COURS WINDEV NUMERO 8

15/02/2015

La Programmation Orientée Objet

Création d'une classe.

Instanciation d'un objet.

Diagramme Uml.

Fonctions de dessins.

…etc…

Cours Windev numéro 8

VERSION 19

L'objectif de cette leçon est de vous familiariser avec la programmation orientée objet (POO).

PROGRAMMATION ORIENTEE OBJET, (UN PEU DE THEORIE, ÇA NE FAIT PAS DE MAL ET C'EST GRATUIT !)

De manière superficielle, le terme « ***orienté objet*** », signifie que l'on organise le logiciel comme une collection d'objets dissociés comprenant à la fois une structure de données (***attributs)*** et un comportement (***méthodes***) dans une même entité (***encapsulation***). Exemple : une voiture peut avoir une certaine couleur et en même temps possède un comportement qui sera le même pour toutes les autres voitures, comme accélérer. Ce concept est différent de la programmation conventionnelle dans laquelle les structures de données et le comportement ne sont que faiblement associés.

Encapsulation.

L'encapsulation est le principe qui permet de regrouper les attributs et méthodes au sein d'une classe. Cette notion est aussi associée au système de protection qui permet de contrôler la visibilité d'une variable ou d'une méthode. En d'autres termes, cela signifie que chaque fois que vous définissez un ***membre*** d'une classe (***attribut*** ou ***méthode***), vous devez indiquer les droits d'accès quant à l'utilisation de ce membre.

Les objets.

Chaque objet a une identité et peut être distingué des autres. Deux pommes ayant les mêmes couleurs, forme et texture demeurent des pommes individuelles; une personne peut manger l'une, puis l'autre. De la même façon, des jumeaux identiques sont deux personnes distinctes, même si elles se ressemblent. Le terme identité signifie que les objets peuvent être distingués grâce à leur existence inhérente et non grâce à la description des propriétés qu'ils peuvent avoir. Nous utiliserons l'expression ***instance*** d'objet pour faire référence à une chose précise, et l'expression ***classe*** d'objets pour désigner un groupe de choses similaires. En d'autres termes, deux objets sont distincts même si tous leurs attributs (nom, taille et couleur par exemple) ont des valeurs identiques (deux pommes vertes sont deux objets distincts).

Les classes.

La classification signifie que les objets ayant la même structure de donnée (***attributs***) et le même comportement (***méthodes***) sont regroupés en une classe. Les objets d'une classe ont donc le même type de comportement et les mêmes attributs. En groupant les objets en classe, on abstrait un problème. Les définitions communes (telles que le nom de la classe et les noms d'attributs) sont stockées une fois par classe plutôt qu'une fois par instance. Les méthodes peuvent être écrites une fois par classe, de telle façon que tous les objets de la classe bénéficient de la réutilisation du code. Par exemple, toutes les ellipses partagent les procédures de dessin, de calcul d'aire ou de test d'intersection avec une ligne.

Les attributs.

Un attribut est une valeur de donnée détenue par les objets de la classe. ***Couleur*** et ***Poids*** sont des attributs des objets relatifs à ***Voiture***. Chaque attribut à une valeur pour chaque instance d'objet. Par exemple, l'attribut ***Couleur*** porte la valeur ***rouge*** dans l'objet ***Mercedes*** alors que pour l'objet ***Clio*** la valeur de l'attribut ***Couleur*** est ***verte***. Les instances peuvent avoir des valeurs identiques ou différentes

pour un attribut donné. Chaque nom d'attribut est unique à l'intérieur d'une classe. Ainsi, la classe ***Voiture*** et la classe ***Pomme*** peuvent avoir chacune un attribut ***Couleur***.

Les méthodes.

