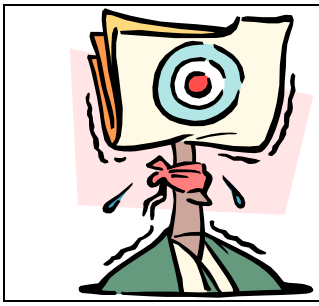
Cours WinDev Numéro 8-3



Objectifs : Les classes composées de tableau d'objets.

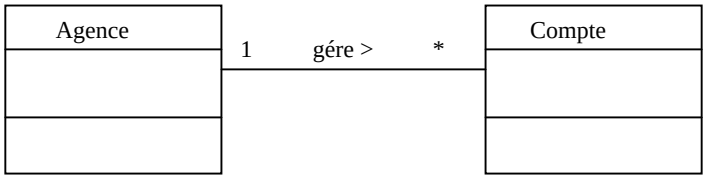
Création de classes.
Instanciation dynamique d'objets.
Utilisation des tableaux.
Un peu d'UML.

Pré-requis : L'acquisition parfaite des cours objets précédents.

Introduction:

On souhaite gérer les comptes bancaires des agences d'une banque.
Une agence gère plusieurs comptes et un compte est géré par une seule agence.

Réalisons le diagramme de classes relatif à cet exemple :



Nous avons vu lors du cours précédent comment gérer dans une classe les multiplicités 1 (= 1..1) et 0..1, c'est à dire comment gérer une référence à la classe Agence dans la classe Compte.
Nous allons voir dans ce cours comment prendre en compte les autres multiplicités (n, m..n, * ou 0..* et 1..*) dans une classe.

Comment représenter dans la classe Agence le fait qu'une agence gère plusieurs comptes ?

La classe agence va contenir un ensemble de références d'objet. Nous allons choisir de représenter cet ensemble par un tableau de références d'objets dans un premier temps puis nous verrons qu'il est possible d'utiliser une collection de références d'objets.

Voyons un peu du côté d'UML …

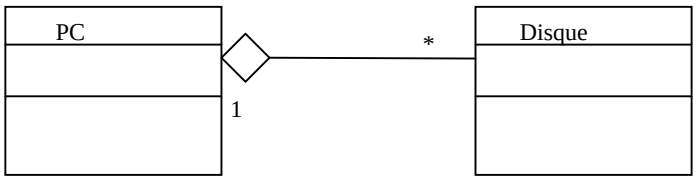
La relation peut être une relation d'association comme la relation Agence-Compte , une relation d'agrégation ou de composition. La multiplicité « plusieurs » sera gérée de manière presque identique.

L'association représente des relations structurelles entres classes : elle possède un nom qui symbolise l'association.

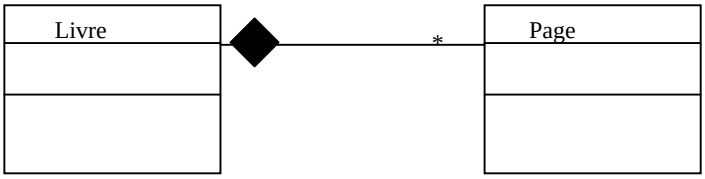
L'agrégation est une relation particulière où une classe joue un rôle prépondérant. Elle traduit une relation d'appartenance. C'est à dire le fait qu'un objet de la classe est composé d'un ou plusieurs objets d'une autre classe.

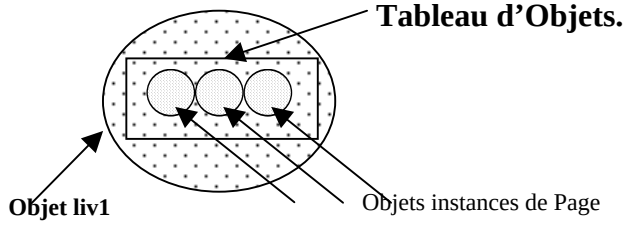
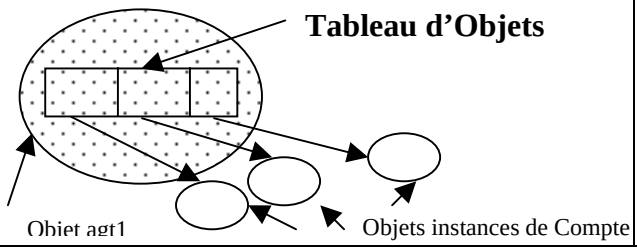
La composition est une agrégation particulière : la vie du composant est liée à celle du composé. L'objet composant a la même durée de vie que le composé.

