Chapitre 2 : L'approche orientée objet

1- Introduction :

La modélisation orientée objet repose sur un ensemble de concepts tirant leur origine du monde des mathématiques entre autres les algèbres et les ensembles.

La modélisation et la conception orientées objets propose une nouvelle façon de penser les problèmes en utilisant des modèles organisés autour des concepts du monde réel. La construction fondamentale est l'objet qui constitue à la fois des structures de données et un comportement dans une seule entité. Les modèles orientés objet sont utiles pour comprendre les problèmes, communiquer avec les experts de l'application, entre rendre les modélisations, préparer la documentation et concevoir les programmes et les bases de données. La modélisation orientée objet repose sur les fondements suivants :

2- Abstraction : C'est la représentation des caractéristiques essentielles d'un objet en omettant les détails non essentiels. L'abstraction peut concerner aussi bien des phénomènes concrets qu'abstraits. Il pourrait bien s'agir de la modélisation d'un système électronique, industriel, logiciel de gestion. Au fait, l'abstraction est une technique que nous utilisons couramment pour décrire des phénomènes de la vie courante.

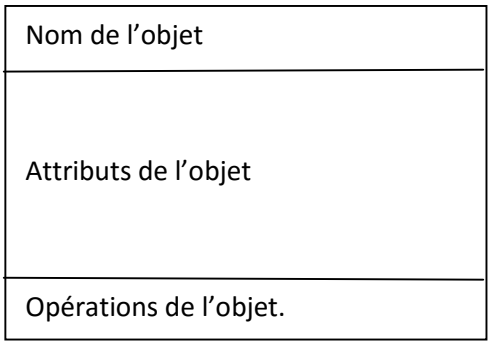
Exemple :- un tableau de peinture représente une abstraction faite par le peintre sur un sujet ou un thème ;

- Une carte géographique est une abstraction d'un territoire ou d'une zone.
- Un fichier comportant les notes des étudiants est une abstraction de leur examinassions.

3- Objet :

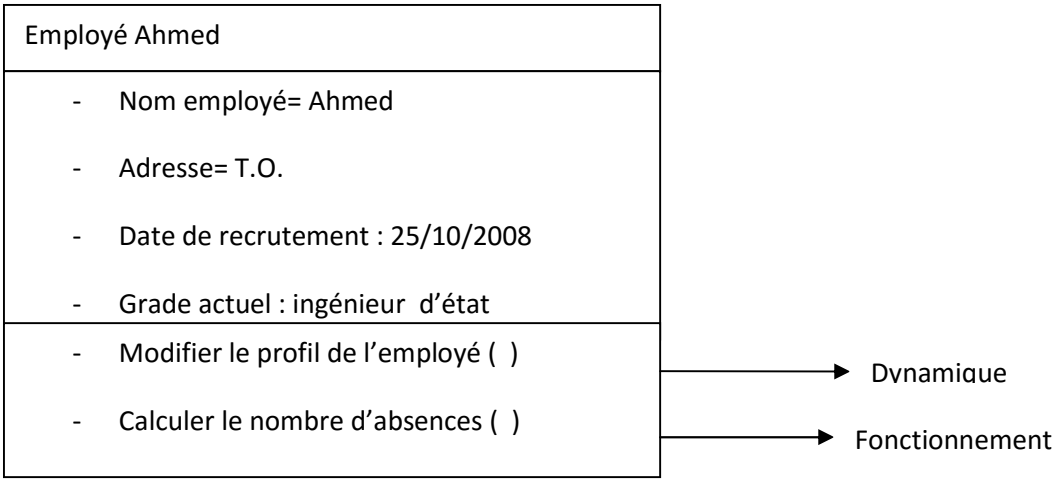
Dans la modélisation orientée objet, le système est décrit par un ensemble d'objets pouvant communiquer entre eux. Chaque objet possède des propriétés (attributs) qui décrivent son état et des opérations qui décrivent son comportement. Contrairement aux approches systémiques où on sépare les deux aspects statique et dynamique d'un système en termes de données et de traitements ;

- Notation graphique :



F1 : Formalisme de représentation de l'objet.

