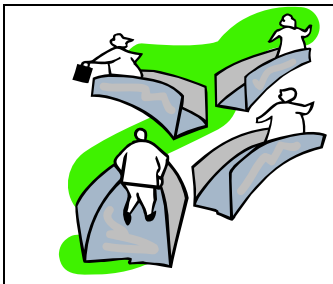
Cours WinDev Numéro 8-5



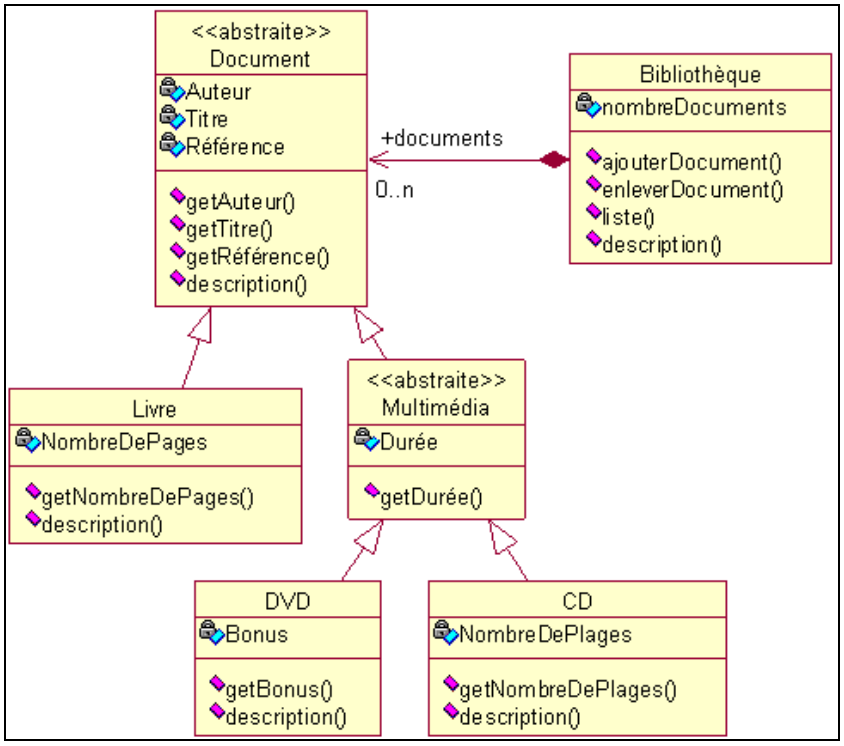
Objectifs : Le Polymorphisme.

Création de classes.
Instanciation dynamique d'objets.
Un peu de théorie.
Génération automatique du diagramme de classe.

Pré-requis : L'acquisition parfaite du principe de l'héritage.

Le polymorphisme.

Le terme polymorphisme indique qu'une entité peut apparaître suivant plusieurs formes. Dans le cas de notre hiérarchie de documents nous remarquons qu'il existe une méthode qui porte le même nom.



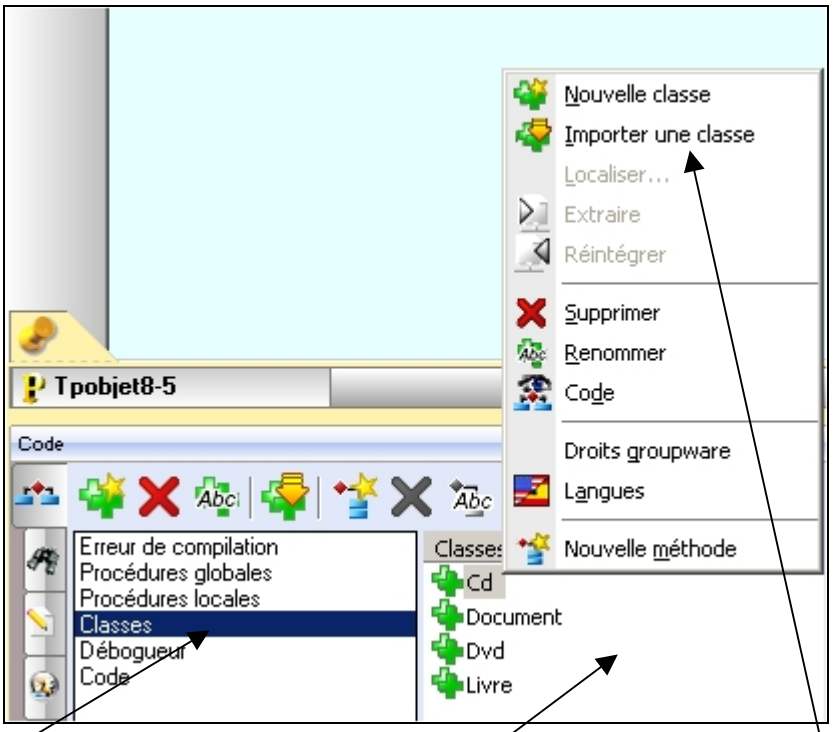
Il s'agit de description(). Elle apparaît sur toutes les classes faisant partie de la hiérarchie. Normalement, le principe même de l'héritage, c'est que lorsqu'une méthode est décrite sur une classe parente, elle est automatiquement héritée par les classes enfants. Si malgré tout, nous re-spécifions le même nom de la méthode sur les enfants, cela signifie que le comportement sera différent. En effet, la description d'un CD est différente de celle d'un livre.

