

Aujourd'hui:

Présentation des concepts de base de la programmation de haut niveau.
Introduction au langage Java.

- Exemples explicatifs: Héritage, polymorphisme ...
- Aspects algorithmiques
- IDE: Eclipse

Le concept Orienté objet (OO)

Exemples explicatifs (1/7)

Notion d'héritage: L'héritage permet de spécialiser une classe qui possédera non seulement les propriétés et méthodes de sa mère mais également d'autres méthodes spécifiques ou redéfinies.

Le terme est *faire dériver la classe en une classe fille*. Dans l'objet fille, on trouve:

- De nouvelles méthodes ou propriétés
- Des méthodes ou propriétés qui *surchargent*, c'est-à-dire redéfinissent celles de la classe mère.
- Les propriétés et méthodes de la classe mère qui n'ont pas été surchargées

Exemples explicatifs (2/7)

Exemple: la classe *Bâtiment* est caractérisée par :

Propriétés = Variables	Comportements = Méthodes
<ul style="list-style-type: none">• les murs ;• le toit ;• une porte ;• l'adresse ;• la superficie.	<ul style="list-style-type: none">• ouvrir le Bâtiment ;• fermer le Bâtiment ;• agrandir le Bâtiment.

Exemples explicatifs (3/7)

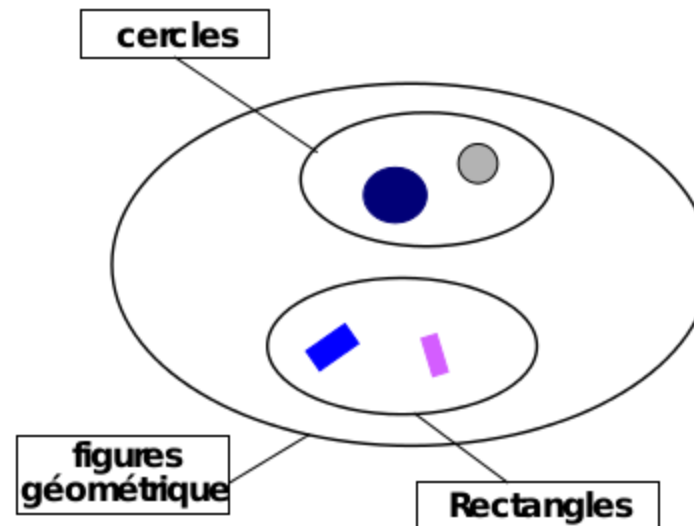
La classe *Maison* hérite de la classe *bâtiment*

Propriétés = Variables	comportements = Méthodes
<ul style="list-style-type: none">• La toiture• le jardin	<ul style="list-style-type: none">• Faire le jardin;• nettoyer la toiture• faire une piscine