- **Exemple :**



Représentation d'un objet employé

3-1- ETAT :

L'état d'un objet est la description de ses propriétés à un instant donné. Par propriété, on entend l'ensemble des attributs décrivant la partie statique de l'objet.

Exemple : Dans l'exemple précédent, la propriété grade de l'employé = « ingénieur d'état » décrit la situation professionnelle actuelle de l'employé « Ahmed » ayant été recruté le 25/10/2008 et habitant à T.O.

3-2- Comportement :

Par comportement, on entend tout aspect dynamique ou fonctionnel d'un objet. Le comportement d'un objet est décrit à travers ses opérations. La dynamique d'un objet concerne les opérations relatives aux changements de son propre état alors que son fonctionnement concerne les opérations relatives aux services qu'il offre à son environnement.

Exemple : Si on reprend l'exemple précédent, tout changement d'adresse ou de grade conduira à un changement d'état de l'objet « employé Ahmed »

- L'opération « modifier le profile de l'employé () permet de modifier la valeur de la propriété « Grade actuel » car celle-ci change selon l'évolution.de la carrière de l'employé « Ahmed » (promotion, rétrogradation). Cette opération concerne la dynamique de l'objet »Employé Ahmed ».

Par contre, l'opération « calculer le nombre d'absences () » permet de délivrer cette information à d'autres objets impliqués dans le calcul de la paye de l'employé « Ahmed » par exemple. Cette opération concerne le fonctionnement de l'objet « employé Ahmed ».

De tout cela, nous pouvons dire :

- Une opération est une transformation qui peut être appliquée à ou par des objets ;
- Une opération possède comme paramètre implicite l'objet pour lequel elle est invoquée ;
- Une opération utilise et/ou modifie des attributs de l'objet ;
- L'implantation effective d'une opération pour une classe s'appelle une méthode (ou encore fonction membre).

3-3- Encapsulation :

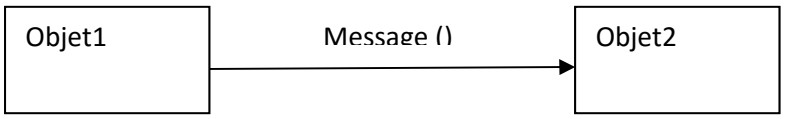
Dans les méthodes systémiques (telle que merise), les données sont séparées des traitements. Ceci implique que tout changement dans la structure d'une donnée doit être effectué au niveau de tous les programmes l'utilisant. Cette duplication des données est évitée dans l'approche objet grâce au concept d' « encapsulation ».

Cette propriété fait que tout objet renferme en son sein ses attributs et ses opérations. Ces derniers sont cachés aux autres objets du système. Tout accès à un service ne peut se faire que par le biais d'envoi de message qui va déclencher l'exécution d'une opération. Ceci rend indépendant tout changement interne de la structure de l'objet invoqué sans avoir à le propager dans tout le système.

3-4- Communication entre objets :

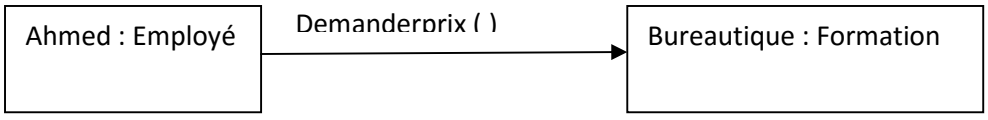
Les objets communiquent entre eux par envoi de messages. Un message représente l'appel d'opération d'un objet pour un objet source. De plus, le système étant constitué d'objets, les messages permettent de relier ces derniers entre eux.

Notation graphique :



Communication entre objets

Exemple : Une demande du prix d'une formation représentée par un message demande prix ( ) entre l'objet employé Ahmed et l'objet Formation bureautique.