Nous sommes là en présence d'un polymorphisme. La description d'un document dans le sens général n'est pas suffisante pour la description d'un livre. Cette technique s'appelle une redéfinition, c'est-à-dire que dans la classe dérivée, nous allons redéfinir une méthode qui porte le même nom avec une signature identique (polymorphisme) que la classe de base.

Il ne faut pas mélanger la redéfinition et la surdéfinition :
1. Une surdéfinition (ou surcharge) permet d'utiliser plusieurs méthodes qui portent le même nom au sein d'une même classe, avec une signature différente, pour que le système puisse s'y retrouver.
2. Une redéfinition permet de fournir une nouvelle définition d'une méthode d'une classe ascendante et ainsi de substituer la description qui en été faite. Nous avons également le même nom que la méthode parente mais surtout avec une signature rigoureusement identique.

Nous allons appliquer cela sur notre projet précédant (celui qui traite de l'héritage pour ceux qui ne suivent pas !).

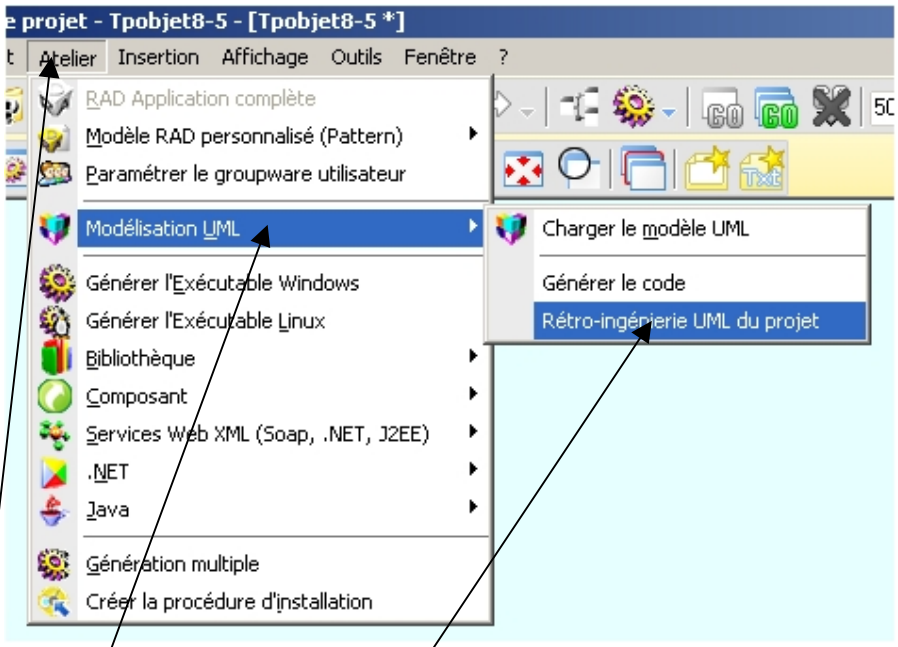
Créez un nouveau projet nommé TpObjet8-5 sans analyse. Pour comprendre un autre intérêt de la programmation orientée objet qu'est la ré-utilisation du code, nous allons réemployer les classes déjà définies dans le support précédent.