Exemple d'agrégation : un PC est composé de plusieurs disques durs.



Exemple de composition : un livre est composé de plusieurs pages.

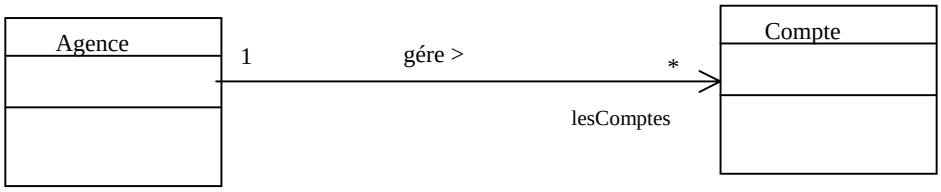


RELATION DE COMPOSITION.	
	Les objets du tableau n'existent pas en dehors de l'objet liv1. Les objets du tableau sont inclus dans l'objet liv1.
RELATION D'ASSOCIATION OU D'AGREGATION.	
	Les objets comptes et l'objet agt1 ont une existence indépendante. Mais ils sont liés.

N.B. : seul le 2ème cas sera étudié dans ce cours.

Rappel sur la navigabilité :

Dans une relation, on peut en général naviguer d'une classe à l'autre. Il est parfois nécessaire à une étape de la conception d'indiquer que la navigabilité n'est plus bidirectionnelle. Dans notre exemple, il est nécessaire que l'agence connaisse les comptes qu'elle gère, mais il n'est pas nécessaire que le compte connaisse l'agence qui le gère. Ce choix est représenté par une navigabilité restreinte.



Voyons comment définir et utiliser les classes décrites dans ce diagramme de classe.

Soit la classe Compte suivante :

```
classe Compte
  privé
        numéro : entier
        solde : réel
  public
        procédure Init(entrée numéroSai : entier )          //initialise les données du compte
        procédure Créditer(entrée montantSai : réel)        //crédite le compte du  montant
        procédure Débiter(entrée montantSai : réel)         //débite le compte du  montant
        fonction GetNum() : entier                          //retourne le numéro du compte
        fonction GetSolde() : réel                          //retourne le solde du compte
fin classe
```