Communication entre objets

4- Classification :

Lorsque le système comporte un ensemble d'objets partageant les mêmes caractéristiques et le même comportement, ces derniers sont regroupés sous un même type ou classe. On dit que chaque objet est une réalisation ou instanciation de la classe**.**

Notation graphique :

Nom de la classe
Liste de ses attributs
Liste de ses opérations

Représentation graphique d'une classe d'objets

Exemple :

L'exemple suivant décrit la classe d'objets « employé » cités dans les exemples précédents :

Employé
• Nom de l'employé • Adresse • Date de naissance • Grade actuel
• Modifier le profile d l'employé (…) • Calculer le nombre d'absences(…)

Classe employé

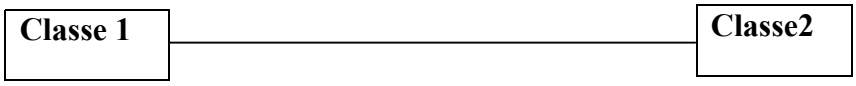
Remarque :

Notons toutes fois qu'un objet (ou une classe) peut être cité(s) avec uniquement son nom, sans en préciser les détails. On parlera alors d'objets (ou de classe) non documenté(s).

5- Association :

On parle d'association, quand on veut décrire un lien statique entre plusieurs objets (ou plusieurs classes).

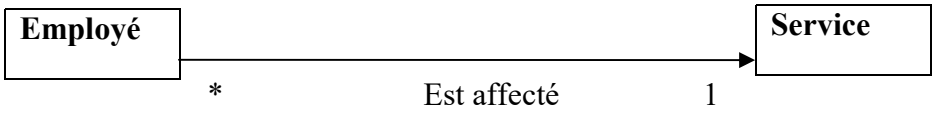
Notation graphique :



-Lien d'association entre deux classes-

Exemple :

Dans cet exemple, un employé est affecté à un seul service mais que ce dernier peut accueillir plusieurs employés à la fois**.**

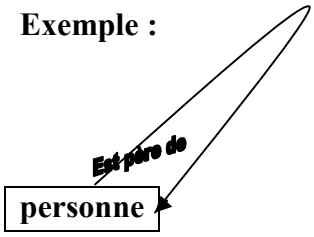


-Affectation d'employé aux services-

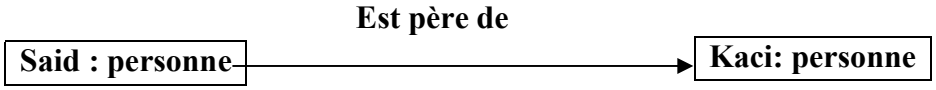
Remarque :

Lorsque le lien existe entre des objets de la même classe, on parle d'association réflexive.

Exemple :



Exemple:



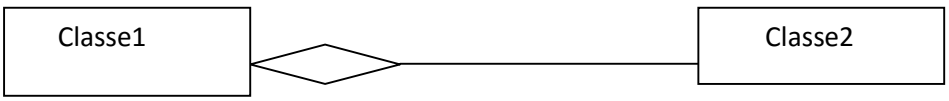
Association réflexive

Il existe des liens d'association particuliers, tel que l'agrégation, la composition et l'héritage.

5-1- L'agrégation :

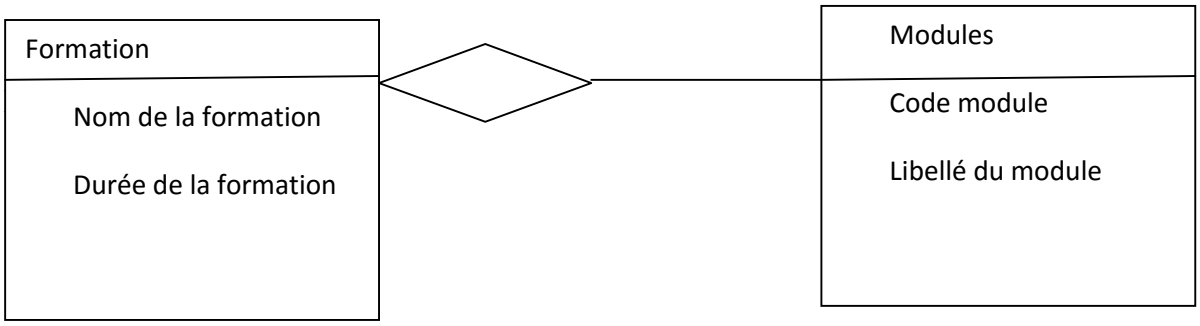
C'est un cas particulier d'association où un tout est relié à ses paries. Le tout (la classe à côté du losange) est appelé <u>agrégat</u> et la classe en opposé est appelé <u>agrégée</u>.