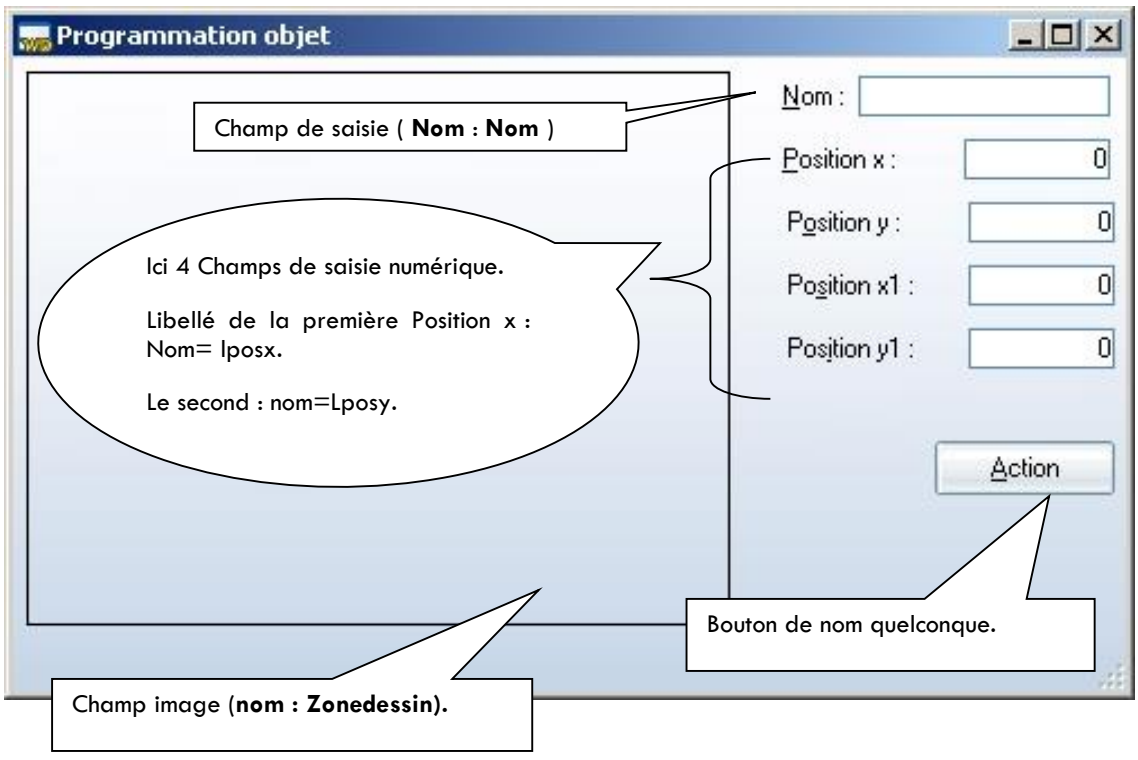
Une méthode est une fonction ou une transformation qui peut être appliquée aux objets ou par les objets dans une classe. ***Accélérer*** et ***freiner*** sont des méthodes de la classe ***Voiture***. Tous les objets d'une même classe partagent les mêmes méthodes. Chaque méthode a un objet cible comme argument implicite (c'est l'objet lui-même ***this***, elle peut donc accéder à chacun des attributs). La même opération peut s'appliquer à de nombreuses classes différentes. On dit qu'elle est ***polymorphe***, c'est-à-dire qu'elle prend différentes formes dans des classes différentes. Une méthode peut avoir des arguments, en plus de son objet cible.

FINI POUR LA THEORIE VOYONS CELA AVEC UN EXEMPLE FAIT AVEC WINDEV

1. Nom du projet : TpObjet1
2. Pas d'analyse.

Créez une nouvelle fenêtre que vous nommerez « **Départ** » et qui sera la première fenêtre du projet.

Voici la fenêtre une fois ajouté tous les éléments nécessaires à ce support :



Le but de cette fenêtre est de faire afficher un cercle dans la zone image avec les coordonnées que nous lui passerons

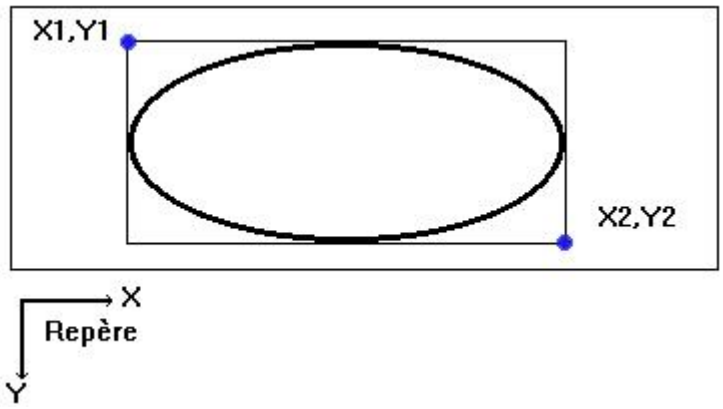
Nom : C'est le nom que nous donnerons à notre objet cercle

Position X : Abscisse du coin haut gauche du rectangle dans lequel le cercle est inscrit. Ces coordonnées sont exprimées en pixels.

