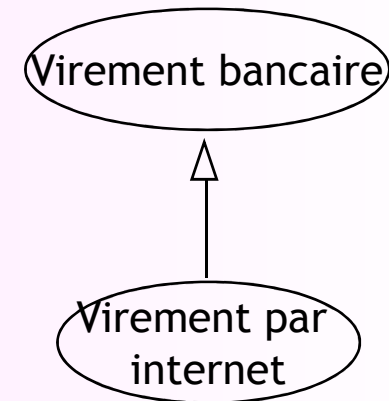
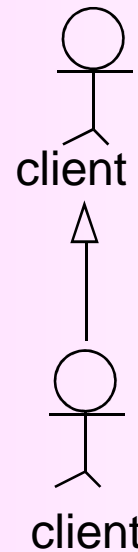
N.B : La même phrase dite inversement.



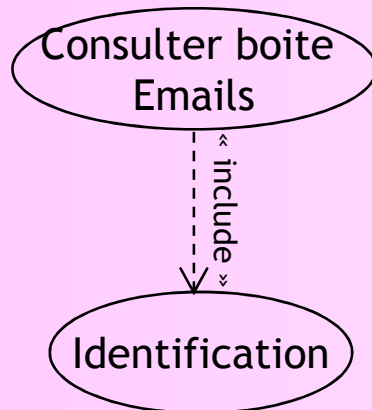
Exemples d'associations



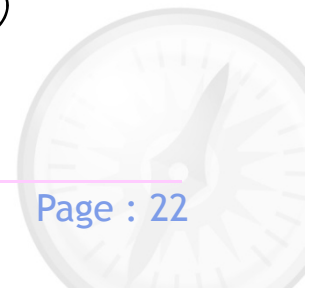
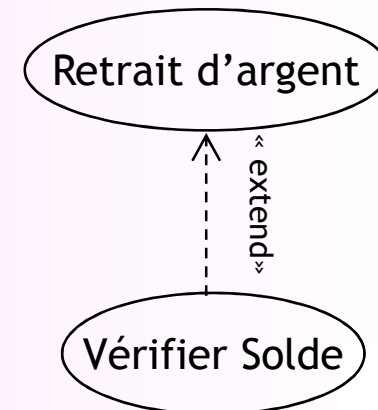
Exemple Généralisation/Spécialisation



Exemple Inclusion



Exemple Extension



Scénarios



Chaque fois qu'une instance d'un acteur déclenche un cas d'utilisation, un scénario est Créé (le cas d'utilisation est instancié). Ce scénario suivra un chemin particulier dans le Cas d'utilisation.

scénario primaire/scénario secondaire:

Il est préférable de commencer à décrire le déroulement normal, basique, c'est à dire La séquence la plus commune: c'est le scénario primaire.

Ensuite, il devient possible d' écrire des alternatives en utilisant des flots d'événements alternatifs. Il est possible d'écrire également les alternatives comme des scénarios secondaires.

Recherche des scénarios secondaires:

Y a t'il quelques autres actions possibles à ce niveau du scénario ?

Y a t'il quelque chose qui pourrait mal fonctionner à ce niveau du scénario ?

Y 'a t'il quelques comportements qui pourraient arriver à ce moment du scénario ?

Message :

Les instances des acteurs communiquent avec le système en envoyant et recevant des instances de messages allant ou venant des instances de cas d'utilisation (au niveau de la réalisation, allant ou provenant des objets).



Description textuelle d'un cas d'utilisation



Description textuelle des cas d'utilisation :

Une description textuelle couramment utilisée se compose de trois parties.

1- La première partie permet d'identifier le cas

Nom : Utiliser un verbe à l'infinitif (ex : Réceptionner un colis).

Objectif : Une description résumée permettant de comprendre l'intention principale du cas d'utilisation.

Acteurs principaux : Ceux qui vont réaliser le cas d'utilisation

Acteurs secondaires : Ceux qui sont sollicités pour des informations complémentaires.

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

Responsable : Le nom des responsables.

Version : Le numéro de version.

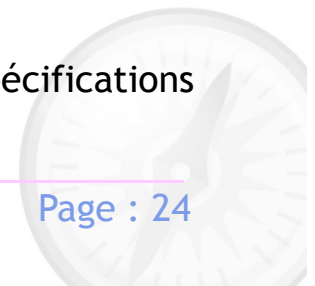
2. La deuxième partie contient la description du fonctionnement du cas sous la forme d'une séquence de messages échangés entre les acteurs et le système. Elle contient Toujours une séquence nominale qui décrit le déroulement normal du cas. À la séquence Nominale s'ajoutent fréquemment des séquences alternatives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

Les préconditions : elles décrivent dans quel état doit être le système (l'application) avant que ce cas d'utilisation puisse être déclenché.

Des scénarii : Ces scénarii sont décrits sous la forme d'échanges d'événements entre l'acteur et le système. On distingue le scénario nominal, qui se déroule quand il n'y a pas d'erreur, des scénarii alternatifs qui sont les variantes du scénario nominal et enfin les scénarii d'exception qui décrivent les cas d'erreurs.

Des postconditions : Elle décrivent l'état du système à l'issue des différents scénarii.

La troisième partie (optionnelle): Elle contient des spécifications non fonctionnelles, (spécifications matérielles et techniques portant sur le temps de réponse, outils, Mono Multitâche...



Description textuelle d'un cas d'utilisation

Exemple :
Scénario nominal

Actions acteurs principal	Actions Système
1	2
	3

Enchaînements alternatifs :

A1 : nom d'enchaînement

- Le point de démarrage à partir d'un point x du scénario nominal
- Les actions numérotées du système avec des numéros
- Le scénario nominal reprend au point y

Enchaînements des erreurs :

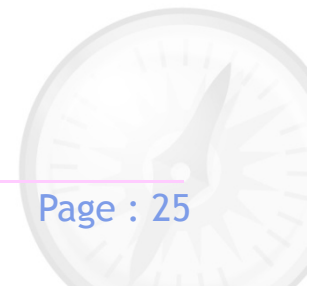
- E1 : Nom d'erreur
- Le point de démarrage à partir d'un point x du scénario nominal
- Les actions numérotées du système suite à cette erreur



Remarque :

Les enchaînements alternatifs et d'erreurs sont causé par l'utilisateur.

Une erreur arrête le scénario nominal, cependant une alternative permet de reprendre le scénario nominal.

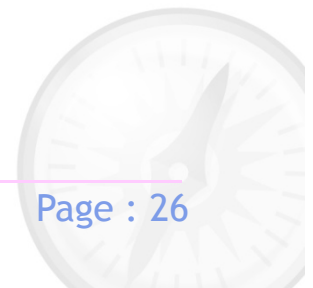


Conclusion

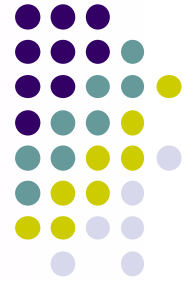
Construction d'un MODELE USES CASES



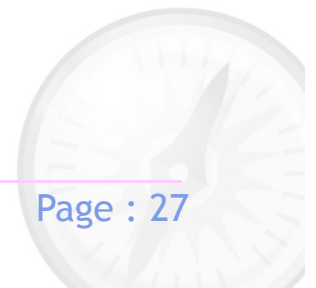
- Identifier les acteurs qui utilisent, qui gèrent, qui exécutent des fonctions spécifiques.
- Organiser les acteurs par relation d'héritage.
- Pour chaque acteur, rechercher les cas d'utilisation avec le système. En particulier, ceux qui modifient l'état du système ou qui attendent une réponse du système.
- Ne pas oublier les différents messages échangés entre acteur et le Système (cas d'erreur, cas alternatifs).
- Organiser associations entre cas d'utilisation par héritage, par inclusion et par extension.



Uses Cases et Devis



- Un uses cases permet de produire un devis au préalable, en déterminant le degré de difficulté de chaque fonctionnalité ou cas d'utilisation et on estime le coût et le délai de réalisation Jour/Homme(JC CORRE & M.Foyaul)

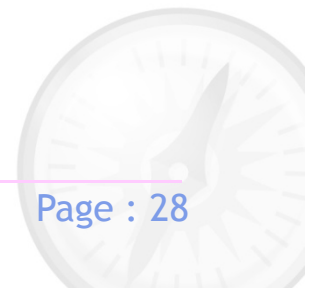




Exercices 1

Considérons une station-service de distribution d'essence. Le client se sert de l'essence de la façon suivante : il prend un pistolet accroché à une pompe et appuie sur la gâchette pour prendre de l'essence.

1. Qui est l'acteur du système ?
2. Proposer un petit diagramme de cas d'utilisation pour modéliser la situation.
3. Le pompiste, peut se servir de l'essence pour sa voiture dans sa station. Pour modéliser cette activité, Comment modélise-t-on ça ?
4. Lorsque le pompiste vient avec son camion citerne pour remplir les réservoirs des pompes, est-il considéré comme un nouvel acteur ? Comment modélise-t-on cela ?
5. Certains pompistes sont aussi qualifiés pour opérer des opérations de maintenance en plus des opérations habituelles des pompistes telles que le remplissage des réservoirs. Ils sont donc réparateurs en plus d'être pompistes. Comment cela ?



Exercices 2



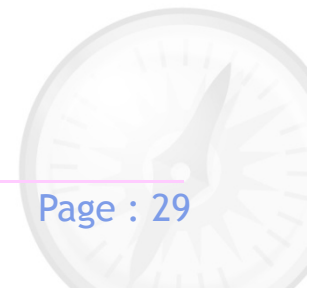
1°) Dans un établissement scolaire, on désire gérer la réservation des salles de cours ainsi que du matériel pédagogique (ordinateur portable ou/et Vidéo projecteur).

Seuls les enseignants sont habilités à effectuer des réservations (sous réserve de Disponibilité de la salle ou du matériel).

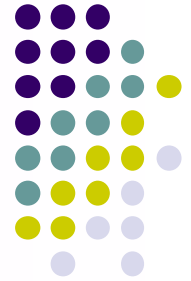
Le planning des salles peut quant à lui être consulté par tout le monde (enseignants et étudiants).

Par contre, le récapitulatif horaire par enseignant (calculé à partir du planning des salles) ne peut être consulté que par les enseignants.

Enfin, il existe pour chaque formation un enseignant responsable qui seul peut éditer Le récapitulatif horaire pour l'ensemble de la formation.



Exercices 3



Dans un magasin, le processus de vente est le suivant : le client entre, passe dans Les rayons, demande éventuellement des renseignements ou procède à des essais, prend des articles (si le stock est suffisant), passe à la caisse où il règle ses achats (avec tout moyen de paiement accepté). Il peut éventuellement bénéficier d'une réduction. *Modéliser cette situation par un diagramme de cas d'utilisation*



Exercices 4



3°) On considère le système suivant de gestion d'un GAB (Guichet automatique de billets) :

- le distributeur délivre de l'argent à tout porteur de carte (carte Visa ou carte de la banque)

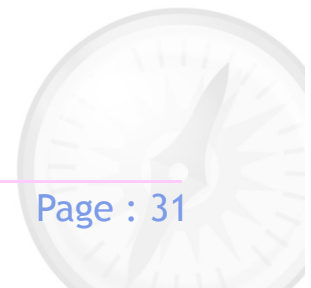
- pour les clients de la banque, il permet :

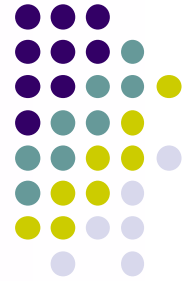
- ☐ la consultation du solde du compte

- ☐ le dépôt d'argent (chèque ou numéraire)

- toute transaction est sécurisée et nécessite par conséquent une authentification

- dans le cas où une carte est avalée par le distributeur, un opérateur de maintenance se charge de la récupérer. C'est la même personne qui collecte également les dépôts d'argent et qui recharge le distributeur.





Analyse



Ph 2 : Analyse

Diagramme de séquences Système «Boite Noire»



Caractéristiques

- ❖ Met en évidence l'aspect temporel (haut vers le bas) des interactions entre les acteurs et le Système
- ❖ Un objet a une ligne de vie représentée par une ligne verticale pointillée.
- ❖ Une flèche reçue par un objet se traduit par l'exécution d'une opération ou d'une action.
- ❖ La durée de vie de l'opération est symbolisée par un rectangle.
- ❖ Certains objets vivent pendant tout le diagramme, d'autres sont activés et/ou meurent pendant la séquence.
- ❖ Un diagramme de séquence doit être enrichi par des notes faisant référence aux enchaînements alternatifs et d'erreurs du scénario nominal.

Diagramme de séquence « Boite noire »

Diagramme de séquence Système est dit aussi un diagramme de séquence boite noire (Système = Boite Noire), il décrit un scénario d'un cas d'étude. Un Diagramme de Séquence « Boite noire » peut être enrichi par des actions internes (Souvent des vérifications) des notes de renvoi aux enchaînements alternatifs et d'erreurs

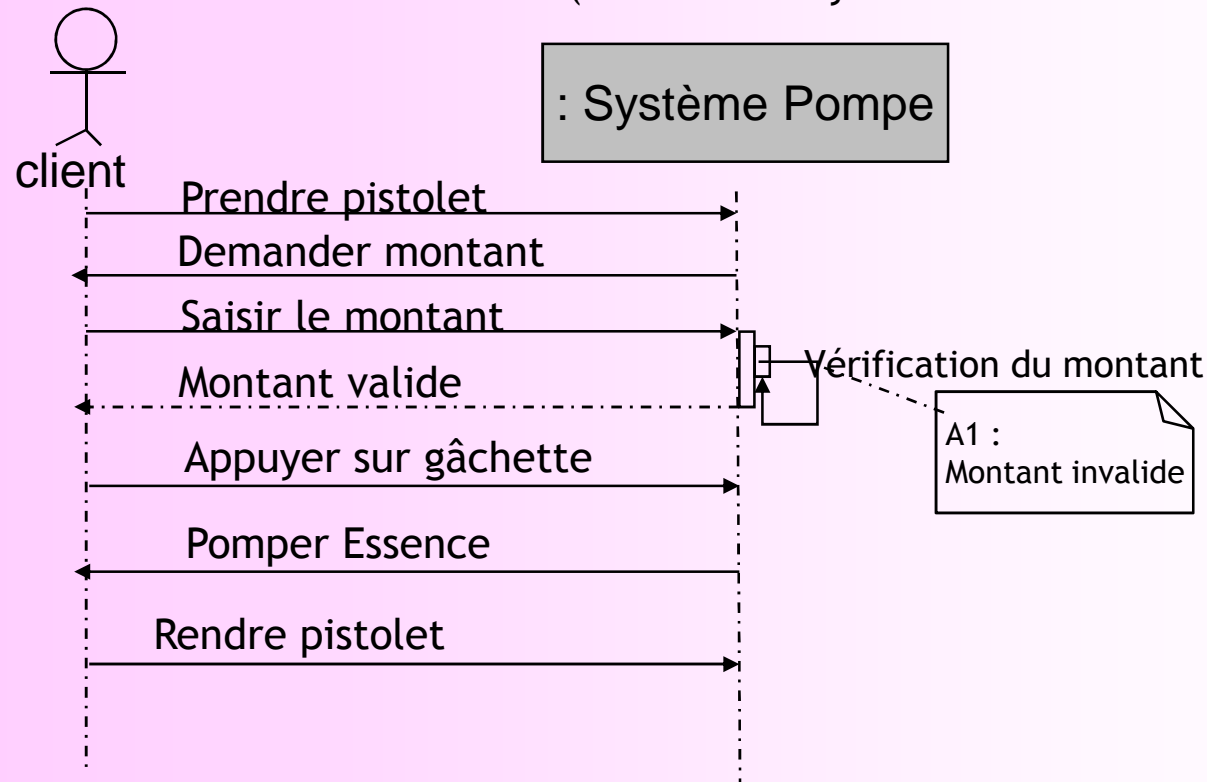


Exemple

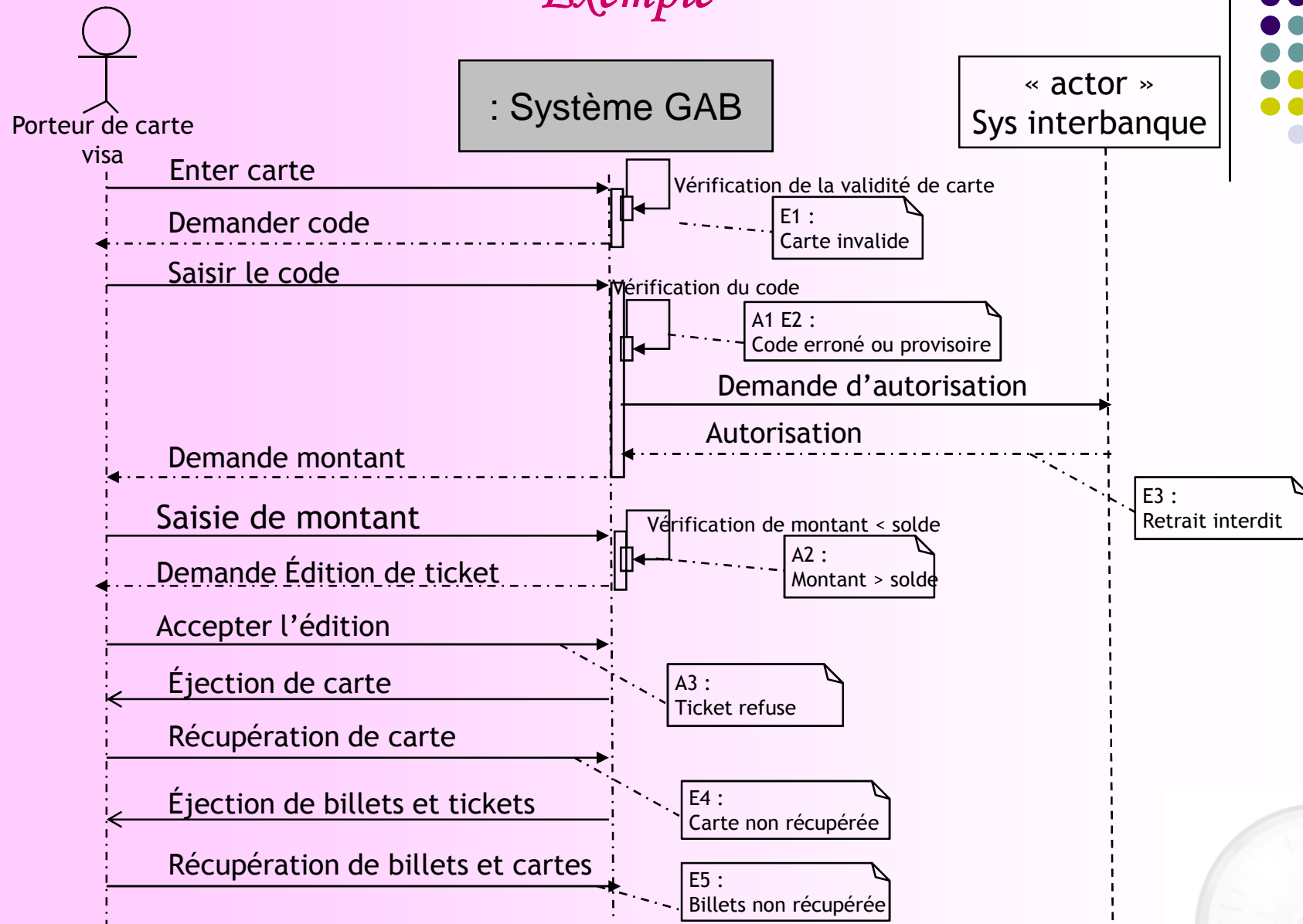
Suite aux descriptions textuelles, le scénario(nominal ou secondaire) peut être représenté en utilisant un diagramme de séquences.