Notation graphique :



Lien d'agrégation

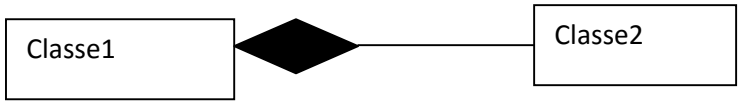
Exemple : Dans le cadre d'une formation, le cursus de cette dernière est une agrégation de modules à enseigner.



Remarquons dans cet exemple que la suppression d'une formation ne conduit pas automatiquement à la suppression des modules étant donné que ces derniers peuvent être enseignés dans d'autres formations.

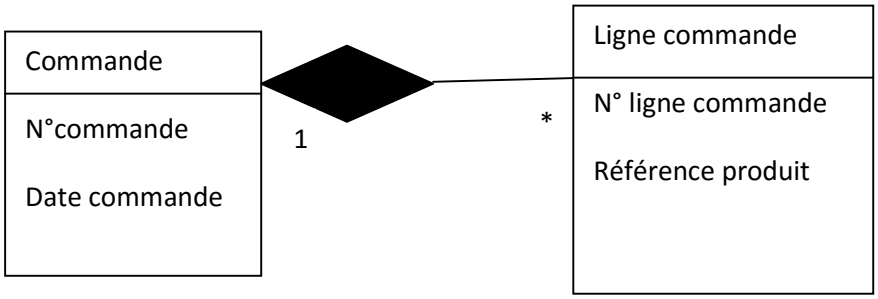
5-2- Composition : Lorsque le lien d'agrégation est « fort », c'est-à-dire que la suppression de l'objet agrégat conduit même à la suppression des objets agrégés, on parle de composition.

Notation graphique :



Lien de composition

Exemple : une commande d'un ensemble de lignes commande décrivant les produits commandés. La suppression d'une commande conduira obligatoirement à la suppression de toutes ses lignes commandes.



1 et * sont des multiplicités (cardinalités).

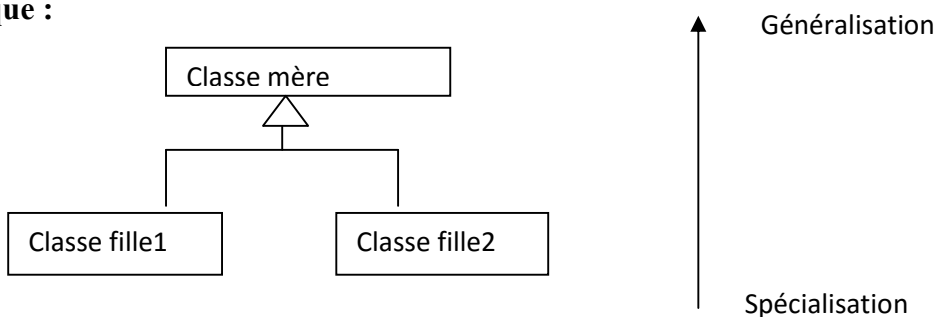
5-3- Héritage :

L'héritage est mis en œuvre grâce à deux propriétés qui sont : la généralisation et la spécialisation.

La généralisation décrit le fait de pouvoir regrouper un ensemble de classes partageant des éléments en commun en une seule superclasse (ou classe mère).

