UML: Introduction à la modélisation Objet

Introduction à la modélisation objet

I. Les principes fondamentaux
II. Quelques propriétés importantes de la modélisation objet.

Les principes fondamentaux de l'orienté Objet.

Notions d'Objet et de Classe.

- Dans l'orientée objet, on va regrouper au sein d'une même entité certaines données et les moyens de traitement de ces données.

- Une telle entité s'appellera un **objet** et possédera donc :
  1. Une identité.
  2. Des variables définissant sont état (**attributs**).
  3. Des sous programmes gérant son comportement (**méthodes**).

I. Les Principes fondamentaux de l'orienté Objet

2. Notions d'Objet et de Classe.

- Des objets ayant des propriétés communes (attributs et méthodes) sont alors regroupés dans une structure abstraite appelée **classe**.

- On retrouve ainsi notre manière de pensée habituelle, où nous « classifions » chaque élément de notre entourage : animaux, véhicules, ordinateurs, étudiants, livres…

I. Les Principes fondamentaux de l'orienté Objet

2. Notions d'Objet et de Classe.

Exemple de classe :

Classe Personne
Début
Attributs :
Nom : CHAÎNE
Prénom : CHAÎNE
AnnéeNaissance : ENTIER
Méthodes :
Age() : ENTIER
RenvoyerNom() : CHAÎNE
Fin

Personne
Nom : CHAÎNE Prénom : CHAÎNE AnnéeNaissance : ENTIER
Age() : ENTIER RenvoyerNom() : CHAÎNE

I. Les Principes fondamentaux de l'orienté Objet

2. Notions d'Objet et de Classe.

Exemple de classe :

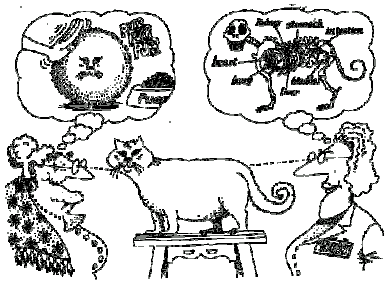
Classe Rectangle
Début
Attributs :
Largeur : RÉEL
Longueur : RÉEL
Méthodes :
Aire() : RÉEL
Périmètre() : RÉEL
Fin

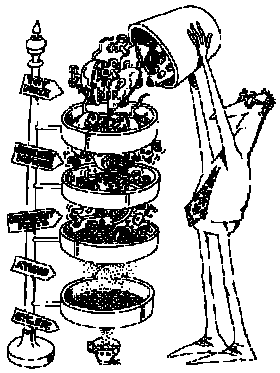
Rectangle
Largeur : RÉEL Longueur : RÉEL
Aire() : RÉEL Périmètre() : RÉEL

I. Les Principes fondamentaux de l'orienté Objet

2. Notions d'Objet et de Classe.

- A noter qu'une démarche d'abstraction est nécessaire pour ne retenir que les propriétés (attributs et méthodes) pertinentes d'un objet pour un problème précis.



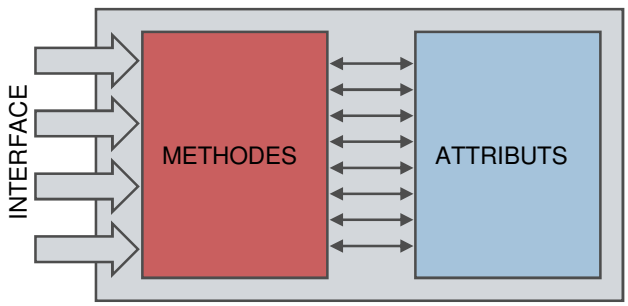


2. Notions d'Objet et de Classe.

- En POO une des questions fondamentales du développeur est donc : sur quoi porte le programme ?
- A opposer à la question fondamentale à se poser en programmation procédurale : à quoi sert le programme ?
- Cette différence se retrouvera aussi lors de la phase de conception, où nous réserverons l'algorithmique classique à l'implémentation des méthodes, alors que pour concevoir les classes nous utiliserons un langage graphique plus adapté : l'**UML**.

3. Encapsulation

- **Encapsulation** : le principe est d'interdire l'accès direct aux attributs. On ne dialoguera avec l'objet qu'à travers une interface définissant les services accessibles aux l'utilisateurs de l'objet. Ce sera le rôle des méthodes.