Le diagramme de séquences permet :

- . de visualiser l'aspect temporel des interactions
- . de connaître le sens des interactions (acteur vers système ou contraire)



Exemple

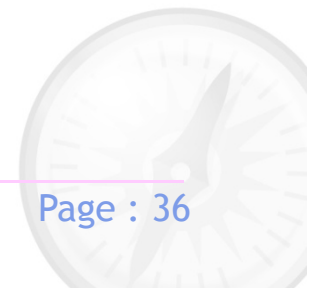
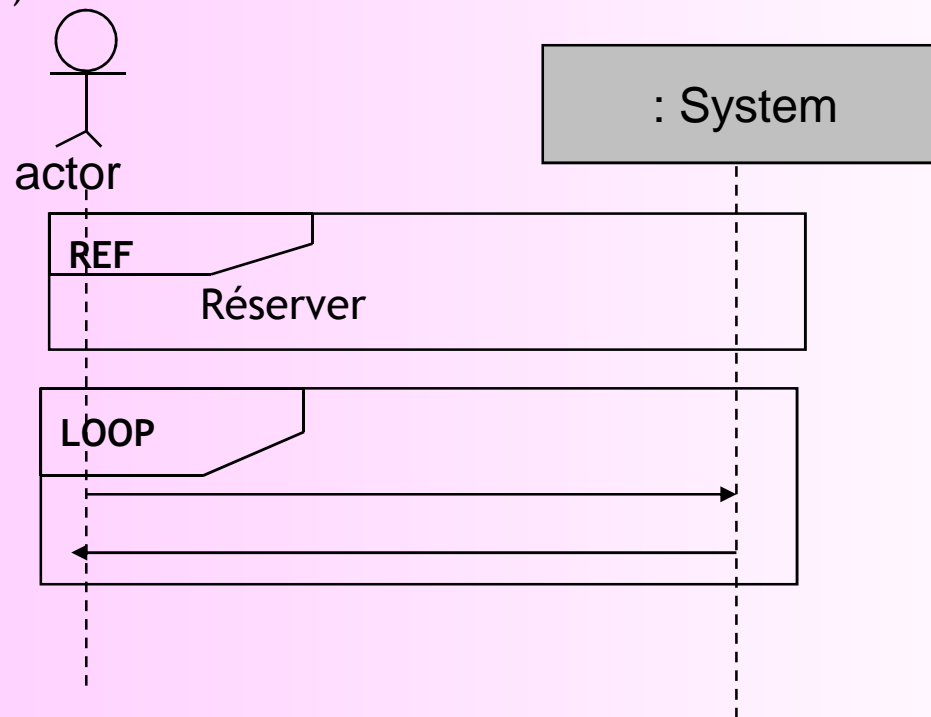


Particularités de UML 2

Cadres d'interactions



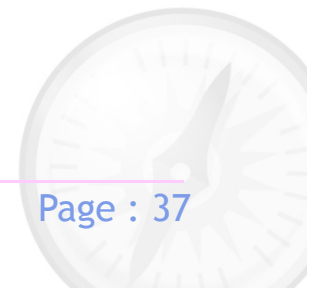
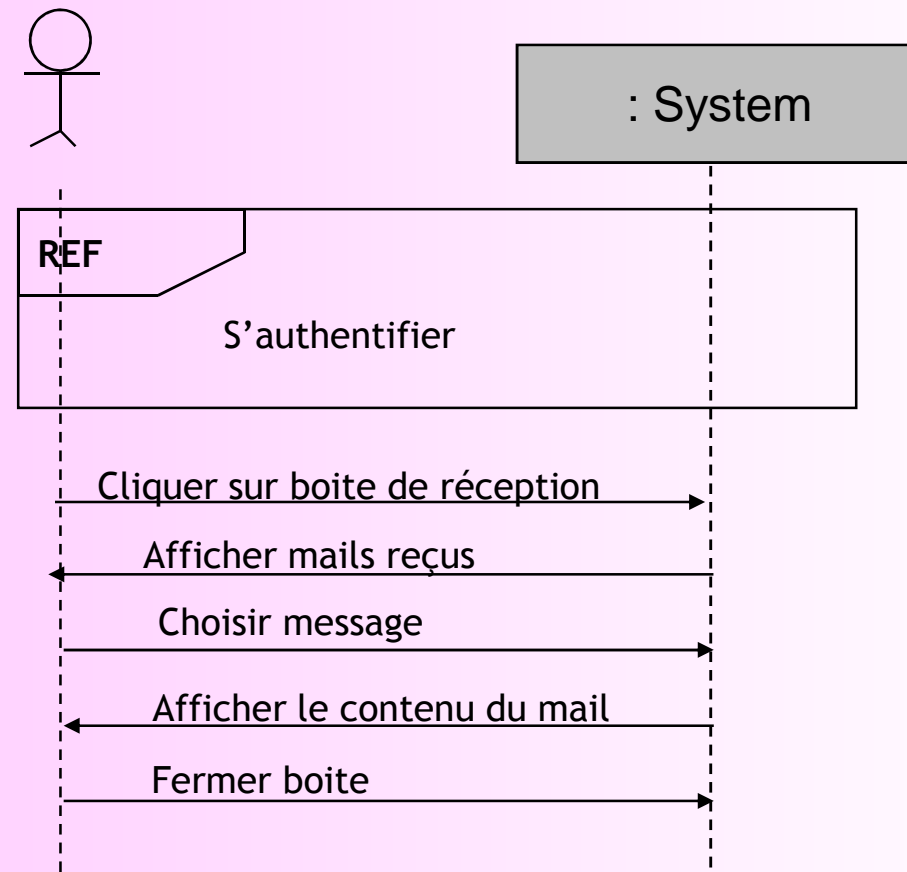
- En UML 2, pour un cas d'utilisation fait référence à autre, qui permet d'alléger un diagramme de séquence
- Des actions d'un cas d'utilisation peuvent se répéter (loop), être optionnel (opt) , ou alternatifs (alt).



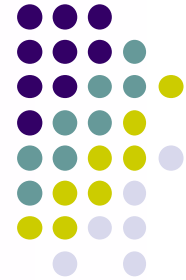
Exemples de cadres d'interactions



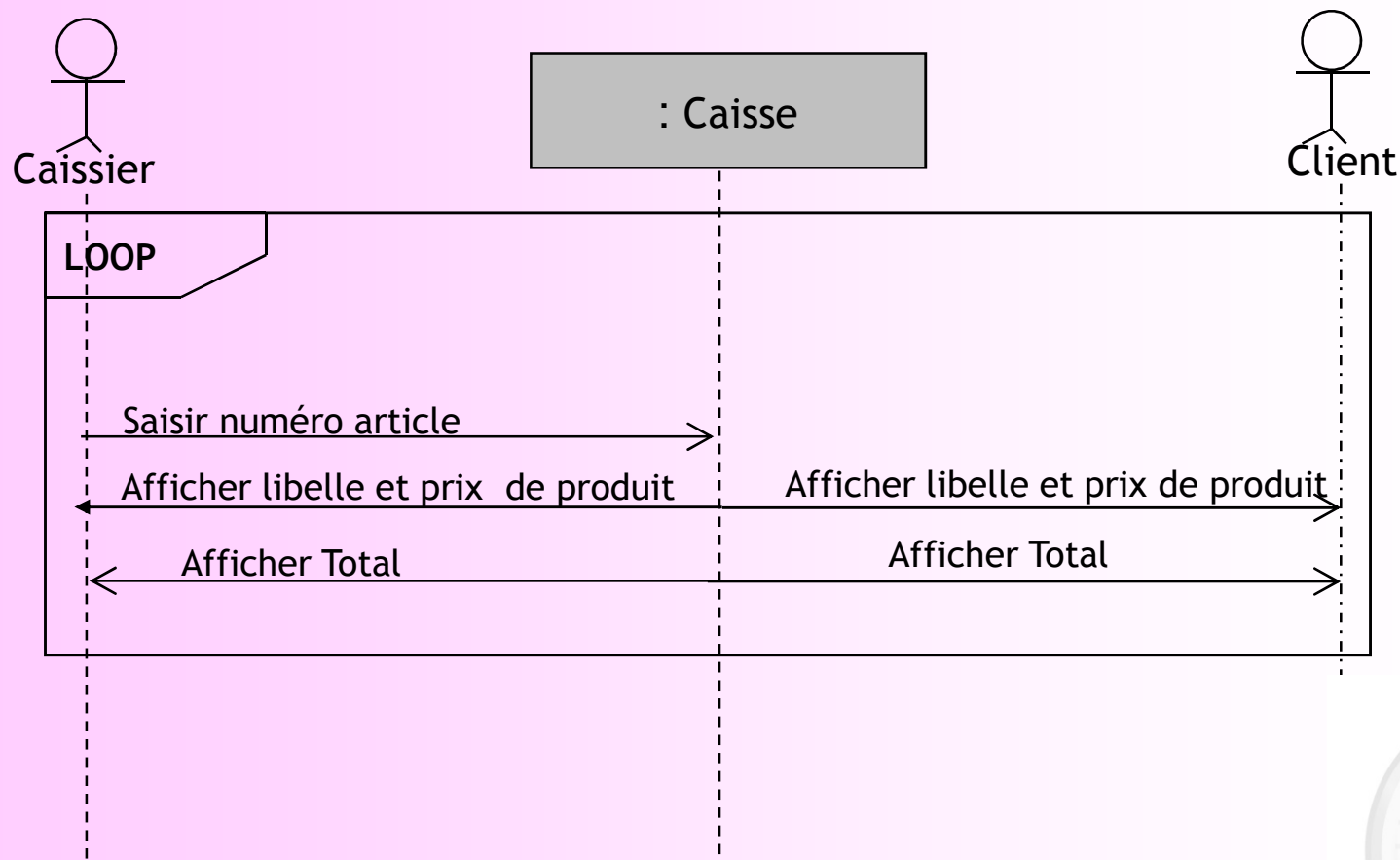
Un internaute veut consulter sa boîte a emails, pour se faire il a besoin de s'authentifier. en UML2 on peut faire appel a ce cas d'utilisation par sa référence voir exemple



EXEMPLE CAISSE

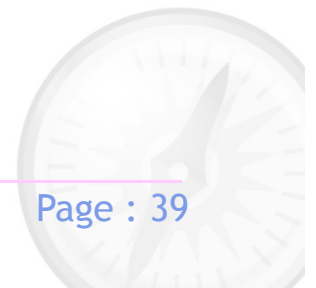


Un client se présente a la caisse pour payer les articles qui les a acheté. Le caissier saisie le num_article et sa quantité, le Système visualise le libelle et le prix produit saisi, ainsi que la total net a payer a la fin

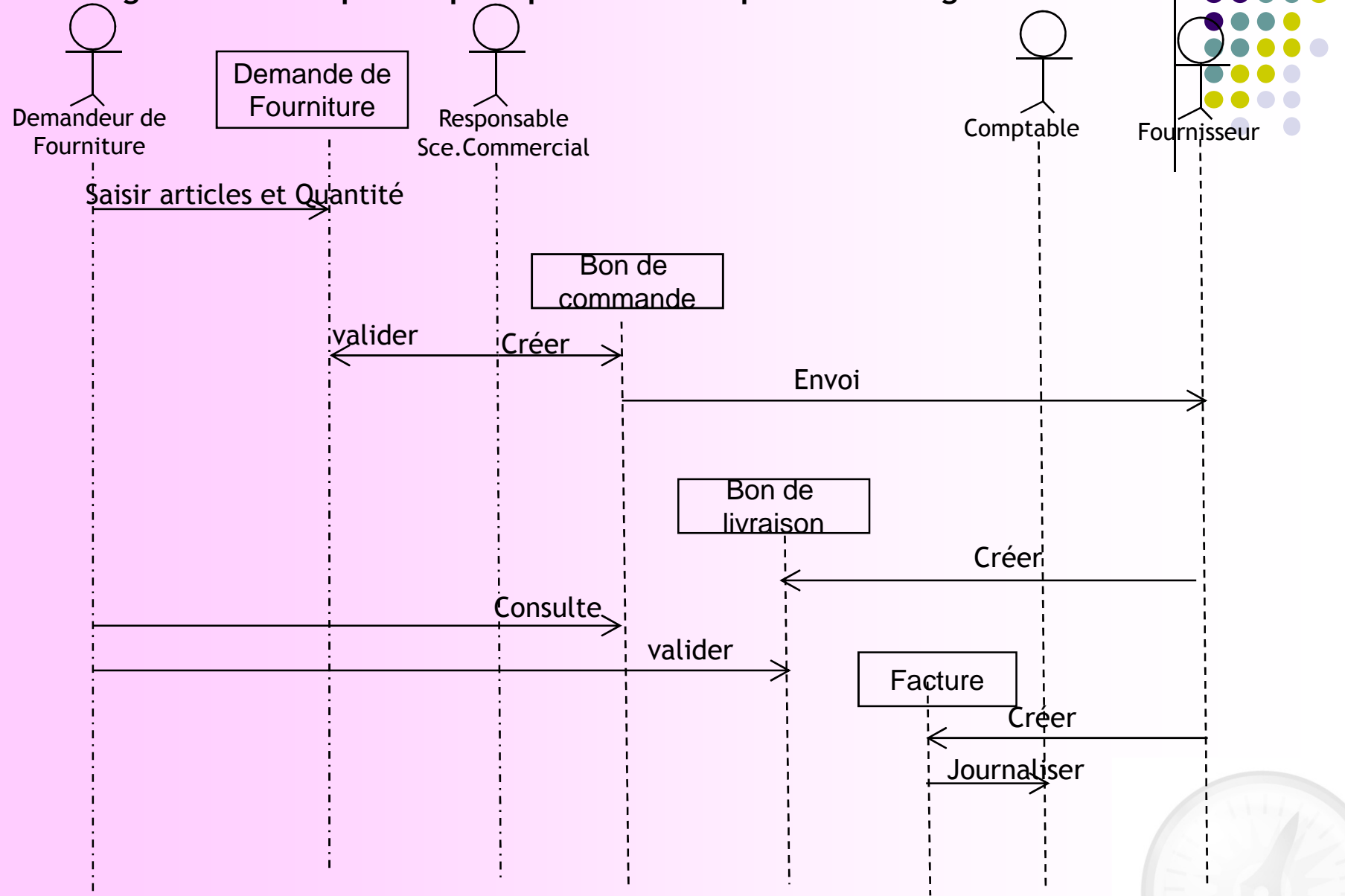


Exercices

Réaliser les diagrammes de séquence concernant Établissement scolaire et le magasin



Un diagramme de séquence peut présenter un processus de gestion



Ph 2 : Analyse

Diagramme d'activité

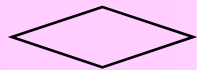
Il est opportun de construire un diagramme d'activité, si les diagrammes de séquence sont très nombreux et trop chargés.

Il décrit la dynamique d'un cas d'utilisation. En présentant les actions système. ils viennent illustrer et consolider la description textuelle des cas d'utilisation

Composants graphique :



Début Système



Condition



Action du système



Fork ou Join (Séparer ou joindre)



Fin Système (Nominale ou Échec Système)

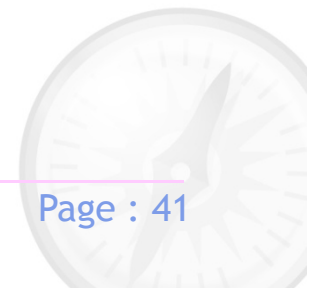
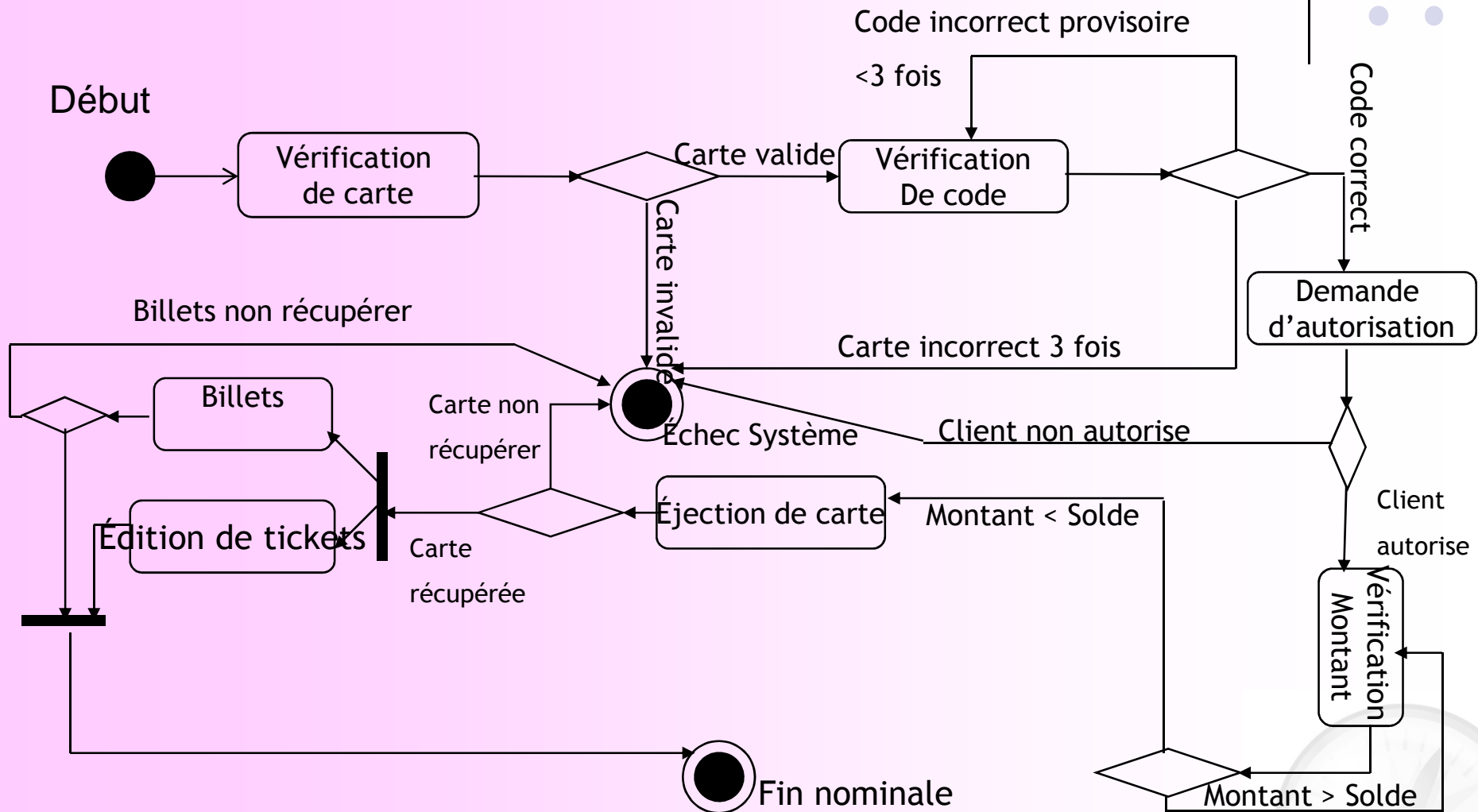


Diagramme d'activité

Exercice GAB

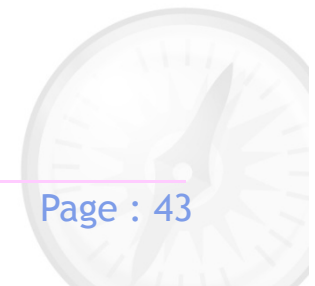
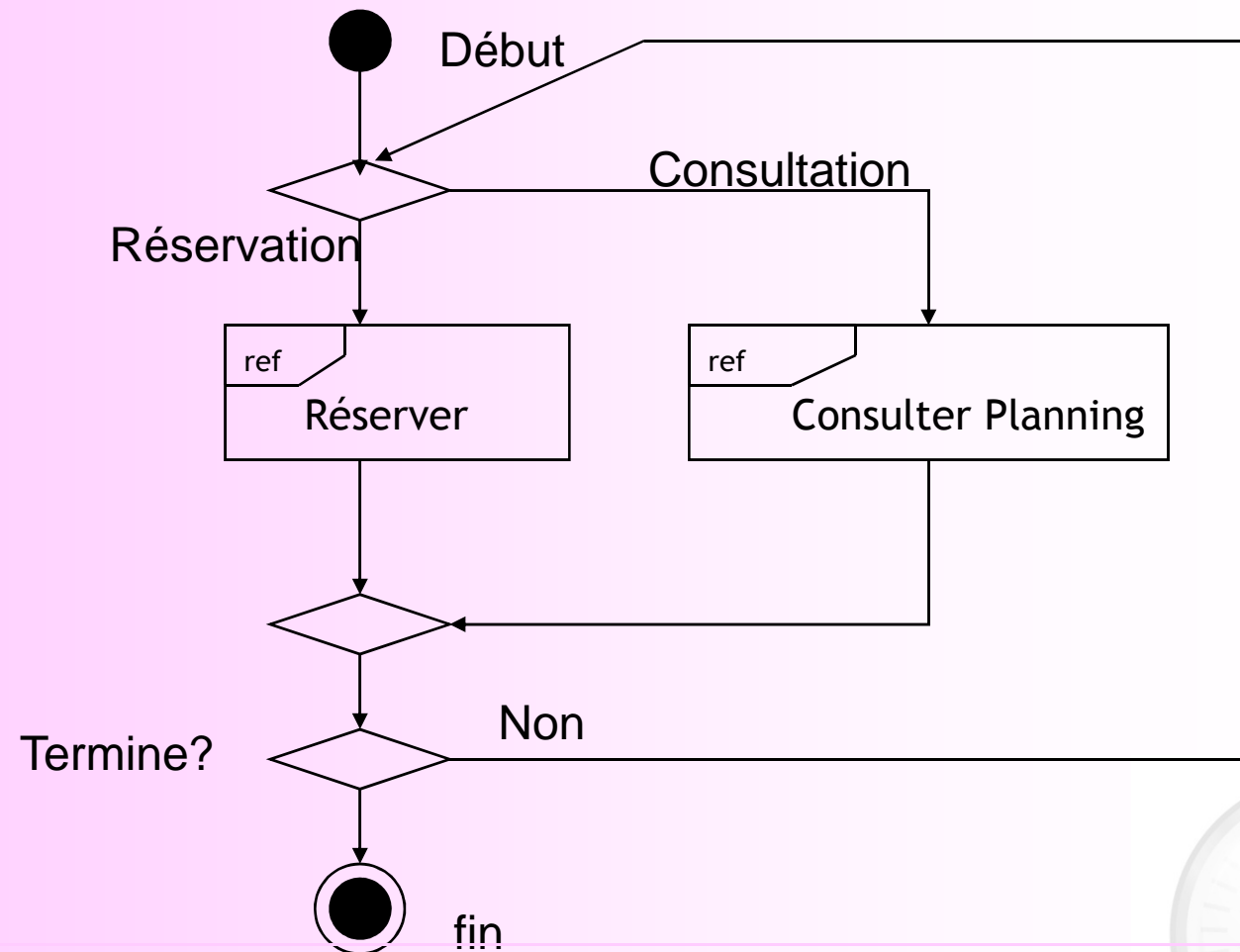


Vue Global d'interaction

Exercice Gestion de réservation

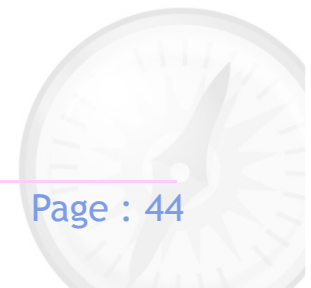
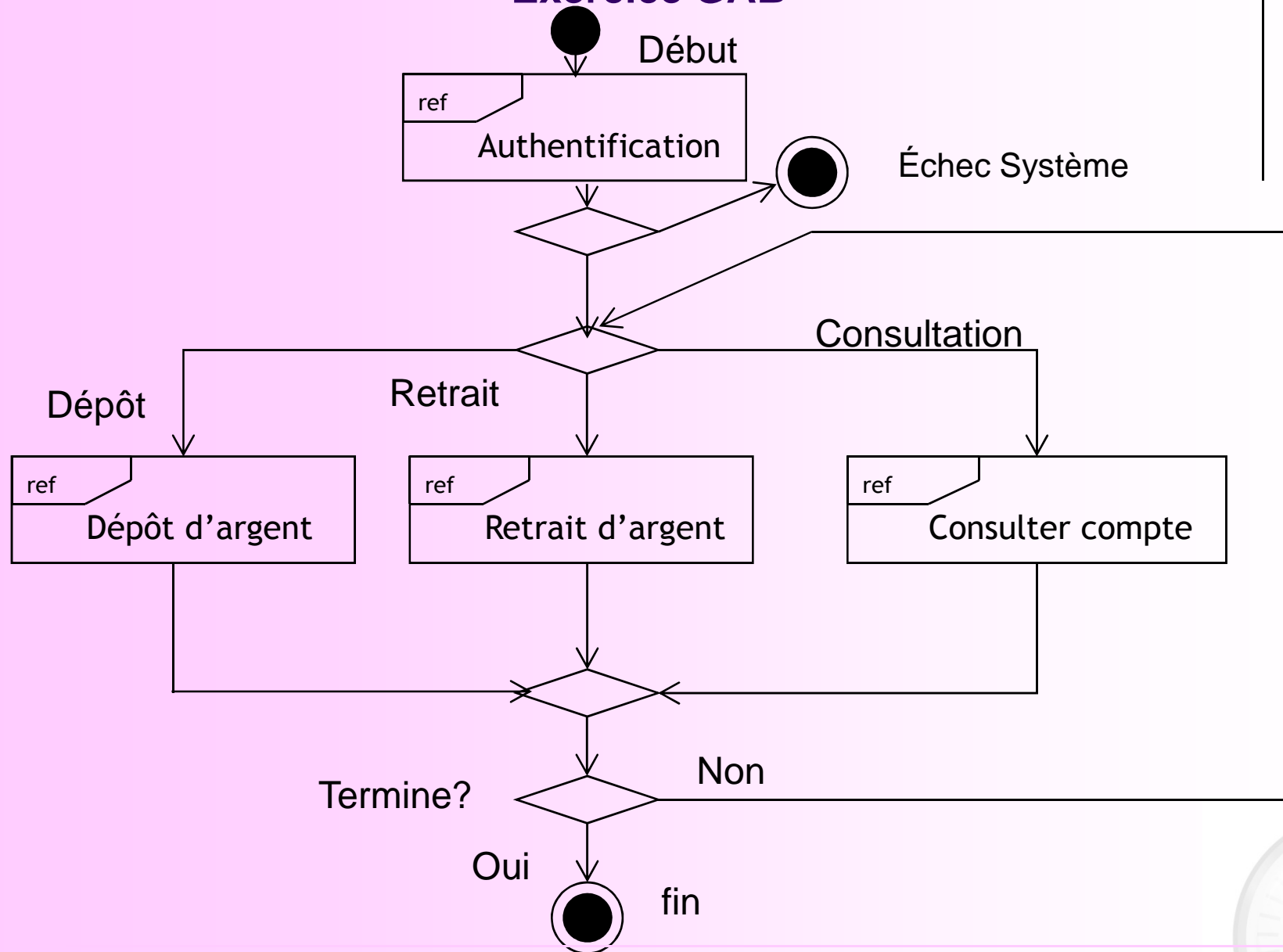


Un Diagramme d'interaction vue global permet de représenter l'ensemble des transactions du système afin de donner une vue globale des fonctionnalités du Système. Ce diagramme regroupe les diagramme de séquence et d'activité.



