C#/Asp.Net – Cours 1

## Calculatrice - Niveau I

Nous allons maintenant commencer un programme que l'on reprendra petit à petit au cours des exercices. Nous souhaitons pouvoir disposer d'une calculatrice pour résoudre des opérations mathématiques simples (addition, soustraction, multiplication, division, puissance...). Pour se simplifier la vie, pour débuter :

* on ne s'intéressera qu'aux nombres entiers
* on ne gèrera pas de suite les éventuelles erreurs de saisies clavier.
* on ne fera que des calculs avec 2 opérandes et un opérateur (a + b, a - b, ab etc.)

Pour débuter par quelque chose de simple, réalisez un petit programme qui :

* Indique à l'utilisateur de saisir un premier nombre (la première opérande)
  + *Via l'utilisation de Console.WriteLine qui prends en paramètre la chaine de caractère à afficher*
* Capture sa saisie clavier et affecte le texte saisi dans une 1ère variable
  + *Via l'utilisation de Console.ReadLine qui attends que l'utilisateur ait appuyé sur "entrée" et renvoit le texte saisi.*
* En fait de même pour la seconde opérande affectée à une autre variable
* Converti chacune des saisies utilisateurs dans une variable de type entier ("int")
  + *Via l'utilisation de Int32.Parse qui prends en paramètre une chaine de caractère (par exemple, celle saisi) et renvoi une nouvelle valeur sous la forme d'un entier.*
* Affiche le résultat de l'addition, soustraction, multiplication, et division des deux opérandes.

Le pseudo code est donc de la sorte

soit une première variable "operandeGauche" de type chaine de caractère, et une seconde variable "operandeDroite" de type chaine de caractère

Afficher("Veuillez entrer l'opérande de gauche")

Affecter à la variable "operandeGauche" la valeur saisie par l'utilisateur

Afficher("Veuillez entrer l'opérande de droite")

Affecter à la variable "operandeDroite" la valeur saisie par l'utilisateur

Soit une variable "valeurOperandeGauche" de type entier, et une variable "valeurOperandeDroite" de type entier

Affecter à la variable "valeurOperandeGauche" la conversion de la chaine de caractère "operandeGauche"

Affecter à la variable "valeurOperandeDroite" la conversion de la chaine de caractère "valeurOperandeDroite"

calculer (valeurOperandeGauche + valeurOperandeDroite) et l'afficher

calculer (valeurOperandeGauche - valeurOperandeDroite) et l'afficher

calculer (valeurOperandeGauche \* valeurOperandeDroite) et l'afficher

calculer (valeurOperandeGauche / valeurOperandeDroite) et l'afficher

calculer (valeurOperandeGauche valeurOperandeDroite ) et l'afficher

# Instructions conditionnelles et opérations booléennes

Dans l'exercice précédent, rajouter différents tests pour gérer certains cas atypiques, notamment :

* Si la valeur de l'opérande de droite vaut 0, on ne tente pas d'effectuer la division mais on affiche un message à l'utilisateur
* Si la valeur de l'opérande de droite est inférieure à 0, on n'effectue par la mise en puissance
* Si la valeur de droite ainsi que la valeur de gauche sont égales à 0, on n'effectue aucun calcul

# Gestion de l’opérateur

Puis, modifiez votre code pour demander à l'utilisateur de saisir l'opérateur à utiliser (ex. "+", "-", "/", "^", "\*"), et n'effectuez que celui qui correspond à son choix, ou ne faites rien si l'opérateur n'est pas reconnu.

# Méthodes

Nous allons procéder à la modification du code source existant pour le clarifier en mettant en place un certain nombre de méthodes.

Le 1er but du développement de méthodes est de clarifier au maximum le code en capturant un comportement technique dans une méthode représentant une « intention ». C’est ce que nous allons faire :

Créer une méthode « DemanderOperandeALUtilisateur » qui :

* Prends en paramètre le message à afficher
* Affiche ce message, puis capture la saisie utilisateur
* Renvoie cette capture

Puis utilisez cette méthode pour demander les 2 opérandes

Le code obtenu à la fin devrait être beaucoup plus lisible. En bonus, il est beaucoup plus maintenable (pas de répétition de code).

Dans un second temps, nous allons aussi extraire le code qui analyse la saisie de l’opérateur par l’utilisateur.

* Rajoutez donc une énumération qui contient les différents opérateurs connus (Addition, Soustraction, Multiplication, etc. ainsi que le choix « OperateurNonReconnu »)
* Créez une méthode « DemanderOperateurALUtilisateur » qui :
  + Demande à l’utilisateur de saisir un opérateur
  + Capture sa saisie
  + Analyse la saisie et renvoie l’opérateur correspondant
* Puis remplacez le code correspondant dans le corps de la méthode Main, et utilisez une instruction switch sur le résultat de l’appel à « DemanderOperateurALUtilisateur » pour exécuter la bonne opération.

Y-a-t’il encore du code en doublon ? Si oui et si c’est pertinent, remplacez-le par une méthode appropriée.