

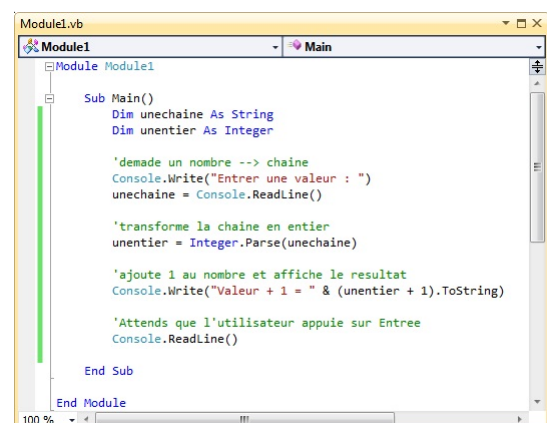
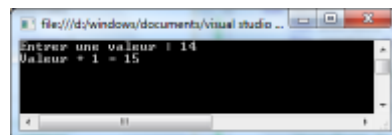
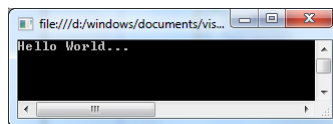
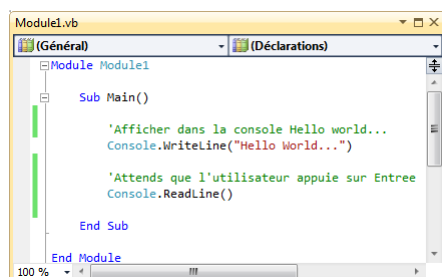
Implémentation de la classe COMPTE

Objectifs : Créer la classe Compte + Surcharge de constructeur + Debug + Documentation du code

1 - Lancer VB.NET

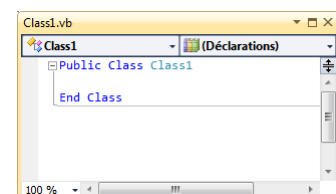
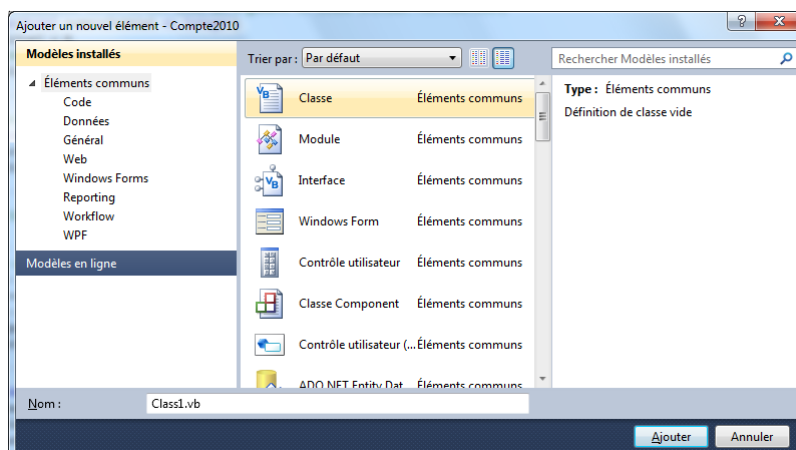
2 - Créer un nouveau projet

3 - Choisir Application CONSOLE



4 - Créer la classe Compte

Ajouter une nouvelle classe au projet :



4.1 - Créer les propriétés avec le méthodes d'accès

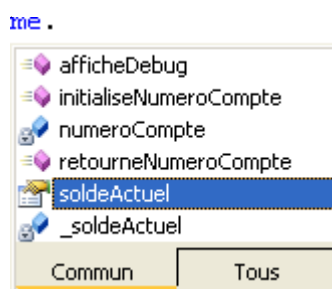
Les champs et les propriétés sont utilisés pour stocker des informations dans un objet. Bien que les champs et les propriétés soient difficiles à distinguer du point de vue de l'application cliente, ils diffèrent en fonction de la manière dont ils sont déclarés au sein d'une classe. Les champs sont tout simplement des variables publiques exposées par une classe, alors que les propriétés utilisent des procédures Property (SET et GET) pour contrôler la façon dont les valeurs sont définies ou retournées.