Vue Global d'interaction

Exercice GAB



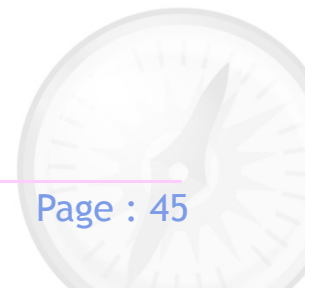
Introduction aux diagramme de classes et séquence « boîte blanche»



Introduction :

Un Être Humain peut être vu comme Système « boîte noire ». Tout système peut être représenté par ses composants statiques. Ainsi, le système « corps humain », peut il être représenté par sa composition en termes de bras, visages, jambes, yeux, etc. Cela donne une certaine compréhension du système. Si l'on ajoute à ces composants statiques, des comportements, par exemple, les yeux peuvent "s'ouvrir", " se fermer" ou encore "pleurer", on obtient alors une vision "objet" de notre système « Boîte blanche ».

Pour les systèmes logiciels, il en est de même. En U.M.L, le diagramme qui permet de décrire, spécifier, documenter ce type de connaissances s'appelle le diagramme de classes et le diagr séquence «boîte blanche ».



Analyse du domaine : (Analyse Statique)

modèle du domaine et classes d'analyse:



Analyse du domaine : modèle du domaine et classes d'analyse:

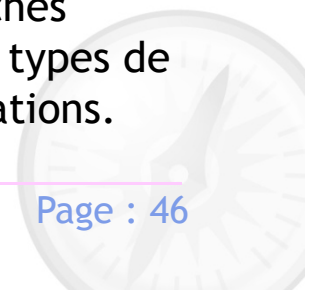
✓ L'élaboration du modèle des classes du domaine permet d'opérer une transition vers une véritable modélisation objet.

✓ La phase d'analyse du domaine permet d'élaborer la première version du diagramme De classes appelée modèle du domaine.

Ce modèle doit définir les classes qui Modélisent les entités ou concepts présents dans le domaine (on utilise aussi le terme de métier) de l'application. Il s'agit donc de produire un modèle des objets du monde réel dans un domaine donné.

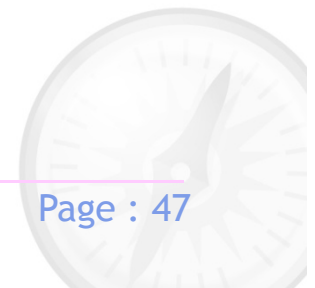
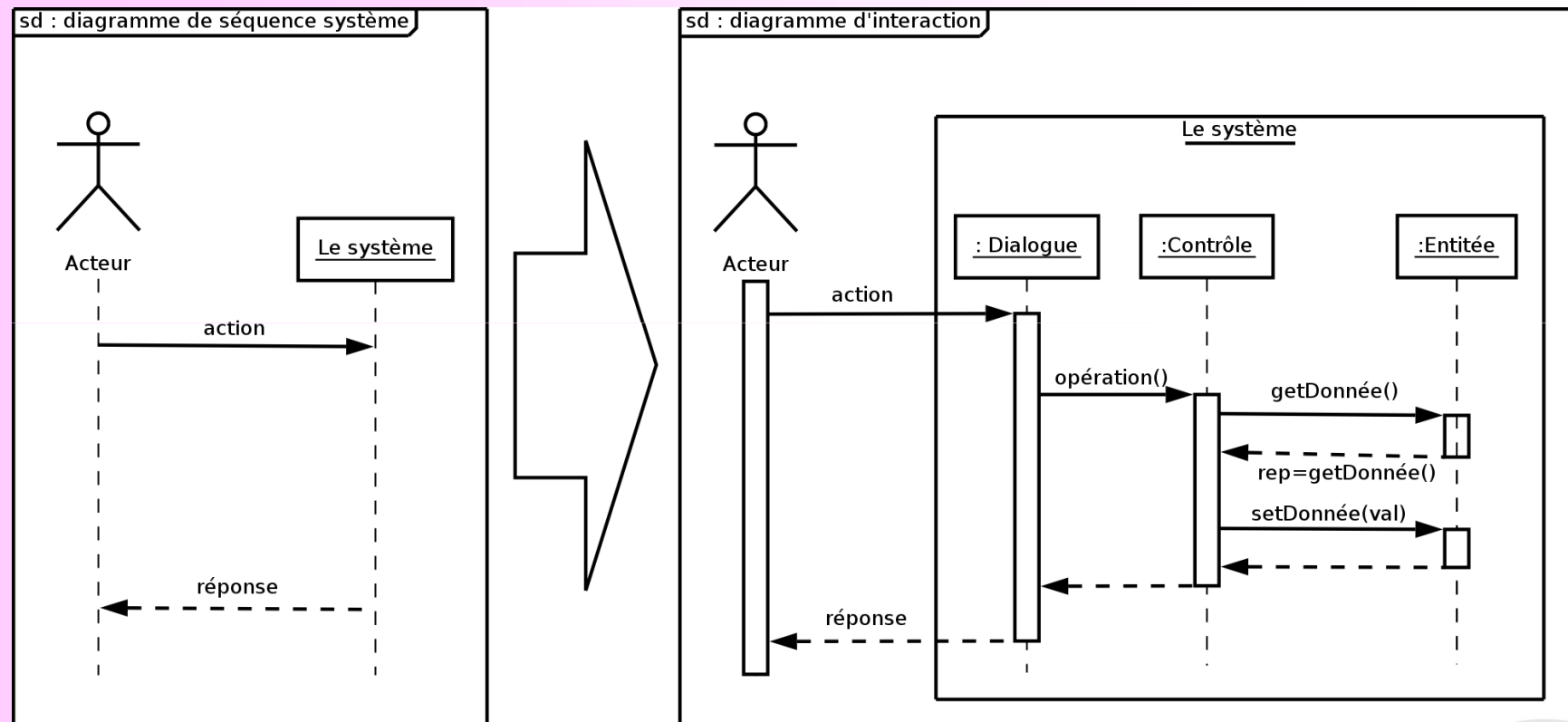
Ces entités ou concepts peuvent être identifiés directement à partir de la connaissance du domaine ou par des entretiens avec des experts du domaine. Il faut absolument utiliser le vocabulaire du métier pour nommer les classes et leurs attributs. Les classes du modèle du domaine ne doivent pas contenir d'opération, mais seulement des attributs.

✓ Il n'est pas souhaitable que les utilisateurs interagissent directement avec les instances des classes du domaine par le biais de l'interface graphique. En effet, le modèle du domaine doit être indépendant des utilisateurs et de l'interface graphique. De même, l'interface graphique du logiciel doit pouvoir évoluer sans répercussion sur le coeur de l'application. C'est le principe fondamental du découpage en couches d'une application. Ainsi, le diagramme de Classes participantes modélise trois types de classes d'analyse, les dialogues, les contrôles et les entités ainsi que leurs relations.





Exemple :



Classes d'analyse



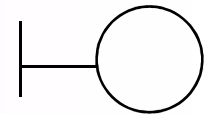
Définition de classes d'analyses

On s'aperçoit que dans l'analyse d'un problème trois types de classes

Apparaissent couramment:

classe Dialogue

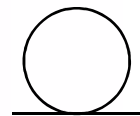
La classe qui permet au système de communiquer avec le monde réel, elle est à la frontière du système, elle se conçoit en général par une interface graphique, nous la représentons par l'icône suivante



Réception de facture

classe entité ou métier

La classe qui mémorise et gère des données, par exemples les livres présents, les prêts effectués, nous la représentons par l'icône suivante



Elles permettent à des données et des relations d'être stockées dans des fichiers ou des bases de données. Lors de l'implémentation, ces classes peuvent ne pas se concrétiser par des classes mais par des relations, au sens des bases de données relationnelles

Stockage de facture

classe contrôle

La classe qui réalise le contrôle nécessaire pour interpréter le scénario
Décrivant un cas d'utilisation



Gestion de facture

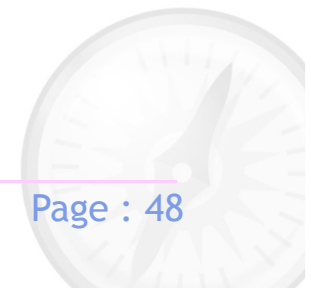


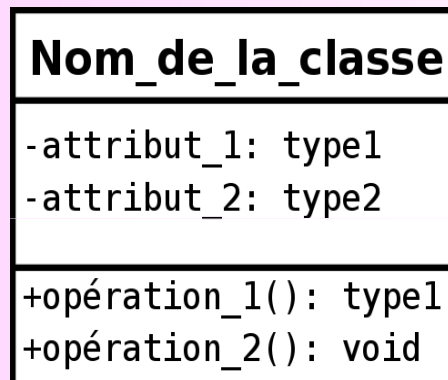
Diagramme de classes métier



Le diagramme de classes permet de modéliser les classes du système et leurs associations.

le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisation. Il s'agit d'une vue statique car on ne tient pas compte du facteur temporel dans le Comportement du système.

Représentation graphique :



Représentation graphique de l'encapsulation :

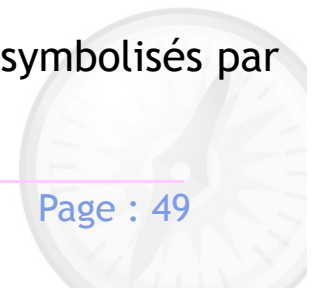
+ public

- private

protected

/ Attributs dérivés

Les attributs dérivés peuvent être calculés à partir d'autres attributs. Ils sont symbolisés par l'ajout d'un « / » devant leur nom.



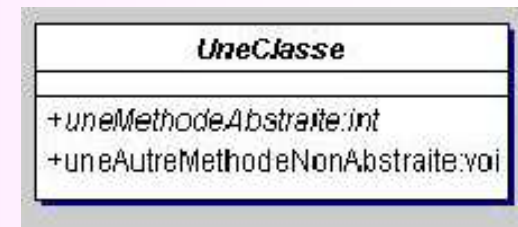
Classes abstraites et Interfaces



Classes abstraites

Une classe abstraite ne peut donc pas être utilisée pour fabriquer des instances d'objets; elle sert uniquement de modèle, que l'on pourra utiliser pour créer des classes plus Spécialisées par dérivation (héritage). Une classe abstraite est assez proche de la notion d'interface; d'ailleurs, la notion d'interface est généralement implémentée par une classe.

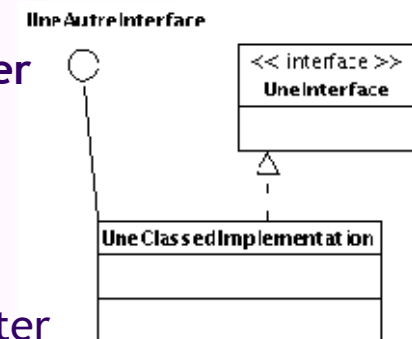
Représentation d'une classe abstraite



Interface

- Un modèle ou prototype de classes, qui définit un contrat a respecter
- Descripteur des opérations
- Sans code
- Pas d'attribut
- Pas d'association

Une classe peut implémenter une interface; elle peut aussi en implémenter plusieurs. En notation UML, cette relation est dénotée par une flèche en pointilles



Associations



Association :

Relation structurelle entre deux classes d'objets

- Relie deux classificateurs

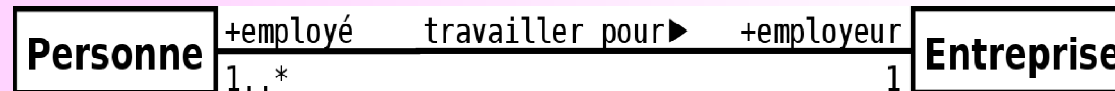
Classes, interfaces

Parfois : association représentée par une classe

Elles représentent soit une appartenance ou une collaboration.

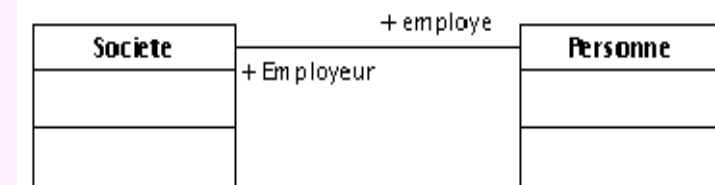
Une association binaire est matérialisée par un trait plein entre les classes associées. Elle peut être ornée d'un nom, avec éventuellement une précision du sens de lecture

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



Rôles

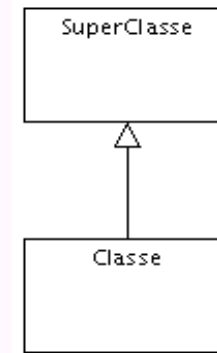
- Extrémité d'une association
- Indication des rôles relatifs des deux classes reliées par association
- Pseudo-attribut de la classe source
- Ex : Employeur est un pseudo attribut de la classe Personne
- Indication de visibilité; Public par défaut , Privé (-) ou protégé (#)



Types d'associations

Héritage

L'héritage constitue une relation de spécialisation. Elle est notée, en UML, par une Flèche allant de la classe spécialisée vers la classe originale (de la classe vers la superclasse).



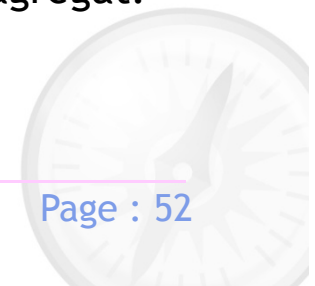
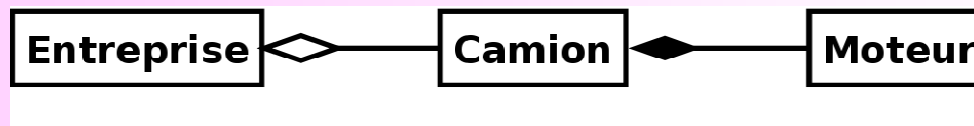
La relation d'agrégation/Composition

Agrégation

Lorsqu'un objet en contient d'autres, on parle d'agrégation. Une agrégation est une association qui représente une relation d'inclusion structurelle ou comportementale d'un élément dans un ensemble. Graphiquement, on ajoute un losange vide du côté de l'agregat, comme indiqué à la figure. Elle n'entraîne pas non plus de contrainte sur la durée de vie des parties par rapport au tout.

Composition

La composition, également appelée agrégation forte, décrit une contenance structurelle entre instances. Ainsi, la destruction de l'objet composite implique la destruction de ses composants. Une instance de la partie appartient toujours à au plus une instance de l'élément composite : la multiplicité du côté composite ne doit pas être supérieure à 1. Graphiquement, on ajoute un losange plein du côté de l'agregat.

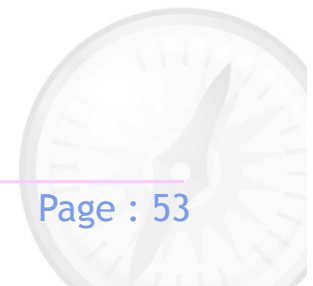
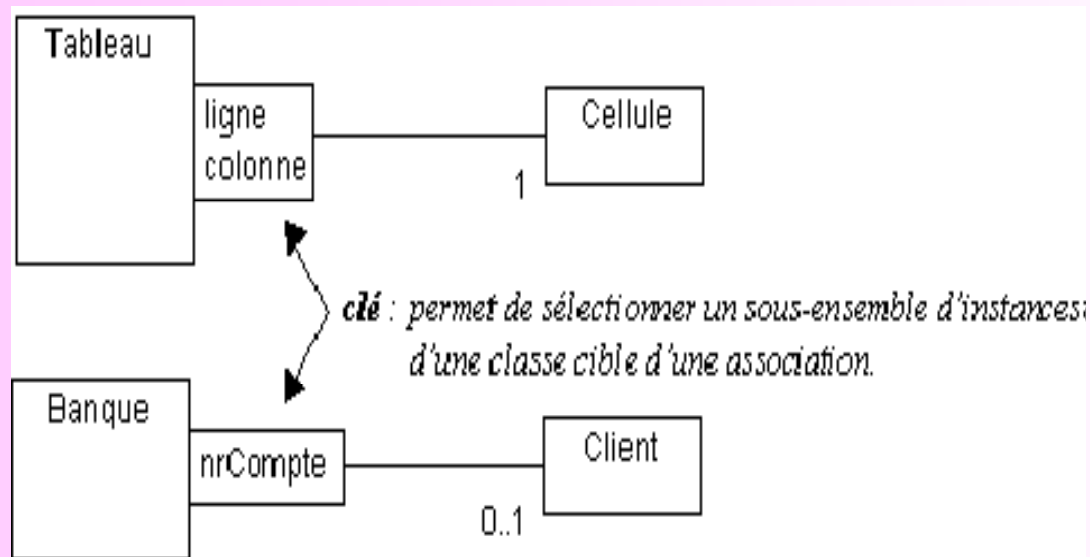


Qualification



Parfois, un élément (un attribut) permet de sélectionner un sous-ensemble d'objets (de 1 à n) participant à l'association. Souvent, cela permet de réduire la cardinalité de l'extrémité de l'association à 1... Il pourra s'agir d'une clé dans une HashTable.

L'attribut qualificatif est la clé du HashTable



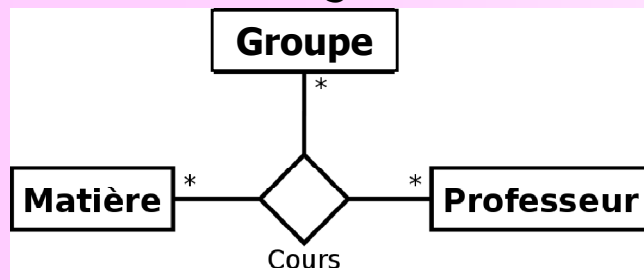
Association n-aire & classe association



Associations n-aire

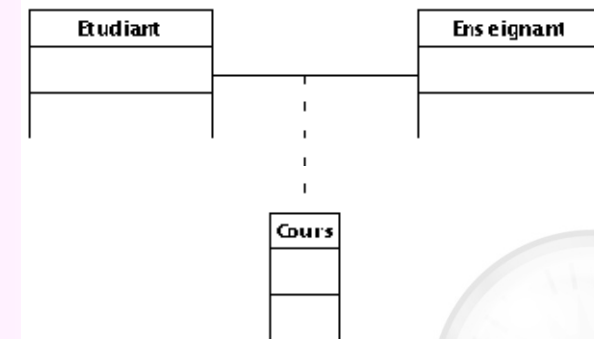
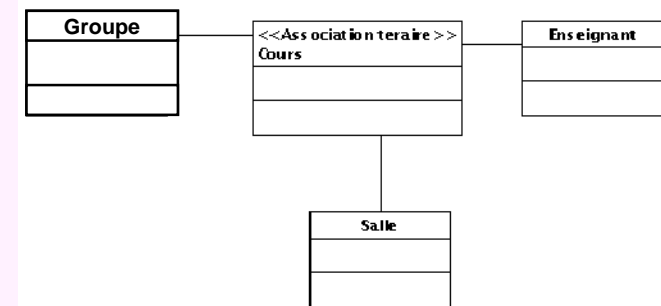
Une association n-aire lie plus de deux classes. La ligne pointillée d'une classe-association peut être reliée au losange par une ligne discontinue pour représenter une association n-aire dotée d'attributs, d'opérations ou d'associations.

