TP C#0 : Un nouveau départ.

1 Introduction

1.1 Objectif

Durant ce TP, vous allez découvrir un nouveau langage, le C# . Vous serez confrontés à une nouvelle façon de coder, avec un nouveau langage, sous un nouvel environnement. Le but n'étant pas de vous perdre tout de suite, nous allons intégrer différentes fonctionnalités liées au C# et à Visual Studio afin de vous guider vers le bon chemin.

Cette semaine nous allons utiliser les WinForms , qui permettent de réaliser des interfaces graphiques, et ce de façon simple et efficace.

1.2 Le langage C#

Le C# est un langage de programmation orienté objet, créé par la société Microsoft, et notamment par un de ses employés, Anders Hejlsberg, le créateur du langage Delphi.

Il a été créé afin que la plate-forme Microsoft .NET soit dotée d'un langage permettant d'utiliser toutes ses capacités. Il est très proche du Java dont il reprend la syntaxe générale, ainsi qu'au niveau des concepts – la syntaxe reste cependant relativement semblable à celle de langages tels que le C++ et le C.

Contrairement au Caml qui est fonctionnel et avec typage inféré, le C# est un language impératif et son typage n'est pas inféré. Vous devrez donc spécifier le type choisi pour chacune des variables créées. Des règles basiques seront précisées plus tard.

1.3 Visual Studio

Microsoft Visual Studio est une suite de logiciels de développement pour Windows conçu par Microsoft. Il s'agit d'un ensemble complet d'outils de développement permettant de générer des applications. Ils permettent de développer des applications dans divers langages s'appuyant sur .NET tels que C#, F#, Visual C++ etVisual Basic. Il est possible de développer des types d'applications très variés : applications graphiques, en console, des bibliothèques de classes, des services Windows ou encore des sites web, le tout grâce à l'environnement de développement intégré IDE – Integrated Development Environment.

Les étudiants d'Epita peuvent télécharger gratuitement Visual Studio en allant sur le site Microsoft DreamSpark for Academic Institutions en passant par epita.net ou en téléchargeant la version express gratuite sur le site Microsoft.

2 Cours

2.1 Les Bases

Voici quelques règles basique de syntaxe utilisées en C#. Notez que ce sont seulement certaines bases, et d'autres notions et règles seront vues dans les semaines suivantes. Cependant, nous vous encourageons à être curieux et à vous renseigner par vous-même sur le langage C# et les notions qu'il propose.

- Chaque bloc d'instruction se trouve entre accolades.
- Une ligne est composée d'une instruction et se termine par un point-virgule. (et non deux,contrairement au caml.)
- L'en-tête d'une fonction se compose de son type de retour, de son nom, et enfin, des paramètres et leurs types.
- Chaque variable déclarée doit être déclarée accompagnée de son type.

Les fonctions respectent ce prototype :

```
1 type_de_retour nom_de_ma_fonction(type1 var1, type2 var2,...)
2 {
3         //Code...
4 }
```

Les blocs de conditions, se font quand à eux suivant ce modèle :

```
if (condition)
{
        //Code si la condition est respectee..
}
else
{
        //Code si la condition n est pas respectee...
}
```

Il est également possible de créer des conversions entre tous les types, grâce à des méthodes présentes de base dans le langage.

```
int i = 42;
string the_Answer = i.ToString();
```

Enfin, il est important de commenter votre code, pour vous-même, mais également pour les personnes qui travaillerons avec vous dans le futur. Il faut dès à présent prendre de bonnes habitudes ! En C#, les commentaires se font grâce au caractère "/".

```
//Ceci est un commentaire!

/*
Et voici un
bloc
de commentaires.
*/
```

2.2 Windows Forms

Les Windows Forms permettent de créer des interfaces graphiques sous Windows assez simplement. Visual Studio fournit une boîte à outils où on trouve la liste des WinForms disponibles. Pour les utiliser, il suffit d'effectuer un simple glisser-déposer.
Lorsqu'un Form est sélectionné, on peut le paramétrer dans la fenêtre propriétés de VS où se trouve différents champs du WinForm. À partir de là, on peut le personnaliser – taille, couleur, position, contenu, etc.

Dans la fenêtre propriétés se trouve également un onglet éclair. On peut alors paramétrer diverses actions de l'utilisateur – clic de souris, appui sur une touche du clavier, etc. En effectuant un double-clique sur le champs Click, on se retrouve dans une fenêtre d'édition de code. Une méthode a été automatiquement créée par VS. Cette méthode permet de créer un callback pour définir le comportement de l'application pour une action utilisateur donnée. Le code source dans la fenêtre d'édition contient les différentes méthodes de la Windows Form.