Une propriété est équivalente à un champ plus deux méthodes d'accès.

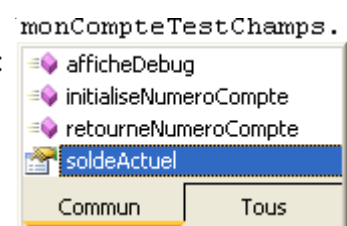
Exemple :	Méthode "CHAMPS"	Méthode "PROPRIETE"
Nom du champ	numeroCompte	_soldeActuel
Nom de la propriété	-	soldeActuel
Méthode pour "initialiser"	initialiseCompte	SET
Méthode pour "récupérer"	recupereCompte	GET

Visibilité des propriétés et méthodes :

Dans la classe :



Dans le programme :



Exemple de code: `Public Class ClassCompteTestChamps`

```

`PROPRIETES
Private _soldeActuel As Integer

`METHODE D'ACCES AUX DONNEES
Public Property soldeActuel() As Integer
    Get
        soldeActuel = Me._soldeActuel
    End Get
    Set (ByVal value As Integer)
        Me._soldeActuel = value
    End Set
End Property

`METHODE UTILISATEUR

...
End Class

```

4.2 - Test en mode console

Les champs et les propriétés sont utilisés pour stocker des informations dans un objet. Bien que les champs et les propriétés soient difficiles à distinguer du point de vue de l'application cliente, ils diffèrent eu égard à la manière dont ils sont déclarés au sein d'une classe. Les champs sont tout simplement des variables publiques exposées par une classe, alors que les propriétés utilisent des procédures Property (SET et GET) pour contrôler la façon dont les valeurs sont définies ou retournées.

4.2.1 Créer le premier constructeur

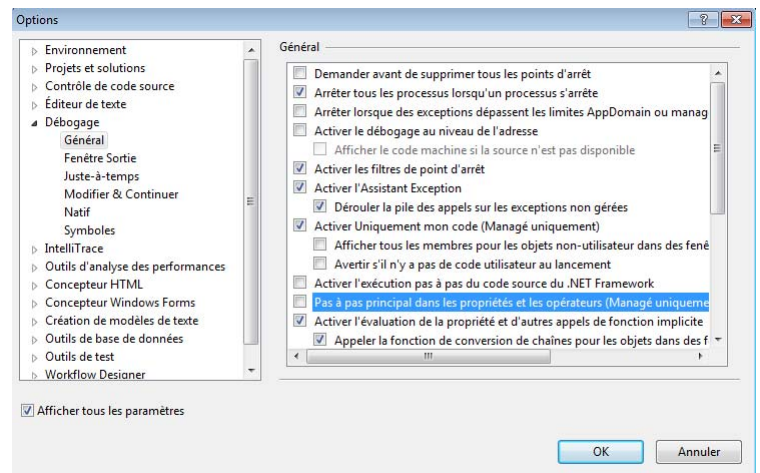
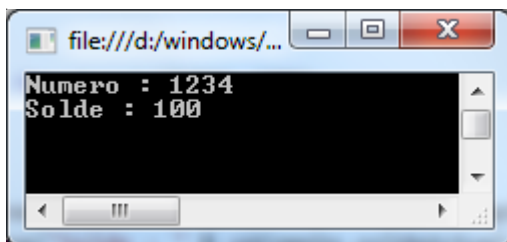
Le premier constructeur permet de renseigner toutes les propriétés de l'objet.

4.2.2 Tester votre code

Instancier un objet de la classe compte.

Tester avec l'exécution "Pas à Pas".

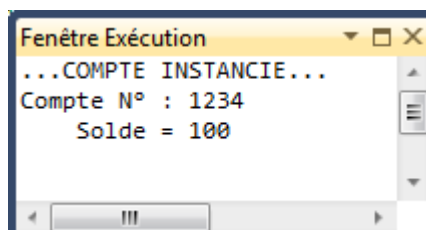
Afficher la le numéro puis le solde du compte.



