Architecture du système d'entreprise (ESA)

cours

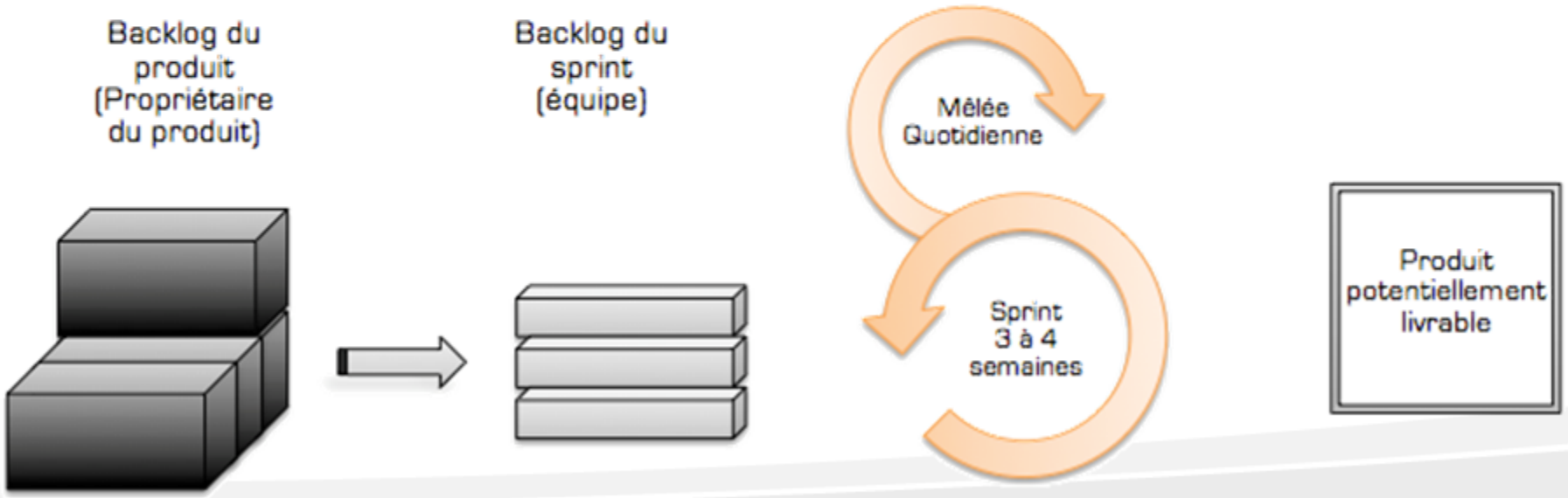
Enseignante: Dr. Maïssa HAMOUDA

Niveau: 3éme Licence Informatique (Génie Logiciel)

A.U. 2021/2022

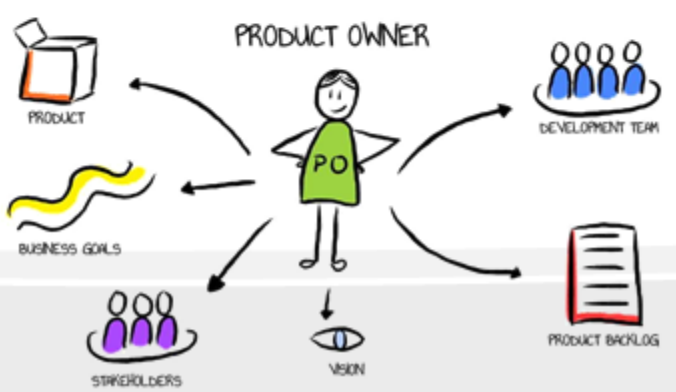
Modèle Scrum (1/3)

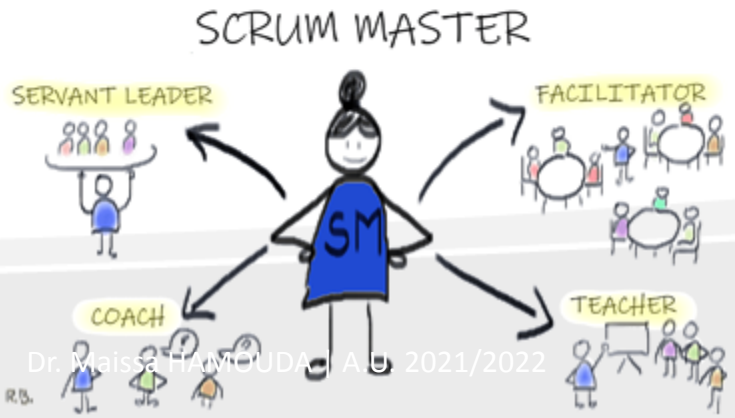
- Aujourd'hui « **Scrum** » est la méthode agile la plus populaire. Elle s'appuie sur des « **sprints** » qui sont des intervalles de temps assez courts pouvant aller de quelques heures jusqu'à un mois.
- Généralement et de préférence un sprint s'étend sur deux semaines. À la fin de chaque sprint, l'équipe présente ce qu'elle a ajouté au produit.



Modèle Scrum (2/3)

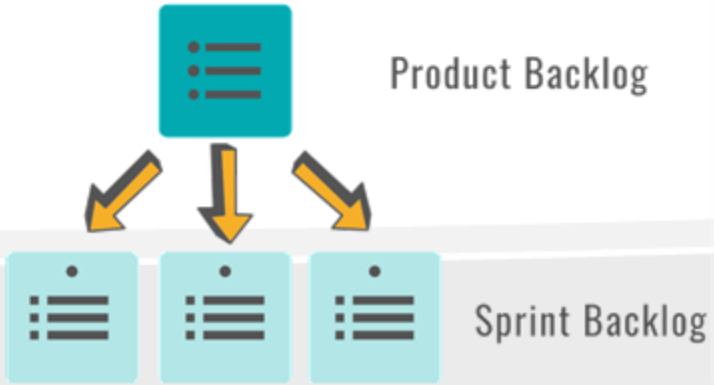
- Scrum regroupe trois acteurs :
  - Le **Product Owner** (ou « Directeur de produit ») : il communique les objectifs premiers des clients et utilisateurs finaux, coordonne l'implication des utilisateurs et des parties importantes, et se coordonne lui-même avec les autres product owners pour assurer une cohérence.
  - Le **Scrum Master** : membre de l'équipe, il a pour but d'optimiser la capacité de production de l'équipe. Pour se faire, le scrum master aide l'équipe à travailler de façon autonome tout en s'améliorant d'avantage.
  - L'**équipe opérationnelle** (qui regroupe idéalement moins de dix personnes): la particularité d'une équipe scrum est qu'elle est dépourvue de toute hiérarchie interne. Une équipe scrum est auto-organisée.





Modèle Scrum (3/3)

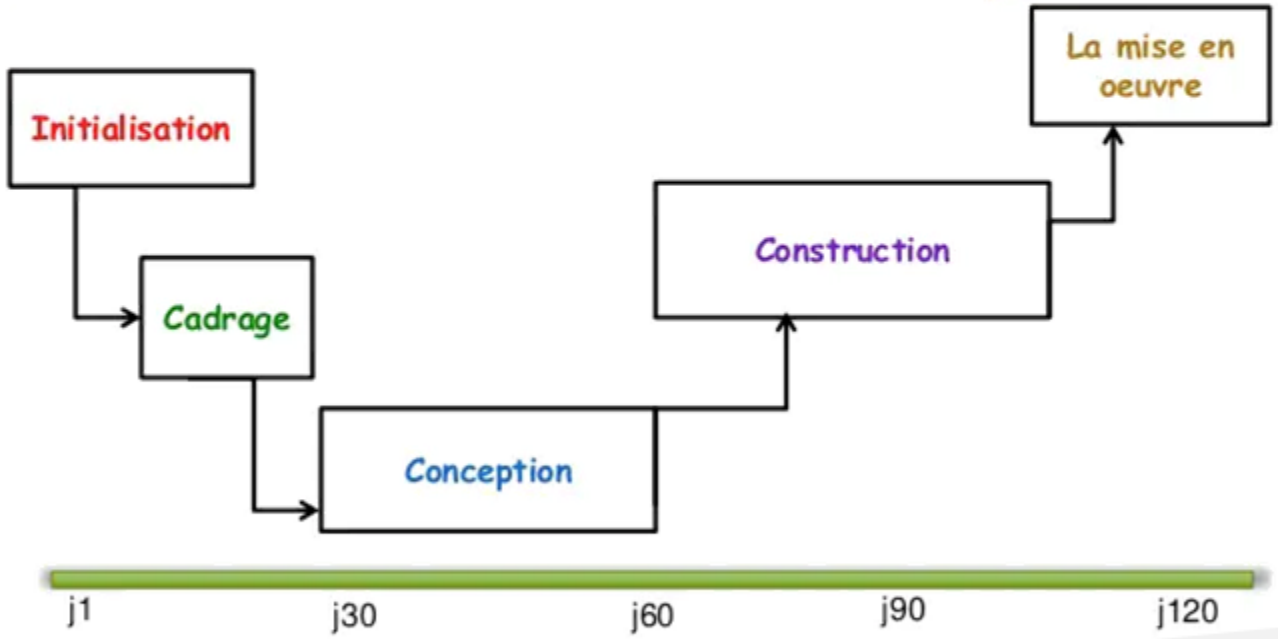
- Autres éléments de Scrum:
  - Le **product backlog** (carnet du produit): ce document contient les exigences initiales dressées puis hiérarchisées avec le client en début de projet. Néanmoins il va évoluer tout au long de la durée du projet, en fonction des divers besoins du client.
  - Le **sprint backlog** (carnet de sprint) : en chaque début de sprint, l'équipe définit un but. Puis lors de la réunion de sprint, l'équipe de développement choisit les éléments du carnet à réaliser. L'ensemble de ces éléments constitue alors le sprint backlog.
  - **User story** : ce terme désigne les fonctionnalités décrites par le client.
  - La **mêlée** (scrum) : c'est une réunion d'avancement organisée de manière quotidienne durant le sprint.



La mêlée

Le modèle RAD (Rapid Application Development)

- Le ***modèle RAD*** associe plusieurs techniques et outils afin d'optimiser les délais de développement à objectif de qualité donnée. Le but est de livrer rapidement un minimum de fonctions opérationnelles.