2.3 Votre premier WinForm

Pour ce premier TP d'approche au C#, nous allons donc créer plusieurs Windows Form. Dans un premier temps, ouvrez Visual Studio si ce n'est pas déjà fait. Créez à présent un nouveau projet.
Fichier -> Nouveau -> Projet -> Application Windows Forms WindowsFormsApplication va être créé dans une solution, représentant un ensemble de projets. Pour lancer un projet, rien de plus simple, il vous suffit d'appuyer sur la touche F5. Vous pouvez remarquer sur le bord droit de votre IDE l'explorateur de solutions. Celui ci permet d'afficher une vue d'ensemble du projet sur lequel vous travaillez. Il contient :

Solution WindowsFormsApplication (1 projet)
- WindowsFormsApplication : l'application.
  - Properties : Ressources du projet.
  - Références : Bibliothèques utilisées dans l'application.
  - Form1.cs : Contient votre code et affiche l'éditeur graphique.
    - Form.Designer.cs : code généré par l'éditeur graphique.
    - Form1.resx : Contient votre code.
    - Program.cs : Point d'entrée du programme.

2.4 Conseils

Pour tout vos projets ou TP C# vos plus grand amis seront : MSDN, StackOverflow et Google ! De plus vérifiez bien que votre AUTHORS contient bien un seul retour à la ligne.

3 Rendu

3.1 Arborescence

A la fin de ce TP, vous devrez rendre une archive respectant l'architecture suivante :

```
rendu-tpcs0-login_x.zip
    |– rendu-tpcs0-login_x/
        |– AUTHORS
        |– HelloWorld/
            |– HelloWorld.sln
            |– HelloWorld/
                |– Tout sauf bin/ et obj/
        |– ImageViewer/
            |– ImageViewer.sln
            |– ImageViewer/
                |– Tout sauf bin/ et obj/
        |– PlusOuMoins/
            |– PlusOuMoins.sln
            |– PlusOuMoins/
                |– Tout sauf bin/ et obj/
```

Bien entendu login_x doit être remplacé par votre login et le code **doit** compiler !

3.2 Fichier AUTHORS

Ce fichier contient votre login sous la forme suivante : une étoile *, un espace, votre login(login_x) et un retour à la ligne – représenté par le caractère $ dans l'exemple ci-dessous.

```
* login_x$
```

4 Excercice 1 : `Hello World`

Dans les exercices qui vont suivre, vous profitez d'une liberté de création. Une fois les exercices terminés, nous vous conseillons d'expérimenter !

4.1 Handling

Vous devez commencer par créer un nouveau projet Windows Forms qui prendra le nom de l'exercice : `Hello World`. Créez une interface graphique qui contiendra - au moins - les éléments suivants :

- Un *Button* `button_say` qui contient le texte `Say`
  - Modifiez le nom du bouton `(Name)` dans les *properties*.
  - Modifiez le contenu du *champ* `Text` dans les *properties*.
- Un *Label* `label_say` qui ne contient pas de texte.
  - Modifiez le nom de l'etiquette `(Name)` dans les *properties*.
  - Supprimez le texte du *champ* `Text` dans les *properties*.

L'interface graphique a été conçue et paramétrée. Il faut maintenant la faire fonctionner en créant des connexions entre les *WinForms*. Double-cliquez sur le bouton `button_say` de l'interface graphique ou dans l'onglet représenté par un *éclair* et double-cliquez sur l'action `Click`. L'éditeur de code s'affiche. Assignez le texte du label `label_say` :

```
1 private void button_say_Click(object sender, EventArgs e)
2 {
3         label_say.Text = "Hello World !";
4 }
```

Appuyez sur **F5** pour lancer l'application avec le *débuggeur*.

4.2 Améliorations

Attention, vous êtes obligés d'implémenter **au minimum** les 2 premières améliorations :

- Dites *bonjour* en plusieurs langues en utilisant une `ComboBox`.
- Récupérez le nom de la personne à qui vous dites *bonjour* en utilisant une `TextBox`.
- Dites *bonjour* en couleur en ajoutant un `Button` et la boîte de dialogue `ColorDialog`.
- Ajoutez des images à votre *Windows Form*.

Vous pouvez aussi rajouter vos propres améliorations ou customisations.

5 Excercice 2 : Image Viewer

5.1 Manipulations

Le but de cet exercice est de réaliser un lecteur d'image. L'interface graphique devra comporter les éléments suivants :

— Un *Button* : `button_open` avec le texte `Open`.

— Une *PictureBox* : `pictureBox_viewer`

— Une boîte de dialogue *OpenFileDialog* : `dialog_open`

Affecter l'action de `button_open` :

```
1 private void button_open_Click(object sender, EventArgs e)
2 {
3         dialog_open.ShowDialog();
4         Image img = Image.FromFile(dialog_open.FileName);
5         pictureBox_viewer.Image = img;
6 }
```

5.2 Améliorez

Liberté de création dans cette partie.

— Permettre au lecteur d'image d'afficher des images haute résolution en effectuant un redimensionnement.