4.2.3 Créer des méthodes d'affichage dans la console

4.3 - Créer une méthode de débogage

```
Public Sub afficheDebug()  
    Debug.WriteLine("COMPTE N° : "  
    Debug.Indent()  
    Debug.WriteLine("Solde : "  
    Debug.Unindent()  
End Sub
```



4.4 - Ajouter deux constructeurs **SURCHAGE**

Avec 0,1 et 2 arguments :

```
Public Sub New()  
.....
```

Me sert de variable objet faisant référence à l'instance en cours d'une classe

4.5 - Tester les trois constructeurs

Dans le programme principal, créer trois instances de la classe Compte :

```
Dim monCompte1 as New ClassCompte(  
    ▲ 1 sur 3 ▼ New ()  
    ▲ 2 sur 3 ▼ New (numeroInitial As Integer)  
    ▲ 3 sur 3 ▼ New (numeroInitial As Integer, soldeInitial As Integer)  
  
Dim monCompte1 As New ClassCompte()  
.....  
Dim monCompte2 As New ClassCompte(....)  
.....  
Dim monCompte3 As New ClassCompte(...., ....)
```

4.6 - Créer et tester les méthodes retrait et dépôt

```
Public Sub depot(ByVal somme As Integer)
```

```
.....
```

```
End Sub
```

```
Public Sub retrait(ByVal somme As Integer)
```

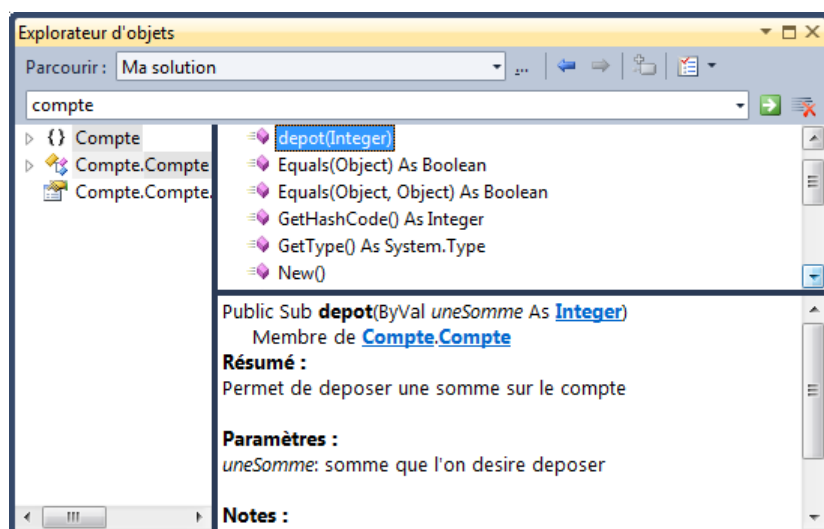
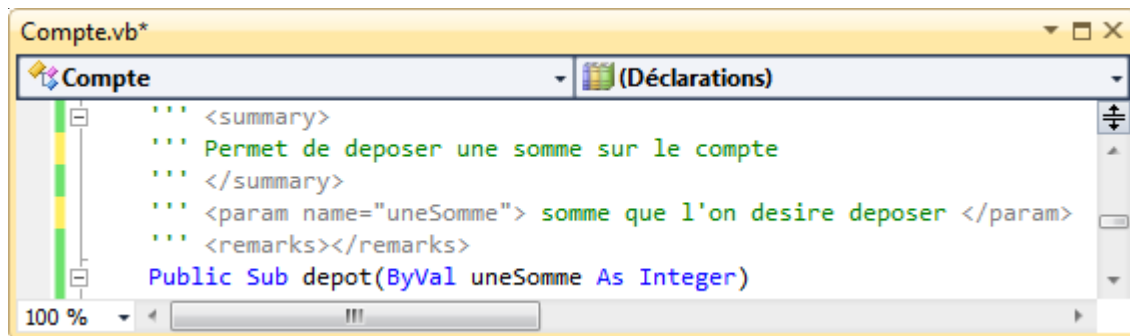
```
.....
```

```
End Sub
```