On représente une association n-aire par un grand losange avec un chemin partant vers chaque classe participante. Le nom de l'association, le cas échéant, apparaît à proximité du losange.



Association-Classe

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



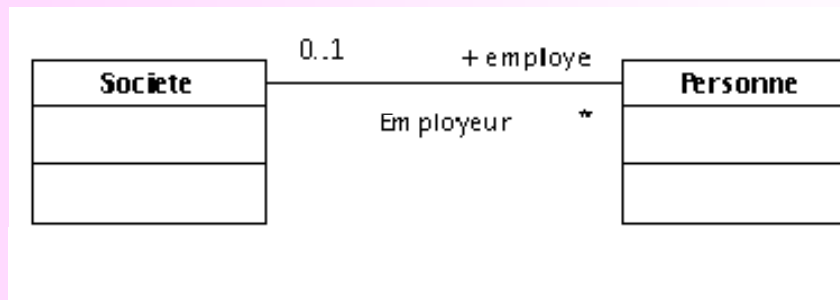
Multiplicité



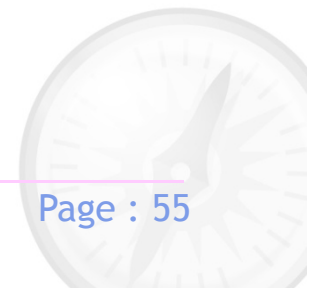
- Multiplicité

Contraintes liées au domaine d'application, elle est Valable pendant toute la vie de l'objet. Pas d'influence sur l'ordre de création des objets (associations simples)

Il faut noter que, pour les habitués du modèle entité/relation, les multiplicités sont en UML« à l'envers »



1	Un seul
0..1	Zéro ou un
N	(entier naturel) exactement N
M..N	De M à N(entiers naturels)
*	De zéro à plusieurs
0..*	De zéro à plusieurs
1..*	D'un à plusieurs



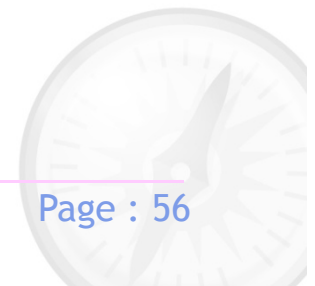
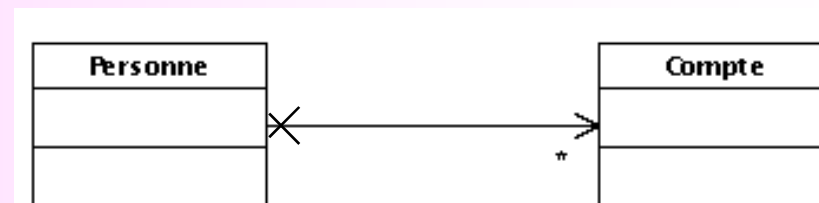


Navigabilité

Navigabilité

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

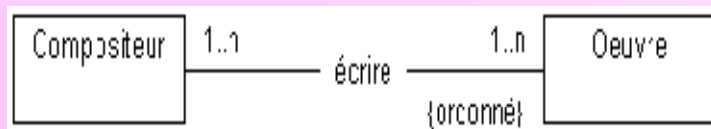
Par exemple, sur la figure, la terminaison du côté de la classe *Personne* n'est pas navigable : cela signifie que les instances de la classe *Compte* ne stockent pas de liste d'objets du type *Personne*. Inversement, la terminaison du côté de la classe *Compte* est navigable : chaque objet *Personne* contient une liste de *Comptes*.



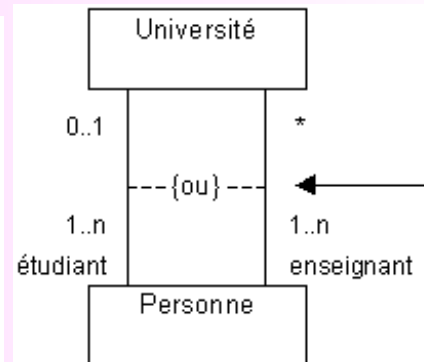
Contraintes sur les associations



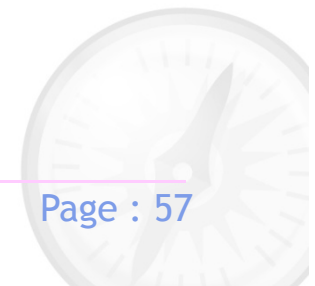
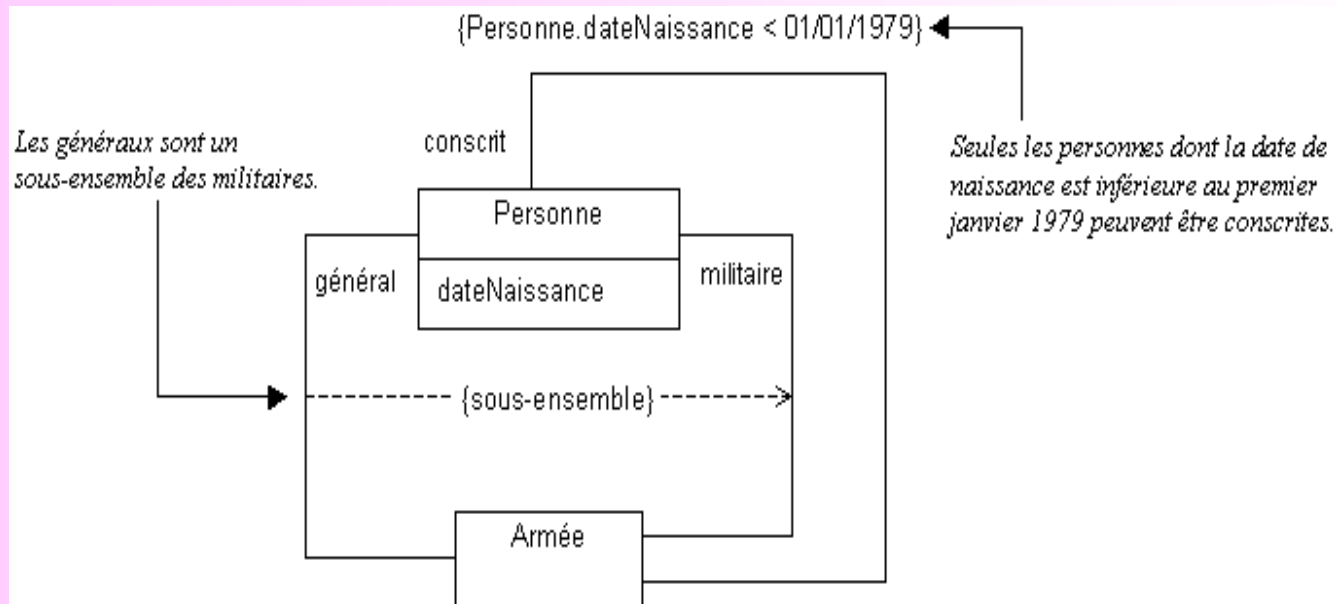
Il est possible d'exprimer des contraintes sur une association, afin de limiter les objets mis en jeu. Cela permet de mieux cadrer l'architecture de l'ensemble.



Ici, on met l'accent sur l'importance de l'ordre de la création des oeuvres d'un compositeur.



Ici, on indique qu'une personne joue soit le rôle d'étudiant, soit le rôle d'enseignant pour une université donnée (« ou » exclusif).



Analyse linguistique



Noms et groupes nominaux : Classe ou un travailleur métier

Article « le/la/un/une » : Multiplicité

Ce, ces, Cette et Synonymes : Classe identique

Possessifs (son/sa/ses): attribut ou une classe. Un attribut si la possession est une simple caractéristique du possesseur. Une Classe si la possession est un concept, le possesseur et la possession sont reliés automatiquement par une association structurelle.

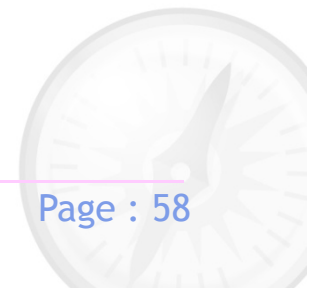
OU : Spécialisation/Généralisation

Verbe : association ou (une dynamique exclus de l'analyse du domaine)

Un verbe entres concepts : association plus n-aire ou association classe générale (neutre) avec une multiplicité plusieurs

Participe présent : association

Liste : Contrainte Ordred



Conclusion



identification des classes (d'objets) :

Recherchez les classes candidates (différentes suivant le niveau d'abstraction) à l'aide de diagrammes d'objets (ébauches).

filtrez les classes redondantes, trop spécifiques ou vagues.

identification des associations entre classes / interactions entre objets (instances) :

recherchez les connexions sémantiques et les relations d'utilisation, documentez les relations (nom, cardinalités, contraintes, rôles des classes).

identification des attributs et les opérations des classes :

recherchez les attributs dans les modèles dynamiques (recherchez les données qui caractérisent les états des objets),

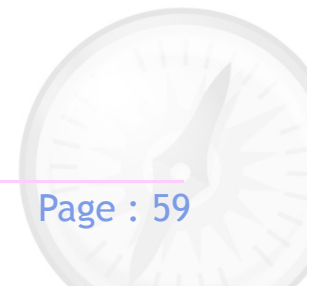
recherchez les opérations parmi les activités et actions des objets (ne pas rentrer dans le détail au niveau spécification).

Encapsuler les caractéristiques des classes

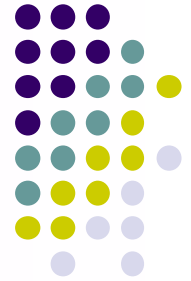
Optimisation les modèles :

choisissez vos critères d'optimisation (généricité, évolutivité, précision, lisibilité, simplicité...),

utilisez l'agrégation la généralisation et la spécialisation (classification),



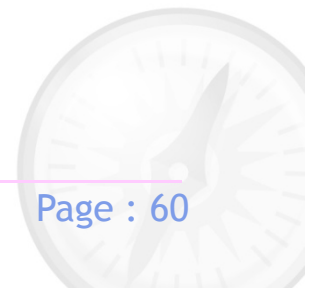
Exercices



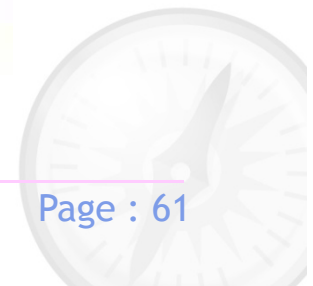
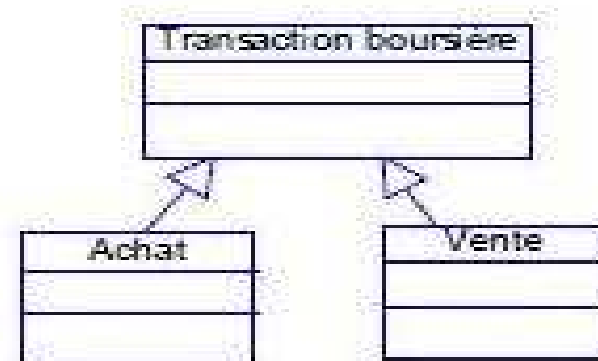
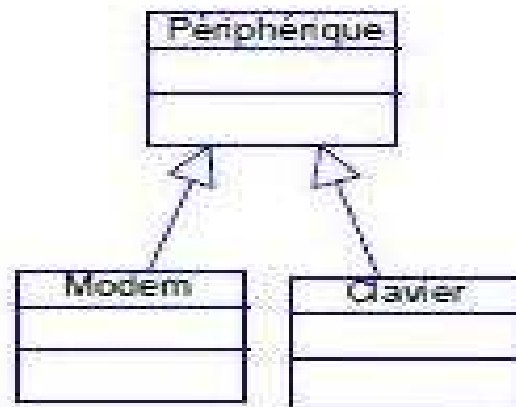
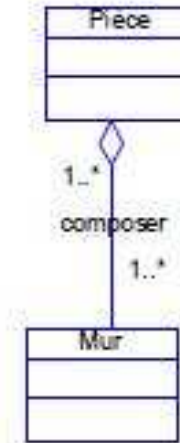
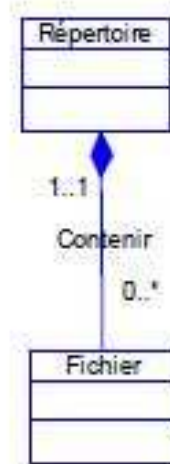
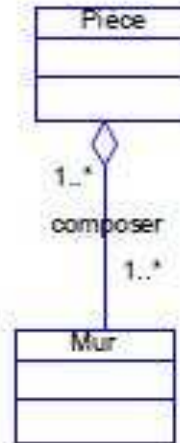
Soient les phrases suivantes :

- Un répertoire contient des fichiers, qui contiennent aussi des lignes;
- Une pièce contient des murs;
- Une droite est tracée par deux points;
- Un groupe d'étudiants est spécialisé réseau ou développement, il contient des stagiaires;
- Les voitures a gaz ou carburant démarrent différemment;
- Les clients identifiés par CIN et pompistes identifiés par matricule peuvent se servir de l'essence pour remplir leurs voiture;
- Les modems et claviers sont des périphériques d'entrée / sortie;
- Une transaction boursière est un achat ou une vente;
- Un compte bancaire peut appartenir à une personne physique ou morale;

Élaborez les diagrammes de classe correspondants en choisissant le type de relation approprié



Exercices –Correction-

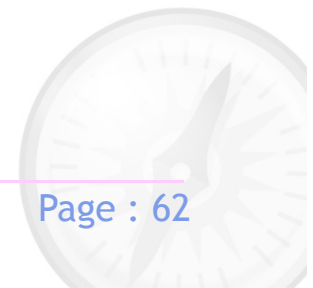


Exercices

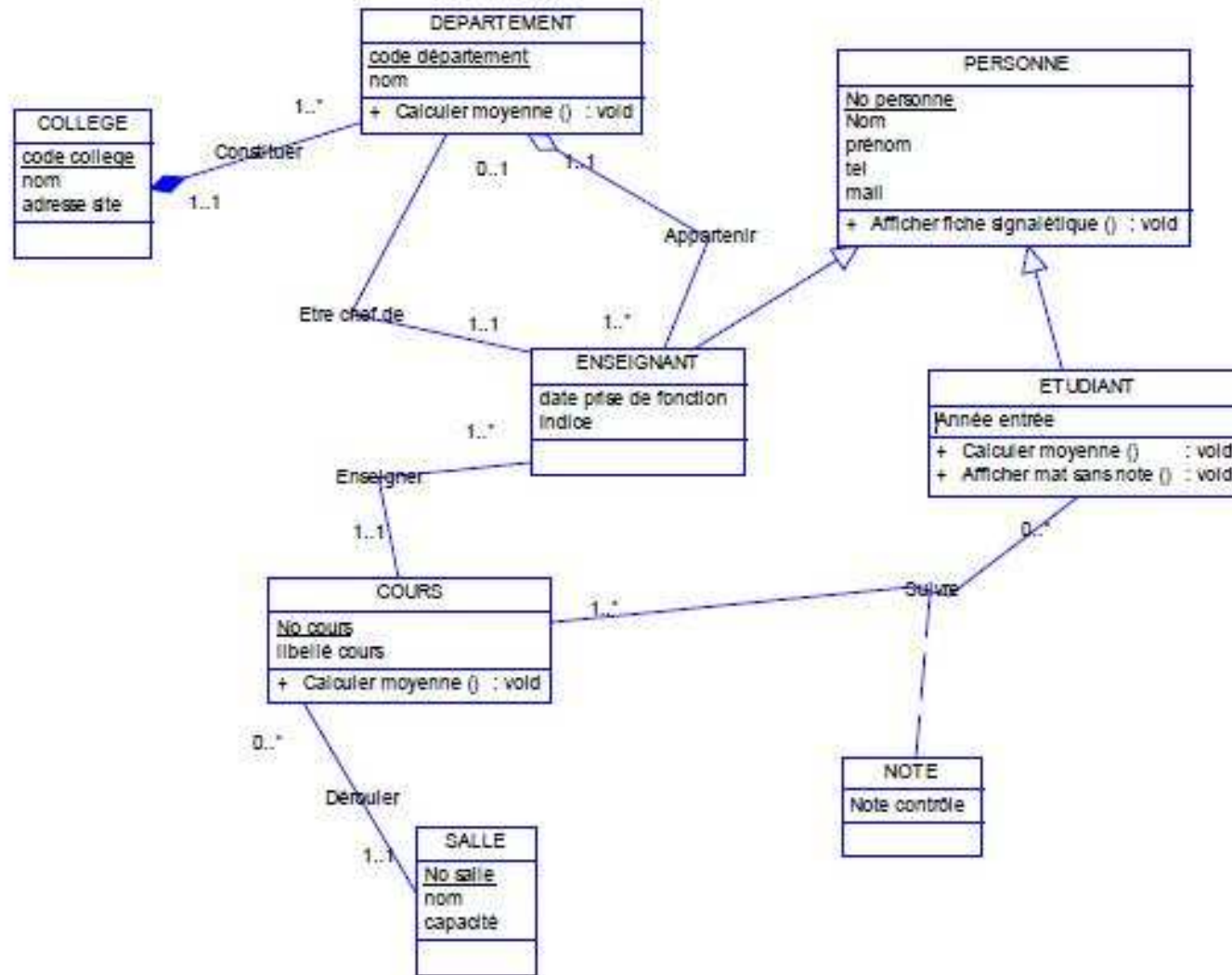


Une académie souhaite gérer les cours dispensés dans plusieurs collèges. Pour cela, on dispose des renseignements suivants :

- Chaque collège possède d'un site Internet
- Chaque collège est structuré en départements, qui regroupent chacun des enseignants spécifiques. Parmi ces enseignants, l'un d'eux est responsable du département.
- Un enseignant se définit par son nom, prénom, tél, mail, date de prise de fonction et son indice.
- Chaque enseignant ne dispense qu'une seule matière.
- Les étudiants suivent quant à eux plusieurs matières et reçoivent une note pour chacune d'elle.
- Pour chaque étudiant, on veut gérer son nom, prénom, tél, mail, ainsi que son année d'entrée au collège.
- Une matière peut être enseignée par plusieurs enseignants mais a toujours lieu dans la même salle de cours (chacune ayant un nombre de places déterminé).
- On désire pouvoir calculer la moyenne par matière ainsi que par département
- On veut également calculer la moyenne générale d'un élève et pouvoir afficher les matières dans lesquelles il n'a pas été noté
- Enfin, on doit pouvoir imprimer la fiche signalétique (, prénom, tél, mail) d'un enseignant ou d'un élève.



Correction -COLLEGE



Exercices

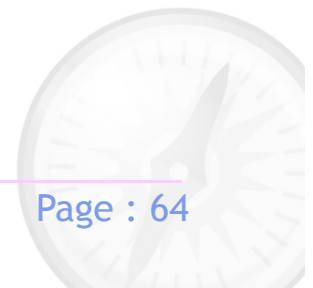


Une agence de location de maisons et d'appartements désire gérer sa liste de logements. Elle voudrait en effet connaître l'implantation de chaque logement (nom de La commune et du quartier) ainsi que les personnes qui les ont loué.

Le loyer dépend d'un logement, mais en fonction de son type (maison, Villa, Appartement...). Pour chaque logement, on veut disposer également de l'adresse, de la superficie.

Quant aux individus qui occupent les logements, on se contentera de leurs noms, prénoms, date de naissance et numéro de téléphone.

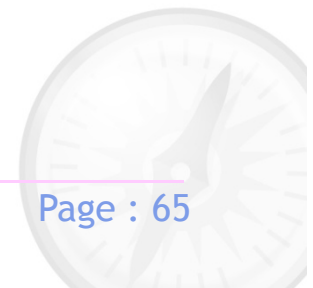
Pour chaque commune, on désire connaître le nombre d'habitants ainsi que la distance séparant la commune de l'agence.



