




# POO

Programmation Orientée Objets



# Définition

- La POO est une méthode d'implémentation dans laquelle les programmes sont organisés sous formes de collections coopératives d'objets, dont chacun représente une instance d'une classe quelconque et dont toutes les classes sont membres d'une hiérarchie de classes unis à travers des relations d'héritage.
- 



# Objet et classe



- Un objet est une entité logicielle :
  - Ayant une identité : son nom
  - Capable de sauvegarder un état c'est-à-dire un ensemble d'information dans des variables internes : Les attributs
  - Répondant à des messages précis en déclenchant des activations internes appropriés qui changent l'état de l'objet. Ces opérations sont appelées méthodes. Ce sont des fonctions liées à des objets et qui précisent le comportement de ces objets.



# Objet



- Un objet est une entité vivante, qui réagit aux messages reçus
- Un objet contient les données (attributs) et les traitements qui permettent d'agir dessus (Méthodes)
- Un objet a des frontières bien définies, des états et des comportements



# Objet

- un objet est une zone mémoire allouée avec un "**new**"
- pour garder un lien vers cette zone mémoire, on utilise une référence
- un objet ne peut pas être manipulé sans référence
- Lorsqu'un objet n'est plus référencé, il est finalisé
- Exemple: `Voiture v = new Voiture();`



# Classe

Une classe définit attributs et méthodes:

- ▀ les attributs sont typés, ce sont des données
- ▀ les méthodes sont décrits par un nom, un type de retour et des paramètres eux-mêmes typés
- ▀ il n'y a pas d'ordre dans les déclarations
- ▀ le corps des méthodes peut utiliser les attributs, les paramètres et les variables locales éventuelles
- ▀ Par convention, une classe commence par une majuscule, un attribut et une méthode par une minuscule puis la capitalisation
- ▀ Une classe peut être publique ou non
- ▀ Une classe peut étendre une autre classe
- ▀ Une classe peut implémenter une interface



# Classes, objets, instances

- Une classe est un ensemble d'objets qui ont en commun :
  - les mêmes méthodes
  - les mêmes types d'attributs
- Une classe est un type d'objet
- Une instance d'une classe est un objet particulier d'une classe qui peut activer les méthodes de la classe et qui a des valeurs particulières de ses attributs.
- On définit l'objet comme l'instance d'une classe. La classe représente pour l'objet ce que représente le type pour la variable





# Attribut



- Les attributs d'un objet sont l'ensemble des informations se présentant sous forme de variable et permettant de représenter l'état de l'objet.
- Les attributs sont champs de donnée typés
- Le nom des attributs suit la convention : commencer par une minuscule et ensuite capitalisation



# Méthode

- Une méthode est une fonction ou procédure liée à un objet qui est déclenchée à la réception d'un message particulier : la méthode déclenchée correspond strictement au message reçu. La liste des méthodes définies au sein d'un objet constitue l'interface de l'objet pour l'utilisateur : ce sont les messages que l'objet peut comprendre si on les lui envoie et dont la réception déclenche les méthodes correspondantes.
- une méthode est un traitement effectué sur un objet
- son nom suit la même convention que les attributs
- elle peut déclarer une liste de paramètres(type nom, type nom,...) ou pas
- elle a un retour, sinon elle déclare "**void**"
- une méthode correspond à un message que l'on peut envoyer à l'objet de ce type



# Signature d'une méthode

- La signature d'une méthode représente la précision de son nom, du type de ses arguments et du type de donnée retournée.
- 



# Constructeurs

Les constructeurs sont des méthodes spéciales permettant de créer un objet. En effet lors de la création d'un objet avec le mot clé new, c'est l'un des constructeur de la classe qui est appelé. Chaque classe a un constructeur par défaut.

- Ils portent le nom de la classe
- Ils n'ont pas de type de retour
- Ils initialisent les attributs
- Ils peuvent avoir des parametres
- L'appel d'un constructeur est implicite



# Encapsulation

L'encapsulation est le fait qu'un objet renferme ses propres attributs et ses méthodes. Les détails de l'implémentation d'un objet sont masqués aux autres objets du système à objets. On dit qu'il y'a encapsulation de données et du comportement des objets. On précise trois modes d'accès aux attributs d'un objet.

- Le mode **public** avec lequel les attributs seront accessibles directement par l'objet lui même ou par d'autres objets. Il s'agit du niveau le plus bas de protection.
- Le mode **private** avec lequel les attributs de l'objet seront inaccessibles à partir d'autres objets : seules les méthodes de l'objet pourront y accéder. Il s'agit du niveau le plus fort de protection.
- Le mode **protected** : cette technique de protection est étroitement associée à la notion d'héritage (suite du cours).
- Par défaut, on a une **visibilité de package** : les membres d'un même package peuvent partager des données