Faire une surcharge des méthodes dépôt et retrait

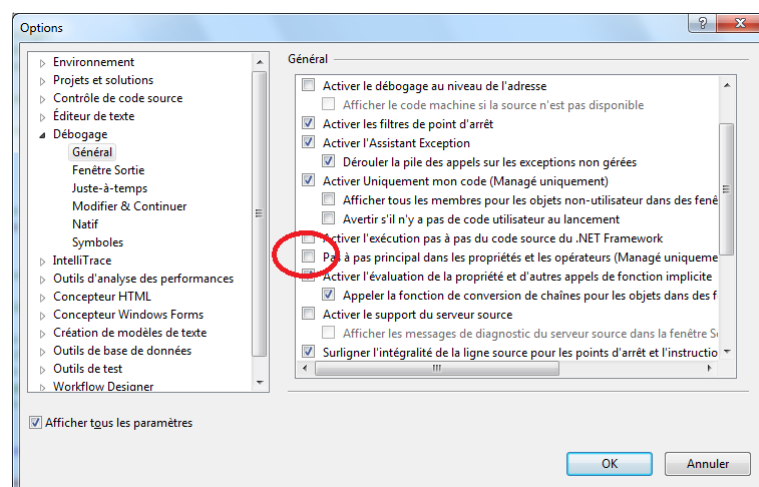
5 - Bien commenter les méthodes

Faire un commentaire ''' puis remplir le XML

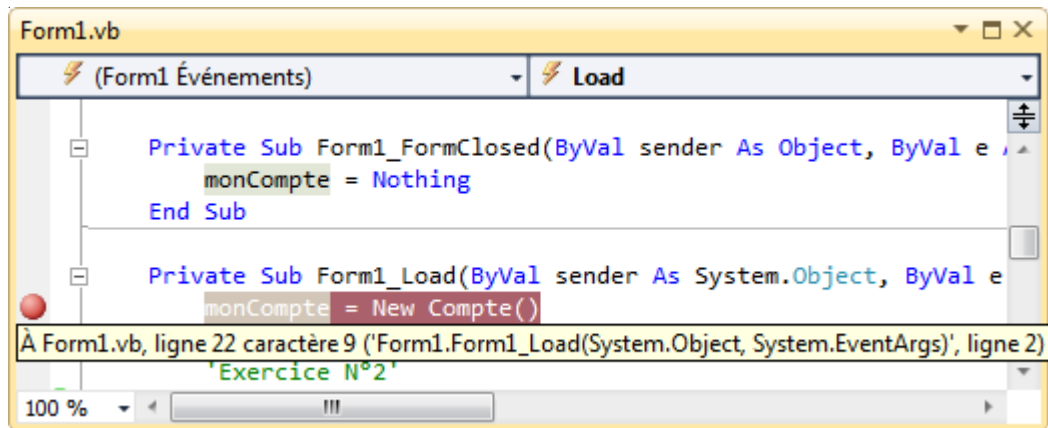


6 - Utilisation du déboguer

Décocher "Pas à pas principal" dans le menu option de Déboguer



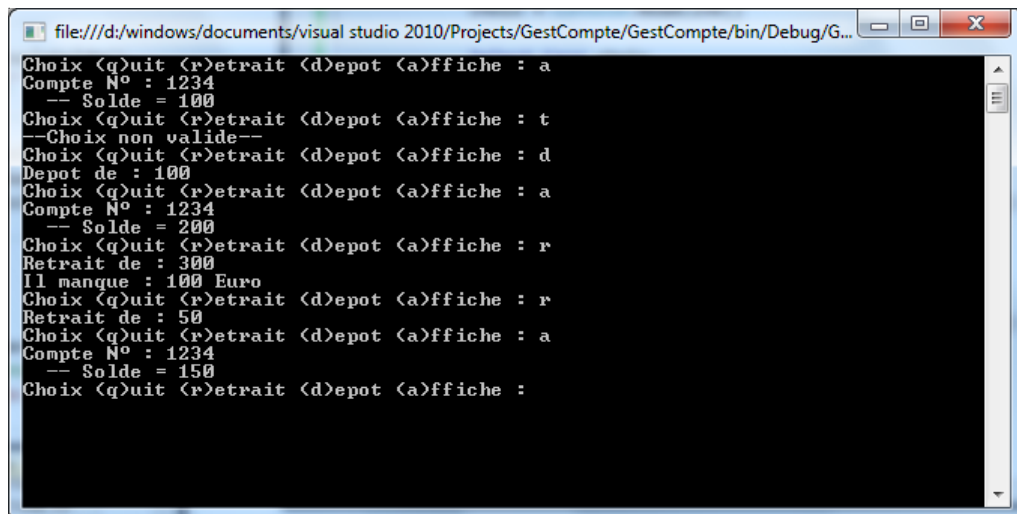
Faire un click droit dans la marge de la ligne ou l'on souhaite s'arreter



F8 Permet de passer à l'instruction suivante.