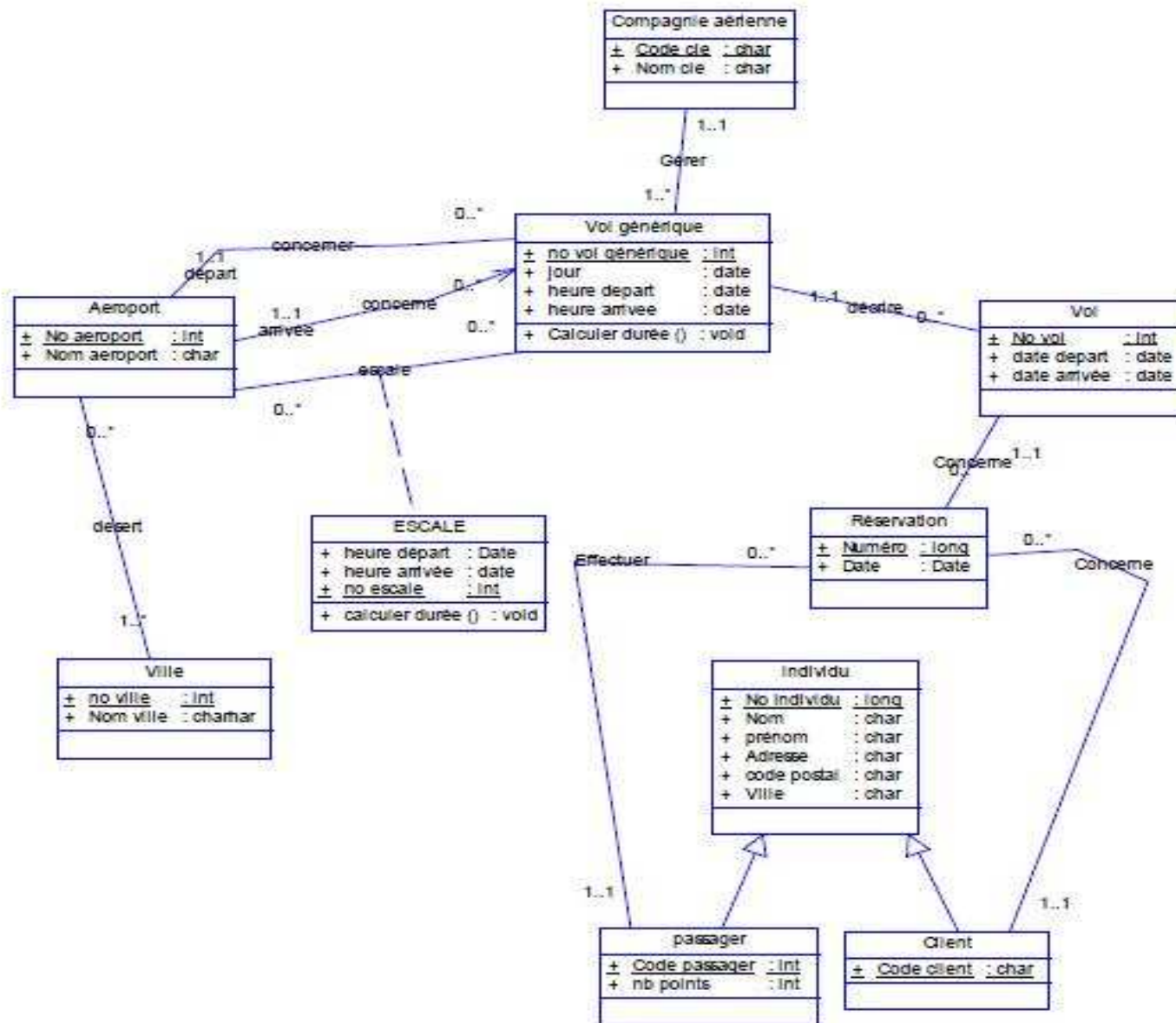
Exercices –Réservation de vols



- On souhaite gérer les réservations de vols effectués dans une agence. D'après les
- interviews réalisées avec les membres de l'agence, on sait que :
- · Les compagnies aériennes proposent différents vols
- · Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
- · Un client peut réserver un ou plusieurs vols, pour des passagers différents
- · Une réservation concerne un seul vol et un seul passager
- · Une réservation peut être confirmée ou annulée
- · Un vol a un aéroport de départ et un aéroport d'arrivée
- · Un vol a un jour et une heure de départ, et un jour et une heure d'arrivée
- · Un vol peut comporter des escales dans un ou plusieurs aéroport(s)
- · Une escale a une heure de départ et une heure d'arrivée
- · Chaque aéroport dessert une ou plusieurs villes



Correction – Réservation de vols



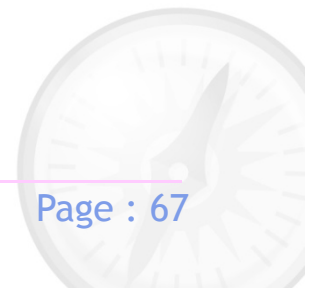
Exercices



Une école désire gérer les stages de ses étudiants à divers entreprises. Chaque étudiant est encadré par un Formateur de l'école. Dans chaque stage, l'étudiant doit réaliser une application de gestion. Les jury accordent toujours une note de stage, qui permet d'établir le classement du stagiaire. On désire connaître les stages auxquels ont participé les stagiaires, le sujet de stage, la note et le classement.

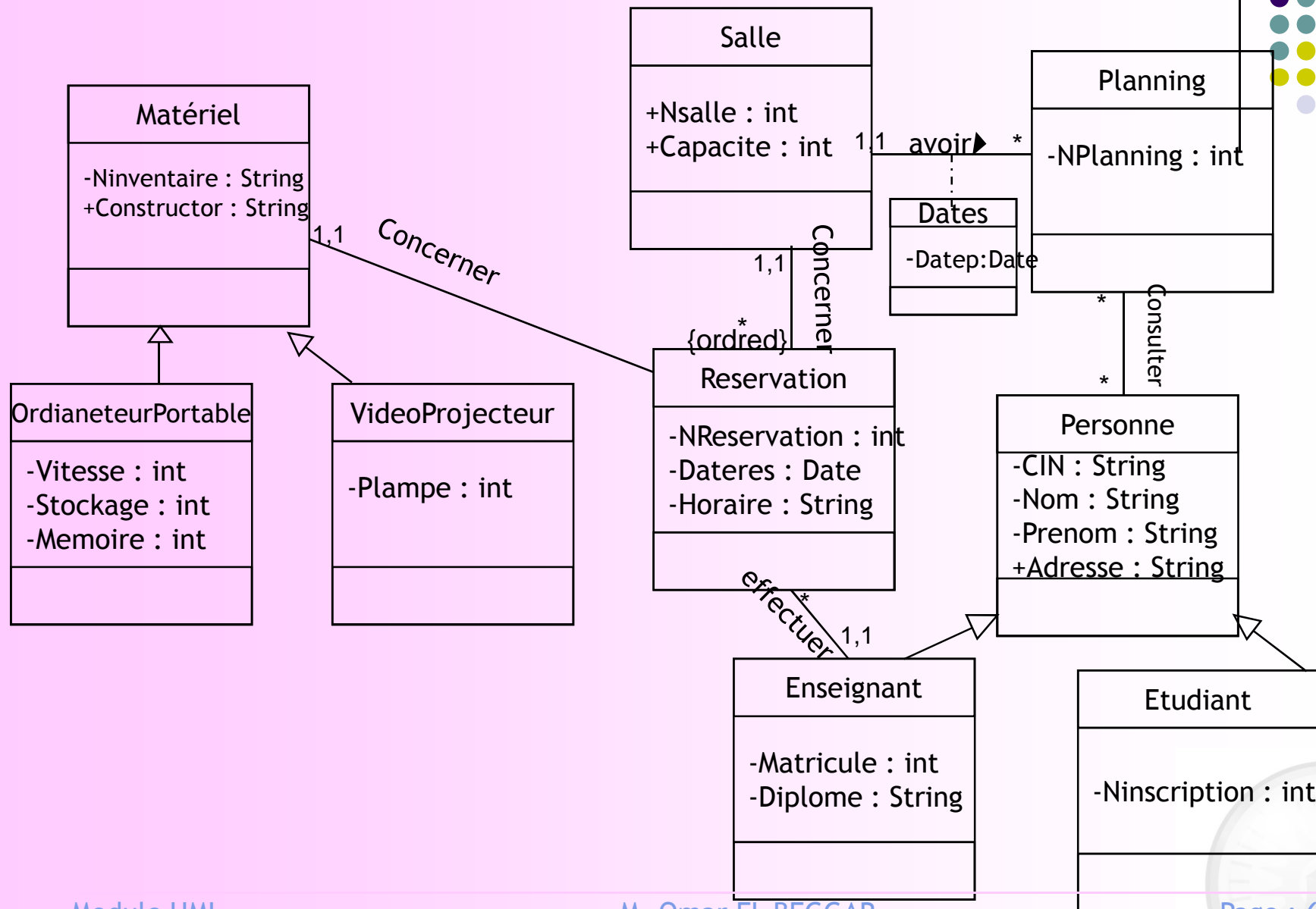
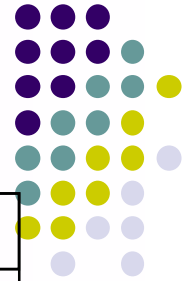
Les informations collectées sont :

- Nom de Stagiaire
- Prénom de Stagiaire
- Nom de l'encadrant
- Prénom de l'encadrant
- Lieu du stage
- Date du stage
- Note
- Classement
- Sujet de stage



Exercice de réservation Salle et matériel

(Diagramme de classe d'analyse)



Introduction



UML modélise les systèmes d'informations pour réaliser un système Informatique.

Système == Logiciel == un ensemble de programmes qui réalisent des Fonctionnalités répondant a des besoins clients.

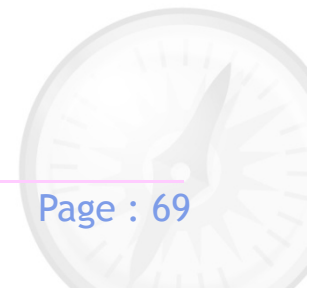
Logiciel est composé de programmes:

IHM : programme de communication ou de dialogue entre l'utilisateur et la machine;

Traitement : programme qui traite et contrôle les données saisies par l'utilisateur;

Data : programme qui stocke et interroge les données métiers du domaine.

Tout programme en O.O est une classe, Donc Système en O.O est composé de classes dialogue, classes contrôle et classes métiers. Ces objets peuvent que interagir entre eux pour réaliser une fonctionnalité eux. Cette dynan système ou interaction et la partie dynamique de l'analyse



Analyse Dynamique (LARMAN)

Opérations Système et contrat d'opérations



Pour Chaque cas d'utilisation, on dégage les opérations Système, et pour chaque opération on détermine son contrat, qui possède les informations suivantes :

Nom d'opération

Responsabilité

Preconditions

Postconditions

Exceptions et notes.

Par exemples : Cas d'utilisation « Réserver une salle de cours », les opérations Système qu'il contient sont :

Opérations Système
Vérifier la disponibilité
Créer une réservation

Contrat d'opération :

Nom d'opération : Créer une réservation

Références : Cas d'utilisation « Réserver une salle de cours ».

Preconditions : les salles et enseignants doivent être enregistrés dans le système.

PostConditions : un Objet réservation : R est crée,
un Objet salle : S et un objet
enseignant :E liés a cette réservation.

Une fois établi le contrat d'opération, on détermine les changements de l'état Système grâce aux Postconditions. Et ceci en utilisant les diagrammes de collaboration ou de séquence « BB »



Analyse Dynamique

Opérations Système et contrat d'opérations



Chaque opération Système va donner lieu a une étude dynamique sous la forme d'un diagramme d'interaction(Communication,Séquence boîte blanche ou collaboration). Les diagrammes d'interaction ainsi réalisés vont permettre d'élaborer le diagramme de clases de conception, et ceci en ajoutant principalement les informations suivantes aux classes issues du modèle d'analyse.

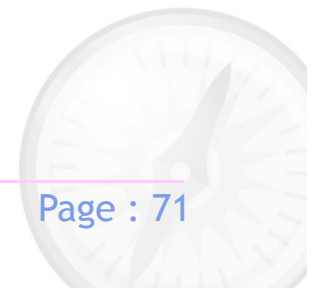
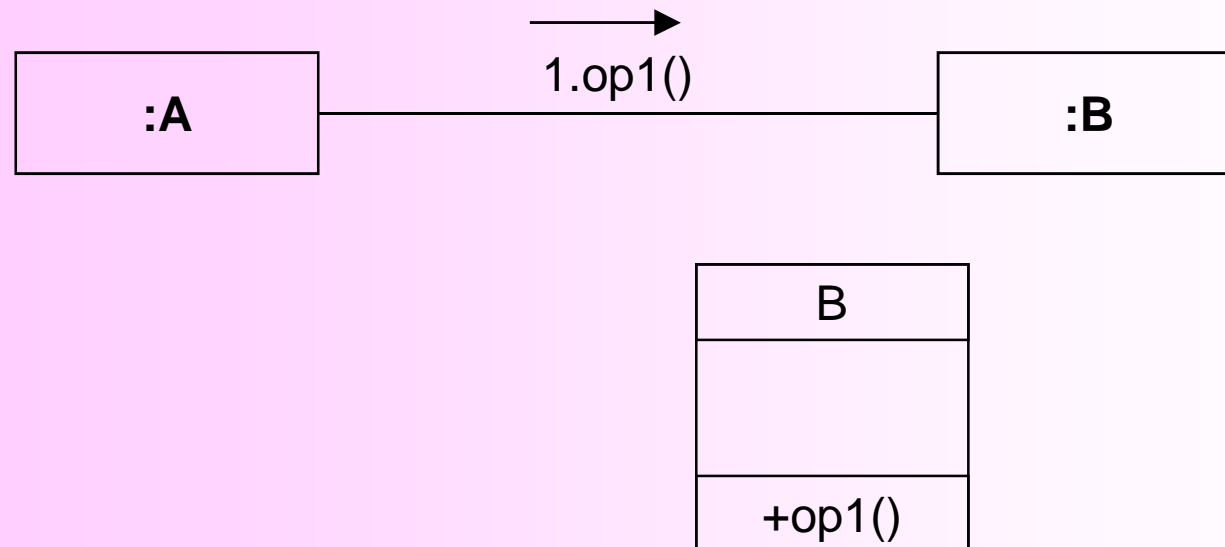


Diagramme de séquence Système boîte blanche



Définition

est un diagramme d'interaction mettant l'accent sur la chronologie de l'envoi des Messages entre les classes d'analyse et les acteurs principaux et secondaires.

Un diagramme de séquence boîte blanche est un diagramme de séquence qui dévoile les composants du système représenté avant dans un diagramme de séquence boîte noire. Le système boîte blanche se compose des objets présentatifs, logiques et métiers (Classes d'analyses).

Les principales informations contenues dans un diagramme de séquence boîte blanche sont les messages échangés entre les lignes de vie des acteurs principaux, acteurs Secondaires et classes d'analyse qui composent le système présentés dans un ordre chronologique. Ainsi, contrairement au diagramme de communication, le temps y est représenté explicitement par une dimension (la dimension verticale) et s'écoule de haut en bas.

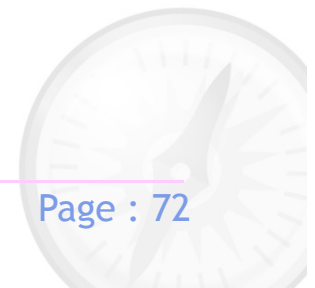


Diagramme de séquence Boite Blanche

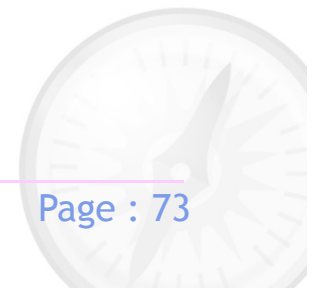
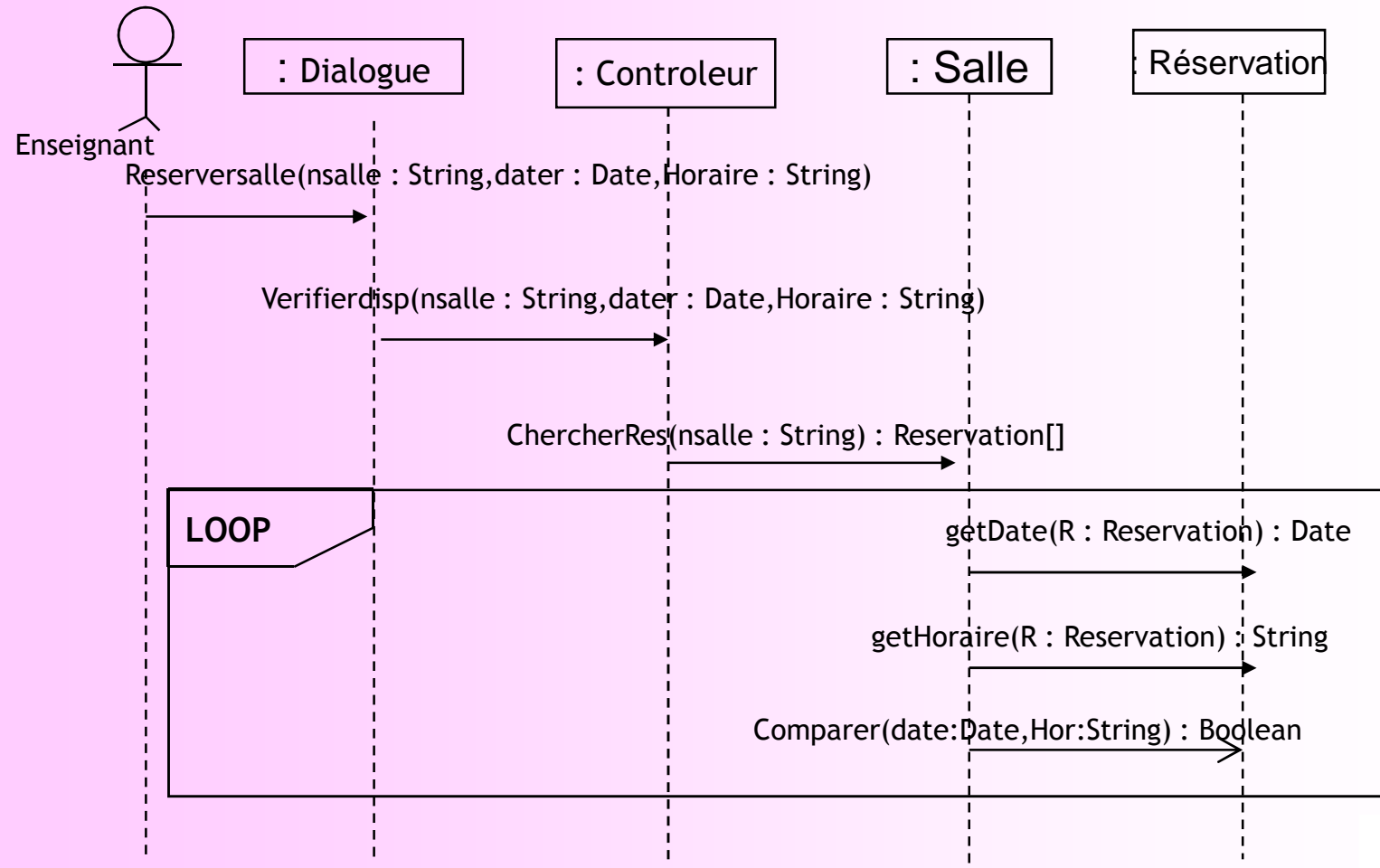


Diagramme de Communication



Définition

Le diagramme de communication est un diagramme d'interaction mettant l'accent sur l'organisation structurelle des objets qui envoient et reçoivent des messages. Contrairement à un diagramme de séquence, un diagramme de communication rend compte de l'organisation spatiale des participants à l'interaction, il est souvent utilisé Pour illustrer un cas d'utilisation ou pour décrire une opération définit par LARMAN. Le diagramme de Communication aide à valider les associations du diagramme de classe en les utilisant comme support de transmission des messages. (*Exemple voir le slide suivant*)

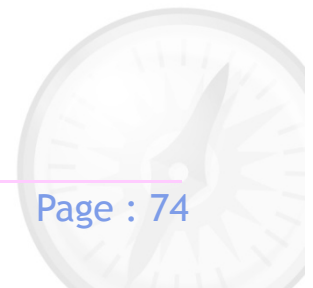
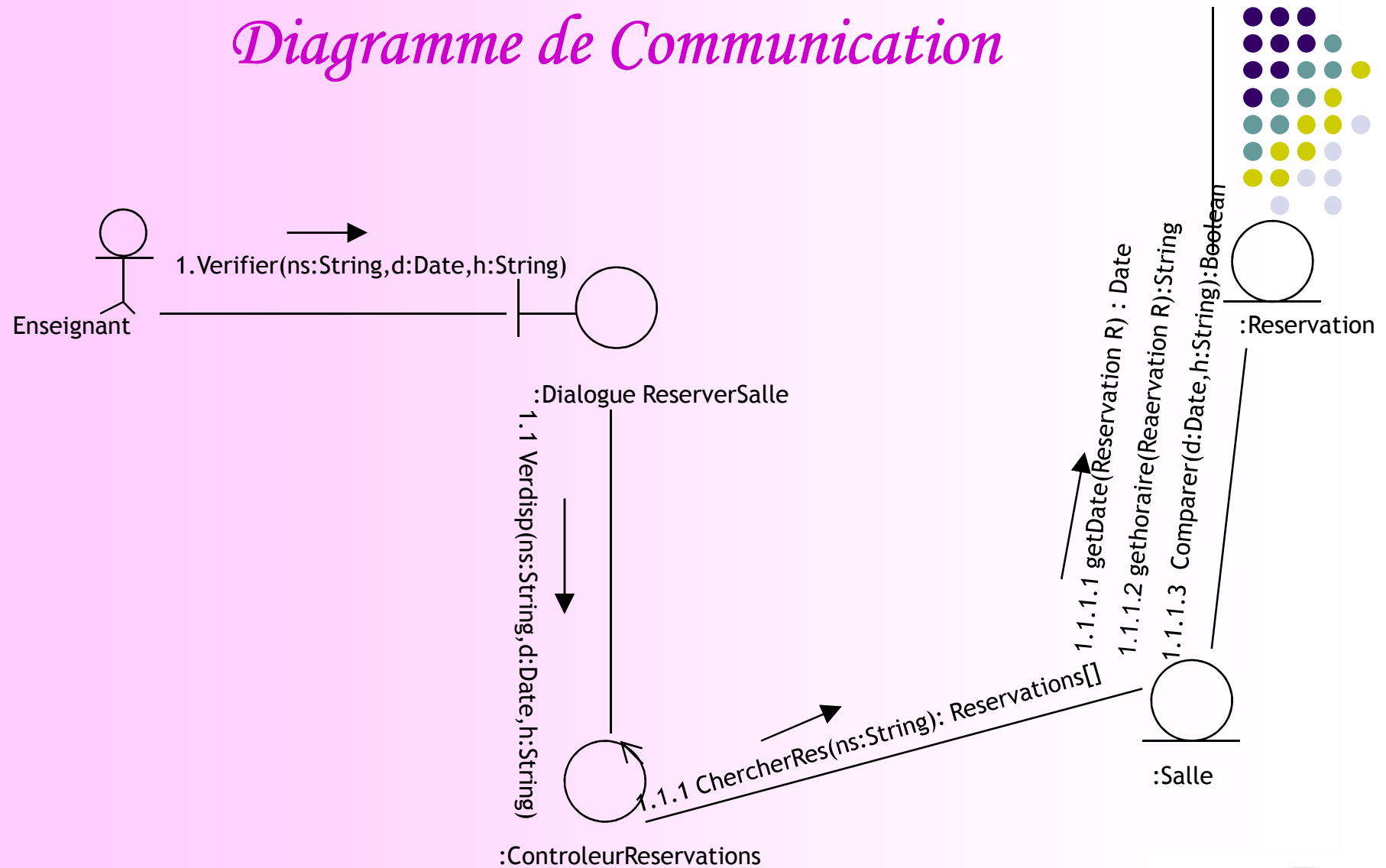
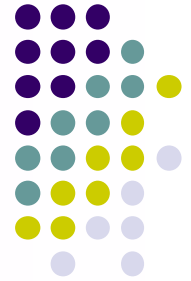


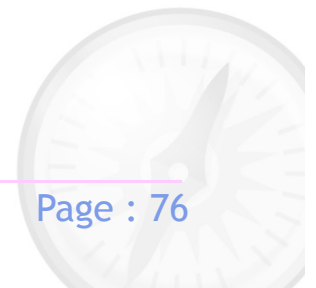
Diagramme de Communication



Le diagramme de communication qu'illustre l'opération « Vérifier disponibilité » du cas d'utilisation « Réserver Salle »



Conception



GRASP PATTERNS

Général Responsibilities Affection Pattern



Contrôleur pattern

Objet responsable d'accueillir les messages de l'extérieur et de les acheminer vers les classes métiers chargées du traitement.