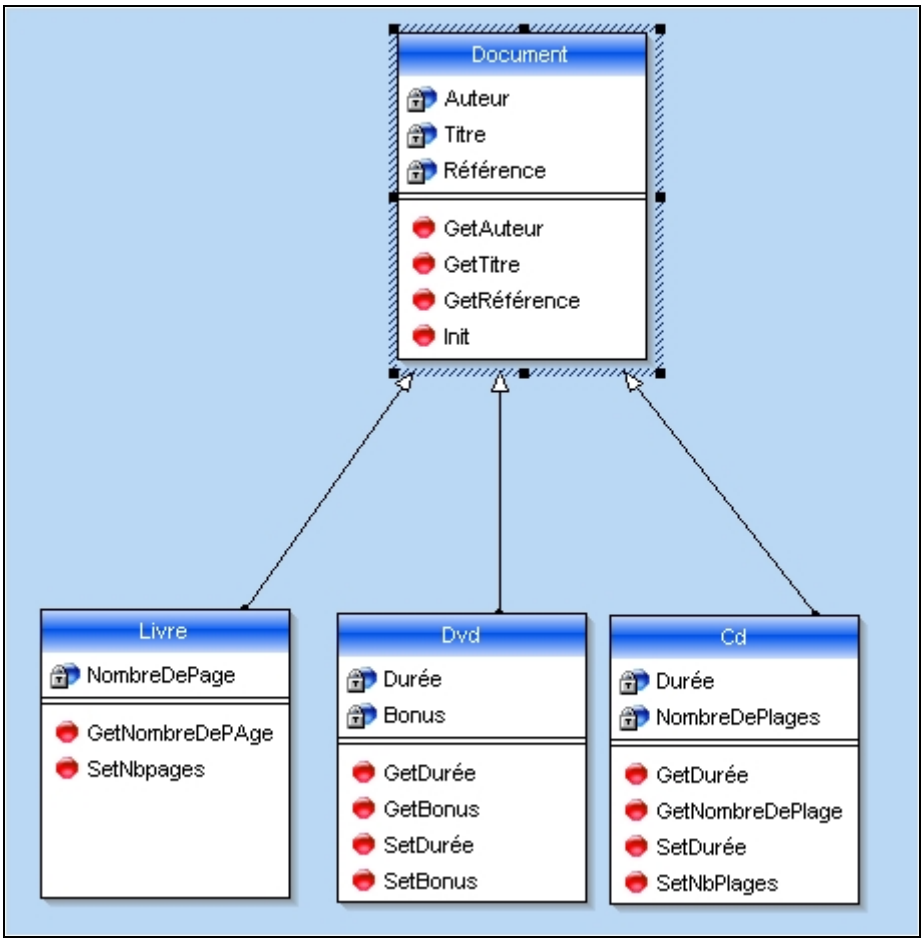
Cliquez sur **Classes** puis faites un clic droit dans la zone **Classes** et ensuite sélectionnez **Importer une classe**.
Il ne vous reste plus qu'a les retrouver et les intégrer à ce nouveau projet.

Ca y est ? on peut continuer ?

Je vais vous montrer quelque chose de sympa, on va demander à WinDev de nous générer un diagramme de classe avec les 4 classes que vous avez créer.

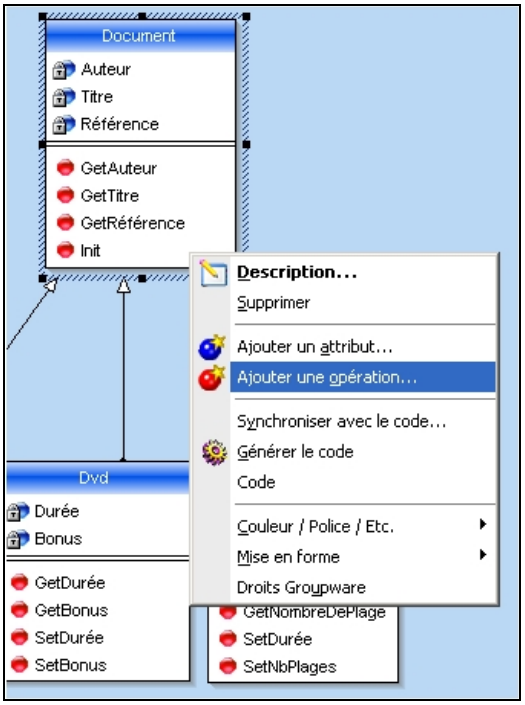


Cliquez sur Atelier->Modélisation Uml->Rétro-Ingénierie UML du projet, dans l'assistant donnez un nom a votre diagramme de classe et cliquez sur terminer.
Là, WinDev vient de vous faire votre diagramme de classe, il faut le mettre en forme pour avoir à peut-prêt ceci :



Désolé pour ceux qui ont des imprimantes sans cartouches couleurs séparées, je viens de vous exploser le bleu !

On va ajouter une méthode description dans la classe document, faites un clic droit sur **Document** puis sur **Ajouter une opération** (j'aurais préféré lire méthode à la place d'opération …!)