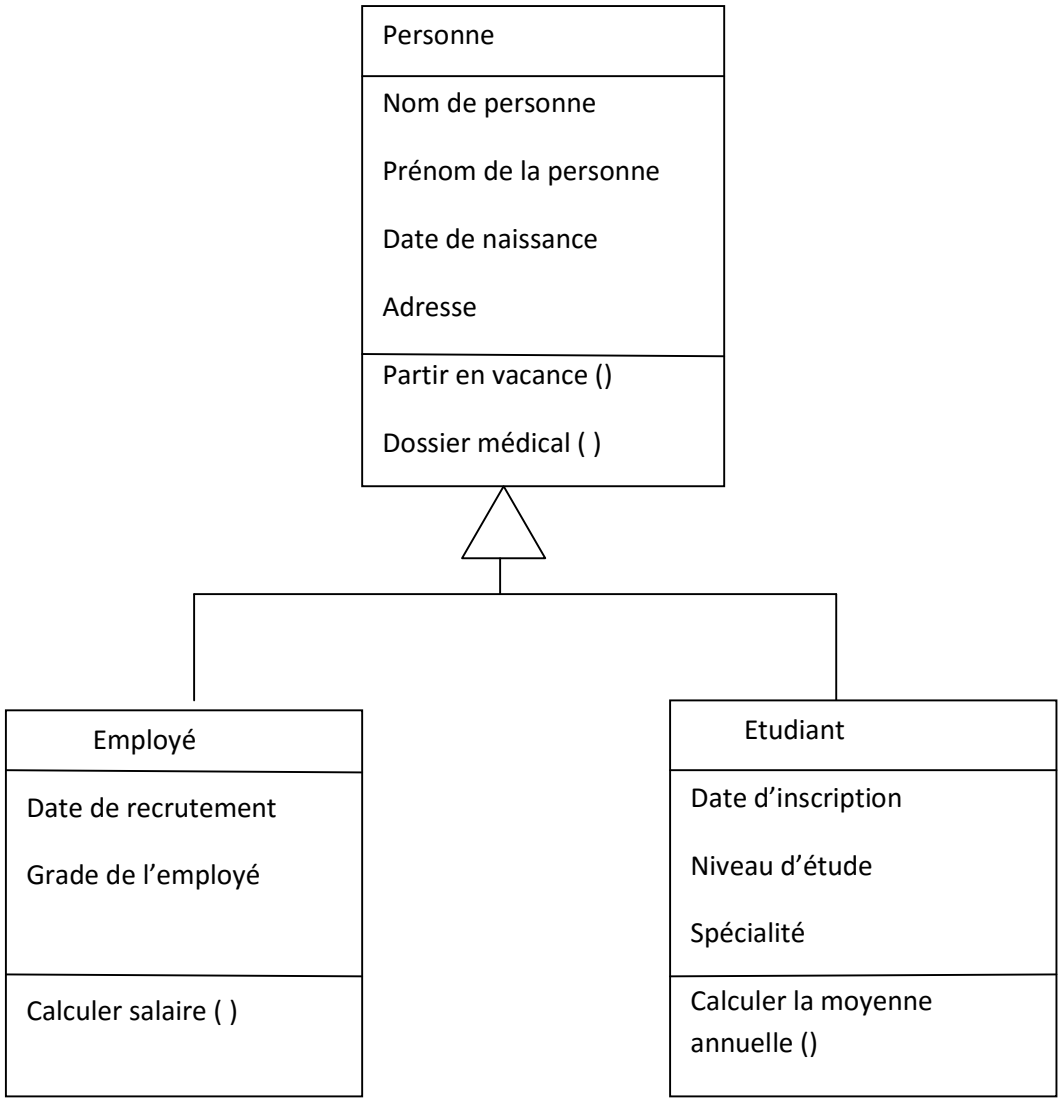
La spécialisation représente le phénomène inverse, c'est-à-dire, pouvoir dériver à partir d'une classe ou superclasse des sous classes (ou classes filles) ayant les propriétés spécifiques les distinguant les unes des autres.

Notation graphique :



Exemple :

Si on considère les classes « Employé » et « étudiant » qui dérivent de la classe « personne » et héritent de cette dernière. Cependant, « employé » se distingue par d'autres attributs tels que la date de recrutement, le grade et le salaire alors que « étudiant » possède une date d'inscription, un niveau d'étude, une spécialité et une moyenne annuelle.



Héritage entre les classes « Employé », « Etudiant » et « Personne »

6- Polymorphisme :

C'est le fait d'utiliser la même expression pour dénoter différentes opérations. En effet, une même opération peut être définie par plusieurs classes. Ceci ne signifie pas que cette opération est implémentée exactement de la même manière (avec la même méthode). Au contraire, on peut lui associer, selon la classe à laquelle elle appartient une méthode différente. Ainsi, un même message vers une opération donnée peut produire des résultats différents selon la classe invoquée.

Cependant, toutes les méthodes qui implantent une opération doivent avoir :

- Le même rôle symbolique ;
- La même signature ;
- Le même type de valeur de retour.