Créateur pattern

Objet responsable de créer des objets, généralement c'est le conteneur, sinon c'est l'objet le plus proche fonctionnellement; exemple l'agrégation ou la combinaison.

Expert pattern : celui qui sait qui fait, l'objet qui contient les informations c'est l'objet qui doit renvoyer l'objet recherche. La classe commande possède l'attribut total, donc elle possédera aussi la méthode getTotal() : double.

Faible couplage :

degré de dépendance, exemple l'héritage présente le forte couplage, la classe fille dépend fortement de la classe mère; Tout changement se répercutera sur la classe fille; Il faut donc éviter ce genre de couplage fort ou d'abuser dans son utilisation. Les classes doivent au maximum être indépendantes.

Forte collusion pattern :

Lorsqu'une par exemple dans une classe se trouve une méthode qui effectue à la fois : la recherche, l'ajout dans la base de données et afficher le total, c'est une forte collusion. Il faut séparer ses traitements en implémentant d'autres méthodes : recherche() : boolean; add(Object) : void et getTotal() : double

Parfois il est inutile d'utiliser le conteneur sur le diagramme de classe d'analyse pour trouver les méthodes. Il est utile d'ajouter une classe collection qui contient qu'une partie des classes composantes dans la bd exemple commande et commandes livres.



Diagramme de Collaboration



Les collaborations sont des interactions entre objets, dont le but est de réaliser un objectif du système (c'est-à-dire aussi de répondre à un besoin d'un utilisateur).

L'élément de modélisation UML "collaboration", représente les classes qui participent à la réalisation d'un cas d'utilisation.

Les diagrammes de collaboration mettent l'accent sur les relations « spatiales » entre objets. Les messages peuvent être numérotés pour introduire une dimension temporelle.

De nombreuses notations annexes permettent de caractériser les messages. Ils sont souvent utilisés pour décrire grossièrement la réalisation des cas d'utilisation.

Exemple : Un objet A envoie un message X à un objet B, puis l'objet B envoie un message Y à un objet C, et enfin C s'envoie à lui même un message Z.

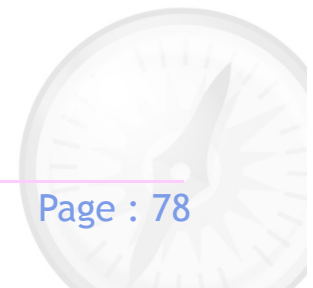
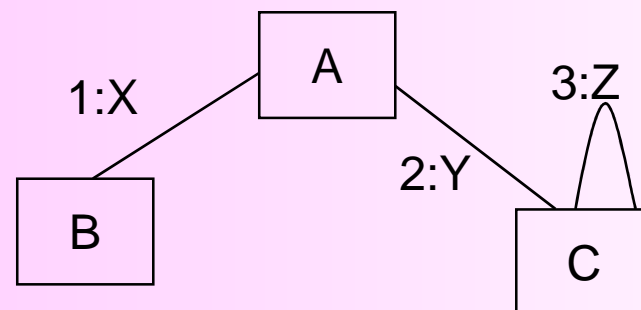
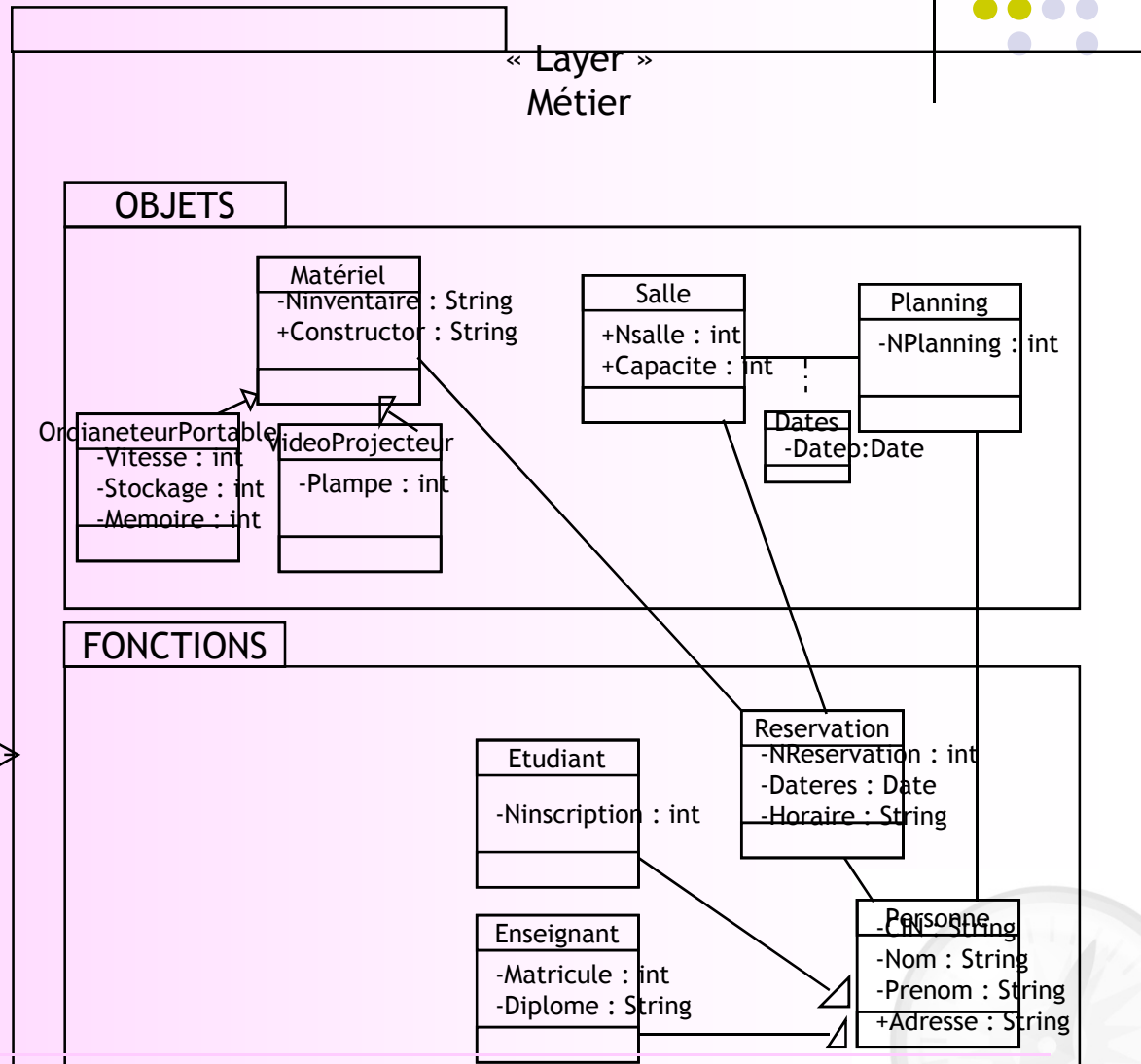
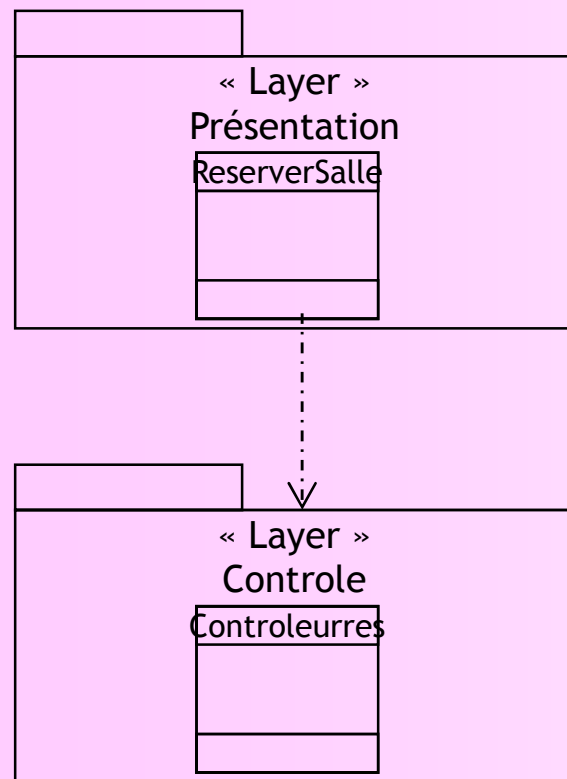


Diagramme de package

Exemple



Diagrammes de déploiement et de composants



Introduction :

Les diagrammes de composants et les diagrammes de déploiement sont des diagrammes de types statique en UML.

-Le diagramme de composants décrit le système modélisé sous forme de composants réutilisables et met en évidence leurs relations de dépendance.

-Le diagramme de déploiement représente les éléments matériels (PC, Modem, Station de travail, Serveur, etc.), leurs dispositions physiques (connexions) et la disposition des exécutable (représentés par des composants) sur ces éléments matériels.



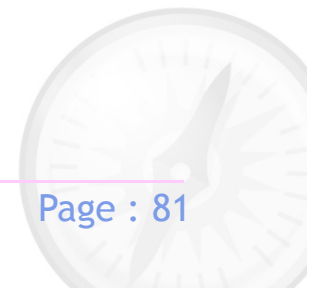


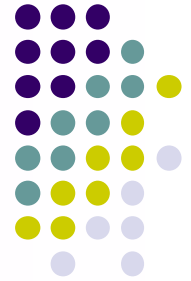
Composant :

Un composant est une unité autonome représentée par un classeur structuré, Stéréotypé «component», comportant une ou plusieurs interfaces requises ou offertes. Son comportement interne, généralement réalisé par un ensemble de classes, est totalement masqué : seules ses interfaces sont visibles. La seule contrainte pour pouvoir substituer un composant par un autre est de respecter les interfaces requises et offertes.

Relation de dépendance :

La relation de dépendance est utilisée dans les diagrammes de composants pour Indiquer qu'un élément de l'implémentation d'un composant fait appel aux services oerts par les éléments d'implémentation d'un autre composant





Implémentation en Java & SQL



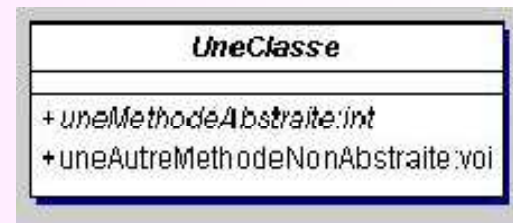
Implémentation du diagramme de classes en Java



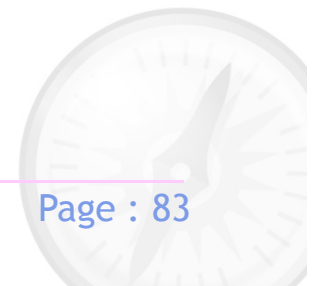
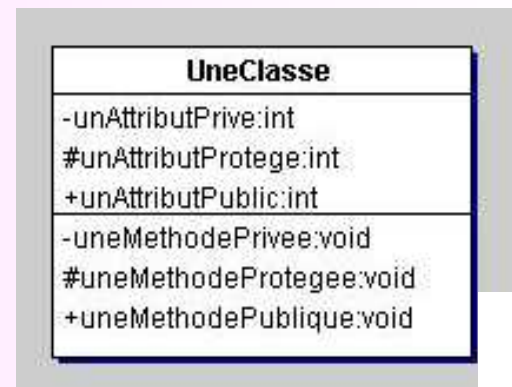
Package exemple;
public Interface Uninterface {
public abstract boolean uneMethodePublique();
}



Package exemple;
public abstract Class UneClasse {
public abstract int uneMethodeAbstraite() ;
public void uneAutreMethodeNonAbstraite(){ }
}



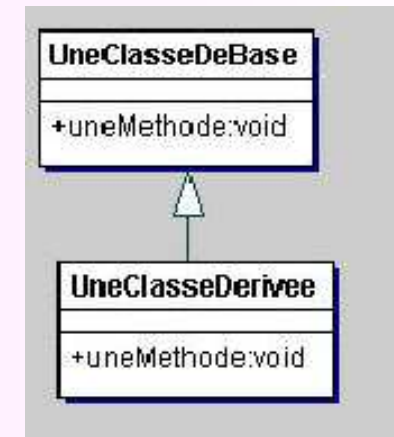
Package exemple;
public Class UneClasse {
private int unAttributPrive;
protected int unAttributProtege;
public int unAttributPublic;
private void uneMethodePrivee(){ }
protected void uneMethodeProtegee(){ }
public void uneMethodePublique(){ }
}



```

Package exemple;
public Class UneClasseDeBase {
public void uneMethode() { }
}
Public Class UneClasseDerivee extends UneClasseDeBase {
public void uneMethode() { }
}

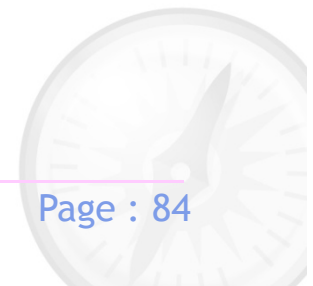
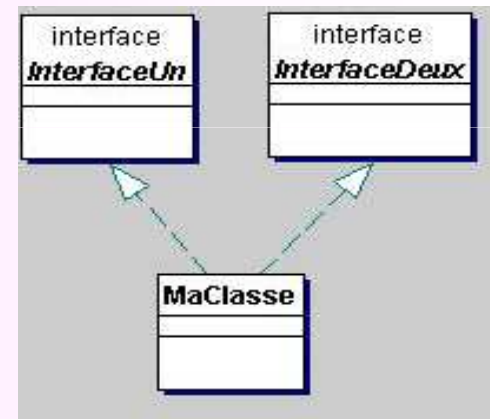
```



```

Package exemple;
public Interface InterfaceUn { }
public Interface InterfaceDeux { }
Public Class MaClasse implements InterfaceUn, InterfaceDeux {
}

```





```

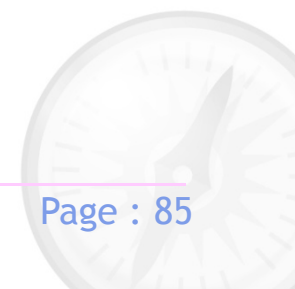
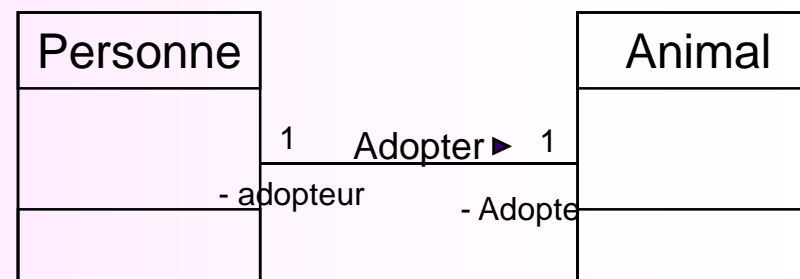
public class Animal {
    public int num;
    private Personne P;
    public Personne getP() {
        return P;
    }
    public void setP(Personne p) {
        P = p;
    }
    public void AddP(Personne p)
    {
        if (p!=null)
        {
            if (p.getA()!=null)
            {
                p.setA(null);
            }
            this.setP(p);
            p.setA(this);
        }
    }
}

```

```

public class Personne {
    public String nom;
    private Animal A;
    public Animal getA() {
        return A;
    }
    public void setA(Animal a) {
        A = a;
    }
    public void AddA(Animal a)
    {
        if (a!=null)
        {
            if (a.getP()!=null)
            {
                a.setP(null);
            }
            this.setA(a);
            a.setP(this);
        }
    }
}

```





Package exemple;

```
public Class A {
    Public B rb;
}
```

```
public Class B {
}
```

Package exemple;

```
public Class A {
    Public Vector<B> rb=new Vector<B>();
}
```

```
public Class B {
    public A ra;
}
```

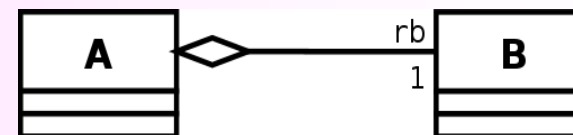
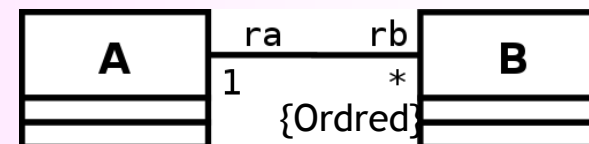
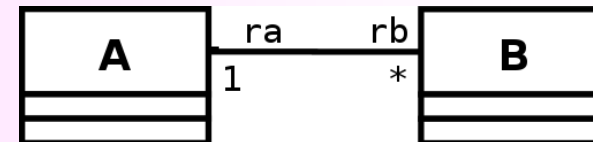
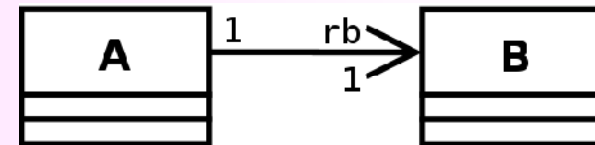
Package exemple;

```
public Class A {
    Public ArrayList<B> rb=new ArrayList<B>();
}
```

```
public Class B {
    public A ra;
}
```

N.B :

L'agrégation et la composition sont implémentées comme les associations en java. Sauf en cas de composition, il faut instancier l'objet ou la collection d'objets composant dans le constructeur de la composite



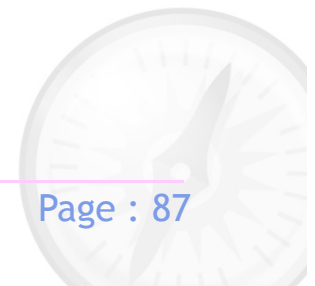
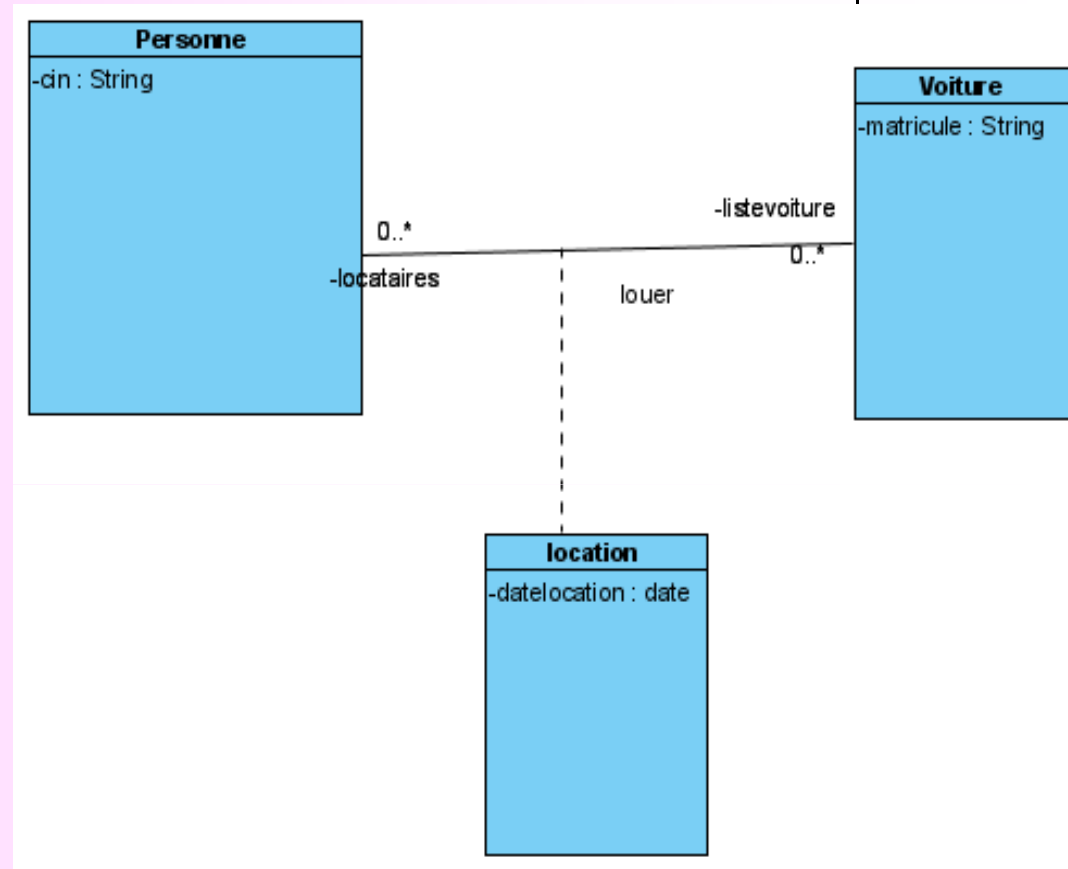
Classe d'association



```
import java.util.Vector;
public class Personne {
    private String _cin;
    Vector<location> listelocation_ = new
    Vector<location>();
}
```

```
import java.util.Vector;
public class Voiture {
    private String _matricule;
    Vector<location> _listelocation = new
    Vector<location>();
}
```