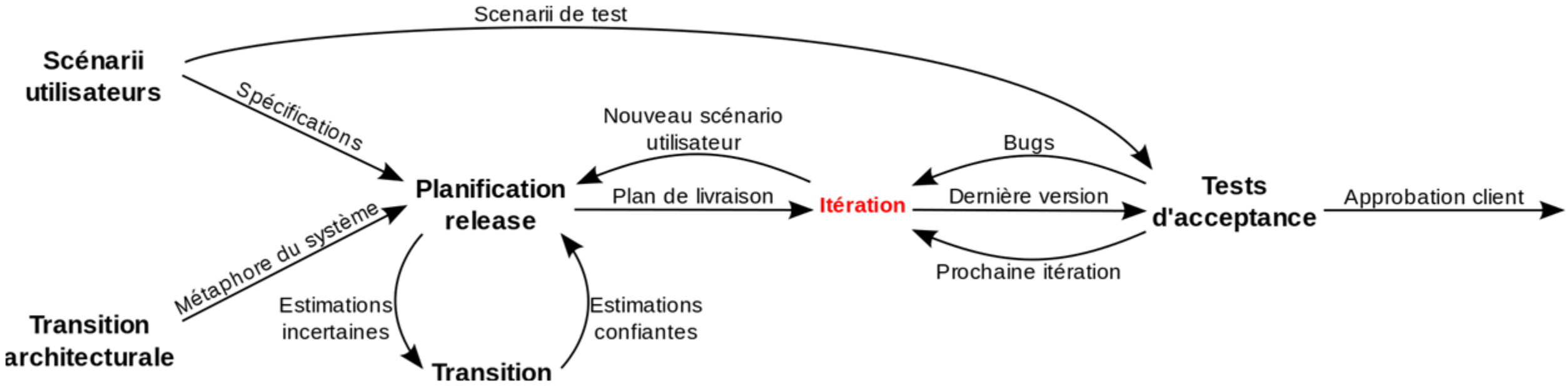
1. La préparation: définit l'organisation du travail et le plan de communication
2. Le cadrage: définit les objectifs, les solutions et les moyens à mettre en œuvre
3. Le prototypage: modélise la solution et valide sa cohérence par l'intermédiaire de prototypes
4. Le développement: réalise la solution
5. La finalisation: constitue une vérification de qualité sur site pilote

Le modèle RAD (Rapid Application Development)

- Il se base sur les principes suivants:
  - **Utilisation de prototypes** (maquette)
  - **Démarche participative de tous les intervenants du projet**
  - **Intégration des outils dans la démarche**
  - **Priorité aux délais**

Le modèle XP (eXtreme Programming)

- Le ***modèle XP*** s'adapte aux petites équipes travaillant dans un contexte changeant.



1. Une phase d'exploration détermine les scénarios « client » qui seront fournis pendant cette itération ;
2. L'équipe transforme les scénarios en tâches à réaliser et en tests fonctionnels ;
3. Chaque développeur s'attribue des tâches et les réalise avec un binôme ;
4. Lorsque le produit satisfait tous les tests fonctionnels, il est livré.

Le modèle XP (eXtreme Programming)

- Il est basé sur quatre principes relativement simples: communication, simplicité, retour d'expérience et livraison.
  - **Communication**: Le programmeur s'adresse directement aux utilisateurs pour permettre à chacun de poser ses questions et de partager l'information. La « présence » des utilisateurs dans le déroulement du développement réduit les délais.
  - **Simplicité**: Il s'agit de choisir la solution la plus simple pouvant satisfaire l'utilisateur. En effet, il coûte moins cher de développer une fonctionnalité simple que de développer un système complexe.
  - **Retour d'expérience:** L'utilisateur est impliqué dans les tests dés le premier jour. Cela permet de repérer et de corriger rapidement les erreurs, mais aussi de donner une vision permanente de l'état d'avancement du projet.
  - **Livraison**: La livraison a lieu dés que possible, en fonction du planning préalablement réalisé avec l'utilisateur, qui aura clairement exprimé ses besoins et priorisé les livraisons.

Langage de Modélisation Unifié

Conception d'une architecture logicielle

- L'UML (Unified Modeling Language) est un langage de modélisation graphique à base de pictogrammes conçu pour la conception orientée objet.
- En UML 2.5, les diagrammes sont représentés sous deux types de vues :
  - ***Statique*** ou ***structurelle*** du domaine avec les diagramme de structure (Structure Diagrams).
  - ***Dynamique*** avec les ***diagrammes de comportement*** (Behavior Diagrams) et les ***diagrammes d'interactions*** (Interaction Diagrams).
- Les diagrammes sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie. Il en existe quatorze depuis UML 2.3.

Diagrammes de structure (1/2)

- Les diagrammes de structure (structure diagrams) ou diagrammes statiques (static diagrams) rassemblent :
  - ***Diagramme de classes (class diagram)*** : représentation des classes intervenant dans le système.
  - ***Diagramme d'objets (object diagram)*** : représentation des instances de classes (objets) utilisées dans le système.
  - ***Diagramme de composants (component diagram)*** : représentation des composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données…)
  - ***Diagramme de déploiement (deployment diagram)*** : représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage…) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.

Diagrammes de structure (2/2)

- ***Diagramme des paquets (package diagram)*** : représentation des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions.

- ***Diagramme de structure composite (composite structure diagram)*** : représentation sous forme de boîte blanche des relations entre composants d'une classe (depuis UML 2.x).

- ***Diagramme de profils (profile diagram)*** : spécialisation et personnalisation pour un domaine particulier d'un meta-modèle de référence d'UML (depuis UML 2.2).

Diagrammes de comportement

- Les diagrammes de comportement (behavior diagrams) rassemblent :
  - ***Diagramme des cas d'utilisation (use-case diagram)*** : représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.
  - ***Diagramme états-transitions (state machine diagram)*** : représentation sous forme de machine à états finis du comportement du système ou de ses composants.
  - ***Diagramme d'activité (activity diagram)*** : représentation sous forme de flux ou d'enchaînement d'activités du comportement du système ou de ses composants.

Diagrammes d'interaction

- Les diagrammes d'interaction (interaction diagrams) ou diagrammes dynamiques (dynamic diagrams) rassemblent :
  - ***Diagramme de séquence (sequence diagram)*** : représentation de façon séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.
  - ***Diagramme de communication (communication diagram)*** : représentation de façon simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets (depuis UML 2.x).
  - ***Diagramme global d'interaction (interaction overview diagram)*** : représentation des enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité) (depuis UML 2.x).
  - ***Diagramme de temps (timing diagram)*** : représentation des variations d'une donnée au cours du temps (depuis UML 2.3).

Conception architecturale

Diagramme de classes (class diagram)

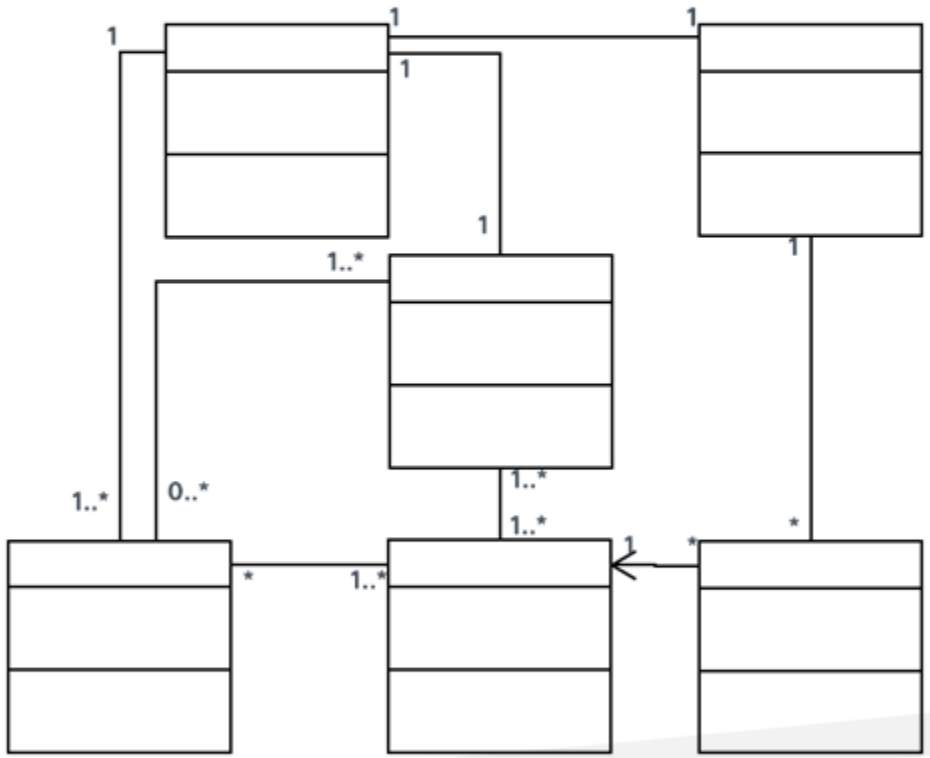
- Un **diagramme de classes** est un diagramme utilisé dans **la conception et la modélisation de logiciels** pour **décrire les classes** et leurs relations.

- Le **diagramme de classes** permet de modéliser des logiciels dans un **haut niveau d'abstraction** et sans avoir à regarder le code source; et est composé de trois sections :

  - **Section supérieure** : contient le nom de la classe. Cette section est toujours nécessaire, que vous parliez du classifieur ou d'un objet.

  - **Section intermédiaire** : contient les attributs de la classe. Utilisez-la pour décrire les qualités de la classe. Elle n'est nécessaire que lors de la description d'une instance spécifique d'une classe.

  - **Section inférieure** : contient les opérations de la classe (méthodes), affichées sous forme de liste. Chaque opération occupe sa propre ligne. Les opérations décrivent la manière dont une classe interagit avec les données.

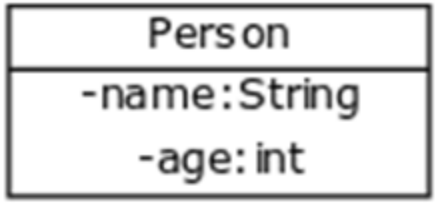


Eléments

Caractère	Rôle	Mot clé	Description
+	accès public	public	Toutes les autres classes ont accès à cet attribut.
#	accès protégé	protected	Seules la classe elle-même et les classes filles (héritage) ont accès à cet attribut.
~	accès package	package	Classe visible uniquement dans le package.
-	accès privé	private	Seule la classe elle-même a accès à cet attribut.