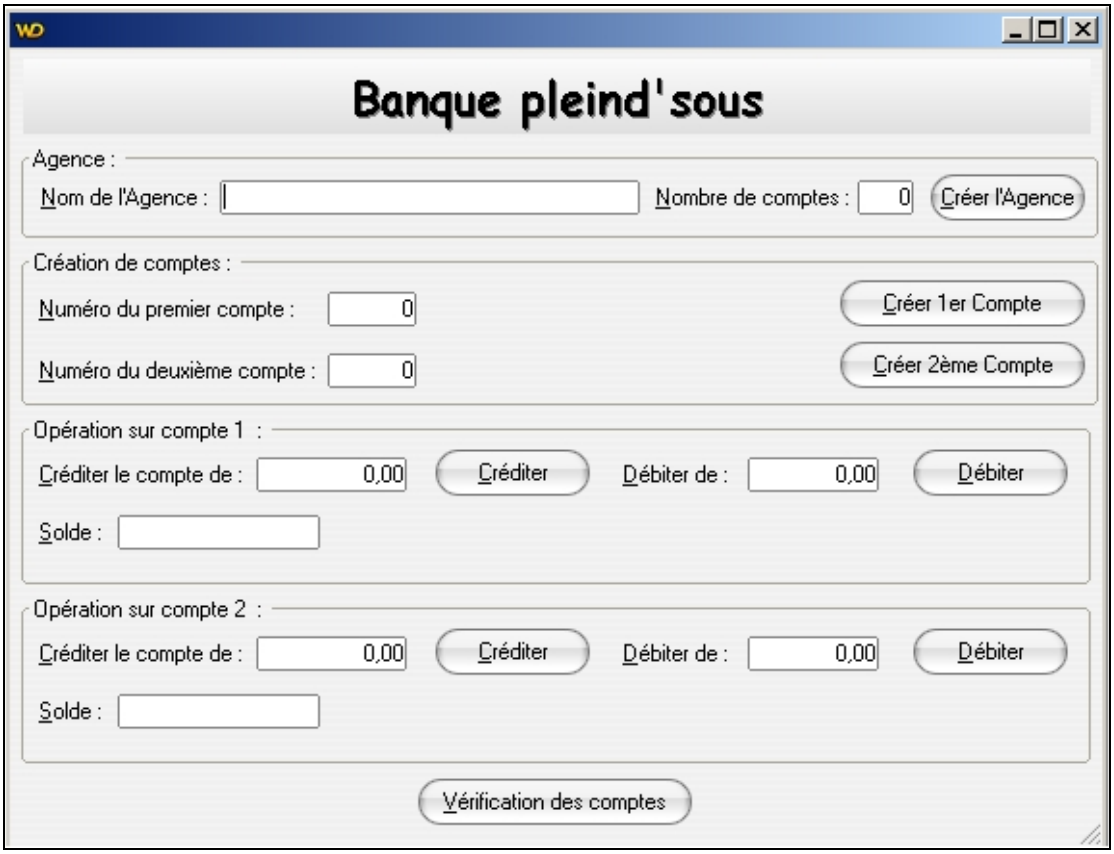
La classe Agence :

```
classe Agence
  privé
        nom : chaîne
        lesComptes : tableau [1..100] de Compte
        nbCpt : entier
  public
        procédure Init(nom : chaîne)                        //initialise les données de l'agence
        fonction GetNom() : chaîne                          //retourne le nom de l'agence
        fonction GetNbCpt () : entier                       //retourne le nombre de comptes gérés par
                                                            l'agence
        fonction ExisteNumCpt (numCpt : entier) : booléen   //retourne vrai si le numéro
                                                            //de compte existe, faux sinon
        fonction ExisteCpt (cpt : Compte) : booléen         //retourne vrai si le compte existe,
                                                            //  faux sinon
        fonction AjoutCpt(cpt: Compte) : booléen            // ajoute le compte si le compte
                                                            // n'existe pas déjà dans le tableau et s'il
                                                            // reste de la place dans le tableau.
                                                            // retourne vrai si le compte a pu être ajouté,
                                                            // retourne faux sinon.
        Procédure Listecompte()                             // Donne des information sur les comptes.
fin classe
```

Voilà, la théorie est posée, maintenant vous allez passer à la pratique.

Créez un nouveau projet nommé Tpobjet8-3, il ne comporte aucune analyse. Ensuite définissez les classes Compte et Agence. Créez les attributs et méthodes de chaque classe (vous écrirez le code des méthodes plus tard).

Voici la seule fenêtre du projet (nommée **départ**):



La zone "Agence" contient :
Un champ de saisie nommé : NomAgence.
Un champ de saisie nommé : NbComptes.
Un bouton nommé : BcréerAgence.

La zone "Création de comptes" contient :
Un champ de saisie nommé : Numducompte1.
Un champ de saisie nommé : Numducompte2.
Un bouton nommé : BcréerCompte1.
Un bouton nommé : BcréerCompte2.

La zone Opération sur compte1 contient :
Un champ de saisie nommé : CréditCompte1.
Un champ de saisie nommé : DébitCompte1.
Un champ de saisie nommé : Vsolde1.
Un bouton nommé : BcréditerCompte1.
Un bouton nommé : BdébiterCompte1.

La zone Opération sur compte2 contient :
Un champ de saisie nommé : CréditCompte2.
Un champ de saisie nommé : DébitCompte2.
Un champ de saisie nommé : Vsolde2.
Un bouton nommé : BcréditerCompte2.
Un bouton nommé : BdébiterCompte2.