Position y : Ordonnée du coin haut gauche du rectangle dans lequel le cercle est inscrit. Ces coordonnées sont exprimées en pixels.

Position x1 : Abscisse du coin bas droit du rectangle dans lequel le cercle est inscrit. Ces coordonnées sont exprimées en pixels.

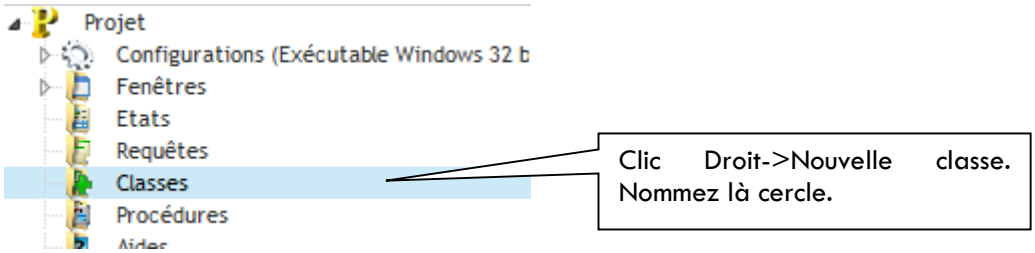
Position y1 : Ordonnée du coin bas droit du rectangle dans lequel le cercle est inscrit. Ces coordonnées sont exprimées en pixels.



Le bouton Action : Il contiendra le code nécessaire au tracé du cercle

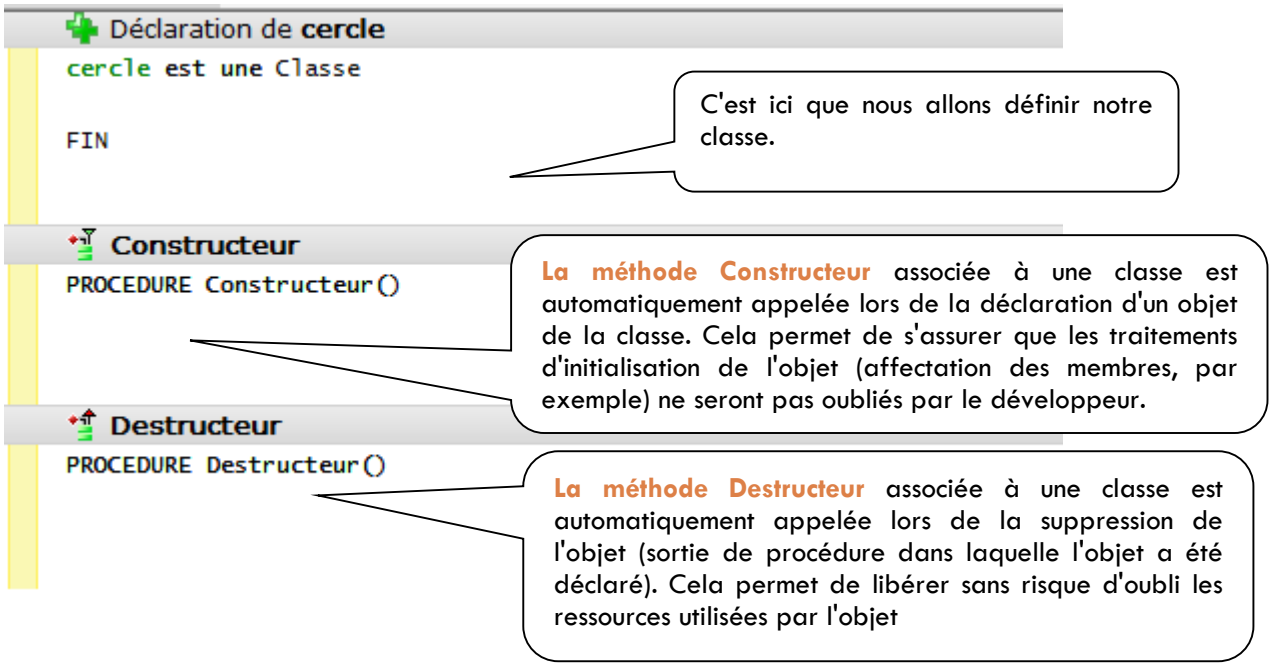
Voilà, maintenant que le décor est planté passons à l'action !

Il nous faut créer une classe nommée cercle qui aura 4 membres : les coordonnées et une méthode dessiner.