- Une ***classe*** représente un objet ou un ensemble d'objets possédant une structure et un comportement communs. On les représente par un rectangle comprenant des lignes pour le nom de la classe, ses attributs et ses opérations.
  - ***Nom*** : première ligne d'une forme de classe.
    - Dans une classe classique, le nom est écrit en romain (exemple : « ClasseClassique »).
    - Le nom des classes abstraites est écrit en italique (exemple : ClasseAbstraite).
  - ***Attributs*** : deuxième ligne d'une forme de classe. Chaque attribut de la classe apparaît sur une ligne distincte.
  - ***Méthodes*** : troisième ligne d'une forme de classe. On les appelle aussi opérations ; elles apparaissent sous forme de liste, chaque opération occupant une ligne différente.
- Dans une ***classe***, seule la ligne supérieure est obligatoire, les autres sont facultatives et ne servent qu'à fournir des détails supplémentaires.
- Les classes template ont, dans leur angle supérieur droit, un rectangle dont la bordure est en pointillé et qui contient les types des paramètres.
- ***Modificateurs d'accès des membres*** : Toutes les classes ont des niveaux d'accès différents, en fonction du modificateur d'accès (indicateur de visibilité), tel que : Public(+), Privé(-), Protégé(#)…
  - Afin de respecter le principe fondamental d'***encapsulation***, tous les attributs devraient être ***privés***.
  - Pour qu'un attribut privé ou protégé soit récupérable, on utilise en général ***un getter*** (ou accesseur); pour qu'il soit modifiable, on utilise en général ***un setter*** (ou mutateur).

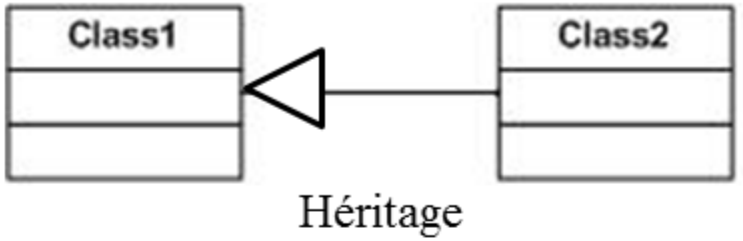
Eléments




```
public class Person {
private String name;
private int age; }
```

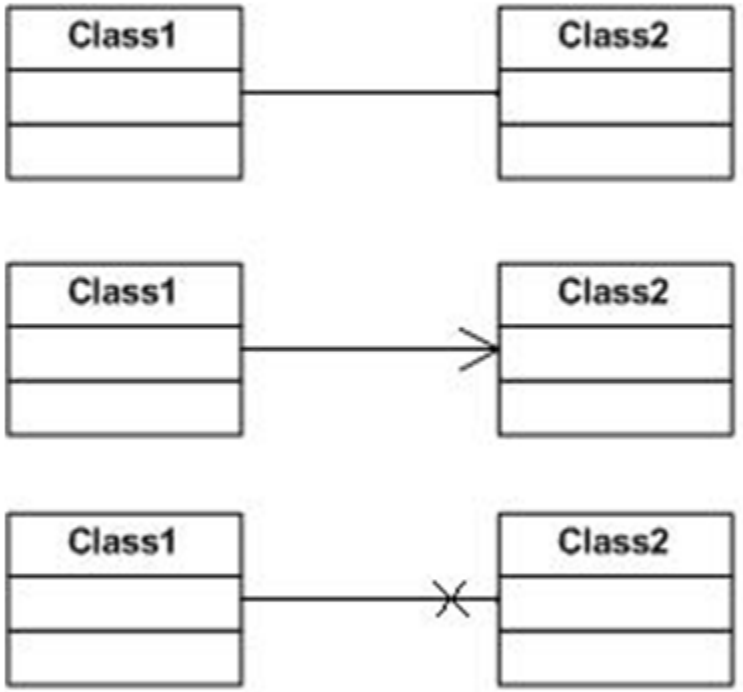
- ***Signaux*** : symboles qui représentent les communications à sens unique et asynchrones entre les objets actifs.
- ***Types de données*** : classifieurs qui définissent des valeurs de données. Les types de données peuvent modéliser les types primitifs et les énumérations.
- ***Interfaces*** : groupe de signatures d'opération et/ou de définitions d'attributs définissant un ensemble cohérent de comportements. Les interfaces sont semblables à des classes, sauf qu'une classe peut avoir une instance de son type et qu'une interface doit compter au moins une classe pour la mettre en œuvre.
- ***Énumérations*** : représentations de types de données définis par l'utilisateur. Une énumération comprend des groupes d'identificateurs qui représentent des valeurs de l'énumération.
- ***Objets*** : instances d'une ou plusieurs classes. On peut ajouter des objets à un diagramme de classes pour représenter des instances concrètes ou prototypiques.

Interactions



Héritage

- ***Héritage*** : connu sous le nom de généralisation, il s'agit du processus par lequel une sous-classe adopte la fonctionnalité d'une super-classe. On le symbolise par une ligne de connexion droite avec une pointe de flèche fermée orientée vers la super-classe.



- ***Association bidirectionnelle*** : relation par défaut entre deux classes. Chacune des deux classes a conscience de l'existence de l'autre et de sa relation avec elle. Cette association est représentée par une ligne droite entre deux classes.

- ***Association unidirectionnelle*** : relation un peu moins courante entre deux classes. Une classe a conscience de l'existence de l'autre et interagit avec elle. Une association unidirectionnelle est représentée par une ligne de connexion droite avec une pointe de flèche ouverte.

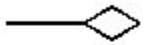
Navigabilité des associations :

1- Bidirectionnelle

2- Mono-directionnelle

3- interdit une association.

Interactions

- ***Agrégation*** 

  L'agrégation est une association avec relation de subordination, représentée par un trait reliant les deux classes et dont l'origine se distingue de l'autre extrémité (la classe subordonnée) par un losange vide. Une des classes regroupe d'autres classes.

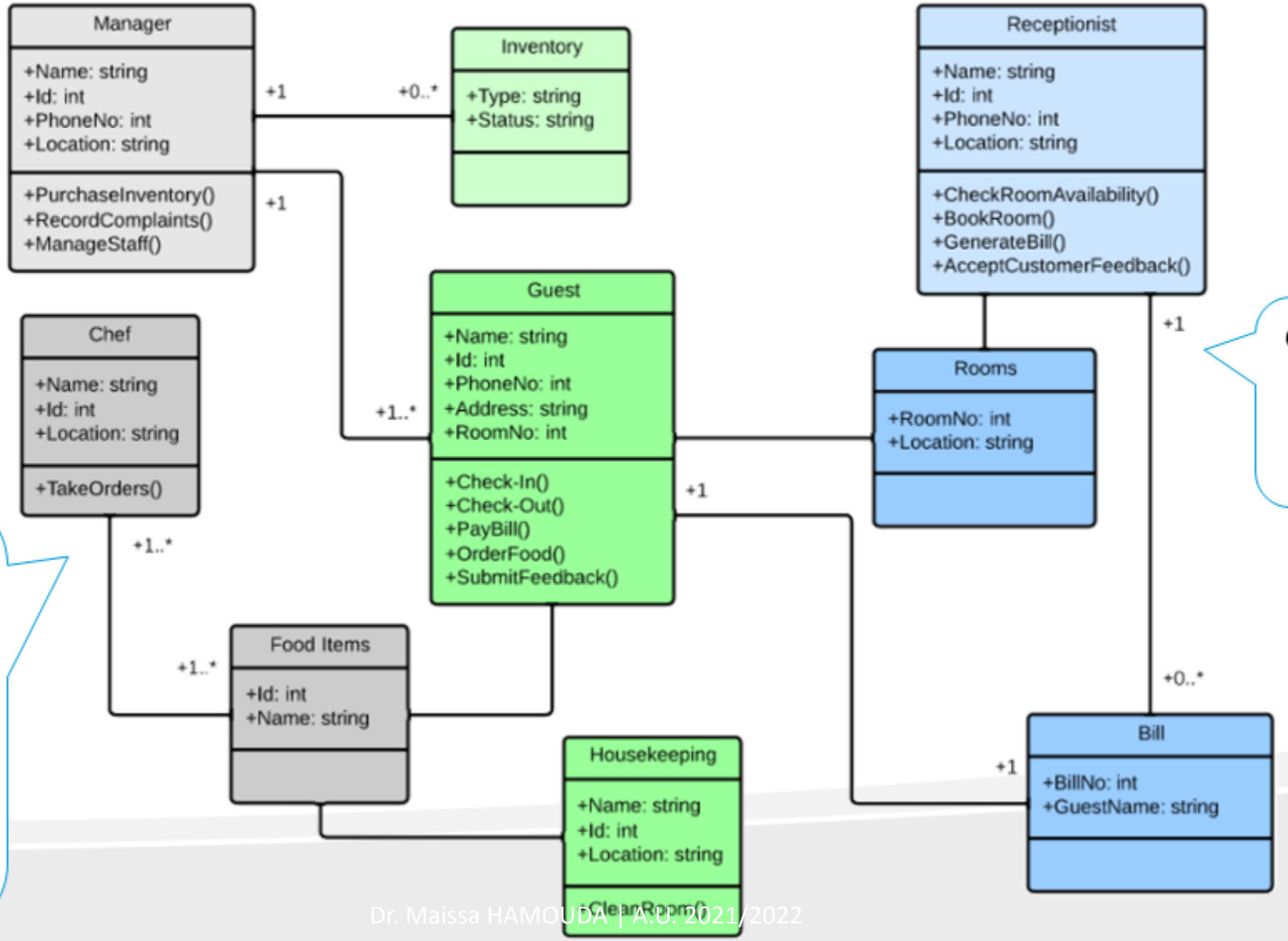
- ***Composition*** 

  La composition est une agrégation avec cycle de vie dépendant : la classe composée est détruite lorsque la classe mère disparait. L'origine de cette association est représentée par un losange plein.

- ***Dépendance*** 

  La dépendance implique qu'une ou plusieurs méthodes reçoivent un objet d'un type d'une autre classe. Il n'y a pas de liaison en ce qui concerne la destruction d'objets mais une dépendance est quand même là. Elle est symbolisée par une flèche en pointillés, dont son extrémité possède trois traits qui se coupent en un même point.