3. Encapsulation

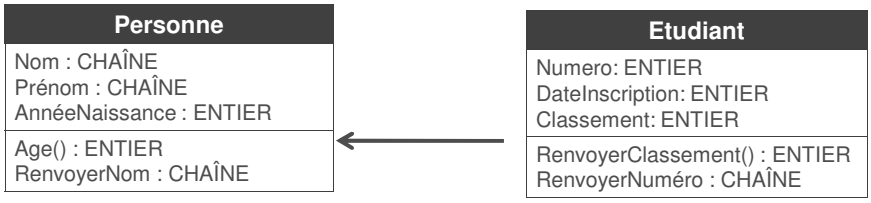
Deux intérêts majeurs :

1. Facilitation de l'évolution d'une application : on peut modifier les attributs d'un objet sans modifier la façon dont il est utilisé.

2. Garantie de l'intégrité des données, car leur accès direct est interdit (ou en tout cas limité et contrôlé).

4. Héritage

- **Héritage** : relation de spécialisation/généralisation entre deux classes. Elle indique qu'une classe est une sous classe d'une autre, i.e. qu'elle possède ses attributs et ses méthodes plus d'autres qui lui sont propres.



4. Héritage

Intérêt :

On construit une hiérarchie de classe. On évite ainsi des répétitions dans le code, en encourageant la réutilisation de classes déjà existantes.

Cela permet bien sûr également de simplifier la conception de la modélisation.

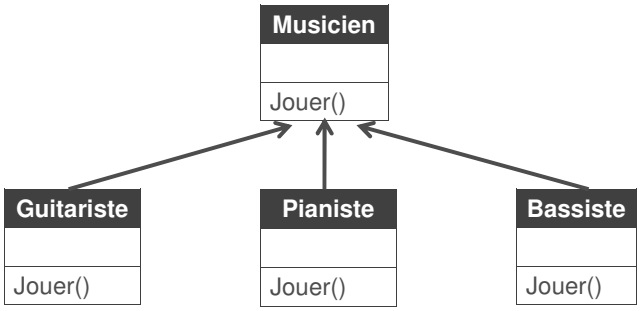
5. Polymorphisme

- **Polymorphisme** : littéralement c'est la faculté de prendre plusieurs formes. En POO, c'est un mécanisme qui permet à une sous classe de redéfinir une méthode dont elle a hérité tout en gardant la même signature.

Selon le contexte, le programme optera pour la « bonne » méthode.

I. Les Principes fondamentaux de l'orienté Objet

5. Polymorphisme



L'appel de la méthode « jouer » sur tous les objets héritant de la classe « musicien » produira un résultat différent selon la sous classe.

I. Les Principes fondamentaux de l'orienté Objet

5. Polymorphisme

Intérêt :

Le code gagne en généricité. On peut appeler des méthodes portant les mêmes noms mais produisant des effets différents selon le type réel de l'objet.

Vos Questions

Ⅱ. Quelques propriétés importantes de la modélisation objet.

Ⅱ. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets.
2. Relations entre classes.
3. Héritage.
4. Polymorphisme.

Ⅱ. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

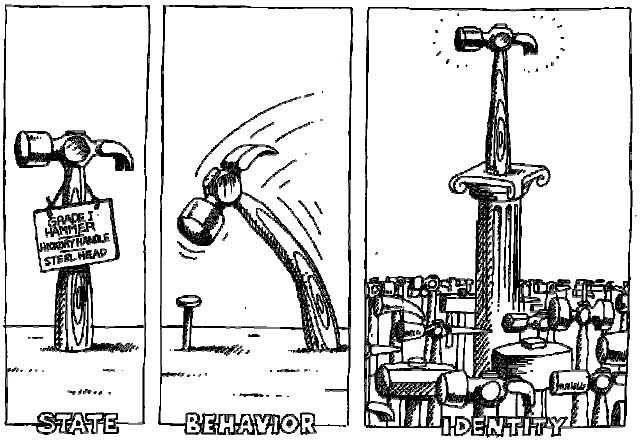
Objet : entité identifiable du monde réel pouvant avoir ou pas une existence physique. Exemples : chat, table, courant de pensée…

Un objet possède **trois composantes** :

1. Une identité.
2. Des variables définissant sont état (**attributs**).
3. Des sous programmes gérant son comportement (**méthodes**).

II. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets



II. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

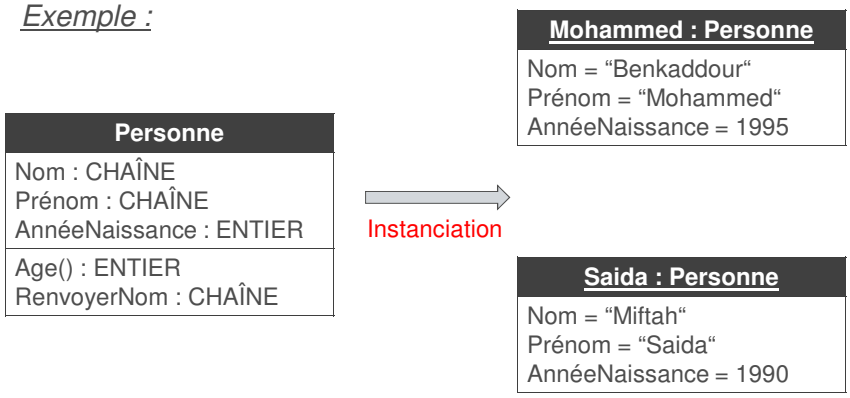
Classe : abstraction regroupant des objets ayant les mêmes attributs et les mêmes méthodes.

Un objet est alors une **instance** de la classe correspondante, et se distingue des autres instances par son identité et la valeur de ses attributs.

II. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

Exemple :



II. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

Visibilité des attributs et méthodes :

Un attribut ou une méthode sont dits **privés** si leur utilisation est interdite en dehors de la classe.

Un attribut ou une méthode sont dits **publics** si leur utilisation est autorisée en dehors de la classe.

Ce choix s'effectue lors de la déclaration de la classe.

Ⅱ. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

Par défaut la visibilité des différents membres vérifie le principe d'**encapsulation** :

- Les attributs sont privés.
- Les méthodes sont publiques.

Rappelons que l'encapsulation est le principe interdisant l'accès direct aux attributs d'une classe.

Ⅱ. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

Cependant, dans certains cas particuliers, on contreviendra à ce principe en imposant une visibilité différente de celle par défaut.

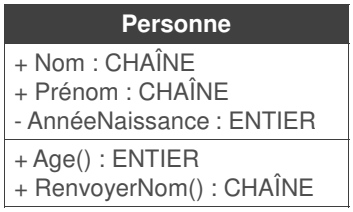
Par exemple, on pourra « découper » certaines méthodes publiques complexes en plusieurs sous méthodes, qui seront déclarées privées car étant internes à l'objet.

II. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

Exemple de déclaration de la visibilité :

Classe Personne
Début
Attributs publics :
Nom : CHAÎNE
Prénom : CHAÎNE
Attribut privé :
AnnéeNaissance : ENTIER
Méthodes publiques :
Age() : ENTIER // *méthode calculant l'âge et le retournant*
RenvoyerNom() : CHAÎNE // *méthode retournant le nom*
Fin



II. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

Comment utiliser tout cela ?

- Vous commencez bien sûr par déclarer votre classe, par exemple la classe personne précédente.
- Ensuite vous pouvez instancier un objet en le déclarant comme une variable et en initialisant ses attributs :

  Mohammed : Personne("Mohammed","Benkaddour",1995)

II. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

Vous avez ensuite accès à ses méthodes et attributs publics :

- ECRIRE(Mohammed.Nom) affichera Benkaddour à l'écran.
- ECRIRE(Mohammed.AnnéeNaissance) est interdit car AnnéeNaissance est un attribut privé.
- ECRIRE(Mohammed.Age()) est par contre autorisé, et affichera 16.

II. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

Attributs et méthodes indépendants des objets :

Attributs de classes : ce sont des attributs particuliers qui ont la même valeur pour toutes les instances de la classe. L'exemple fondamental est un attribut qui compte le nombre d'objets instanciés de la classe.

Méthodes de classes : ce sont des méthodes qui ne dépendent pas des attributs propres de chaque objet mais qui portent sur les attributs de classes.

II. Quelques propriétés importantes de la modélisation objet

1. Objets et classes d'objets

La déclaration d'attribut ou de méthode de classe s'effectue en rajoutant le suffixe **statique** :

Classe Rectangle
Début
Attributs privés :
Largeur : RÉEL
Longueur : RÉEL
NbRect : ENTIER statique
Méthodes publiques :
Aire() : RÉEL
RenvoyerNbRect() : ENTIER statique
Fin

Rectangle
- Largeur : RÉEL - Longueur : RÉEL - NbRect : ENTIER
+ Aire() : RÉEL + RenvoyerNbRect() : ENTIER

II. Quelques propriétés importantes de la modélisation objet

2. Relations entre classe

Les classes sont donc les éléments de base d'une modélisation orientée objet.

Après les avoir définies, il convient de voir comment les relier entre elles, et quels types d'interactions elles peuvent avoir.