— Afficher le nom de l'image et ses caractéristiques – hauteur, largeur, format de fichier en récupérant l'extension, etc.

— Habiller l'interface graphique avec des couleurs.

— ...

6 Exercice 3 : More Or Less

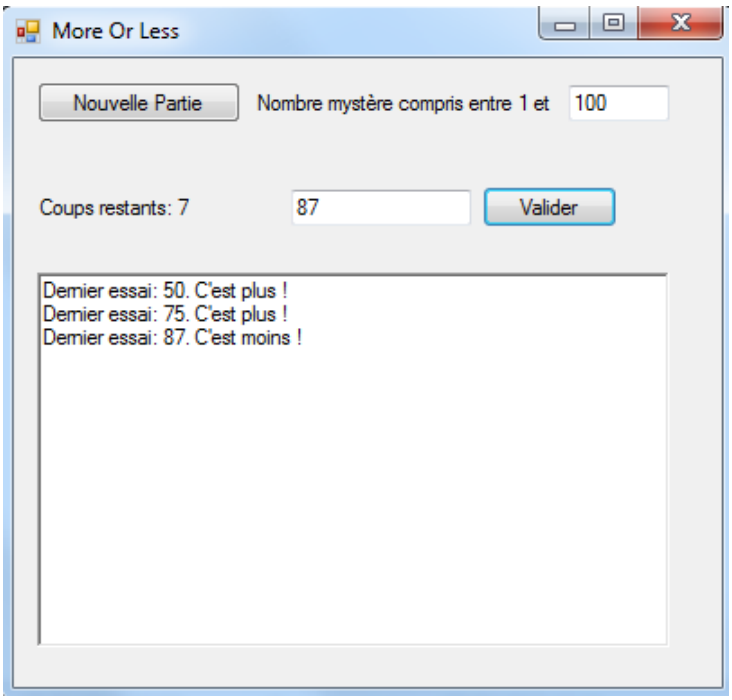
Maintenant que vous avez quelques bases en C# sous Visual Studio, vous allez enfin pouvoir mettre en pratique vos acquis pour un vrai mini-projet. Vous devez normalement maintenant savoir utiliser des **boutons**, des **textbox**, etc. L'objectif de ce mini projet est de faire un jeu du "plus ou moins".

6.1 L'interface

L'interface devra contenir un minimum d'éléments obligatoires :

— Deux **button** :
  — 'Nouvelle partie'
  — 'Valider'
— Deux `TextBox`, l'une pour choisir la borne maximum, l'autre pour que l'utilisateur puisse entrer son choix.
— Un `Label` pour montrer le nombre restant de coups.
— Une `RichTextBox` pour afficher l'historique des choix de l'utilisateur.

Voilà un exemple d'interface :



6.2 But du jeu

Les règles du 'plus ou moins' sont simples : l'ordinateur choisit un nombre aléatoire dans un certain intervalle, puis l'utilisateur doit retrouver ce nombre en un minimum de coups. Voici quelques indications sur la procédure à suivre pour la conception du mini projet :

— 'Nouvelle partie' :

- Initialisation du nombre mystère (vous allez devoir utiliser le type *Random*, n'hésitez pas à demander de l'aide à vos assistants).
- Remise du nombre de coups restants à 10 (par exemple).
- Changer la *propriété* du bouton 'Valider' à *Enabled*.
- Effacer le contenu de l'historique.

— 'Valider' :
- Réduire le nombre d'essais restants de 1.
- Inscrire le choix de l'utilisateur dans l'historique.
- Si le nombre d'essais restants est inférieur à 1 et que le nombre testé par l'utilisateur n'est pas le bon :
  - Inscrire 'Perdu' et afficher le nombre mystère dans l'historique.
  - Changer la *propriété* du bouton 'valider' à *Disabled*.
- Sinon, vérifier si le nombre entré par l'utilisateur est supérieur ou inférieur au nombre mystère, et afficher l'information correspondante dans l'historique. Si ce nombre est égal au nombre mystère, afficher 'Gagné !' et changer la *propriété* du bouton 'Valider' à *Disabled*.

6.3 Bonii

Vous êtes libres sur cette partie de rajouter les fonctionnalités que vous voulez dans votre jeu à condition d'y laisser les parties obligatoires.
Quelques idées :

- Ajoutez d'un panneau d'options pour changer le nombre maximal de coups restants.
- Vérifiez que l'utilisateur entre bien des chiffres dans les *TextBox* prévues à cet effet et pas autre chose.
- Ajoutez un temps limité pour gagner la partie (renseignez vous du côté des *Stopwatch* et des *Timers*).
- Ajoutez un système de meilleur score : les scores sont enregistrés dans un fichier texte et le jeu vous indique si vous arrivez dans le top 10 (renseignez vous du côté de la lecture et l'écriture dans les fichiers).