Diagramme de classes d'un système de gestion hôtelière

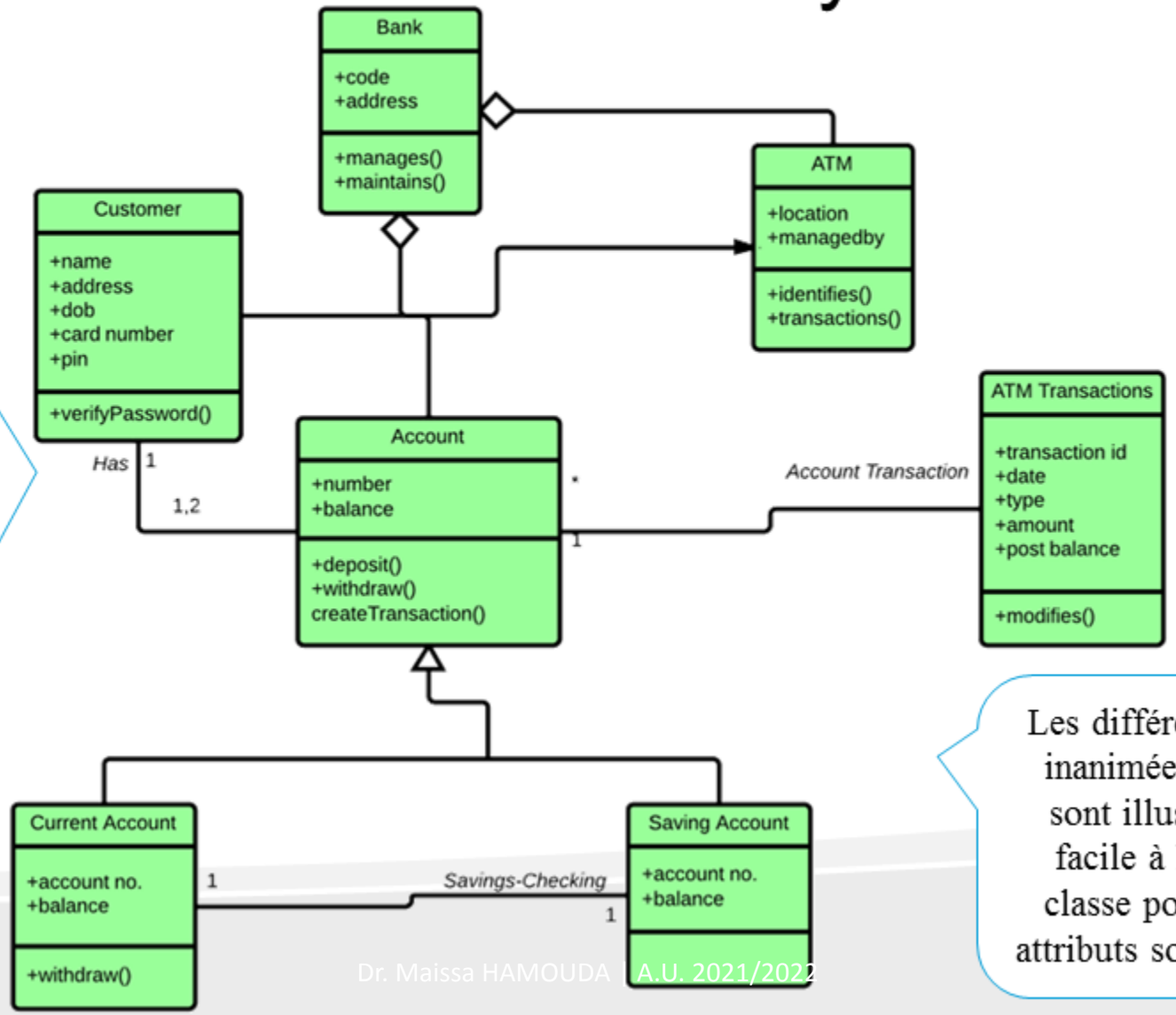


Un diagramme de classes peut montrer les relations entre chaque objet dans un système de gestion hôtelière, y compris les informations des clients, les tâches du personnel et l'occupation des chambres.

Cet exemple donne un aperçu utile du système de gestion hôtelière.

Diagramme de classes d'un système de DAB

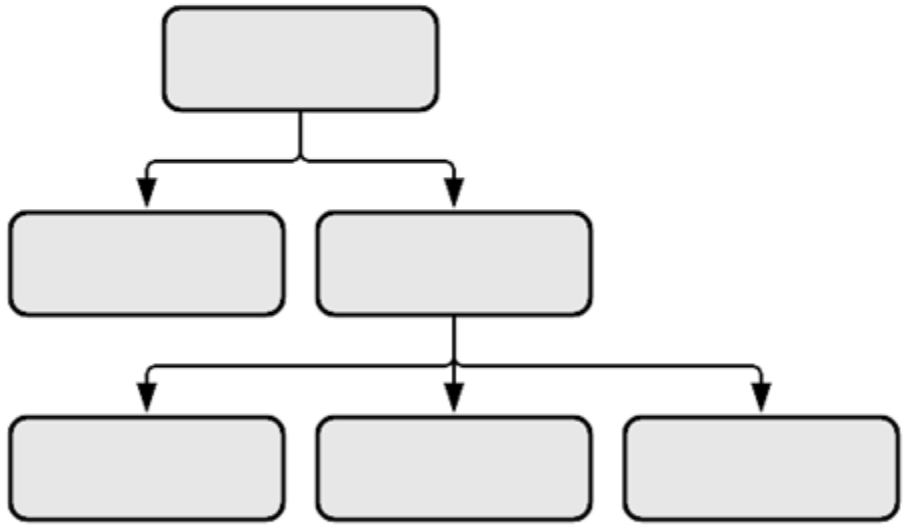
Les distributeurs automatiques de billets (DAB) sont d'une simplicité trompeuse : même s'il suffit aux clients d'appuyer sur quelques boutons pour obtenir de l'argent, un DAB sécurisé et efficace doit traverser de nombreuses couches de sécurité pour éviter les fraudes et offrir une réelle valeur ajoutée aux clients de la banque.



Les différentes parties humaines et inanimées d'un système de DAB sont illustrées par ce diagramme facile à lire, dans lequel chaque classe possède son titre, et où les attributs sont répertoriés au-dessous.

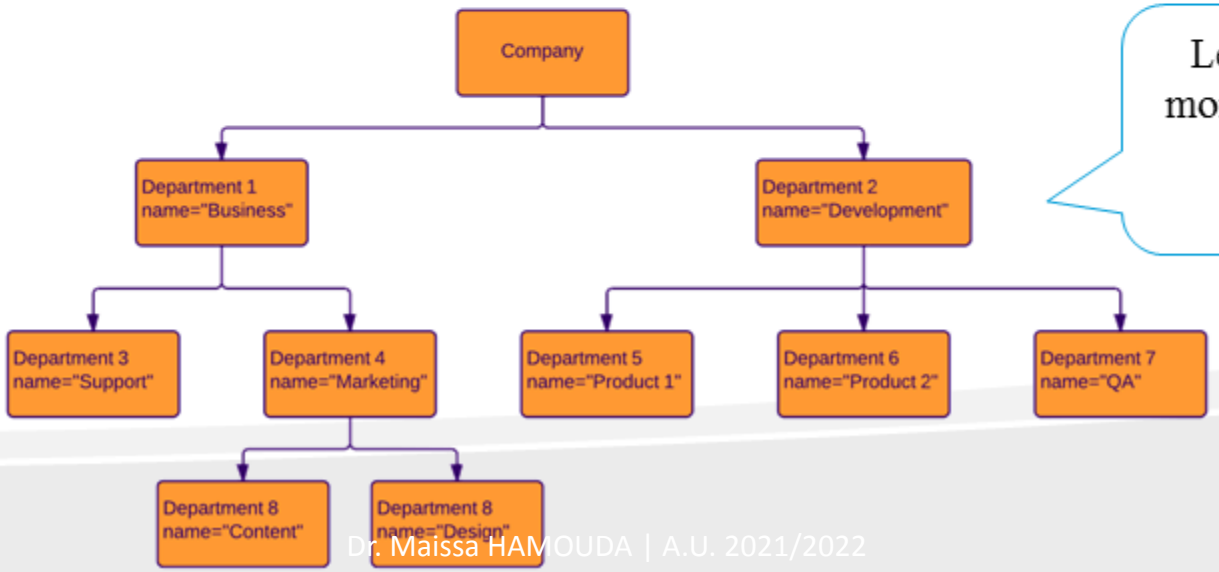
Diagramme d'objets (object diagram)

- Un diagramme d'objets UML représente une instance spécifique d'un diagramme de classes à un moment précis.
- Dans sa représentation visuelle, il est très similaire à un diagramme de classes.
- Un diagramme d'objets se concentre sur les attributs d'un ensemble d'objets et sur la façon dont ils interagissent les uns avec les autres.



Eléments

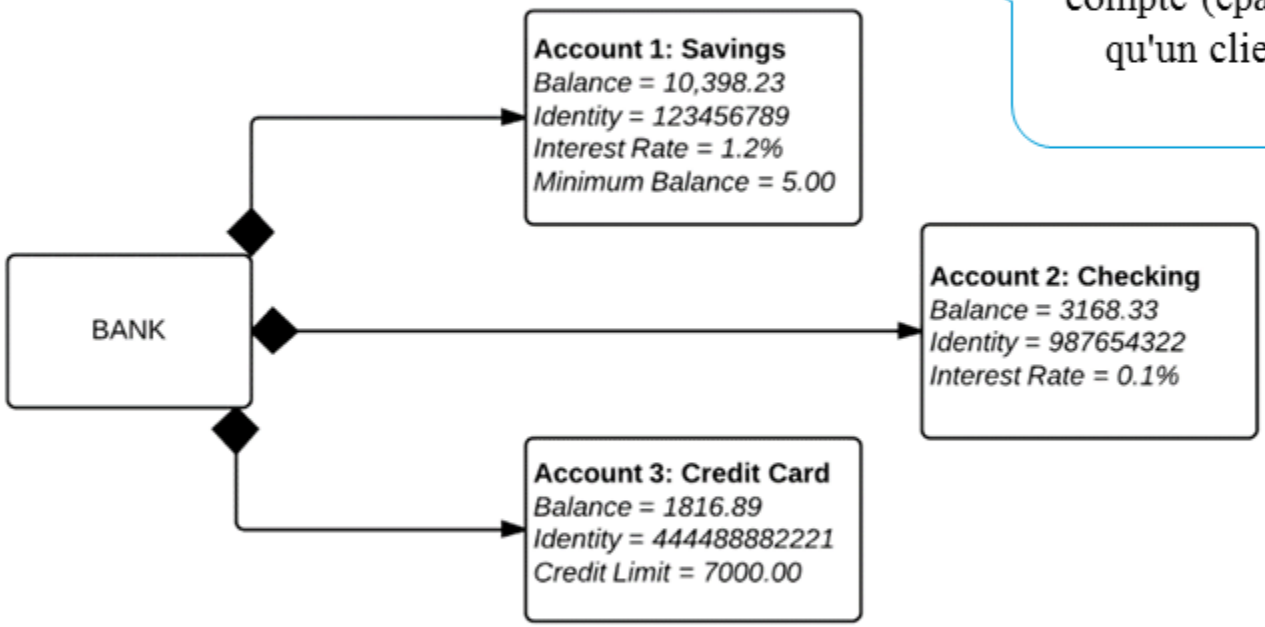
- **Objets :** Les objets désignent les instances d'une classe. Par exemple, si « voiture » est une classe, « Nissan » est un objet de cette classe.
- **Titres de classe :** Les titres de classes sont les attributs spécifiques d'une classe donnée.
- **Attributs de classe :** Les attributs de classe sont représentés par un rectangle avec deux onglets qui indiquent un élément de logiciel.
- **Liens :** Les liens correspondent aux lignes qui relient deux formes d'un diagramme d'objets.