Voilà le résultat :

L'éditeur de code pour la classe cercle devrait ressembler à cela :



Nous allons commencer à coder les membres (où attributs de la classe) :

```
cercle est une Classe

PRIVÉ
      posx1 est un entier
      posy1 est un entier
      posx2 est un entier
      posy2 est un entier
      lenom est un chaîne

FIN
```

Comme vous le voyez Cercle est une classe ayant 5 attributs privés (par analogie vous pouvez imaginer 5 variables) elles sont privées c'est à dire qu'elles sont inaccessibles de l'extérieur de la classe (C'est le principe de l'encapsulation).

- **Privé** : accès autorisé depuis un code de la classe

- **Protégé** : accès autorisé depuis un code de la classe ou un code d'une classe dérivée

- **Public** (par défaut) : accès autorisé depuis n'importe quel code de la classe ou du projet.

Modifions le constructeur :

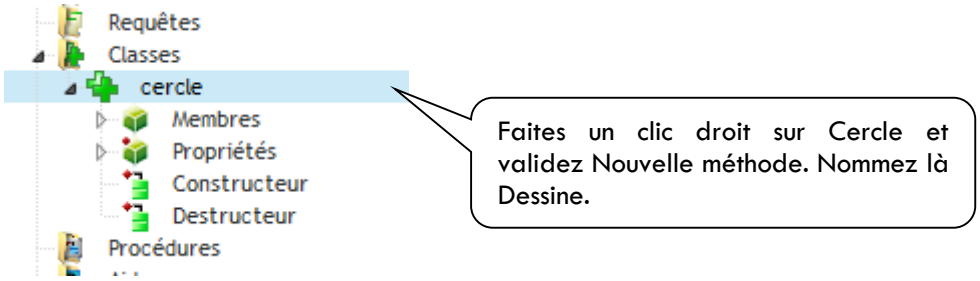
```
PROCEDURE Constructeur(x1,y1,x2,y2,nom)
:posx1=x1
:posy1=y1
:posx2=x2
:posy2=y2
:lenom=nom
```

Le rôle du constructeur est d'initialiser le futur objet avec ou sans paramètre.

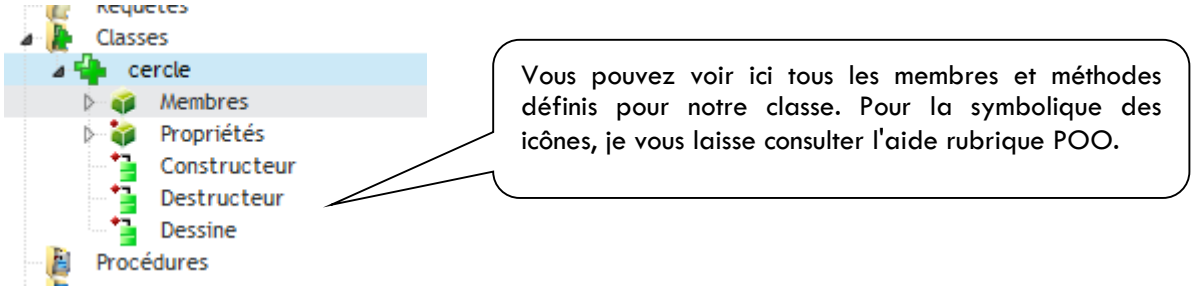
Ici 5 paramètres sont passés (c'est les 4 coordonnées que l'on fera saisir à l'utilisateur ainsi que le nom donné à notre cercle).

Nous avons donc x1,y1,x2,y2 et nom. Ces variables vont être affectées à des variables de classe privées (posx1,posy1…). Comme ces variables sont privés, il faut les préfixer avec le caractère ":".

Créons la méthode de dessin :



Vous devriez, maintenant avoir ceci :



Voici le code de la méthode Dessine :

```
PROCEDURE Dessine()
dDébutDessin(Depart.Zonedessin)
dCercle(:posx1,posy1,posx2,posy2,iRougeClair,iVertClair)
dTexte(:posx2,posy2,:lenom,iJauneFoncé)
```

Explication du code :

// pour pouvoir dessiner on initialise la zone image

dDébutDessin(depart.Zonedessin)

