C# - FICHE PRATIQUE N° 1 – Solutions et compléments

Gestion d'une liste déroulante
Projet, formulaire, contrôle, programmation événementielle.

A/ Présentation (très) rapide de DotNet et C#

- Il est intéressant de faire créer, compiler et exécuter quelques programmes simples à l'aide d'un éditeur de texte et du compilateur en ligne de commandes.
- Une présentation rapide de l'outil Visual studio (à l'aide d'un vidéo-projecteur par exemple) permettra de étudiants de repérer plus rapidement le rôle des différentes fenêtres.
- Le choix de C# n'est nullement obligatoire, les fiches proposées peuvent être facilement réalisées en VB.net par exemple.

B/ Une première application

Il est souhaitable d'analyser tout de suite le code généré avec les étudiants. L'exemple du projet fiche1 après création du formulaire et ajout de la seule liste déroulante est présenté page suivante. Les points suivants méritent d'être mentionnés :

- La directive « using » permet d'inclure des espaces de noms. Un espace de noms regroupe des définitions de classes, des énumérations, et d'autres espaces de noms… On inclut donc les espaces contenant les classes nécessaires à l'application.
- Le mot-clé « namespace » indique l'espace de noms dans lequel la classe correspondant au formulaire est créée. Chaque projet possède un espace de noms par défaut, celui-ci peut être modifié dans les propriétés du projet (clic droit sur le projet dans l'explorateur de solutions).
- On trouve ensuite la déclaration de la classe du formulaire. Il est commode de donner le même nom au fichier source de cette classe. Ceci peut être fait dans l'explorateur de solutions. Chaque instance de formulaire créée dans l'application sera un objet de cette classe (souvent un seul en pratique).
- La liste déroulante est devenue un membre donnée de cette classe. Son type est « ListBox », cette classe étant définie dans l'espace de noms « System.Windows.Forms ». Cet espace de noms étant inclus dans le source, on pourrait simplement écrire « private Listbox lb_liste ; ».
- Chaque formulaire dispose d'un « Container » qui contient l'ensemble de ses contrôles graphiques.
- Le constructeur de la classe fait appel à la méthode « InitializeComponent » qui permet d'instancier chaque contrôle, de renseigner ses propriétés en fonction des valeurs indiquées en mode graphique, et de l'ajouter aux contrôles du formulaire. Il ne faut pas modifier cette méthode car elle est régénérée automatiquement par le concepteur graphique (ces modifications seraient donc perdues).
- On trouve ensuite la méthode « Dispose » qui sera exécutée par le ramasse-miettes lors de la destruction de l'objet.
- La méthode « Main » est le point de démarrage de l'application. La méthode statique « Run » de la classe « Application » permet de lancer l'exécution en créant une instance de la classe « formulaire principal ». Une erreur fréquente survient lorsque le formulaire n'est pas correctement

nommé dès sa création : il n'y a pas cohérence entre le nom de la classe, le nom du constructeur et le nom de la classe instanciée dans la méthode « main ». Ce point doit donc être signalé aux étudiants (j'y ajoute la cohérence avec le nom du fichier source qui n'est pas obligatoire).

Lorsqu'un projet contient plusieurs formulaires, soit un seul d'entre eux contient une méthode « main », soit il faut indiquer l'objet de démarrage dans les propriétés du projet. D'autre part, lorsqu'une solution contient plusieurs projets, il faut préciser le projet de démarrage dans les propriétés de la solution (clic droit sur la solution dans l'explorateur de solutions).

Le mécanisme événementiel est mis en œuvre de manière assez simple (exemple du clic sur un bouton) :

- Définition d'une méthode « MonBouton_Click » avec deux paramètres : l'expéditeur de l'événement (ici le bouton lui-même) et un objet de la classe « EventArgs » ou de l'une de ses classes dérivées (ce paramètre permet de récupérer un certain nombre d'informations (coordonnées de la souris pour un « MouseEventArgs » par exemple).
- Mise en place de la gestion d'événement dans l'initialisation du contrôle (méthode « InitializeComponent ») par l'instruction ci dessous :
- this.bt_ajouter.Click += new System.EventHandler(this.bt_ajouter_Click);
- Remarque : une erreur fréquente consiste à supprimer la méthode sans supprimer l'instruction de gestion de l'événement.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace Fiche1
{

    public class Fm_principal : System.Windows.Forms.Form
    {
        private System.Windows.Forms.ListBox lb_liste;
        private System.ComponentModel.Container components = null;

        public Fm_principal()
        {
            InitializeComponent();
        }

        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }
```

```
        #region Code généré par le Concepteur Windows Form
        private void InitializeComponent()
        {
            this.lb_liste = new System.Windows.Forms.ListBox();
            this.SuspendLayout();
            this.lb_liste.Location = new System.Drawing.Point(16, 40);
            this.lb_liste.Name = "lb_liste";
            this.lb_liste.Size = new System.Drawing.Size(176, 160);
            this.lb_liste.Sorted = true;
            this.lb_liste.TabIndex = 0;

            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(292, 273);
            this.Controls.Add(this.lb_liste);
            this.Name = "Fm_principal";
            this.Text = "Premier programme";
            this.ResumeLayout(false);

        }
        #endregion

        [STAThread]
        static void Main()
        {
            Application.Run(new Fm_principal());
        }
    }
}
```