Le diagramme d'objets d'entreprise montre la façon dont les services sont reliés, dans le style d'un organigramme classique.

Exemple de comptes en banque

Les titres de classes indiquent le type de compte (épargne, courant ou carte de crédit) qu'un client donné peut avoir dans cette banque.



Dans ce diagramme d'objets, les trois comptes en banque sont reliés à la banque elle-même.

Les attributs de classes sont différents pour chaque type de compte. Ainsi, l'objet carte de crédit dispose d'une limite de crédit, alors que les comptes d'épargne et courant disposent de taux d'intérêt.

Diagramme de composants (component diagram)

- Un diagramme de composants a pour objectif d'illustrer la relation entre les différents composants d'un système.
- Dans le cadre de l'UML 2.0, le terme « composant » fait référence à un module de classes qui représentent des systèmes ou des sous-systèmes indépendants ayant la capacité de s'interfacer avec le reste du système.

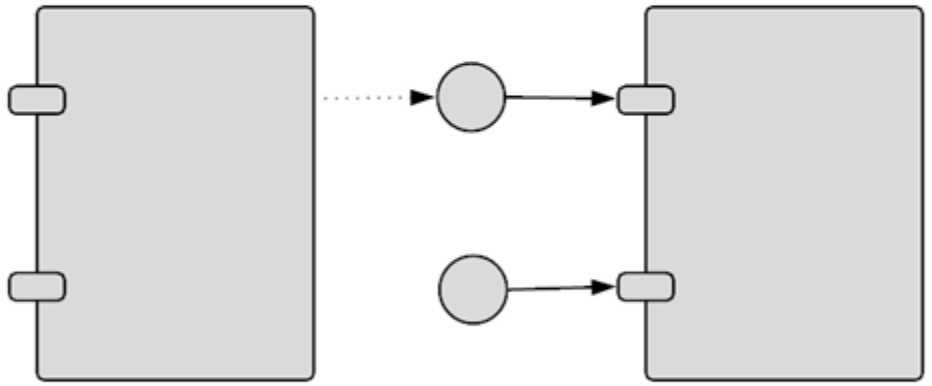
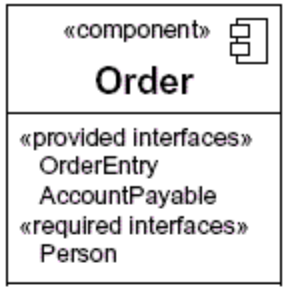


Diagramme de composants (component diagram)



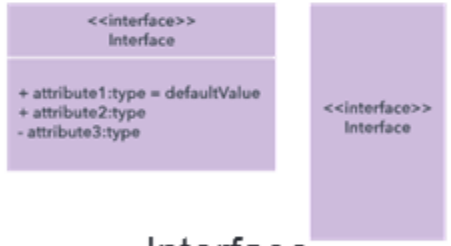
- Dans les diagrammes de composants, UML prévoit que les composants et les packages (ou paquetages, en français) soient reliés les uns aux autres par des lignes représentant les connecteurs d'assemblage et les connecteurs de délégation.
  - Un **connecteur de délégation** est un connecteur qui relie le contrat externe d'un composant (spécifié par ses ports) à la réalisation de ce comportement par les parties internes du composant.
  - Les **connecteurs d'assemblage** représentent les chemins de communication par le biais desquels vos classificateurs se demandent et se fournissent des services.
- Pour créer un compartiment pour le nom d'un composant, on doit toujours inclure le texte du composant entre des chevrons et/ou le logo du composant. La distinction est importante, car un rectangle indiquant uniquement un nom à l'intérieur est réservé aux classificateurs (éléments de classe).
- Comme pour les classes, les composants disposent également d'un espace facultatif pour énumérer les interfaces, de la même manière que vous ajoutez des attributs et des méthodes à la notation de classe.
- Les interfaces représentent les endroits où les groupes de classes du composant communiquent avec les autres composants du système.

Eléments

- ***Composant*** : Fournit et consomme un comportement par le biais d'interfaces, ainsi que par le biais d'autres composants.
- ***Nœud*** : Représente des objets matériels ou logiciels situés à un niveau supérieur aux composants.
- ***Interface*** : Indique les entrées ou les données qu'un composant reçoit ou fournit. Les interfaces peuvent être représentées par des notes textuelles ou des symboles, tels que des formes de sucette, de douille ouverte et d'articulation.

Composant

Nœud



Interface

Eléments

- ***Port*** : Spécifie un point d'interaction distinct entre un composant et son environnement. Les ports sont symbolisés par un petit carré.
- ***Paquetage*** : Regroupe plusieurs éléments du système. Tout comme les dossiers de fichiers regroupent plusieurs feuilles, les packages peuvent englober plusieurs éléments.
- ***Remarque*** : Permet d'ajouter une méta-analyse sur le diagramme de composants.
- ***Dépendance*** : Indique les relations de dépendance entre les différentes parties de votre système. Les dépendances sont représentées par des lignes pointillées reliant un composant (ou élément) à un autre.

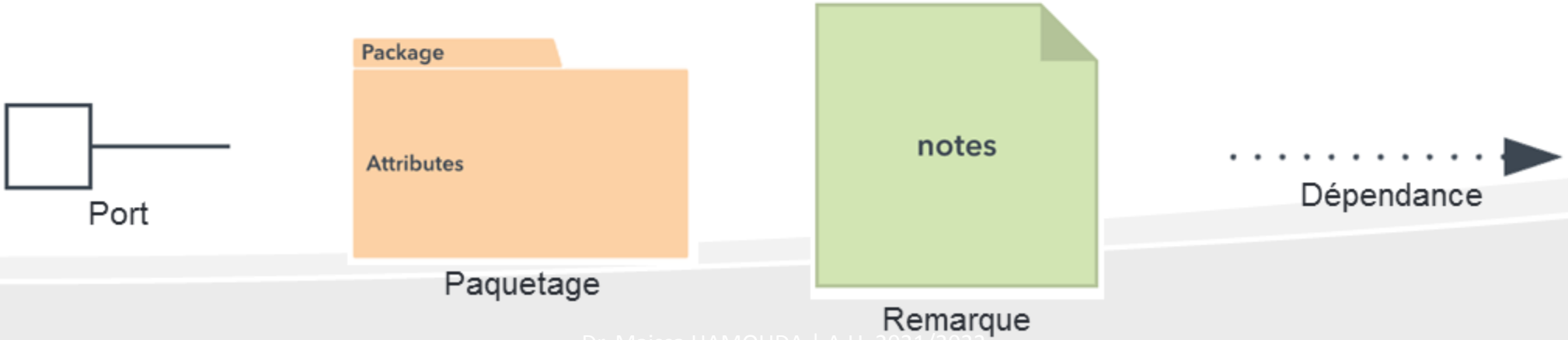
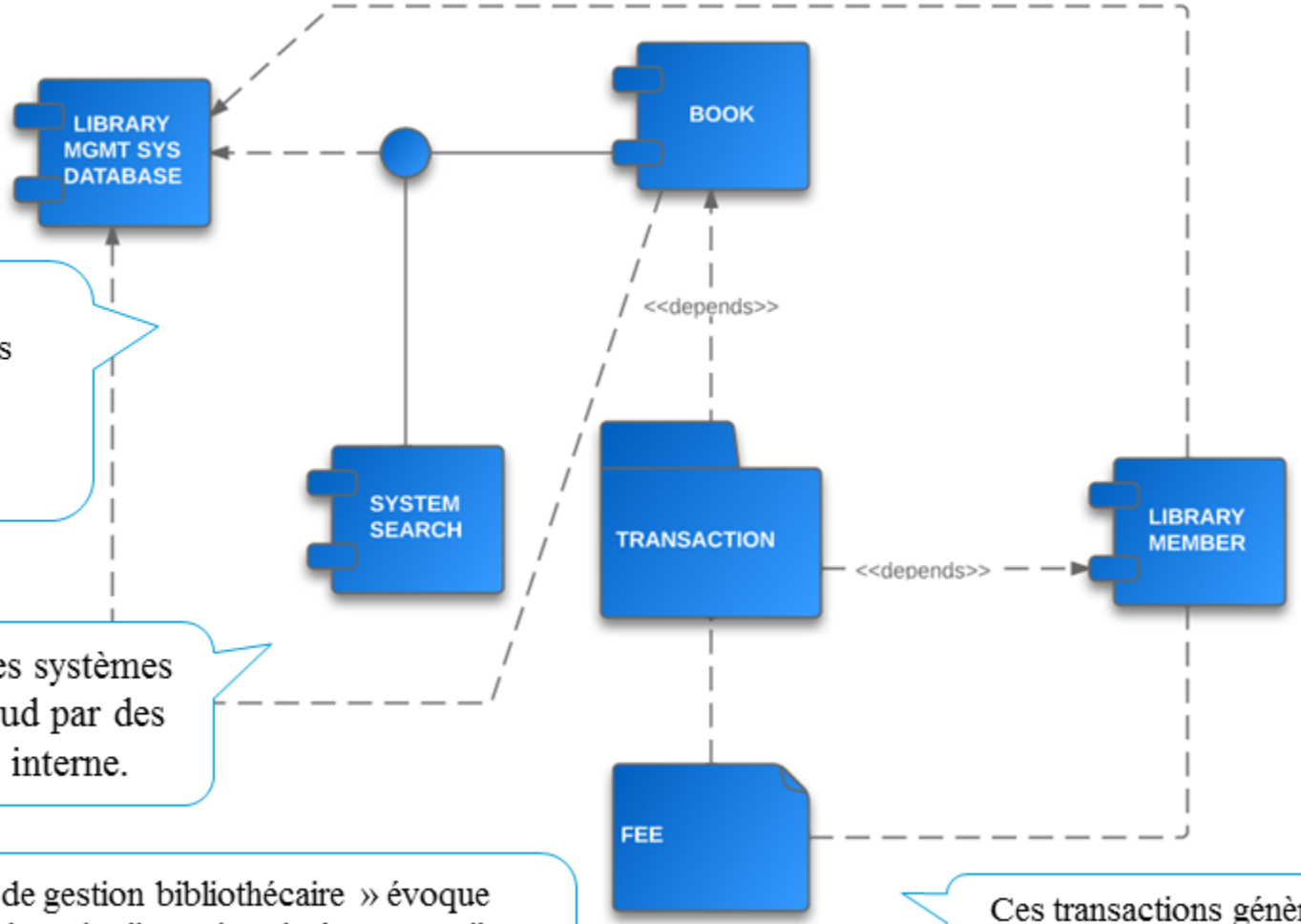