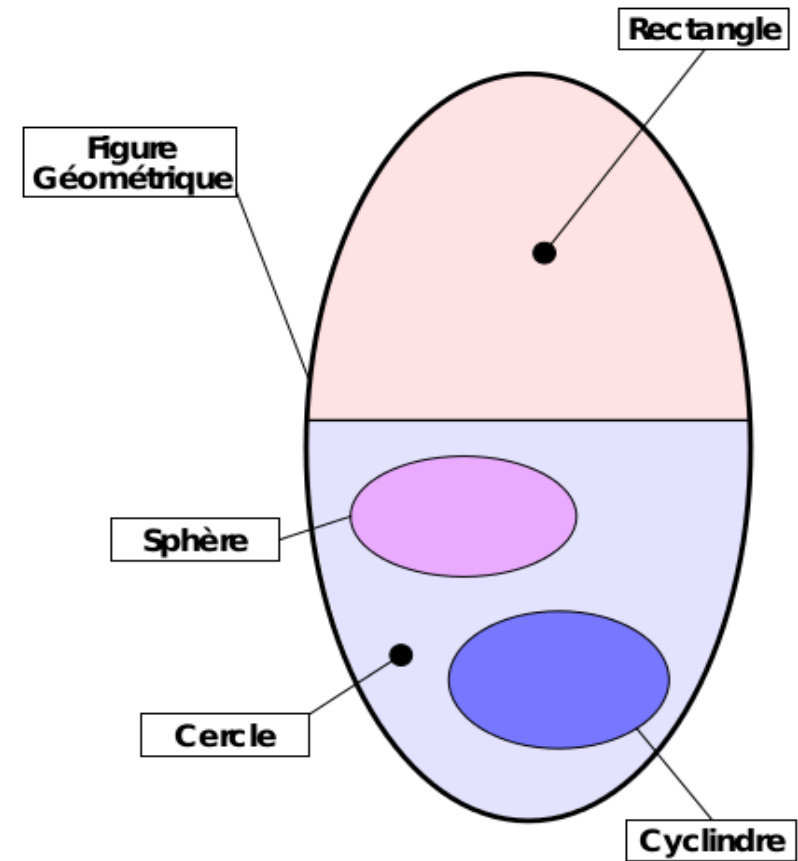
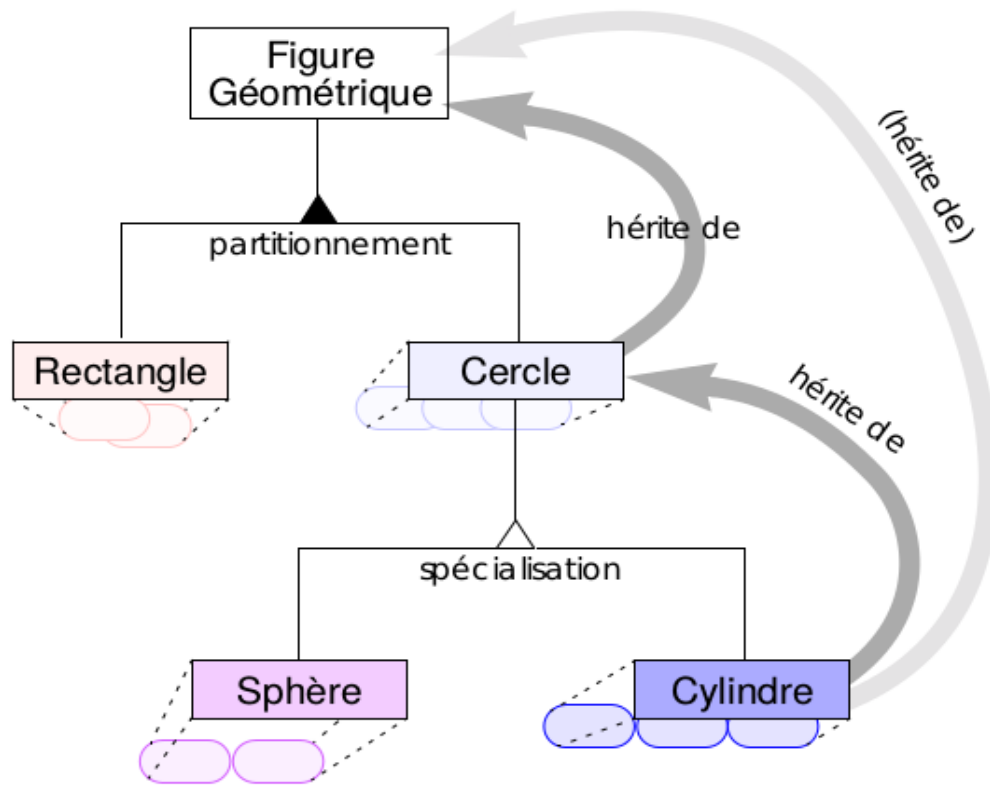
Exemples explicatifs (4/7)

Exemple:

- Classe Sphère;
- Classe Figures géométriques;
- Classe Cercle;
- Classe rectangle;
- Classe Disque.



Exemples explicatifs (5/7)



Les Propriétés de base de l'OO (6/7)

Notion de polymorphisme: Les objets sont dits *polymorphes* car ils possèdent plusieurs types: le type de leurs classes et les types des classes ascendantes.

Exemple: *uneHonda*, instance de *Moto* aura comme type initial *Moto* mais une *Moto* possède par héritage le type *Véhicule*. L'objet *uneHonda* est bien un *Véhicule*.

Exemple :

- Classe Sphère;
- Classe Figures géométriques;
- Classe Cercle;
- Classe Disque.

Les Propriétés de base de l'OO (7/7)

Notion de transtypage: Il est possible de forcer le programme à 'voir' un objet comme un type différent de son type initial, c'est le *transtypage* ou *cast*.

Le transtypage ne modifie pas l'objet mais indique seulement la façon de le voir. Il y a transtypage implicite de la fille vers la mère: une *Voiture* est implicitement de type *Véhicule*.

Exemple :

- Classe Sphère;
- Classe Figures géométriques;
- Classe Cercle;
- Classe Disque.