Exemple : Si on reprend l'exemple précédent, l'opération « partir en vacance ( ) » dont héritent les classes « employé » et « Etudiant » pourraient avoir des implémentations différentes pour ces deux sous classes car le départ en vacances pour l'employé aura des incidences financières sur son salaire alors que ce ne sera pas le cas pour l'étudiant. D'où, il serait préférable d'implémenter cette opération par deux méthodes différentes : une pour la sous classe « Employé » et une pour la sous classe « Etudiant ».

7- Les cardinalités :

La cardinalité indique le nombre d'objets d'une classe qui peuvent participer à l'association. Elle est généralement définie avec une borne inférieure et une borne supérieure. La borne inférieure est un entier positif ou nul et la borne supérieure est un entier strictement positif ou * (pour infini). Si les deux bornes sont égales, on utilisera un seul nombre (1..1) est équivalent à 1. Comme * est équivalent à 0..*.

8- Présentation d'UML (Unified Modeling Language)

UML (*Unified Modeling Language*) est un langage de modélisation orientée objet développée en réponse à l'appel à propositions lancé par l'OMG (*Object Management Group*) dans le but de définir la notation standard pour la modélisation des applications construites à l'aide d'objets. Elle est héritée de plusieurs autres méthodes telles que OMT (*Object Modeling Technique*) et OOSE (*Object Oriented Software Engineering*) et Booch. Les principaux auteurs de la notation UML sont Grady Booch, Ivar Jacobson et Jim Rumbaugh.

Elle est utilisée pour spécifier un logiciel et/ou pour concevoir un logiciel. Dans la spécification, le modèle décrit les classes et les cas d'utilisation vus de l'utilisateur final du logiciel. Le modèle produit par une conception orientée objet est en général une extension du modèle issu de la spécification. Il enrichit ce dernier de classes, dites techniques, qui n'intéressent pas l'utilisateur final du logiciel mais seulement ses concepteurs. Il comprend les modèles des classes, des états et d'interaction. UML est également utilisée dans les phases terminales du développement avec les modèles de réalisation et de déploiement.

UML est un langage de représentation graphique. L'usage d'une représentation graphique est un complément excellent à celui de représentions textuelles. En effet, l'une comme l'autre sont ambigües mais leur utilisation simultanée permet de diminuer les ambiguïtés de chacune d'elle. Un dessin permet bien souvent d'exprimer clairement ce qu'un texte exprime difficilement et un bon commentaire permet d'enrichir une figure.

Il est nécessaire de préciser qu'une méthode telle que UML ne suffit pas à produire un développement de logiciel de qualité à elle seule. En effet, UML n'est qu'un formalisme, ou

plutôt un ensemble de formalismes permettant d'appréhender un problème ou un domaine et de le modéliser, ni plus ni moins. Un formalisme n'est qu'un outil. Le succès du développement du logiciel dépend évidemment de la bonne utilisation d'une méthode comme UML mais il dépend surtout de la façon dont on utilise cette méthode à l'intérieur du cycle de développement du logiciel.

8-1- Eléments et mécanismes généraux d'UML :

Un modèle est une représentation simplifiée d'un problème. UML permet d'exprimer les modèles objets à travers un ensemble de diagrammes. Ces derniers sont des moyens de description des objets ainsi que des liens qui les relient.

8-2- Les diagrammes d'UML :

Un diagramme est une représentation graphique qui s'intéresse à un aspect précis du modèle.

Dans UML, il existe plusieurs formalismes ou « modèles » :

- le modèle *des classes ;*
- le modèle *des états ;*
- le modèle *des cas d'utilisation ;*
- le modèle *d'interaction ;*
- le modèle *de réalisation ;*
- le modèle *de déploiement ;*

Le modèle des classes est le plus utile. C'est un formalisme pour représenter les concepts usuels de l'orienté objet. Le modèle des états et le modèle d'interaction permettent de représenter la dynamique des objets. Le modèle des cas d'utilisation permet de décrire les besoins de l'utilisateur final du logiciel. Le modèle de réalisation et le modèle de déploiement, moins importants que les autres modèles de UML, ne seront pas décrits par ce module.