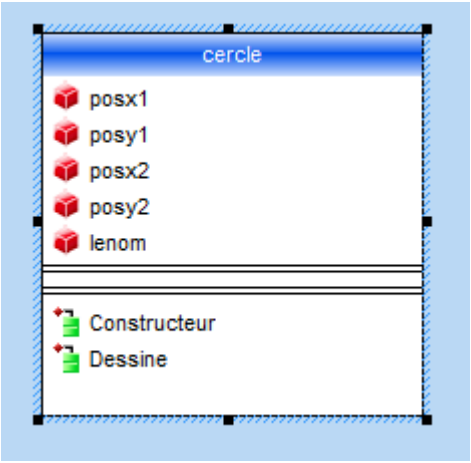
// La procédure dCercle créé le cercle de coordonnées et couleurs définies

dCercle(:posx1,:posy1, :posx2, :posy2,*iRougeClair*,*iVertClair*)

//La fonction dTexte dessine un texte à une position donnée relative à la zone image

dTexte(:posx2, :posy2, :lenom, *iJauneFoncé*)

Voilà notre classe est finie. Vous savez que WinDev vous permet de la visualiser au format UML ? On va voir ? C'est parti : Cliquez sur Projet->Modélisation UML->Rétro-ingénierie du projet. Suivez l'assistant et vous verrez le diagramme de classe correspondant à votre classe.



Vous retrouverez ce diagramme dans l'Explorateur de projet.

La modification (ou l'ajout) d'un membre dans le diagramme sera répercutée immédiatement dans le code et réciproquement, essayez voir !

Il nous faut maintenant créer une instance de cette classe pour la mettre en œuvre. C'est le code du bouton Action qui va réaliser tout ça.

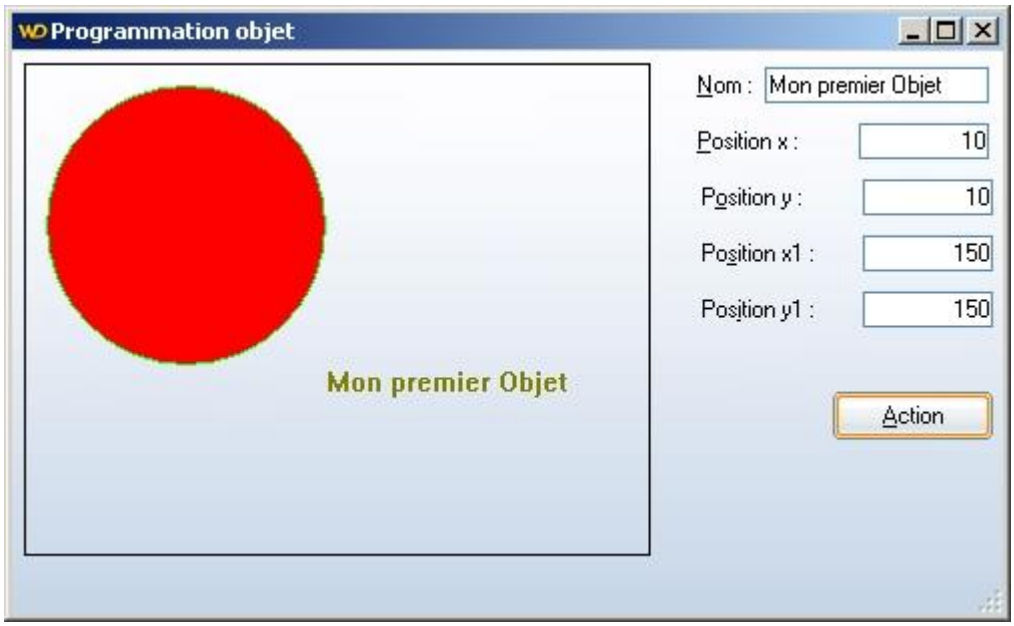
```
moncercle est un objet cercle(Lposx,Lposy,Lposx1,Lposy1,Nom)
moncercle:Dessine()
```

Etudions ce code :

La première ligne instancie un objet moncercle de type Cercle en passant au constructeur les 5 paramètres.

Ensuite, nous exécutons la méthode "dessine" de l'objet.

Passons à la pratique :



Renseignez les champs, cliquez sur le bouton et oh….!!! Le Zouli dessin. Vous êtes vraiment très fort !

Trêve de plaisanterie, vous avez tout compris ? Comme vous le voyez, ce n'est pas très compliqué de créer et manipuler une classe avec WinDev. Pour le prochain cours nous allons monter d'une gamme dans la théorie objet, donc refaites ce cours si tout n'est pas clair et au plaisir de vous revoir !