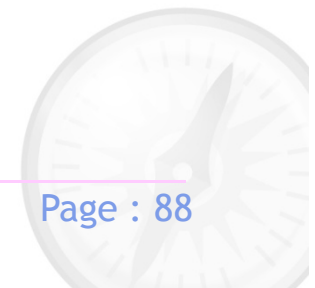
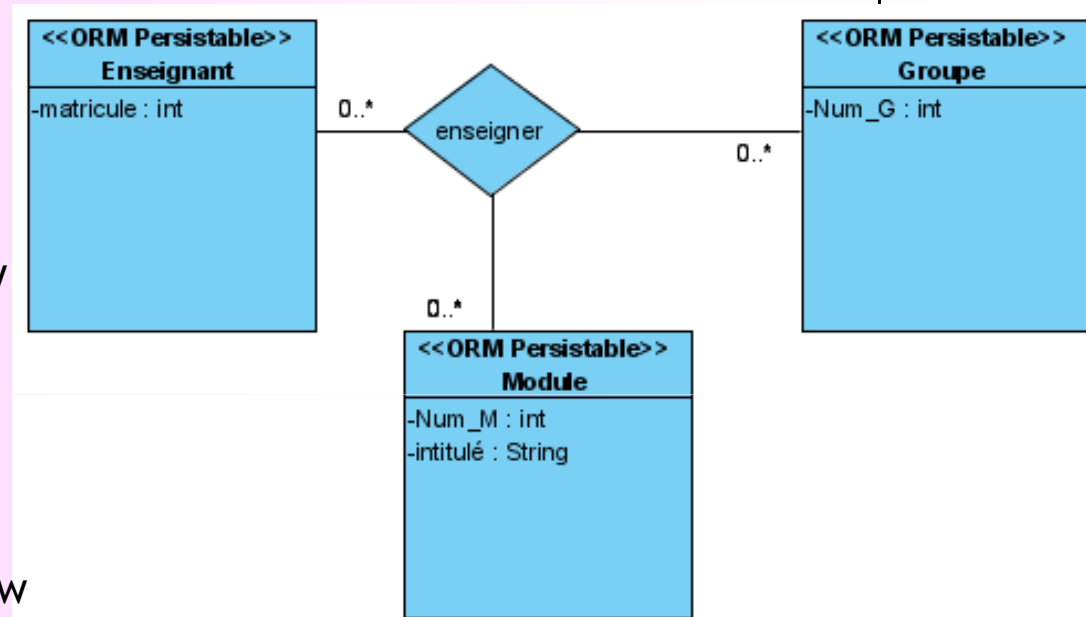
```
public class location {
    private date datelocation;
    private Personne locataire;
    private Voiture voiture;
}
```



Associations



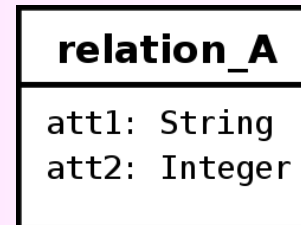
```
import java.util.Vector;
public class Enseignant{
private int matricule;
Vector<Enseigner> enseignements= new
    Vector<Enseigner>();
}
import java.util.Vector;
public class Groupe{
private int Num_G;
Vector<Enseigner> enseignement= new
    Vector<Enseigner>();
}
import java.util.Vector;
public class Module{
private int Num_M;
Vector<Enseigner> enseignements= new
    Vector<Enseigner>();
}
public class Enseigner{
private Enseignant enseignant;
private Module module;
private Groupe groupe;
}
```



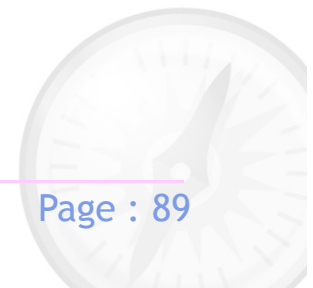
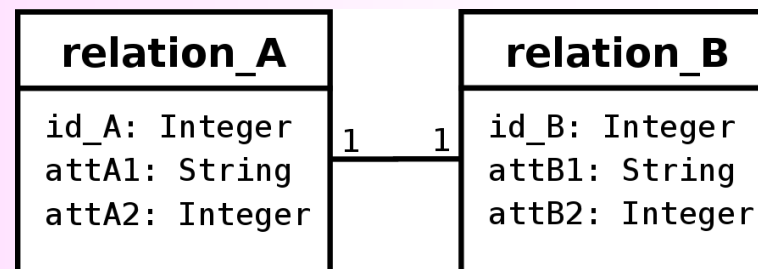
Implémentation du diagramme de classes en SQL



```
create table relation_A (  
  num_relation_A int primary key,  
  att1 text,  
  att2 int);
```



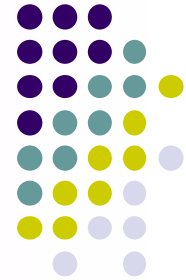
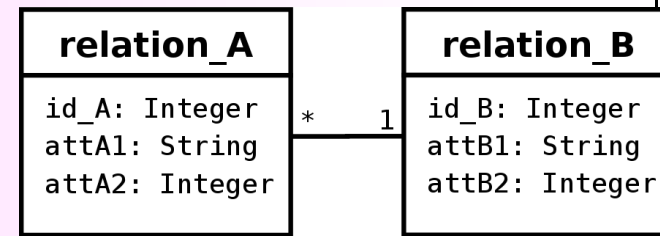
```
create table relation_A (  
  id_A int primary key,  
  attA1 text,  
  attA2 int);  
create table relation_B (  
  id_B int primary key,  
  num_A int references relation_A,  
  attB1 text,  
  attB2 int);
```



```

create table relation_A (id_A int primary key,
num_B int foreign key references
relation_B(id_B),attA1 text,attA2 int)
create table relation_B (id_B int primary key,
attB1 text,attB2 int)

```



```

create table relation_A (id_A int primary key,attA1 text,
attA2 int)

```

```

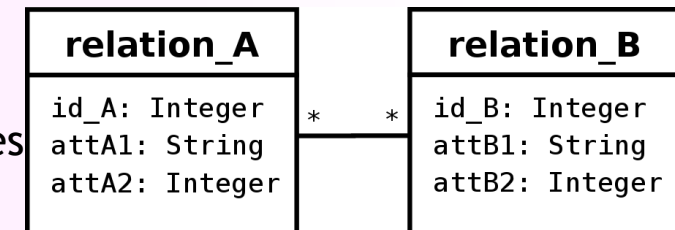
create table relation_B (id_B int primary key,attB1 text,
attB2 int)

```

```

create table relation_A_B (num_A int foreign key references
relation_A(id_A),
num_B int foreign Key references relation_B(id_B),
primary key (num_A, num_B))

```



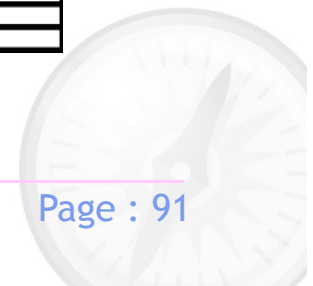
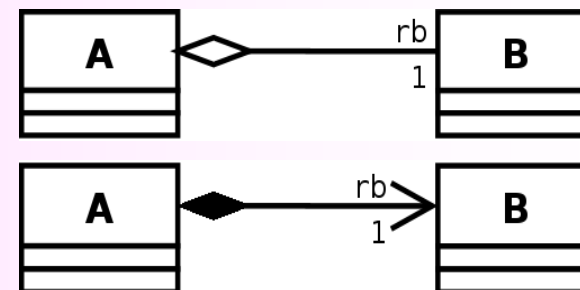
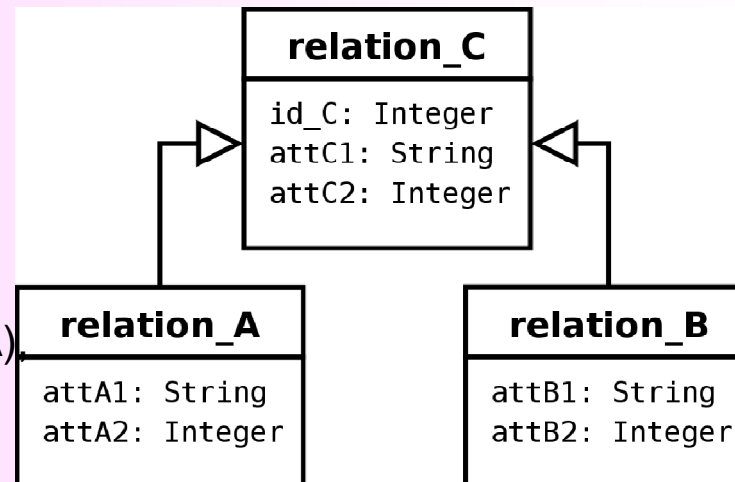


```
create table relation_C (  
id_C int primary key,  
attC1 text,  
attC2 int,  
type text);
```

```
create table relation_A (  
id_A foreign Key references relation_C(id_C),  
attA1 text,  
attA2 int,  
primary key (id_A));
```

```
create table relation_B (  
id_B foreign Key references relation_C(id_C),  
attB1 text,  
attB2 int,  
primary key (id_B));
```

L'agrégation et la composition sont implémentées
comme les associations en SQL



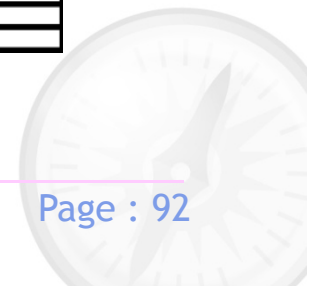
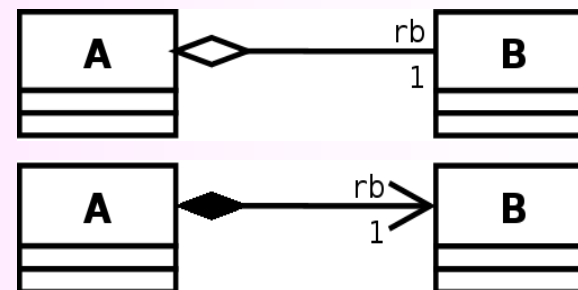
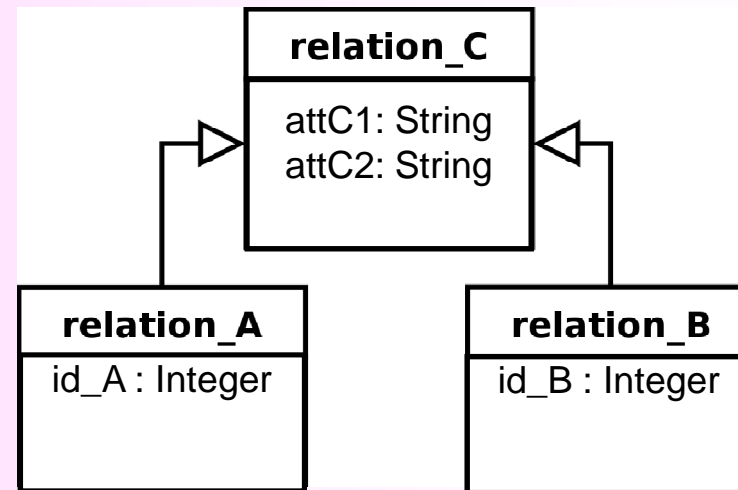


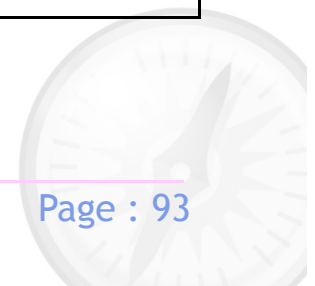
```
create table relation_C (  
id_C int primary key,  
attC1 text,  
attC2 int,  
type text);
```

```
create table relation_A (  
id_A foreign Key references relation_C(id_A),  
attA1 text,  
attA2 int,  
primary key (id_A));
```

```
create table relation_B (  
id_B foreign Key references relation_C(id_C),  
attB1 text,  
attB2 int,  
primary key (id_B));
```

L'agrégation et la composition sont implémentées
comme les associations en SQL





Démonstration du réflexive à l'héritage avec association entre classes dérivées (1)



Association réflexive

```
public class Personne{  
    private Personne encadrant;  
    private Vector<Personne> etudiants;  
}
```

A priori les classes Encadrant et Etudiant ont des attributs en commun (cin,nom,prenom...)

On applique le concept d'héritage (la spécialisation), on obtient deux classes dérivées Encadrant et Etudiant et la classe mère Personne :

```
public class Personne {  
}  
  
Public class Encadrant extends Personne{  
}  
  
Public class Etudiant extends Personne {  
}
```

Association d'héritage avec association entre classes dérivées

```
public class Personne {  
}  
  
Public class Encadrant extends Personne{  
    private Vector<Etudiant> etudiants;  
}  
  
Public class Etudiant extends Personne {  
    private Encadrant encadrant;  
}
```

Démonstration du réflexive à l'héritage avec association entre classes dérivées (1)



Association réflexive

L'association encadrer indique qu'un encadrant encadre plusieurs étudiants et inversement un étudiant a un seul encadrant, donc :

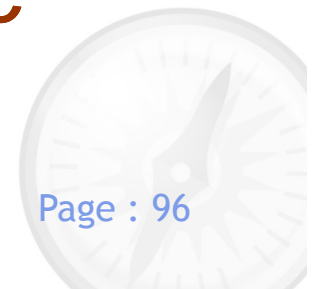
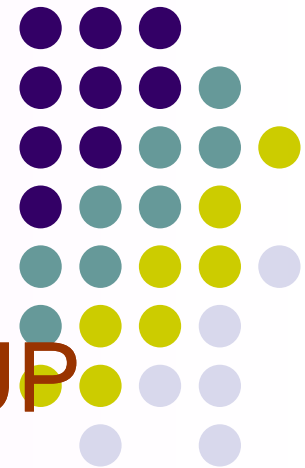
```
public class Personne {  
}  
  
Public class Encadrant extends Personne{  
private Vector<Etudiant> etudiants;  
}  
  
Public class Etudiant extends Personne {  
private Endrant encadrant;  
}
```

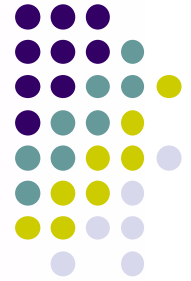
On obtient finalement une Association d'héritage avec association entre classes dérivées



U.M.L.

Démarche UP Résumé





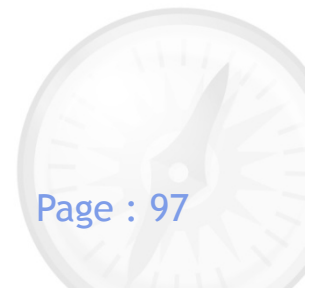
UML

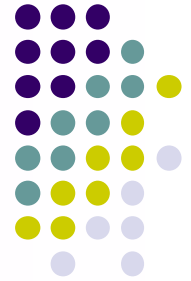
- **Des modèles (diagrammes) : UML**

- Pour représenter la réalité
- Pour avoir un langage commun entre les différents intervenants du projet

- **Une démarche : UP**

- Pour décomposer et découper le système, (ce qui permet de réduire la complexité, répartir le travail, et faciliter la réutilisabilité des sous-systèmes)
- Pour faciliter l'organisation, le pilotage et le suivi du projet





Démarche d'analyse et de conception objet

- **Inception : Définition des besoins**

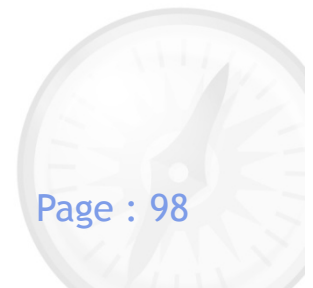
- Phase courte, de 'débroussaillage'
- Non itérative : couvre l'ensemble du projet
- Identifier les principales fonctionnalités

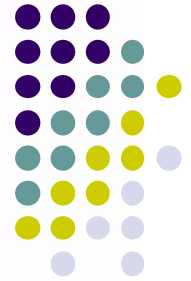
- **Analysis : Phase d'analyse**

- Phase itérative.
- Approfondir les besoins de manière exhaustive : examiner les différents scénarios, tenir compte des traitements exceptionnels et cas d'erreur
- Rester au niveau fonctionnel

- **Design: Phase de conception**

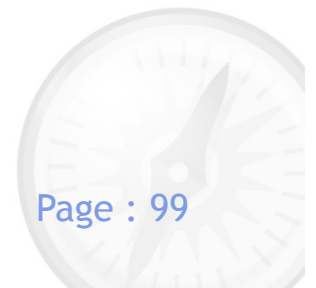
- Intégrer les choix d'architecture et les outils utilisés
- comprendre la logique de fonctionnement du système
- En général, seuls certains éléments de la conception sont modélisés a





Phase d'Inception

- **Lister les exigences du client** issues du cahier des charges, ou issues d'une démarche préalable de collecte d'information
- **Identifier les différents acteurs du système**
- **Associer les exigences aux acteurs** en élaborant un diagramme de contexte
- **Regrouper les exigences en intentions d'acteurs**
- **Elaborer le diagramme des Uses-Cases**
- **Effectuer une description sommaire (*de haut niveau*) de Cases**



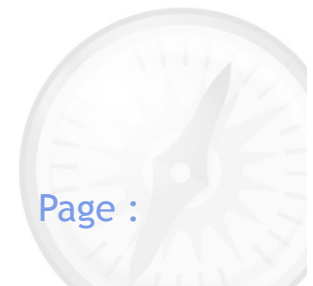


Phase d'Inception

Les exigences du client

- On s'occupe des exigences fonctionnelle
- Les exigences sont explicites ou implicites
- Les exigences seront numérotées

Référence	Fonction
R1	Modifier le prix d'un produit
R2	Calculer le total d'une vente
R3	Rentrer une quantité par article
R4	Calculer le sous total d'un produit
R5	Retourner une marchandise
R6	Payer l'achat
R7	Editer un ticket de vente
R8	Editer un rapport succinct
R9	Editer un rapport détaillé
R10	Se connecter à la caisse
R11	Se connecter en gérant
R12	Définir les droits des usagers
R13	Entrer un produit au catalogue
R14	Supprimer un produit du catalogue
R15	Enregistrer un produit à la caisse
R16	Initialiser la caisse

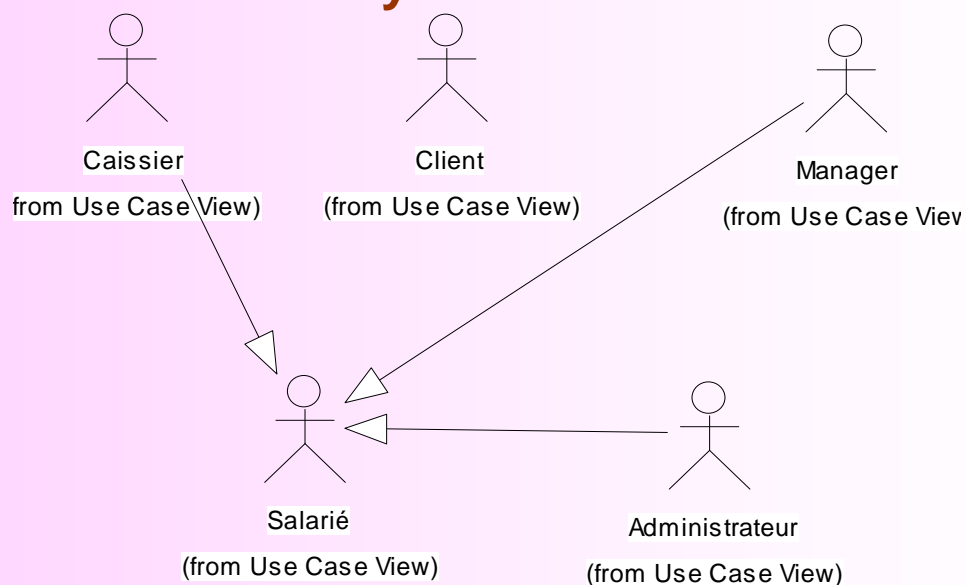




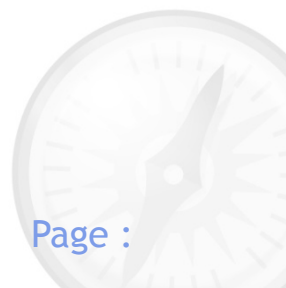
Phase d'Inception

Les acteurs

- Les acteurs représentent un rôle, et non pas une personne.
- Acteurs humains ou système informatique pré-existant qui s'interface avec notre système



les acteurs de l'application

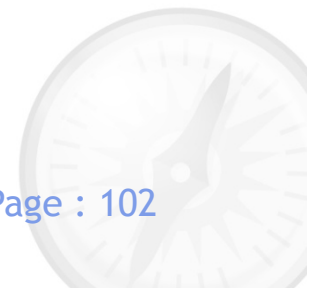
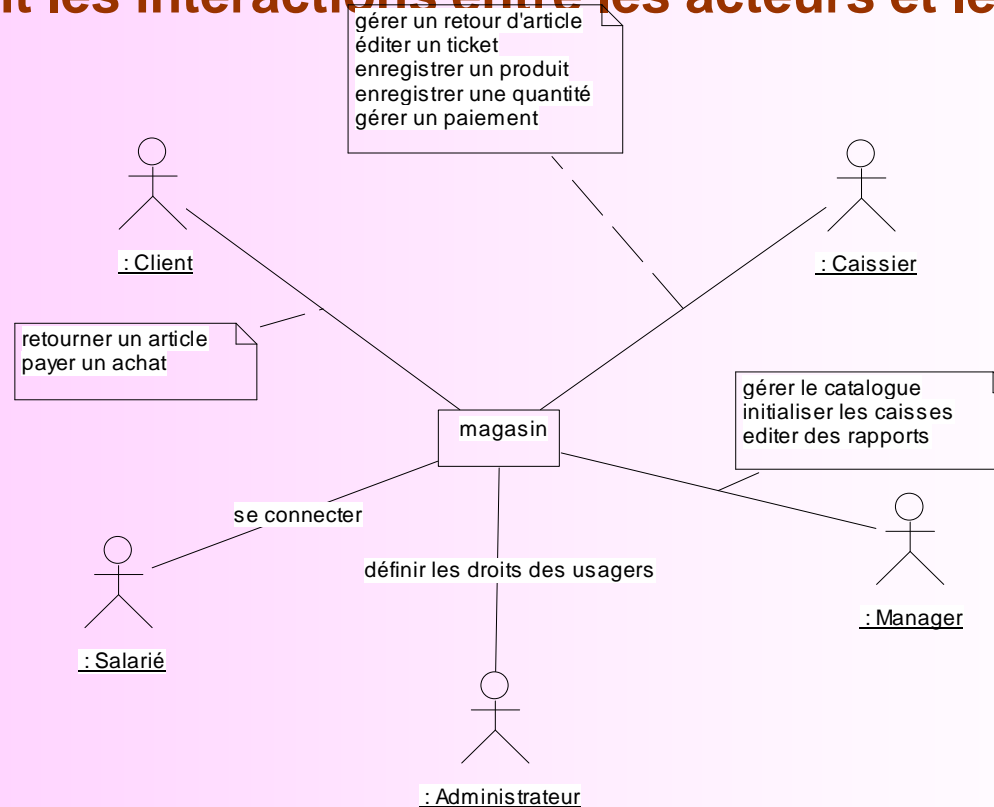


Phase d'Inception

Le diagramme de contexte



- Définit les interactions entre les acteurs et le système.