Et enfin un Bouton nommé : BVérifCompte

Vous arriverez à vous y retrouver ? Je pense quoi vous avez compris le rôle de cette fenêtre, nous allons créer un objet agence et 2 objets compte qui vont êtres manipulés par certaines méthodes de la classe agence et contenus dans un tableau d'objet compte dynamique.

Passons au code de la classe agence :

```
Déclaration de Agence
Agence est une classe
    PRIVÉ
        Nom est une chaîne
        LesComptes est un tableau de 100 Compte dynamique
        NbCpt est un entier
FIN
```

La classe Agence contient 3 attributs :

Nom a qui nous passerons le nom de l'agence
LesComptes est un tableau d'objets Compte, il est déclaré dynamique pour que ce soit la référence des objets compte qui soit inscrites dans le tableau et non une recopie des valeurs. Le gros avantage c'est que si l'objet est modifié hors tableau ses valeurs dans le tableau seront à jour. Le tableau contient l'adresse de l'objet et non une copie de l'objet.

```
Méthode Init
PROCEDURE Init(Lenom)
  :Nom=Lenom
  :NbCpt=0

Méthode GetNom
FONCTION GetNom()
  RENVOYER :Nom

Méthode GetNbCpt
FONCTION GetNbCpt()
  RENVOYER :NbCpt
```

Plutôt que d'utiliser le constructeur, une méthode Init a été crée. Elle récupère le nom de l'agence et l'affecte à l'attribut privé Nom, elle initialise NbCpt (le compteur de compte) à zéro.

On trouve ensuite 2 accesseurs GetNom et GetNbCpt, je pense que vous comprenez leurs usages !

```
Méthode ExisteNumCpt
FONCTION ExisteNumCpt(numcpt)
  ind est un entier=1
  trouvé est un booléen=Faux

TANTQUE ind<=:GetNbCpt() ET trouvé=Faux
    SI :LesComptes[ind]:GetNum()=numcpt ALORS
        trouve=Vrai
    FIN
    ind++
FIN
  RENVOYER trouvé
```

Cette méthode ExisteNumCpt recherche dans le tableau d'objet "LesComptes" si un numéro de compte existe déjà.
Remarquez ceci ":LesComptes[ind]:GetNum()=Numcpt".
La méthode GetNum() de l'objet à l'indice [ind] du tableau d'objet est activée. C'est hyper élégant la programmation objet, vous ne trouvez pas ?

Méthode **ExisteCpt** Erreur : manuel

```
FONCTION ExisteCpt(cpt est un objet Compte)
  ind est un entier=1
  trouvé est un booléen=Faux

TANTQUE ind<=:GetNbCpt() ET trouvé=Faux
     SI :LesComptes[ind]:GetNum()=cpt:GetNum() ALORS
         trouvé=Vrai
     FIN
     ind++
FIN
  RENVOYER trouvé
```

Méthode **AjouteCpt** Erreur : manuel

```
FONCTION AjouteCpt(cpt est un objet Compte )
SI :ExisteCpt(cpt) OU :GetNbCpt()=100 ALORS
     RENVOYER Faux
  SINON
     :NbCpt=:NbCpt+1
     :LesComptes[:NbCpt]=cpt
     RENVOYER Vrai
FIN
```

Méthode **Listecompte** Erreur : manuel

```
PROCEDURE Listecompte()
  i est un entier
POUR i=1 A :GetNbCpt()
     Info ("Pour le compte "+:LesComptes[i]:GetNum()+" Le solde est de : "+:LesComptes[i]:GetSolde())
FIN
```

La méthode ExisteCpt a un fonctionnement différent de la méthode ExisteNumCpt pour un résultat identique. Je vous laisse la comprendre.

La méthode Ajoutecpt ajoute un objet compte dans le tableau d'objet compte.

La méthode Listecompte scanne le tableau et affiche le détail de chaque compte.

Voilà pour la classe Agence, voici la classe Compte :

Déclaration de **Compte**

```
Compte est une classe
    PRIVÉ
        numéro est un entier
        solde est un réel
FIN
```

Rien de particulier pour la déclaration de la classe. Ni pour les 2 accesseurs ci dessous.