II. Quelques propriétés importantes de la modélisation objet

2. Relations entre classe

Relation d'association : elle indique qu'un lien conceptuel existe entre deux classes. Ces classes se connaîssent donc et leurs instances peuvent communiquer en s'envoyant des messages (messages qui ne sont en fait la plupart du temps que des invocations de méthodes).

II. Quelques propriétés importantes de la modélisation objet

2. Relations entre classe

Relation d'agrégation : elle modélise une relation d'inclusion entre les instances de deux classes. Les objets de la classe « conteneur » possèdent donc des attributs qui sont des objets de la classe « contenue ».

Ⅱ. Quelques propriétés importantes de la modélisation objet

2. Relations entre classe

Relation de composition : c'est une relation d'agrégation particulière, où les instances de la classe contenue ne peuvent exister sans celles de la classe conteneur; elles sont crées et détruites par celle ci.

Ⅱ. Quelques propriétés importantes de la modélisation objet

3. Héritage

La plus importante des relations entre classes est celle d'héritage :

Héritage : relation de spécialisation/généralisation entre deux classes. Elle indique qu'une classe dite classe fille spécialise une autre classe dite classe mère, i.e. qu'elle possède les attributs et les méthodes de la classe mère plus d'autres qui lui sont propres.

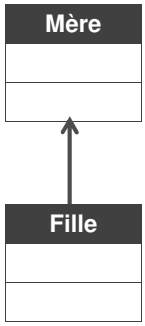
On parle aussi de **super classe** et de **sous classe**.

II. Quelques propriétés importantes de la modélisation objet

3. Héritage

Exemple de déclaration d'une relation d'héritage :

Classe Mère
Début
Attributs :
Méthodes :
Fin
Classe Fille hérite de **Classe** Mère
Début
Attributs : // nouveaux attributs
Méthodes : // nouvelles méthodes
Fin



II. Quelques propriétés importantes de la modélisation objet

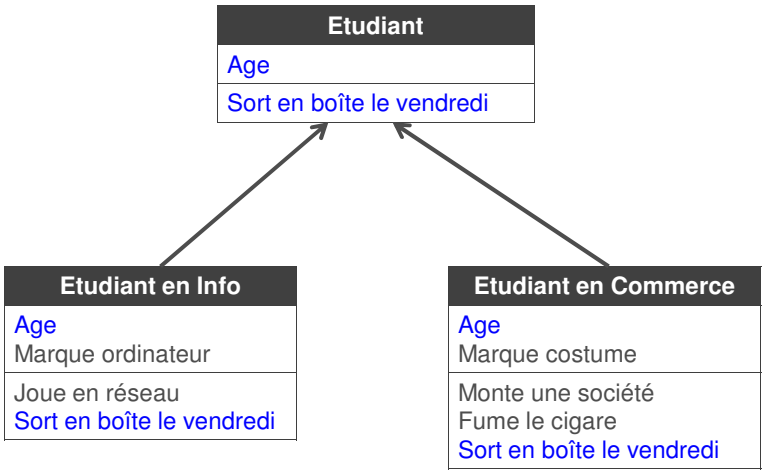
3. Héritage

Deux visions pour un même concept :

1. Une vision « ascendante », en procédant par généralisation : on décèle des attributs et des méthodes communs à des classes différentes, l'héritage permet alors de les factoriser afin de faciliter la conception et la maintenance du code.

Ⅱ. Quelques propriétés importantes de la modélisation objet

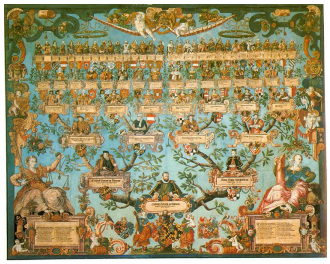
3. Héritage



Ⅱ. Quelques propriétés importantes de la modélisation objet

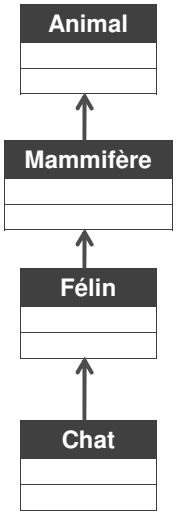
3. Héritage

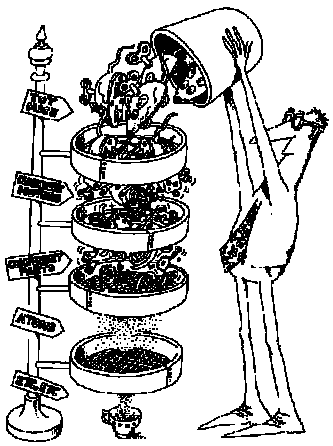
2. Une vision « descendante », en procédant par spécialisation : on crée des classes spécialisées à partir d'une classe de base. Le niveau de spécialisation dépend du niveau d'abstraction que l'on souhaite. On procède souvent ainsi quand on veut réutiliser des classes déjà existantes.