Vous lancez un assistant qui vous demande de donner un nom à l'opération (grrr… méthode !) Inscrivez **Description**, pour le libellé je vous laisse faire.
L'écran suivant demande si notre méthode va renvoyer un résultat et si elle prend des paramètres d'entrée, en fait ne touchez à rien car la méthode description va juste renvoyer un texte. Donc passez à l'écran suivant sans rien modifier.
Ce nouvel écran vous demande la visibilité de la méthode Description, pour nous elle reste publique.

L'opération est abstraite (virtuelle), restez avec non.

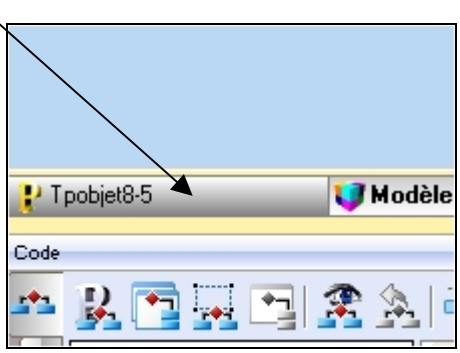
Attention pour les puristes de l'objet ici il y a un amalgame ou abus de langage entre les mots clés Abstrait et Virtuel. Dans d'autres langages 100% objet la vision est différente : Une classe est abstraite ou virtuelle, ce sont 2 vues d'esprit différentes.

La méthode Description est-elle partagée par toutes les instances … Non

En fait, vous ne touchez à rien ici aussi !

Maintenant notre diagramme Uml comporte un objet Document qui a une nouvelle méthode.

Repassez sur le projet en cliquant ici :



Ouvrez la classe Document et voyez ce que l'analyse Uml a fait : Il vous a écrit la méthode et rajouté quelques commentaires…Comme vous le voyez les modifications faites dans l'analyse Uml sont impactés directement dans le code.

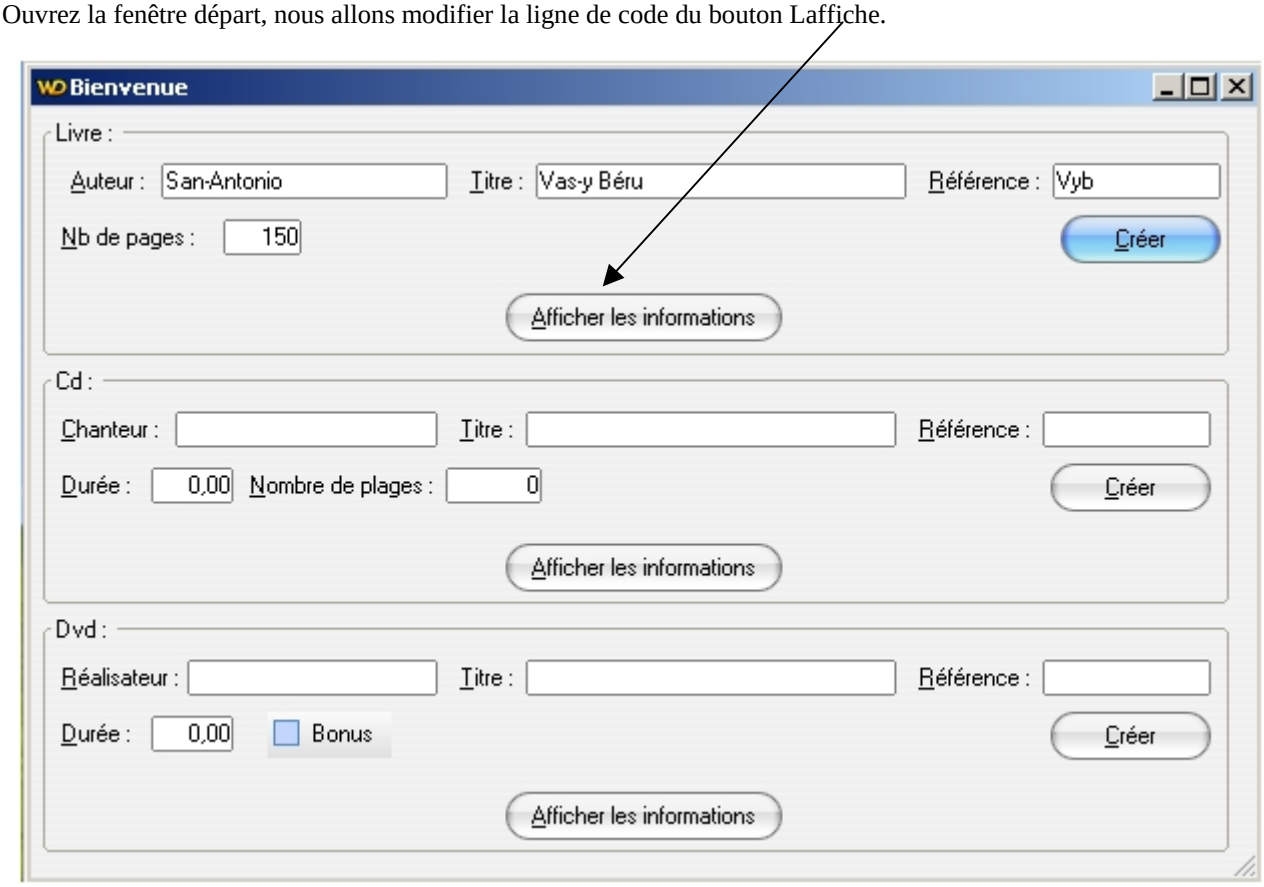
Modifiez le code de la méthode **Description** de la classe **Document** de la façon suivante :

```
Méthode Description *
PROCEDURE Description ( )
  Info("L'auteur : "+:Auteur+" le titre : "+:Titre+" la référence : "+:Référence)
```