Méthode **GetSolde**

```
FONCTION GetSolde()
  RENVOYER :solde
```

Méthode **GetNum**

```
FONCTION GetNum()
  RENVOYER :numéro
```

```
Méthode Init
PROCEDURE Init(NuméroSai)
:numéro=NuméroSai
:solde=0

Méthode Créditer
PROCEDURE Créditer(MontantSai)
:solde=:solde+MontantSai
Info("Votre compte "+:numéro+" vient d'être crédité de : "+MontantSai)

Méthode Débiter
PROCEDURE Débiter(MontantSai)
:solde=:solde-MontantSai
Info("Votre compte "+:numéro+" vient d'être débité de : "+MontantSai)
```

Ces 3 méthodes permettent :

L'initialisation des attributs numéro et solde
Crédite le compte d'un montant passé en paramètre.
Débite le compte d'un montant passé en paramètre.

Les classes sont maintenant définies voyons dans le code de la fenêtre comment les mettre en œuvre.

Il faut déclarer les objets de façon globale à la fenêtre :

```
Déclarations globales de depart                                   Erreur
cpt1 est un objet Compte dynamique//Non, il n'y a pas de constructeur !
cpt2 est un objet Compte dynamique//Non, il n'y a pas de constructeur !
lagence est un objet Agence  // Non, il n'y a pas de constructeur !
```

Comme vous le voyez les 2 objets compte sont dynamiques, donc on travaille dessus par référence et non par valeur.

Voici maintenant, tous les codes de la fenêtre. Je vous laisse coder.

```
Clic sur BCréerAgence                                   Erreur : manuel
lagence:Init(NomAgence) // A la place du constructeur il y a une méthode Init !
```

```
Clic sur BCréerCompte1                                  Erreur : manuel
cptl=allouer un Compte
cptl:Init(Numducomptel) // on crée le compte
SI lagence:AjouteCpt(cptl) =Vrai ALORS
    Info("Votre compte "+Numducomptel+" a été créé")
SINON
    Info("Votre compte  "+Numducomptel+" n'a pas été créé")
FIN
NbComptes=lagence:GetNbCpt() // on fait afficher le nombre de comptes de l'agence
```

Clic sur **BCréerCompte2** Erreur : manuel

```
cpt2=allouer un Compte
cpt2:Init(Numducompte2) // on crée le compte

SI lagence:AjouteCpt(cpt2) =Vrai ALORS
    Info("Votre compte "+Numducompte2+" a été créé")
SINON
    Info("Votre compte "+Numducompte2+" n'a pas été créé")
FIN
NbComptes=lagence:GetNbCpt() // on fait afficher le nombre de comptes de l'agence
```

Clic sur **BCréditerCompte1**

```
cpt1:Créditer(CreditComptel)
Vsoldel=cpt1:GetSolde()
```

Clic sur **BDébiterCompte1**

```
cpt1:Débiter(DébitComptel)
Vsoldel=cpt1:GetSolde()
```

Clic sur **BCréditerCompte2**

```
cpt2:Créditer(CreditCompte2)
Vsolde2=cpt2:GetSolde()
```

Clic sur **BDébiterCompte2**

```
cpt2:Débiter(DébitCompte2)
Vsolde2=cpt2:GetSolde()
```

Clic sur **BVérifCompte**

```
lagence:Listecompte()
```

Maintenant vous pouvez lancer le projet.
Saisissez un nom d'agence par exemple : "Agence Oncle Picsou" et cliquez sur le bouton Créer l'Agence.
Saisissez un numéro de compte pour le premier et un autre pour le second, puis cliquez sur les 2 boutons de création des comptes.

Je vous laisse débiter ou créditer les comptes selon vos préférences. N'oubliez pas la vérification des comptes.

Pour tester la différence entre le passage par référence ou par valeur enlevez "dynamique" à la définition du tableau dans la classe agence. Relancer le projet, mettez les valeurs vérifiez les comptes et oupssss !!

Note pour les puristes, j'aurais pu faire différemment, n'avoir qu'un seul objet compte et le copier par valeur dans le tableau.

Merci à ceux sur le forum "fr.comp.developpement.agl.windev" qui m'ont donné des éclairages sur certains points objets théoriques

A bientôt, pour un cours sur l'héritage.