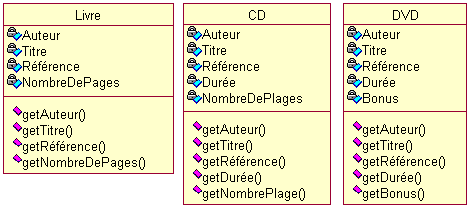
# 

# Héritage

# 

# Généralisation.

Imaginons que nous devions fabriquer un logiciel qui permet de gérer une bibliothèque. Cette bibliothèque comporte plusieurs types de documents ; des livres, des CDs, ou des DVDs. Une première étude nous emmène à mettre en œuvre les classes suivantes.



Nous remarquons que dans les trois types de documents, un certain nombre de caractéristiques se retrouvent systématiquement.

Afin d'éviter la répétition des éléments constituant chacune des classes, il est préférable de factoriser toutes ces caractéristiques communes pour en faire une nouvelle classe plus généraliste.

En effet, nous pouvons dire que, d'une façon générale, et quel que soit le type de document, il comporte au moins un titre, un auteur, etc. il semble aller de soi, que le nom de cette nouvelle classe générale s'appelle justement  Document.

Il faut ensuite proposer une relation entre les classes afin de montrer la filiation. Par exemple, il faut bien préciser qu'un Livre est aussi un Document.

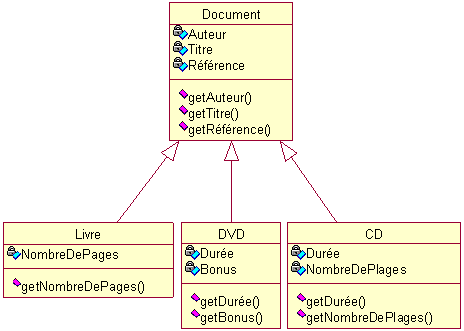
La généralisation se représente par une flèche qui part de la classe fille vers la classe mère.

Par exemple, Un Livre possède, certes un nombre de page, mais en suivant la flèche indiquée par la relation de généralisation, elle comporte également un nom d'auteur, un titre, une référence.

En fait, la classe Livre hérite de tout ce que possède la classe Document, les attributs comme les méthodes.

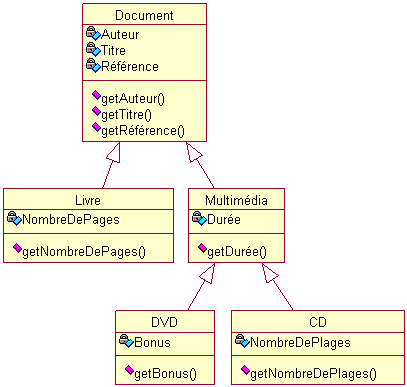
Finalement, cela correspond exactement à ce que nous avions avant sans l'héritage, sauf qu'avec cette technique, nous évitons toutes les duplications.

Toutes les caractéristiques communes n'apparaissent plus dans chacune des classes filles, alors qu'elles sont bien présentes implicitement.



Dans cet exemple, nous avons un seul niveau d'héritage, mais il est bien entendu possible d'avoir une hiérarchie beaucoup plus développée. D'ailleurs, si nous regardons de plus près, nous remarquons que nous pouvons appliquer une nouvelle fois la généralisation en factorisant la durée du support CD et du support DVD.

En fait, il s'agit dans les deux cas d'un support commun appelé Multimédia.



# Classes abstraites

Lorsque nous allons mettre en œuvre notre programme de gestion de bibliothèque, nous allons avoir finalement un certain nombre d'objets relatifs à chacun de ces documents.

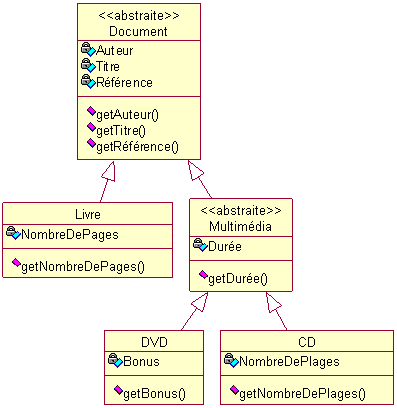
Toutefois, si nous regardons de plus près, nous n'aurons, par exemple, aucun objet relatif à la classe Document.

En effet, les objets qui intéressent le bibliothécaire, sont les livres, les CDs et les DVDs.

Déclarer un objet de la classe Document n'aurait pas de sens. Il s'agit d'une abstraction. Cette classe n'existe que pour les classes filles qui elles correspondent à quelque chose de concret.

Nous pouvons d'ailleurs appliquer le même raisonnement à la classe Multimédia. Il est d'ailleurs de notre devoir d'empêcher que ces classes puissent fournir des objets. De telles classes sont appelées **abstraites**.

Du coût, les classes classiques sont appelées **classes concrètes**. Nous ne pouvons déclarer (instancier) des objets que sur des classes concrètes.



En UML les éléments (classes ou méthodes) sont souvent représentés soit à l’aide d’un <<stéréotype>> comme ici, ou en utilisant de l’*italique*.

# Spécialisation.

Dans la conception des classes, nous pouvons avoir une démarche inverse de la généralisation, c'est-à-dire, cette fois-ci, de partir plutôt de la classe mère pour aboutir ensuite aux classes filles.

Le concept d'héritage constitue l'un des fondements de la programmation orientée objet. En particulier, il est à la base des possibilités de réutilisation des composants logiciels (en l'occurrence de classes). En effet, il vous autorise à définir une nouvelle classe, dite dérivée, à partir d'une classe existante dite de base. La classe dérivée héritera donc des potentialités de la classe de base, tout en lui ajoutant de nouvelles sans remettre en question la classe de base. Il ne sera pas utile de la recompiler, ni même de disposer du programme source correspondant (exception faite de sa déclaration).

Cette technique permet donc de développer de nouveaux outils en se fondant sur un certain acquis, ce qui justifie le terme d'héritage. Comme nous venons de le voir, plusieurs classes peuvent être dérivées de la même classe de base. En outre, l'héritage, n'est pas limité à un seul niveau : une classe dérivée peut devenir à son tour classe de base pour une autre classe. Nous voyons apparaître la notion d'héritage comme outil de spécialisation croissante.

Par exemple, imaginons que nous disposions déjà d'une classe concrète « **Personne** ». Nous pouvons alors spécialiser cette classe afin d'obtenir une classe « **Elève** » qui reste bien entendu une personne mais qui possède en plus un certain nombre de **spécificités** comme, par exemple, la gestion des notes. Dans cet exemple, vous remarquez d'ailleurs que les deux classes sont des classes concrètes.