II. Quelques propriétés importantes de la modélisation objet

3. Héritage

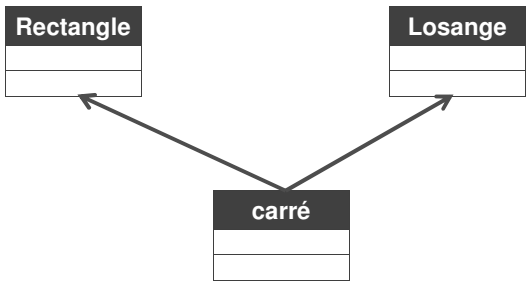




II. Quelques propriétés importantes de la modélisation objet

3. Héritage

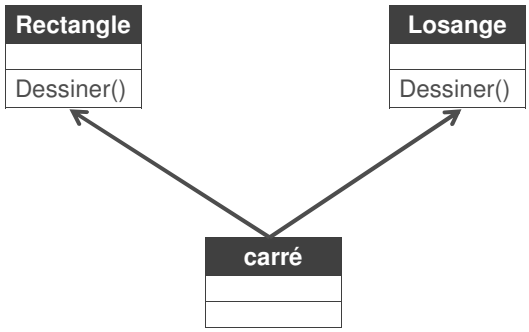
Héritage multiple : possibilité pour une classe de posséder plusieurs classes mères.



II. Quelques propriétés importantes de la modélisation objet

3. Héritage

Des difficultés peuvent apparaître quand les classes mères possèdent des méthodes de mêmes noms qui ne sont pas redéfinies au sein de la classe fille.



II. Quelques propriétés importantes de la modélisation objet

3. Héritage

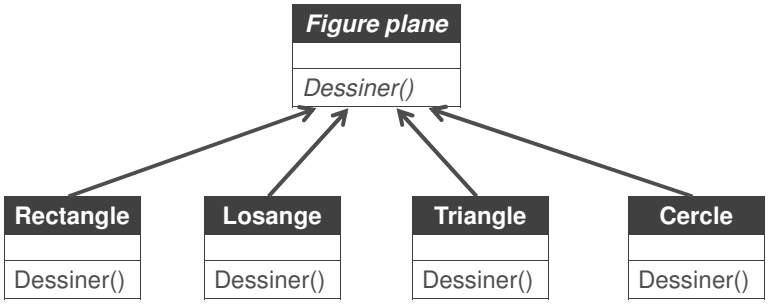
Classe abstraite : classe qui ne peut être instanciée, car elle contient des méthodes abstraites, c'est à dire des méthodes non implémentées.

Une classe abstraite sert essentiellement à factoriser des méthodes et attributs communs à plusieurs classes, et ce dans une relation d'héritage.

Cela permet de clarifier la conception du code.

II. Quelques propriétés importantes de la modélisation objet

3. Héritage



La classe « Figure plane » est une classe abstraite, elle n'est pas destinée à être instanciée. Ses quatre sous classes sont elles concrètes et produiront des objets.

II. Quelques propriétés importantes de la modélisation objet

4. Polymorphisme

Les classes abstraites ont un intérêt grace au concept de **polymorphisme**.

C'est le mécanisme qui permet à une sous classe de redéfinir une méthode dont elle a hérité tout en gardant la même signature.

II. Quelques propriétés importantes de la modélisation objet

4. Polymorphisme

Reprenons l'exemple précédent et imaginons que l'on possède un tableau constitué d'une centaine d'objets dérivant de la classe « Figure plane ». On souhaite les dessiner un par un. La puissance du **polymorphisme** permettra d'utiliser une routine de la forme :

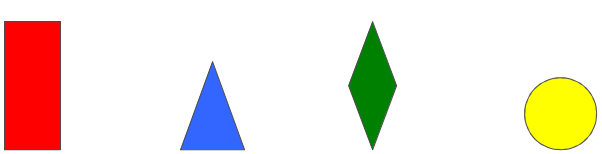
```
POUR i ALLANT de 1 à 100 AU PAS DE +1
FAIRE
    tab[i].Dessiner()
FINPOUR
```

II. Quelques propriétés importantes de la modélisation objet

4. Polymorphisme

A chaque itération, le programme choisira la méthode Dessiner() correspondant au type d'objet en question, que ce soit un rectangle, un triangle, un losange ou un cercle.

Et ce sans lui spécifier directement la nature de l'objet.



Vos Questions