Norme de codage C#
Projet P1, HE-Arc / HES-SO

1. Avant-propos

Objectifs

Le but de ce document est de décrire les principes de codages utilisés dans ce projet. Il est fortement inspiré du document présentant les conventions de codage à utiliser en Java présent sur la forge

Domaine d'application

Ce document est valable pour le langage C#.

2. Langue de programmation

Excepté pour la BDD où le nom des tables et champs sont en français, le code et les commentaires de tous les projets sont écris en anglais ! Les noms des variables, des fonctions, etc., sont tous définis en anglais !

3. Conventions de nommage des différentes entités du code

Classes

Les classes commencent toujours par une majuscule. Chaque nouveau mot commence également par une majuscule. Le reste des lettres qui composent chaque mot est écrit en minuscules.

```
public class MyClass : MySuperClass
{
}
```

Les classes sont préfixées par un identificateur propre au projet.
Ce projet contenant plusieurs projets C#, chacun à son propre préfixe :

- GestionTeleski → GT
- SimulateurLecteur → SL
- SimulateurTourniquet → ST
- CommonClasses
  - Pour les classes : Common
  - Pour les interfaces : IRemote

```
public class GTMyClass : GTMySuperClass
{
}
```

Méthodes

Le nom des méthodes commence toujours par une majuscule. Chaque mot qui suit est composé d'une majuscule. Toutes les autres lettres qui composent les mots sont des minuscules. Seules les fonctions correspondant à des événements (clics sur un bouton, chargement d'un formulaire) peuvent commencer par une minuscule.

```
private void TestAndWrite(CommonCard card)
{…}

private void btnBackToMenu_Click(object sender, EventArgs e)
{…}

private void GTMyUserControlEraseCard_Paint(object sender, PaintEventArgs e)
{…}
```

Constantes

Les variables déclarées en tant que constantes sont écrites en majuscule, avec éventuellement des soulignés entre chaque mot.

```
public static final int MY_CONST = 13;
```

Paramètres reçus des fonctions

Les paramètres reçus d'une fonction ne sont pas distingués particulièrement.

```
public void MyFunc(int param1, int param2)
{…}
```

Variables membres d'une classe

Les variables membres d'une classe ne sont toujours préfixés d'un « _ ».[1]

```
public class MyClass : AnotherClass
{
  int _myVar = 0;
}
```

Format des champs, privé et public

Les attributs privés débutent toujours par une minuscule (« camelCase »).

```
public void MyFunc()
{
  int myVar = 0; // private variable
  MyClass myClass = MyClass(); // instanciation of a private object
  MySingletonClass mySingleton = MySingleton.GetInstance(); // access to a private singleton object
}
```

les attributs publics (« propriétés » de classe) commencent toujours par une majuscule (il est à noter que l'accesseur *get/set* automatise ce procédé).

```
public class MyClass : AnotherClass
{
  int _myVar = 0; // private variable
  public int MyVar {get{…}set{…}};
  public MySdClass mySdClass = mySdClass (); // instanciation of a public object
}
```

Ressources

Les préfixes suivants doivent être ajoutés aux noms des ressources :

Préfixes	Contrôle visuel
Btn	Bouton
Tbx	TextBox
Lb	Label
Bmp	Image (bitmap)
Sc	Scrollbar
Rb	Bouton radio
Chk	Case à cocher (checkbox)
Frm	Formulaire
Cb	Combobox
Gbx	GroupBox
Pnl	Panel

[1] Modification des conventions java où les variables membre d'une classe n'avaient pas de préfixe et les variables reçues des fonctions avaient le "_". Cette modification vient du fait des propriétés en C# : Si les variables membre d'une classes n'avaient pas de préfixe, alors la variable et sa propriété n'auraient différés que d'une majuscule ce qui n'est pas une bonne pratique.

4. Indentation

L'indentation est un principe fondamental de la programmation et n'est pas uniquement imposée par cette convention. Son caractère obligatoire ne fait ici aucun doute.

Les accolades sont déclarées sous l'instruction précédent le bloc.

```
if(toto == 2)
{
}
```

La première instruction du bloc est décalée et l'indentation générale est deux espaces.

```
if(toto == 2)
{
  déclaration;

  instruction;
  instruction;

  if(titi == 1)
  {
    instruction;
  }
}
```

5. Squelettes des fichiers « .cs »

Les fichiers contenant les classes C# (« .cs ») suivent la nomenclature suivante :

```
/*=============================================================================*
 * Class: <MyClass>
 * Version/date: <x.y.z>[2] <yyyy.mm.dd>
 *
 * Description: <"This class…">
 * Specificities: <e.g. "This class is a singleton">
 *
 * Authors:   <FirstName> <LAST_NAME>
 * Copyright: HES-SO/HE-Arc, all rights reserved
 *
 *=============================================================================*/
```

Les constructeurs apparaîtrons en premier dans le fichier, suivis des destructeurs, des fonctions d'initialisation et enfin des autres fonctions.

6. Commentaires

Toutes les fonctions et propriétés implémentées sont précédées d'un tag XML expliquant ce qu'elle fait.

```
/// <summary>
/// This ...
/// </summary>
/// <remarks>The...<c>blab bla bla</c></remarks>
private void TestAndWrite(CommonCard card)
{
  ...
}
```

[2] « x » pour la version, *e.g.* v1 ou v2 ; « y » pour les modifications majeures de la version, *e.g.* v2.3 ; « z » pour les modifications mineurs de cette version, *e.g.* v2.3.5