Il nous manque une fenêtre pour dynamiser ce code, vous savez ce que l'on va faire ? On va récupérer la fenêtre départ du projet précédent (Tpobjet8-4 – L'héritage) et on va juste modifier quelques lignes de code.

Cliquez sur **Fichier**->**Importer**->**Des éléments WinDev et leurs dépendances…** Recherchez votre dossier de stockage du projet et sélectionnez la fenêtre **départ**.

Ouvrez la fenêtre départ, nous allons modifier la ligne de code du bouton Laffiche.



Enlever tout et remplacez par ceci :

```
Clic sur Laffiche
livl:Description()
```

Par cette méthode on active l'affichage de l'auteur, du titre et de la référence du livre, donc remplissez les champs cliquez sur créer pour activer l'instanciation de l'objet **liv1** et cliquez sur **Afficher les informations**.

Voyez le résultat : c'est génial, non ? Il y a un inconvénient majeur, c'est que cette méthode va très bien pour afficher les informations relatives à un livre, mais en est-il de même pour un Cd ou un Dvd ? La réponse est non, car on peut afficher le terme auteur pour l'écrivain du livre, mais il serait plus judicieux d'afficher Chanteur pour la description du Cd et Réalisateur pour le Dvd.

Nous allons donc faire du polymorphisme, comment ? En réécrivant la méthode Description pour les classes Cd et Dvd.
Ouvrez la classe Cd et créez une méthode description :

```
Méthode Description                                   Erreur : manuel
PROCEDURE VIRTUELLE Description()
  Info("Le chanteur : "+:GetAuteur()+" le titre : "+:GetTitre()+" la référence : "+:GetRéférence())
```

Dans le code du bouton Afficher les informations saisissez :

```
Cd1:Description()
```

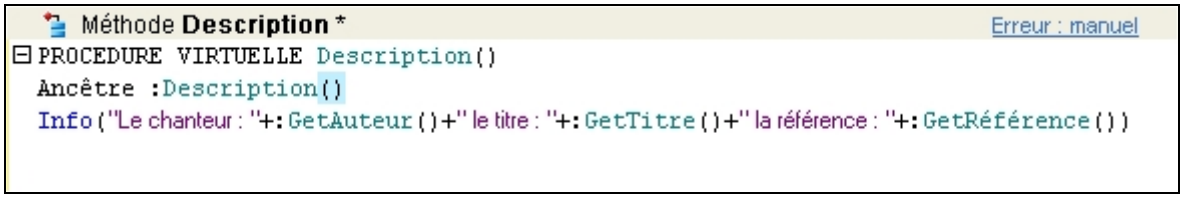
Faites de même pour dvd.

Dans le code du bouton Afficher les informations saisissez :

```
Dvd1:Description()
```

Maintenant, inscrivez des informations dans chaque rubrique de chaque item (Livre, Cd, Dvd) créez les instances de chaque objet et cliquez sur les boutons **Afficher les informations**. Vous remarquez que l'appel de la méthode est le même, mais le comportement diffère.

Essayez ceci : dans le code description de la classe cd rajoutez la ligne suivante ;



Le fait de mettre **ancêtre** fait que la méthode de la classe Mère (document) va s'exécuter.

Lancez la fenêtre et testez.

Vous venez de comprendre le polymorphisme (du moins, je l'espère !).

A bientôt.