Diagramme de composants UML pour un système de gestion bibliothécaire



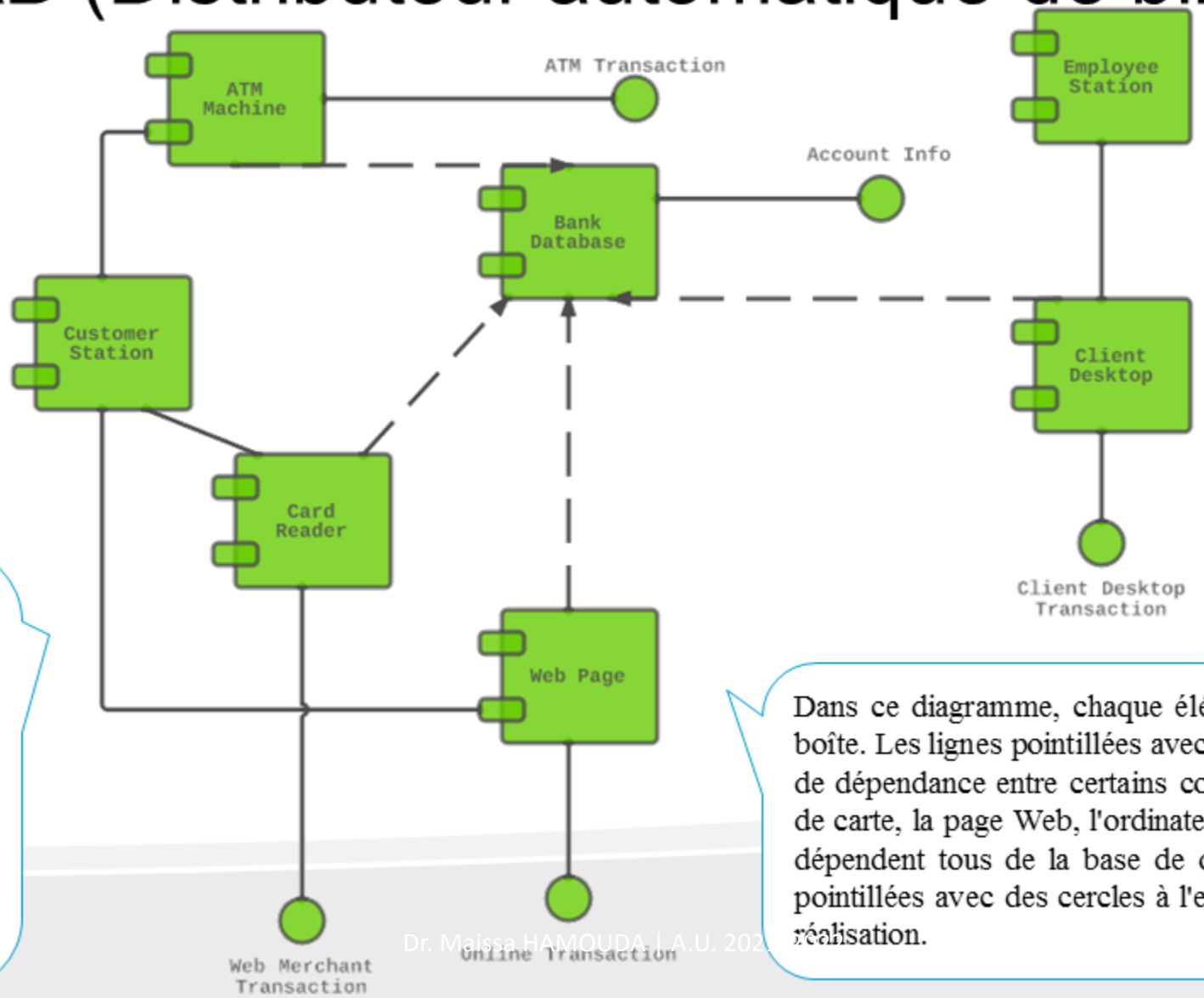
Les systèmes de gestion bibliothécaire ont été parmi les premiers au monde à être principalement gérés par informatique.

Aujourd'hui, beaucoup de ces systèmes sont administrés dans le cloud par des services tiers plutôt qu'en interne.

Bien que le terme « système de gestion bibliothécaire » évoque généralement un moyen de suivre des livres imprimés, ces outils organisent aujourd'hui toutes sortes de données enregistrées et consultées par les utilisateurs.

Ces transactions génèrent un réseau de relations entre les composants du système de gestion bibliothécaire.

Diagramme de composants UML d'un système de DAB (Distributeur automatique de billets)

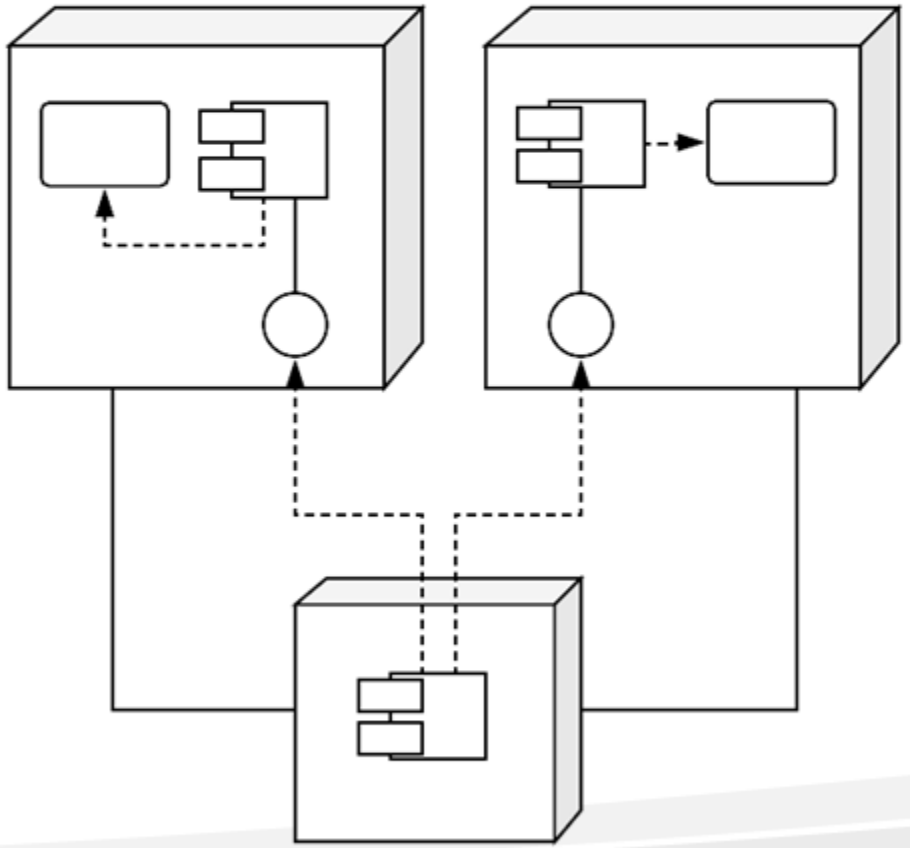


Un diagramme de composants est similaire à un diagramme de classes dans la mesure où il illustre les relations entre les éléments d'un système donné, mais les diagrammes de composants représentent des relations plus complexes et variées que la plupart des diagrammes de classes.

Dans ce diagramme, chaque élément est contenu dans une petite boîte. Les lignes pointillées avec des flèches indiquent les relations de dépendance entre certains composants. Par exemple, le lecteur de carte, la page Web, l'ordinateur du client et le système de DAB dépendent tous de la base de données de la banque. Les lignes pointillées avec des cercles à l'extrémité indiquent une relation de réalisation.

Diagramme de déploiement (deployment diagram)

- Le diagramme de déploiement décrit le déploiement physique des informations générées par le logiciel sur des composants matériels.



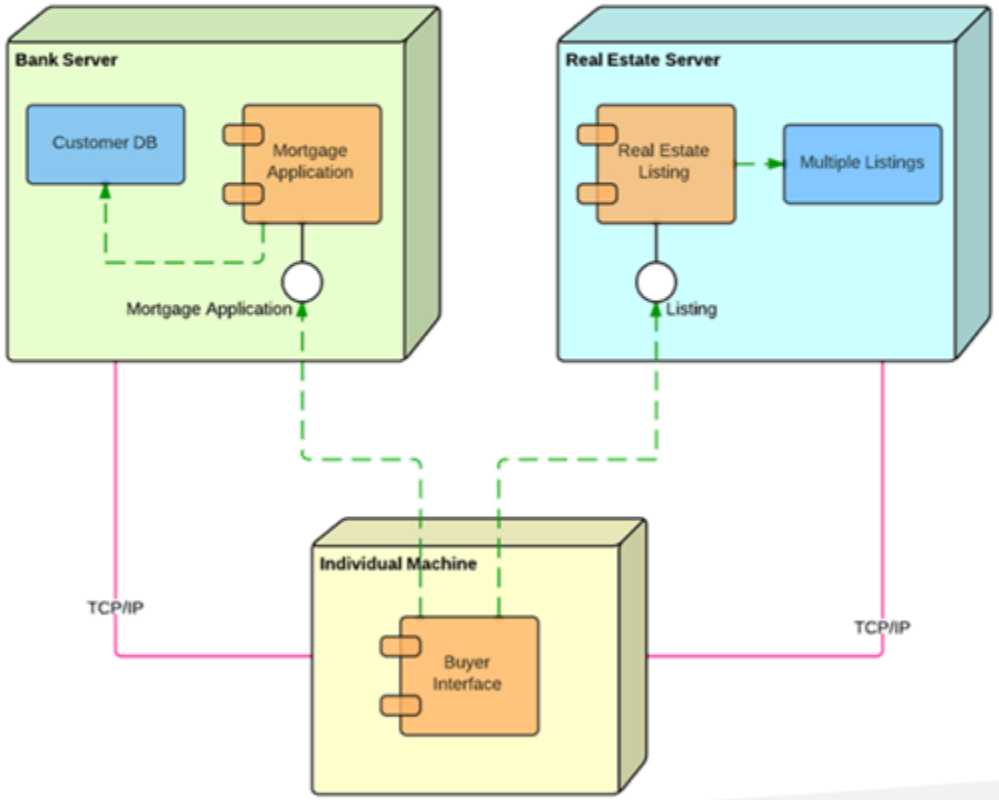
Eléments

- Artefact : produit développé par le logiciel, symbolisé par un rectangle avec le nom et le mot « artefact » entourés de flèches doubles.
- Association : ligne indiquant un message ou tout autre type de communication entre deux nœuds.
- Composant : rectangle avec deux onglets indiquant un élément logiciel.
- Dépendance : ligne en pointillés terminée par une flèche, qui indique qu'un nœud ou composant est dépendant d'un autre.
- Interface : cercle qui indique une relation contractuelle. Les objets qui réalisent l'interface doivent remplir une sorte d'obligation.