Programmation Orientée Objet

Quelques notions (1/4)

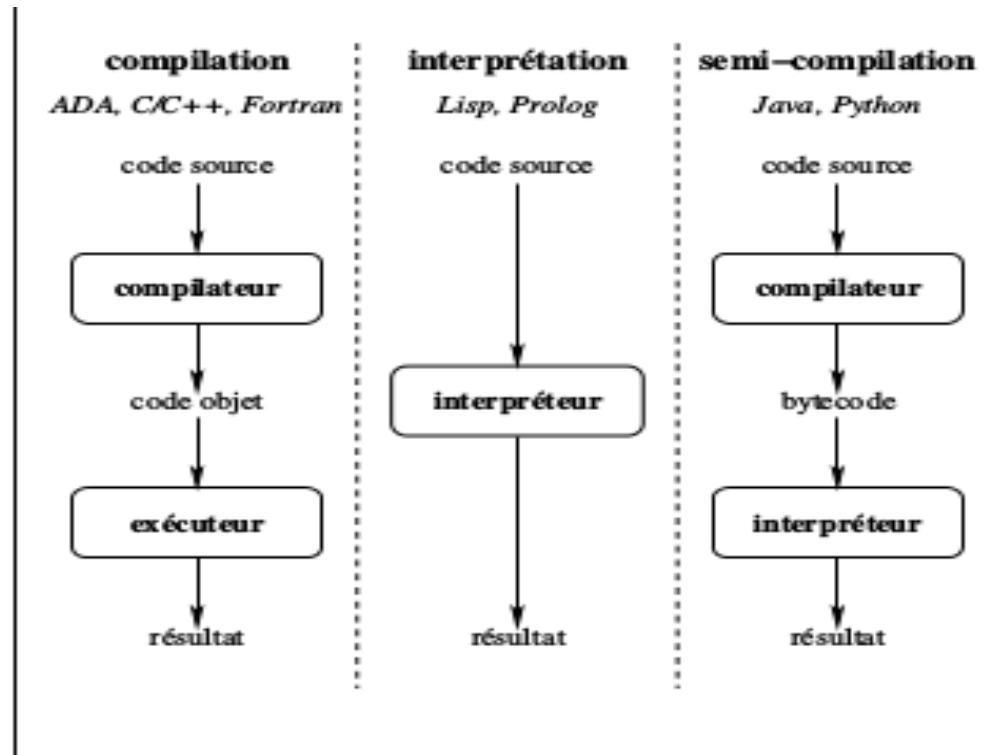
Programme ----> Traduction ----> Exécution

Un compilateur est un programme informatique qui traduit un langage, le langage source, en un autre, appelé le langage cible.

Un interpréteur est un outil informatique (logiciel ou matériel) ayant pour tâche d'analyser et d'exécuter un programme écrit dans un langage source

Programmation Orientée Objet

Quelques notions (2/4)



Programmation Orientée Objet

Quelques notions (3/4)

Compilation : La traduction de la totalité du code source en une fois. Le compilateur lit toutes les lignes du programme source et produit une nouvelle suite de codes appelé programme objet (ou code objet). Les langages Ada, C, C++ et Fortran sont des exemples de langages compilés.

L'interprétation consiste à traduire chaque ligne du programme source en quelques instructions du langage machine, qui sont ensuite directement exécutées au fur et à mesure (« au fil de l'eau »). Aucun programme objet n'est généré. Les langages Lisp et Prolog sont des exemples de langages interprétés. Un programme compilé fonctionnera toujours plus vite que son homologue interprété.

Programmation Orientée Objet

Quelques notions (4/4)

Semi-compilation : Certains langages tentent de combiner les deux techniques afin de retirer le meilleur de chacune. C'est le cas des langages Python et Java par exemple. De tels langages commencent par compiler le code source pour produire un code intermédiaire, similaire à un langage machine (mais pour une machine virtuelle), que l'on appelle bytecode, lequel sera ensuite transmis à un interpréteur pour l'exécution finale. Du point de vue de l'ordinateur, le bytecode est très facile à interpréter en langage machine. Cette interprétation sera donc beaucoup plus rapide que celle d'un code source.

Interpréteur Java = machine virtuelle Java (JVM)+ bibliothèques de classes (API)

Structuration et Conception d'un programme (1/3)

Le cycle de développement d'un programme (ou d'une application) informatique " peut se résumer ainsi :

**Problème --> Analyse --> Algorithme --> Programme -->
Compilation --> Exécution**

Structuration et Conception d'un programme (2/3)

Les interrogations clés

1. Quelles sont les données sur lesquelles le programme travaille.
2. Quels sont les services qu'il doit offrir (service = répondre à une requête d'un utilisateur / d'un autre programme).
3. Quels sont les traitements qu'il doit effectuer pour offrir ces services
4. Comment faut-il organiser et représenter les données pour que les traitements soient possibles (structure abstraite)

Structuration et Conception d'un programme (3/3)

Les interrogations clés - Suite-

5. Comment les traitements s'appellent-ils les uns – les autres / sont-ils appelés par l'interface de requête
6. Comment peut-on construire au mieux (de la façon la plus efficace en temps / en espace) les différents traitements.
7. Implémenter les structures de données dans le langage choisi / Implémenter les traitements.