7 - Debugage fonctionnel

Ecrire un programme permettant de vérifier le bon fonctionnement de votre classe `Compte`.



8 - Modification de la classe compte

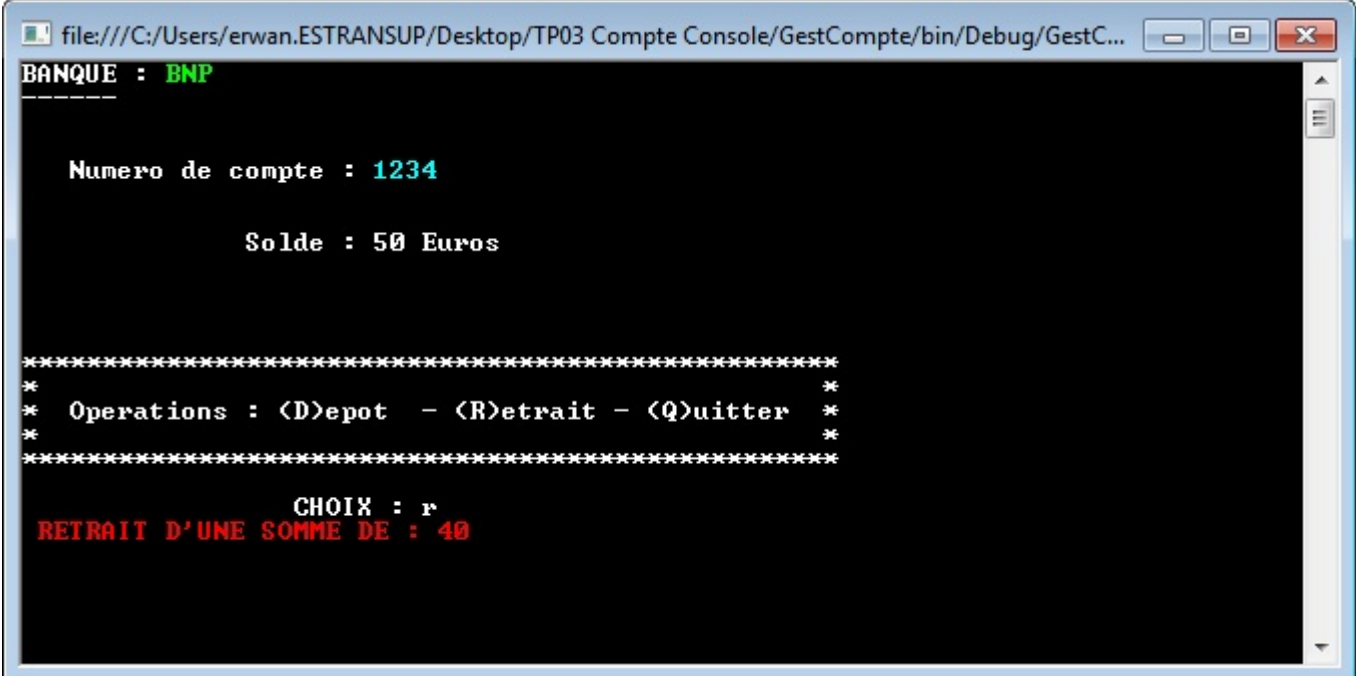
Modifier la méthode `retrait()`

* Contrainte : on ne peut retirer que l'argent qui est disponible sur le compte.

Attention aux méthodes surchargées.....

9 - Création d'une IHM

Afin d'utiliser votre classe compte, on demande de créer l'interface suivante :



```
file:///C:/Users/erwan.ETRANSUP/Desktop/TP03 Compte Console/GestCompte/bin/Debug/GestC...
BANQUE : BNP
-----

Numero de compte : 1234

Solde : 50 Euros

*****
* Operations : <D>epot - <R>etrait - <Q>uitter *
*
*****

CHOIX : r
RETRAIT D'UNE SOMME DE : 40
```