Eléments

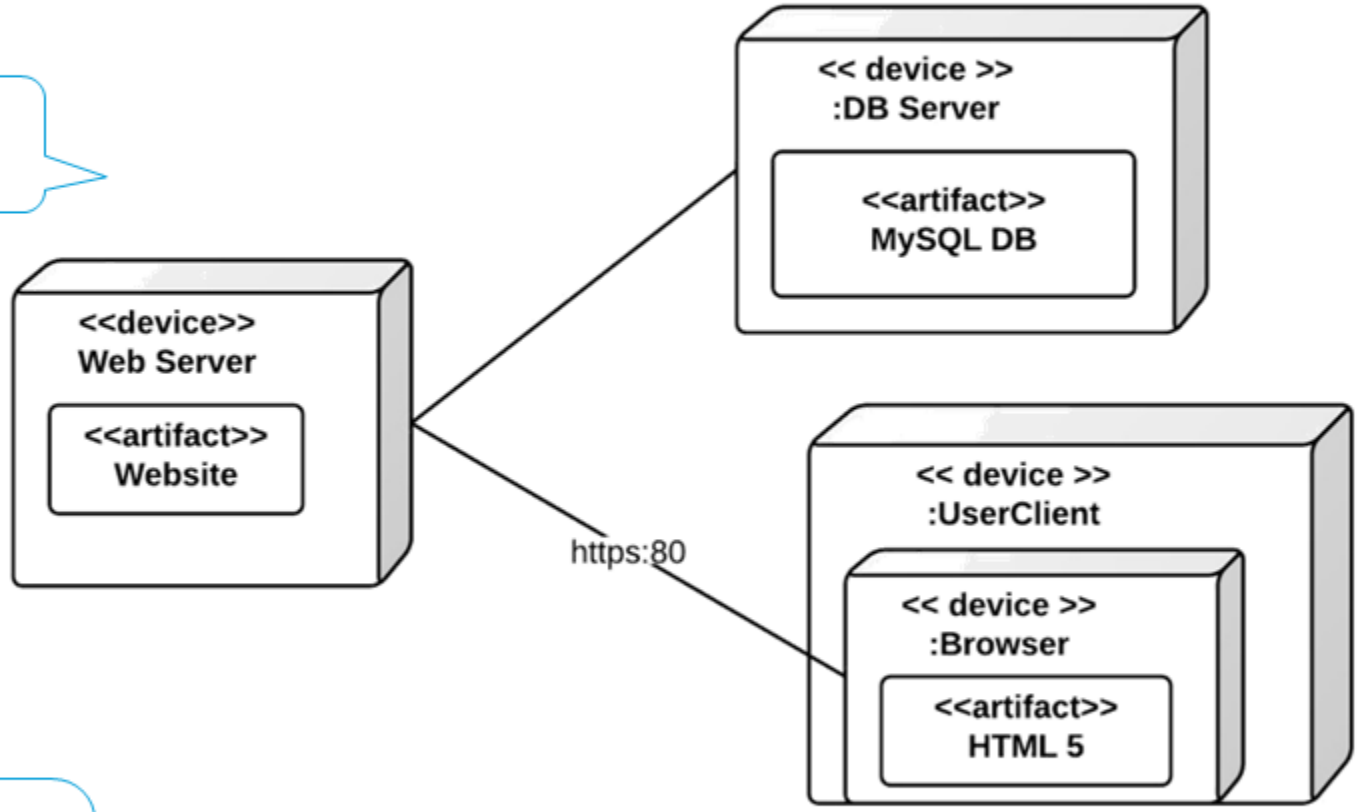
- Nœud : élément matériel ou logiciel représenté par une boîte en relief.
- Nœud conteneur : nœud qui en contient un autre, comme dans l'exemple ci-dessous où les nœuds contiennent des composants.
- Stéréotype : dispositif contenu dans le nœud, présenté dans la partie supérieure du nœud et dont le nom est entouré de flèches doubles.
- Voie de communication : ligne droite qui représente la communication entre deux nœuds de périphériques.
- Paquetage : une boîte en forme de dossier qui regroupe tous les nœuds de périphériques pour contenir l'intégralité du déploiement.

Exemple de diagramme de déploiement (Bank server)



Exemple de diagramme de déploiement (web server)

Cet exemple montre un diagramme de déploiement.

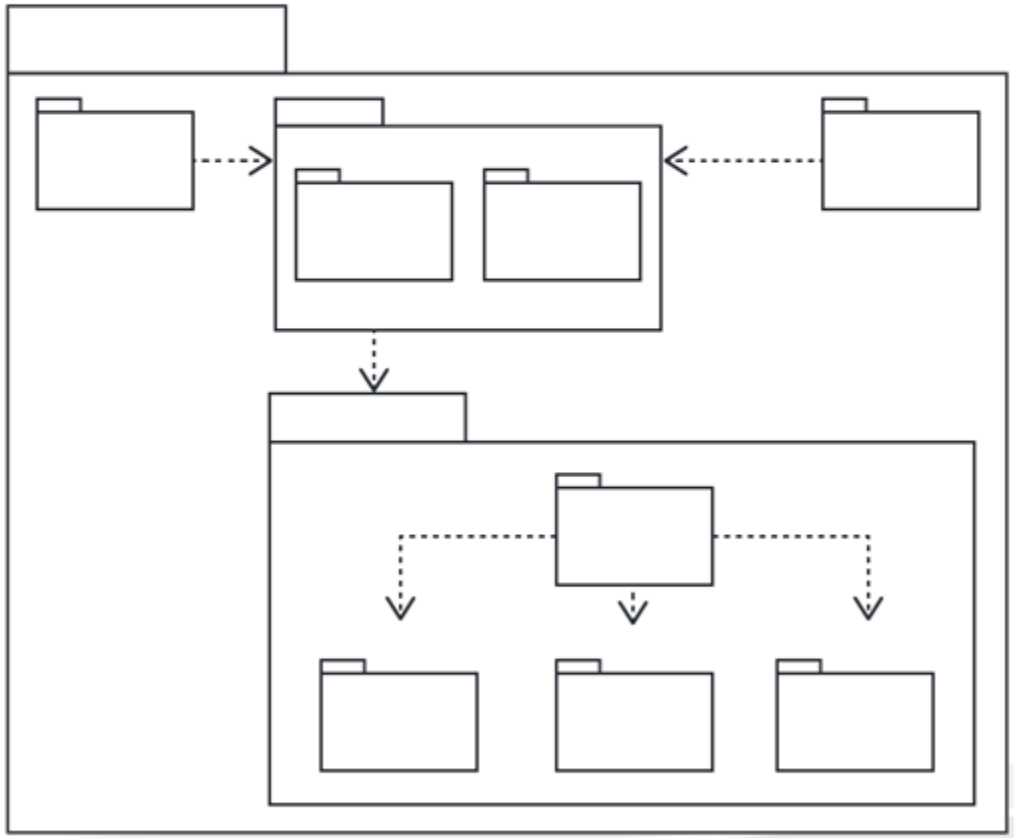


On y voit un serveur web, un serveur de base de données et la machine depuis laquelle l'utilisateur consulte le site web.

On peut complexifier ce diagramme en montrant les différentes parties du serveur web et les interactions entre JavaScript et le client, mais cet exemple nous donne une idée de l'apparence d'un diagramme de déploiement avec la notation UML.

Diagramme des paquets (package diagram)

- Un paquetage est un regroupement d'éléments UML apparentés, tels que des diagrammes, des documents, des classes ou même d'autres paquetages.
- Tous les éléments du diagramme sont imbriqués dans des paquetages, qui sont eux-mêmes représentés sous forme de dossiers de fichiers et organisés de manière hiérarchique.
- Les diagrammes de paquetages sont le plus souvent utilisés pour donner un aperçu visuel de l'architecture en couches d'un classifieur UML, tel qu'un système logiciel.



Eléments

- Paquetage : espace de noms utilisé pour regrouper un ensemble d'éléments liés de manière logique au sein d'un système. Tous les éléments contenus dans un paquetage doivent être empaquetables et porter un nom unique.
- Élément empaquetable : élément nommé, appartenant éventuellement de manière directe à un paquetage. Il peut s'agir d'événements, de composants, de cas d'utilisation et même de paquetages. Les éléments empaquetables peuvent également être représentés sous la forme d'un rectangle à l'intérieur d'un paquetage, marqués avec le nom correspondant.
- Dépendances : représentation visuelle de la façon dont un élément (ou un ensemble d'éléments) dépend d'un autre ou l'influence. Les dépendances sont divisées en deux groupes : les dépendances d'accès et les dépendances d'importation.

Eléments

- Element import : relation dirigée entre un espace de noms d'importation et un élément empaquetable importé. Ce symbole est utilisé pour importer des éléments particuliers sans avoir recours à l'importation de paquetages et sans les rendre publics dans l'espace de noms.
- Package import : relation dirigée entre un espace de noms d'importation et un paquetage importé. Ce type de relation dirigée ajoute le nom des membres du paquetage importé à son propre espace de noms.
- Package merge : relation dirigée dans laquelle le contenu d'un paquetage s'ajoute au contenu d'un autre. En d'autres termes, le contenu de deux paquetages fusionne pour former un nouveau paquetage.

Nomenclature des dépendances dans un diagramme de paquetages

- Les diagrammes de paquetages sont utilisés, en partie, pour représenter les dépendances d'importation et d'accès entre les paquetages, les classes, les composants et les autres éléments nommés de votre système.
- Chacune des dépendances est représentée par une ligne de connexion avec une flèche symbolisant le type de relation entre deux éléments ou plus.
- Il existe deux principaux types de dépendances :
  - **Dépendance d'accès** : indique qu'un paquetage nécessite le soutien d'un autre paquetage. Exemple :

  - **Dépendance d'importation** : indique que la fonctionnalité a été importée d'un paquetage à un autre. Exemple :

Dépendances

- Les dépendances peuvent également être subdivisées dans les catégories suivantes :
  - **Utilisation** : intervient lorsqu'un élément nommé donné en nécessite un autre pour sa définition complète et son déploiement. Exemple : client et fournisseur.
  - **Abstraction** : relie deux éléments représentant le même concept à différents niveaux d'abstraction au sein du système (généralement une relation entre un client et son fournisseur).
  - **Déploiement** : représente le déploiement d'un artefact sur une cible.

Exemple de diagramme de package

- Le modèle suivant présente un diagramme de paquetages modélise les paquetages d'une application Web de commerce électronique de base.

