C# / ASP.Net – Cours 2

I. Listes

Pour les listes, nous allons débuter un nouveau projet (gardez l'ancien projet quelque part, nous en aurons besoin plus tard). Le projet sera à nouveau un projet Console en C#, comme celui débuté hier.

Nous allons créer un premier programme simplifié qui :
- demande à l'utilisateur de saisir des chaînes de caractères
- conserve ces saisies dans une liste de saisies
- les affiche à nouveau dans le sens inverse de la saisie.

II. Classes

Dans le projet calculatrice à nouveau, nous voulons rajouter une nouvelle fonctionnalité fictive : nous souhaitons laisser la possibilité à l'utilisateur d'effectuer plusieurs calculs, puis lorsqu'il le souhaite, lui afficher cette liste.

Classe Operation

Comme nous allons devoir plus tard conserver les informations des opérations effectuées, nous allons rajouter une classe qui permettra de les modéliser, que nous allons utiliser en remplacement du code actuel.

Nous allons donc créer une classe "Operation" qui aura comme champs :
- l'opérateur
- l'opérande de droite
- l'opérande de gauche
- une méthode "GetResult" qui calcule et renvoie le résultat de cette opération.

Le résultat étant un simple calcul, il n'y a pas de raison de le conserver dans l'état de la classe.

Puis dans le corps de la fonction Main, plutôt que d'exécuter le calcul, on construira une nouvelle instance de la classe "Operation", et ce sera elle qui sera capable de déterminer le résultat.

Historique des opérations

Nous allons rajouter maintenant le nouveau comportement qui permet d'effectuer plusieurs calculs, et d'afficher l'historique
-Modifier le corps de votre fonction pour qu'elle fasse des calculs sans s'arrêter sauf si l'utilisateur saisie un opérateur vide, auquel cas, on sortira de cette boucle
- Avant de rentrer dans cette boucle, instancier une nouvelle liste qui contiendra l'historique des opération effectuées.
- Lorsqu'une opération a été effectuée, rajouter cette opération dans la liste d'historique

- Lorsque l’on sort de la boucle (l’utilisateur a tapé un opérateur vide), afficher dans la console l’intégralité des opérations effectuées
- La conversion d’une Opération en chaîne de caractère affichable peut faire l’objet d’une méthode spécifique dans la classe Opération, par exemple “GetRepresentationTextuelle”.

III. Relations entre classes

L’exemple actuel de la calculatrice n’est pas la plus appropriée pour illustrer les relations entre classes. Nous partir sur un autre projet (toujours temporairement, nous reviendrons sur l’exemple calculatrice plus tard).

Cet exemple se basera sur les formes géométriques. Le but sera de modéliser via des classes des notions de formes géométriques (disque, rectangle, etc.) et de disposer de méthodes permettant de les manipuler (par exemple pour en calculer son aire, et son périmètre).

Première classe : le disque

Dans un nouveau projet console C#, créer une nouvelle classe “Disque” qui servira à modéliser la notion d’un disque géométrique (on s’en doutait).
Un disque :
- est constitué d’un simple rayon
- expose deux méthodes : GetArea et GetPerimeter qui renvoient respectivement l’aire et le périmètre de ce cercle.
    (pour information : la valeur de Pi en .Net : Math.Pi)

Instancier quelques disques dans le corps de main et affichez leurs aires et leurs périmètres respectifs pour vérifier que le code est fonctionnel

Seconde classe : le rectangle

Modéliser la notion de “Rectangle” via une nouvelle classe.
Un rectangle :
- est constitué d’une largeur et d’une hauteur
- expose les deux mêmes méthodes : GetArea et GetPerimeter

Troisième classe : le triangle

Un triangle est un peu plus complexe à modéliser car il n’a pas de hauteur ou de largeur, mais 3 points qui le constituent. Pour pouvoir réaliser ce triangle, nous allons donc créer tout d’abord la classe Point

Troisième vrai classe : le point

Le point est constitué de deux coordonnées X et Y, et c’est à peu près tout :)

Quatrième classe : le triangle

Un triangle est composé de 3 points (pA, pB, et pC par exemple).

Son aire est calculé selon la formule suivante :

$$S = \frac{1}{2}\left|\det\begin{pmatrix} x_B - x_A & x_C - x_A \\ y_B - y_A & y_C - y_A \end{pmatrix}\right| = \frac{1}{2}|(x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)|.$$

Son périmètre correspond à la somme des distances entre les points. Ce calcul de distance entre points pourrait être codé directement dans la classe triangle mais il est probable que cette notion soit réutilisée plus tard ailleurs, en dehors de l'utilisation dans les triangles.

Quand on y réfléchi, on se rend compte qu'une distance entre deux points est un concept qui n'a de lien qu'avec les points et devrait donc être présent dans cette dernière classe. Il serait alors bon d'exposer dans "Point" une méthode qui prendra en paramètre un second point et qui calculera la distance entre l'instance courante et le point spécifié en paramètre.

Ps : Dans un second temps, le fait qu'on prenne un point plutôt qu'un autre pour calculer la distance provoquera probablement des questions, pourquoi l'un plutôt que l'autre. Dans la réalité, la distance entre 2 points n'est pas reliée à un point précis, la méthode correspondante ne devrait du coup pas être une méthode d'instance, mais une méthode de classe (statique) qui prendrait 2 points en paramètre.