Phase d'Inception

Les intentions d'acteur

- On regroupe les exigences en 'intentions d'acteur'
- enchaînement de fonctions qui rend un service complet à un acteur

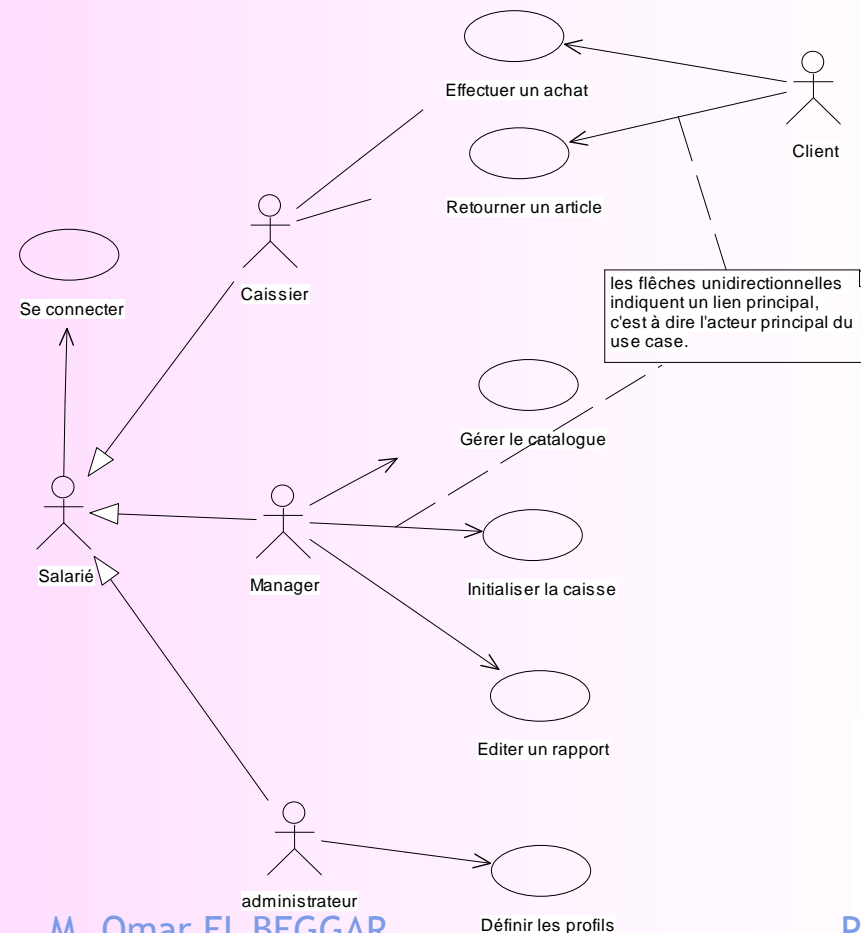
Référence	Fonction	Intention d'acteur	Acteurs
R1	Modifier le prix d'un produit	Gérer le catalogue	Manager
R13	Entrer un produit au catalogue	Gérer le catalogue	
R14	Supprimer un produit du catalogue	Gérer le catalogue	
R16	Initialiser la caisse	Initialiser la caisse	Manager
R5	Retourner une marchandise	Retourner un article	Client, Caissier
R10	Se connecter à la caisse	Se connecter	
R11	Se connecter en gérant	Se connecter	
R8	Éditer un rapport succinct	Éditer un rapport	Manager
R9	Éditer un rapport détaillé	Éditer un rapport	Administrateur
R12	Définir les droits des usagers	Définir les profils	
R7	Éditer un ticket de vente	Effectuer un achat	
R6	Payer l'achat	Effectuer un achat	Client, Caissier
R2	Calculer le total d'une vente	Effectuer un achat	
R3	Rentrer une quantité par article	Effectuer un achat	
R15	Enregistrer un produit à la caisse	Effectuer un achat	
R4	Calculer le sous total d'un produit	Effectuer un achat	



Phase d'Inception

Le diagramme des Uses-Cases

- Un use case est une intention d'acteur
- enchaînement de fonctions qui rend un service complet à un acteur





Phase d'Inception

Description de haut niveau des Uses-Cases

- Description textuelle assez succincte
- Sert à borner le Use-Case, et notamment à définir :
 - l'évènement déclencheur du Use-Case (quand il commence),
 - la terminaison du Use-Case (quand il finit)
 - Le rôle du Use-Case (enchaînement sommaire des opérations du Use-Case)

Nom du Use Case:

Acteur principal:

Acteurs secondaires:

Évènement déclencheur du Use Case:

Rôle du Use Case:

Terminaison du Use Case:

Effectuer un achat

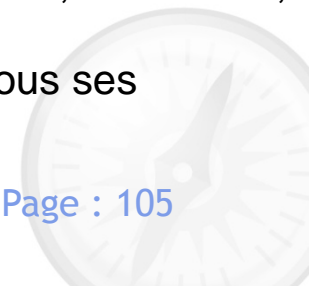
Client

Caissier

Un client arrive à la caisse avec ses achats

Le caissier passe tous les articles, fait la note, et encaisse le paiement.

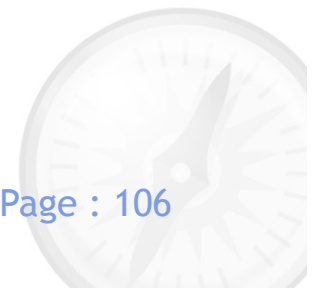
Le client a payé et a ramassé tous ses articles.



Phase d'Analysis



- **Effectuer une description détaillée (*de bas niveau*) des Uses-Cases concernés par l'itération :**
 - Identifier les différents scénarios
 - Eventuellement représenter ces scénarios sous forme de diagramme d'activité
 - Représenter les enchainements d'opérations
 - Lister les différents cas d'anomalies et d'exception
- **Représenter chaque scénario sous forme d'un diagramme de séquence (boîte noire)**
- **Faire un diagramme de classe par use case traité**
- **Elaborer les contrats d'opérations (impact de chaque opération sur les classes de notre système)**
- **Eventuellement, élaborer un diagramme d'état pour classes complexes**





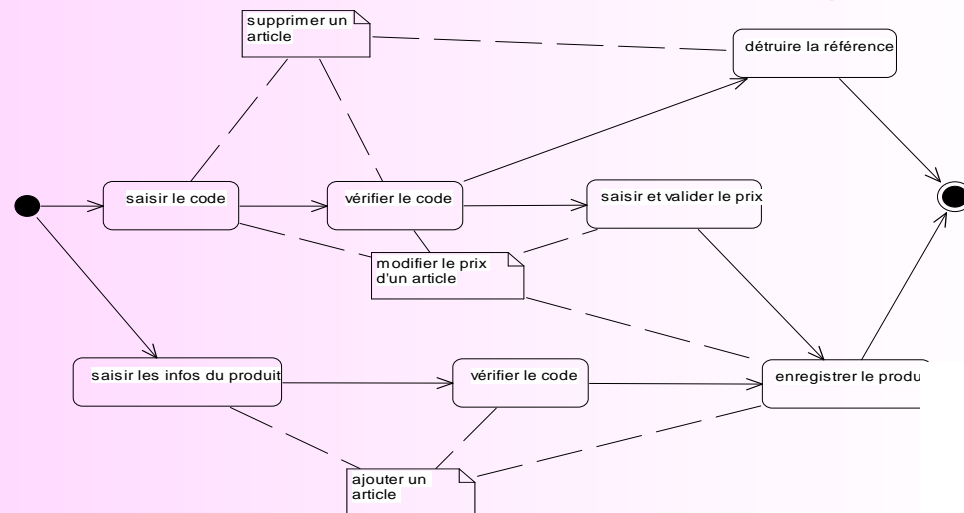
Phase d'Analysis

Description de bas niveau des Uses-Cases

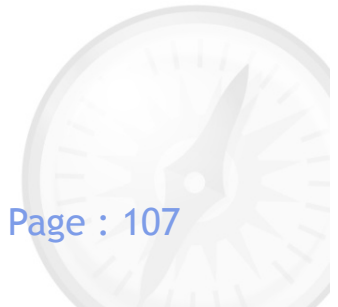
- Liste d'éventuels scénarios différents
- Identification des pré-conditions nécessaires au bon fonctionnement du Use-Case

Les pré conditions nécessaires au fonctionnement du use case: La caisse est allumée et initialisée. Un caissier est connecté à la caisse

- Représentation des scénarios dans un diagramme d'activité

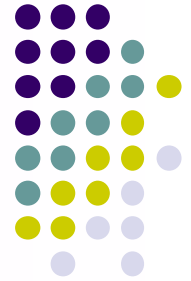


réalisation d'un diagramme d'activité (ici ce n'est pas la norme UML car la version de rose utilisée ne supporte pas les diagrammes d'activité). Ce schéma a été construit à partir d'un diagramme d'état.



Phase d'Analysis

Description de bas niveau des Uses-Cases



- Description des échanges entre le système et les acteurs
- ainsi que la chronologie et l'enchaînement des actions, dans le cas nominal (cas idéal)
- Identification des cas d'erreurs et des exceptions

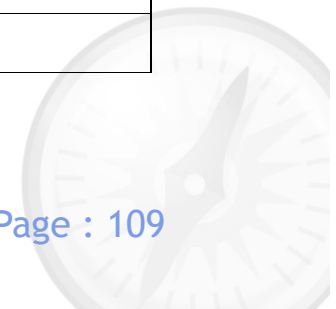


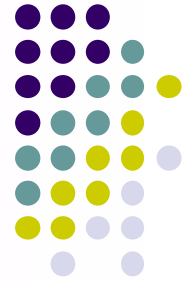


Phase d'Analysis

Description de bas niveau des Uses-Cases

Actions du client	Actions du caissier	Réponses du système
1) Le client arrive à la caisse avec les articles qu'il désire acheter.	2) Le caissier enregistre chaque article.	3) Le système détermine le prix de l'article, les informations sur l'article et ajoute ces informations à la transaction en cours. Il affiche ces informations sur l'écran du client et du caissier.
	4) Le caissier indique la fin de vente quand il n'y a plus d'article.	5) Le système calcule et affiche le montant total de l'achat.
	6) Le caissier annonce au client le montant total.	
7) Le client donne au caissier une somme d'argent supérieure ou égale à la somme demandée.	8) Le caissier enregistre la somme donnée par le client.	9) Le système calcule la somme à rendre au client.
	10) Le caissier encaisse la somme remise par le client et rend la monnaie au client.	11) Le système enregistre la vente effectuée
		12) Le système édite le ticket de caisse
	13) Le caissier donne le ticket de caisse au client	
14) le client s'en va avec les articles qu'il a achetés.		

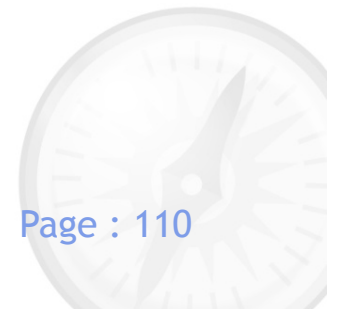




Phase d'Analysis

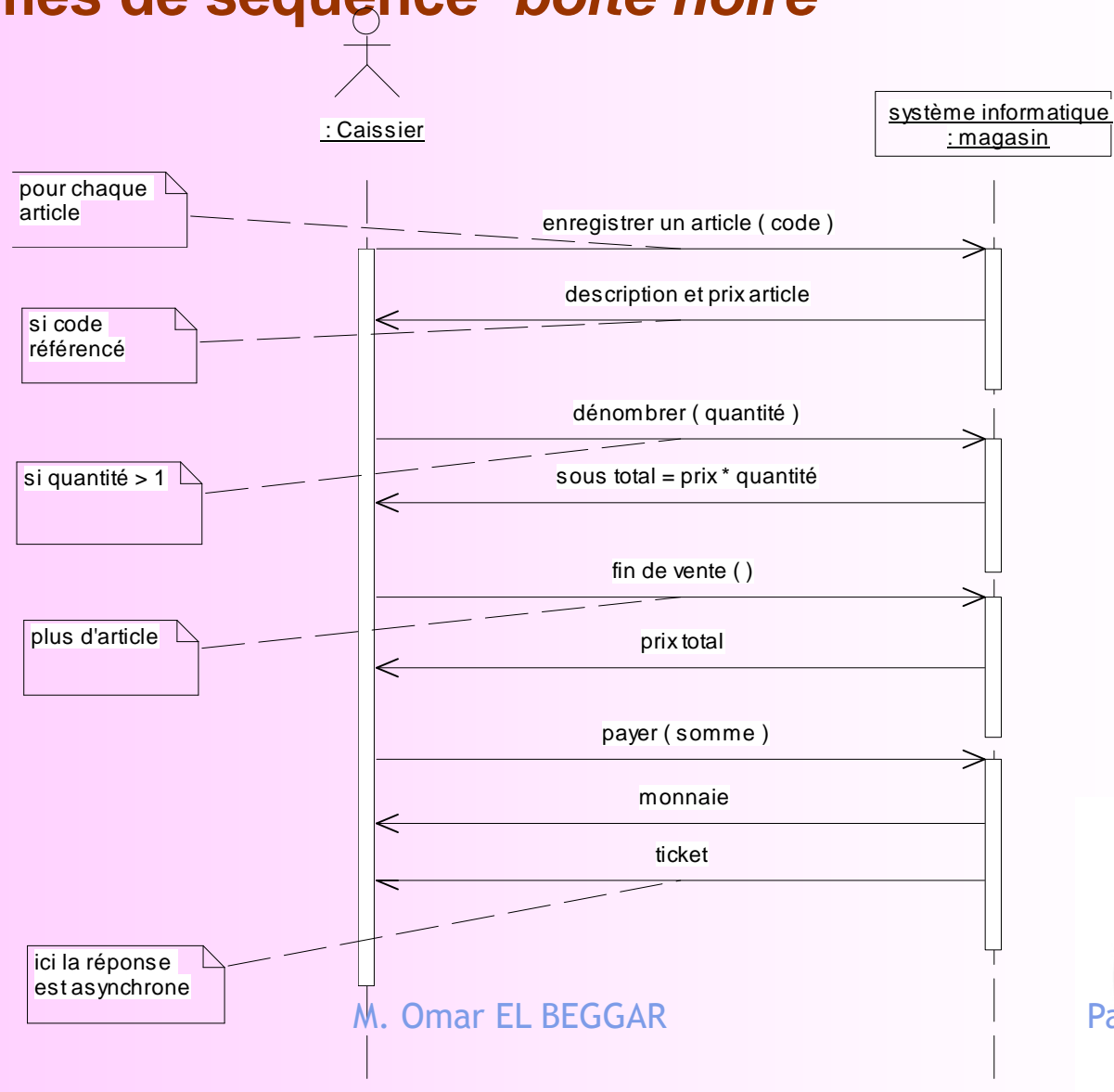
Diagrammes de séquence '*boîte noire*'

- Pour chaque scénario
- Uniquement le cas nominal (cas idéal)
- échanges entre les acteurs utilisateurs du système informatique et le système proprement dit.
- Les '*flèches qui rentrent*' dans le système sont des opérations
- Faire apparaître les paramètres
- On commence à voir apparaître des interfaces graphiques fonctionnelles



Phase d'Analysis

Diagrammes de séquence 'boîte noire'



Phase d'Analysis

Diagramme de classe d'analyse

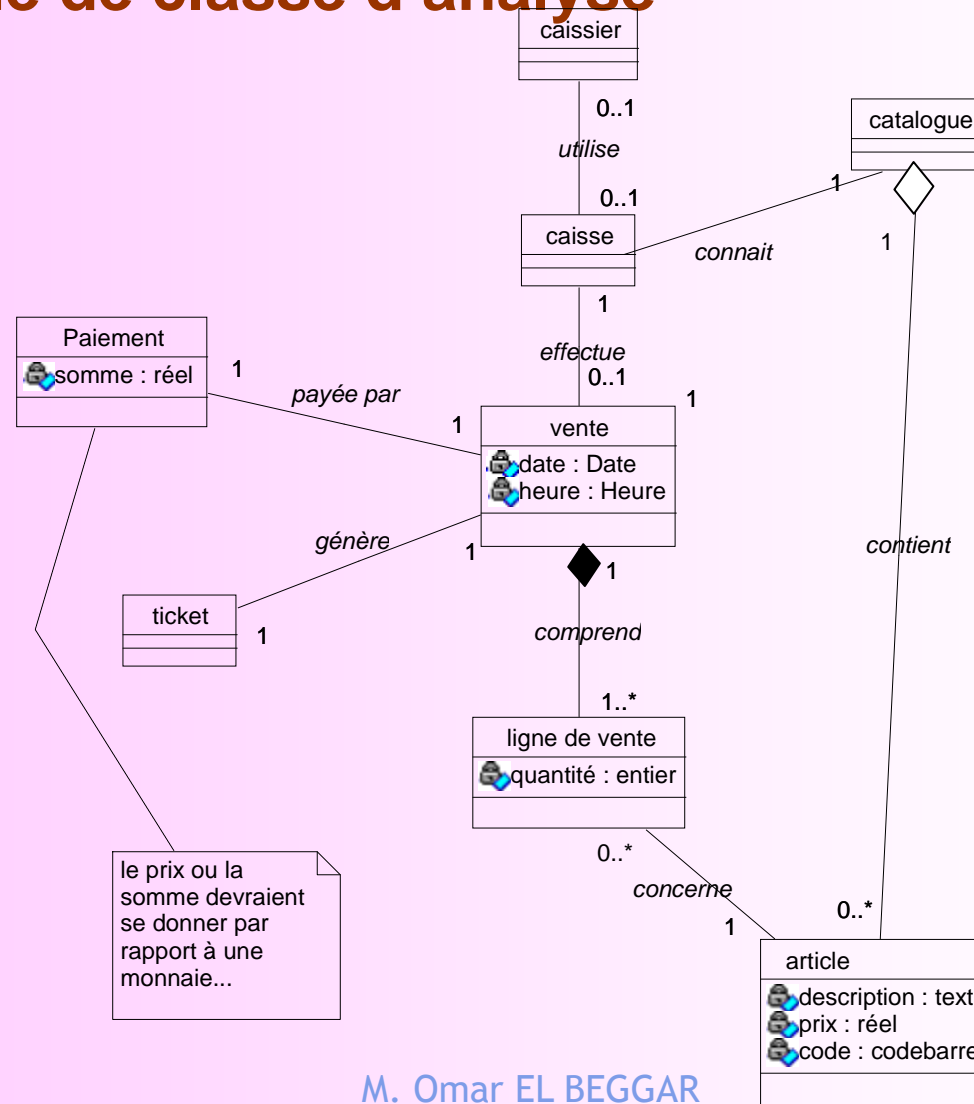


- **Pour chaque Use-Case**
- **but de ce diagramme : clarifier le domaine métier**
- **Seules apparaissent les classes métier.**
- **Les méthodes ne sont pas représentées**
- **Démarche :**
 - Identifier les classes métier
 - Etablir des associations –associations, héritage, composition, agrégation...) entre ces classes
 - Mettre des multiplicités
 - Placer des attributs



Phase d'Analysis

Diagramme de classe d'analyse



Phase d'Analysis

Diagramme de classe d'analyse



- Le diagramme de classe d'analyse va nous aider à élaborer le MCD (se poser la question : quelles vont être les classes métier persistantes?)
- Donc en parallèle (cela ne fait pas partie d'UML, mais cela fait partie intégrante de l'analyse du projet), il faut élaborer le Modèle conceptuel des données, et en déduire le MLD.



Phase d'Analysis

Les contrats d'opérations



- Pour chaque opération (message entrant dans notre système) :
- Décrit, de manière textuelle, l'impact de chaque opération sur le système
 - Objets créés
 - Objets supprimés
 - Associations créées
 - Associations supprimées
 - Attributs modifiés
 - Attributs affichés.
- Bonne pratique : forme '*has-been*'
- On entr'ouvre la boîte noire.....



Phase d'Analysis

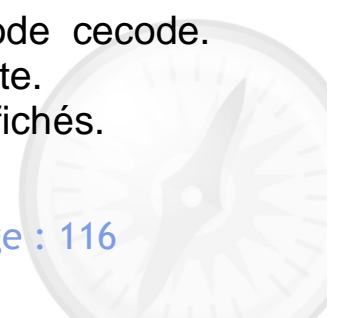
Les contrats d'opérations



- **Exemple :**

Enregistrer un article.

- Nom: enregistrer un article (cecode).
- Responsabilités: enregistrer un article lors d'une vente, et l'ajoute à la vente en cours.
- Exigences: R15 pour le use case effectuer un achat.
- Notes: Si l'article est le premier de la vente, il débute la vente.
Si le code cecode n'est pas référencé dans le catalogue, un message d'erreur est envoyé au caissier.
- Pré conditions: Il y a un article correspondant au code donné.
Il y a un caissier à la caisse.
- Post conditions: Si c'est le premier article de la vente, il faut qu'un objet vente ait été créé et associé à la caisse.
Une ligne de vente (ldv) a été créée. Elle a été associée à une description d'article correspondant au code cecode.
La ligne de vente ldv a été associée à la vente.
Le prix et la description de l'article ont été affichés.
L'attribut quantité a été mis à 1.



Phase d'Analysis

Diagramme d'état



- Le diagramme d'état peut être fait éventuellement pour détailler un objet complexe (cycle de vie de l'objet)