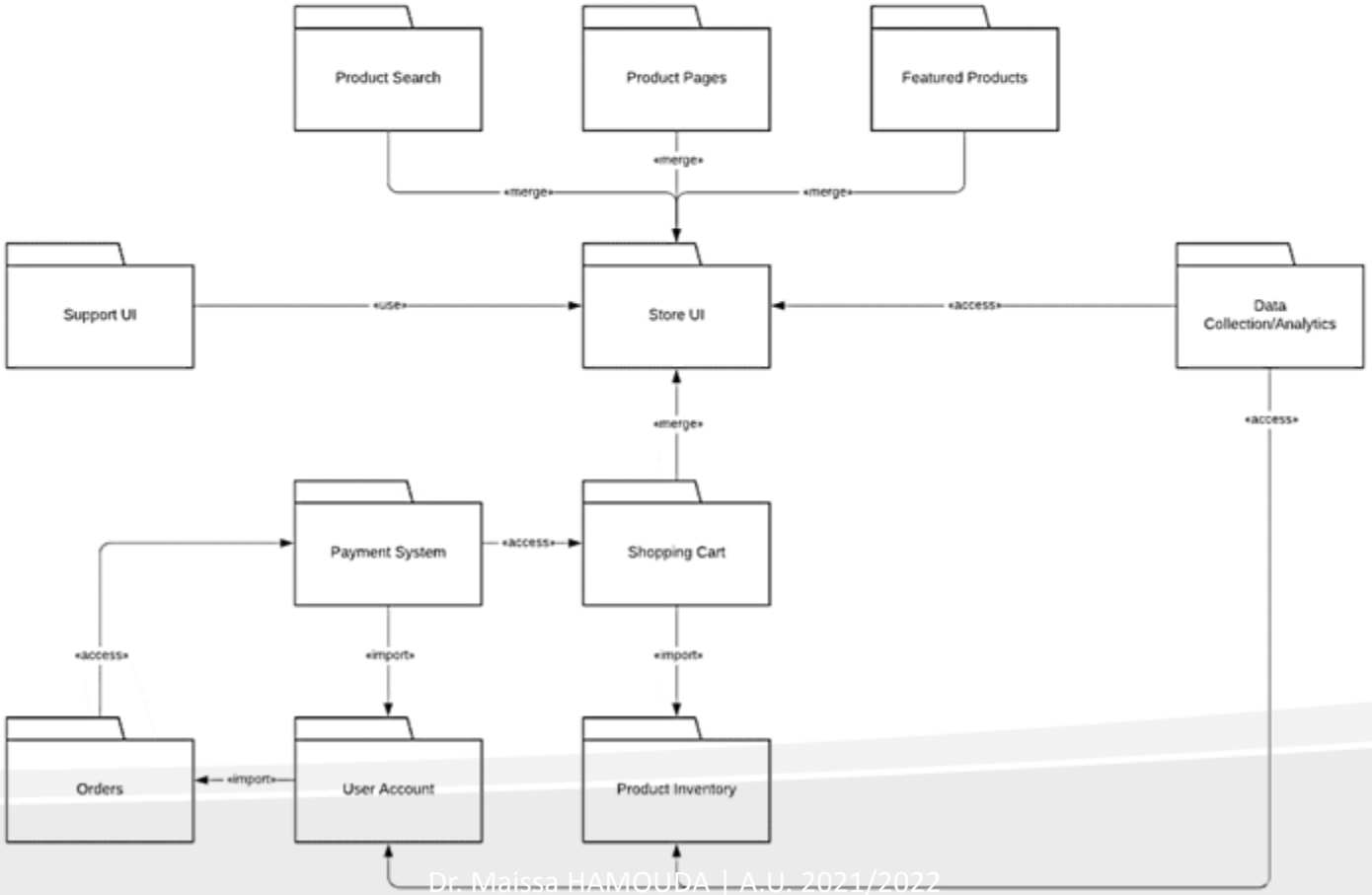
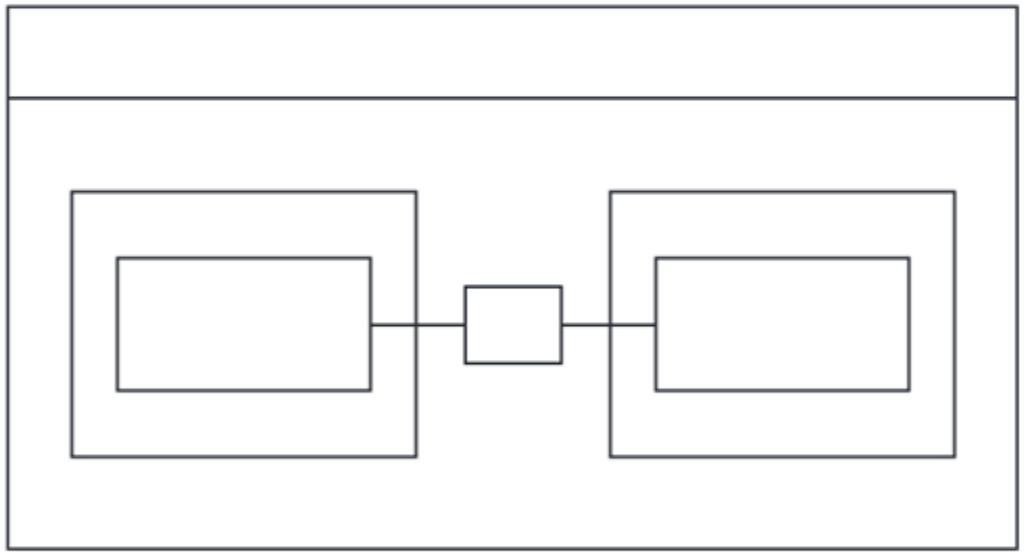
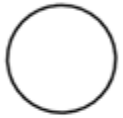


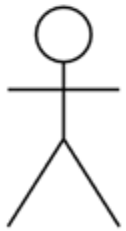
Diagramme de structure composite (composite structure diagram)

- Un diagramme de structure composite est un diagramme structurel UML qui fournit une vue d'ensemble logique de l'ensemble ou d'une partie d'un système logiciel.
- Il permet de visualiser un classifieur structuré donné, en définissant ses classes de configuration, ses interfaces, ses packages et les liens qui les unissent dans le détail.

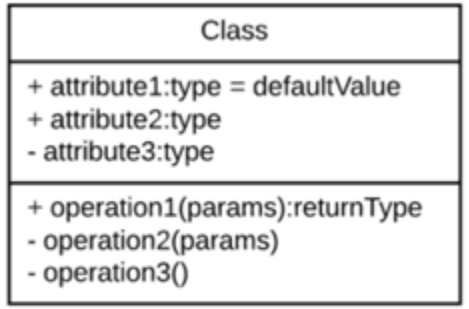


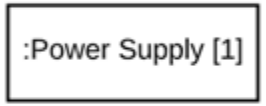
Eléments

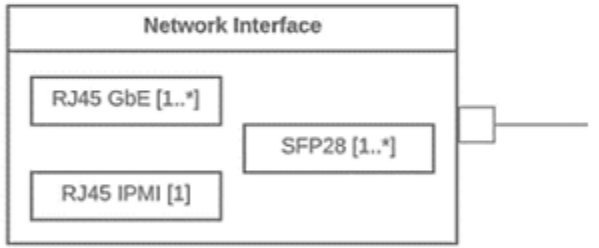




- Terminator : Indique les points de départ et de fin
- Nœud (circulaire) : Indique les événements ou les jalons et contient des chiffres.
- Nœud (rectangulaire) : Indique les événements ou les jalons et contient des chiffres.
- Acteur : Interagit avec le système depuis l'extérieur (p. ex. une personne, un équipement, etc.)
- Classe : Regroupe les objets ayant des propriétés ou des comportements communs (exemple des opérations, des paramètres, des attributs, etc.)
- Partie : Agit comme une instance d'exécution des classes ou des interfaces.
- Port : Agit comme un point d'interaction entre une instance de classifieur (ou son comportement) et son environnement
- Interface : Définit le comportement que l'exécutant accepte de respecter.
- Raccord : Représente les liens de communication entre les parties.







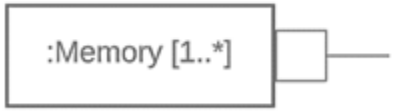
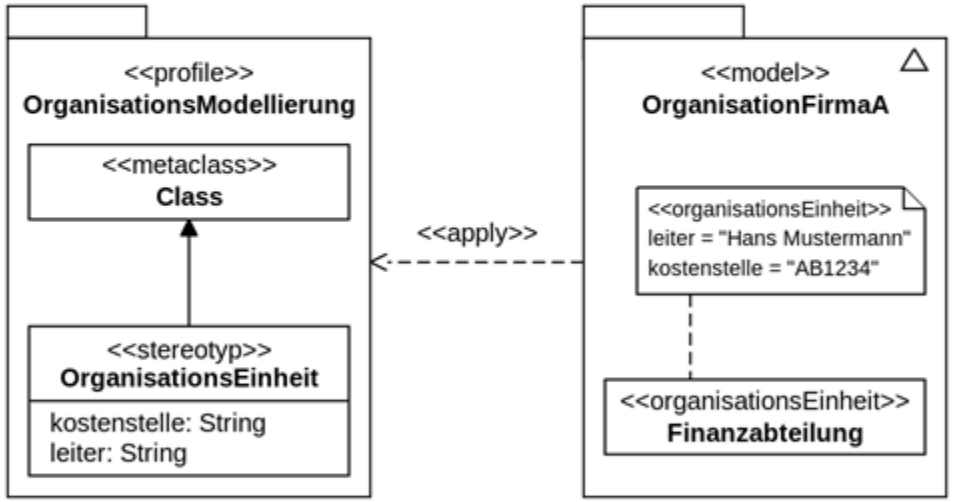


Diagramme de profils (profile diagram)

- Un diagramme de profils est un diagramme de structure permettant l'utilisation de profils pour un métamodèle donné.
- Apparu avec UML 2.2, ce diagramme fournit une représentation des concepts utilisés dans la définition des profils (packages, stéréotypes, application de profils, etc.)



Eléments

- Éléments :
  - Stéréotype (représenté comme une classe avec le stéréotype <<stereotype>>) ;
  - Métaclasse (représentée comme une classe avec le stéréotype <<metaclass>>) ;
  - Profil (représenté comme un package avec le stéréotype <<profile>>).

- Relations entre les éléments :
  - Extension (d'un stéréotype à une métaclasse, représentée comme une flèche pleine) ;
  - Application de profil (d'un profil à un package de métamodèle, représentée comme une flèche pointillée avec le stéréotype <<apply>>) ;
  - Référence (d'un profil à un package ou une métaclasse référencés par le profil, représenté comme une flèche pointillée avec le stéréotype <<reference>>).