C#

Règles de nomenclature

Le *C#* étant un langage propriétaire, **Microsoft** émet des recommandations de nomenclature. Les règles suivantes reflètent ces recommandations.

- Variables (incluants paramètres de méthodes) : `camelCase`
- Attributs de classe : `_camelCase` (préfixé par un souligné `'_'`)
- Tout le reste (classes, méthodes, propriétés, constantes, namespaces, etc.) : `PascalCase`
- Interface : `IPascalCase` (préfixé par la lettre `'I'` en majuscule)

La notation hongroise autre que les cas indiqués ci-dessus est explicitement interdite.

Il est fortement recommandé de ne pas utiliser de caractères accentués dans les noms d'identifiants.

Choix des noms

Les noms des identifiants sont significatifs et représentent bien leur utilité : l'objet modélisé pour une classe, la valeur contenue pour une variable ou une constante, la tâche effectuée pour une méthode.

Exception : pour les boucles dont le compteur ne représente pas une valeur particulière, mais sert uniquement à compter le nombre d'itérations, il est correct d'utiliser une seule lettre, traditionnellement `i`, `j` et `k`.

Classes

- **Nom** : pour représenter un objet (ex. : `Personne`).

Variables et propriétés

- **Nom** : pour représenter une valeur (ex : `largeur`, `dateDeNaissance`).
- **Adjectif** ou **expression descriptive** : pour représenter une condition booléenne (ex. : `estVide`).

Méthodes

- **Verbe** : pour effectuer une tâche (ex. : `Dessiner`).
- **Adjectif** ou **expression descriptive** : pour représenter une condition booléenne (ex. : `EstVide`).

Ne pas répéter le nom de la classe dans les noms des attributs et propriétés. Par exemple, une classe `Rectangle` a un attribut `_largeur`, et non pas `_largeurDuRectangle`.

Règles de codage

- Chaque classe est définie dans son propre fichier qui porte le même nom que la classe.
- Dans la définition d'une classe, les éléments publics sont définis en premier, suivis des éléments protégés (`protected`), et finalement des éléments privés.
- Une méthode ne contient pas plus de 30 lignes de code (sans compter les commentaires).
- Les accolades sont seules sur une ligne, l'ouverture et la fin du bloc sont alignées (sauf exception décrite plus bas).

```
// Faire
if (laCondition)
{
   AfficherMessage();
}

// Ne pas faire
if (laCondition) {
   AfficherMessage();
}
```

- Les blocs d'instructions sont toujours délimités par des accolades.

```
// Faire
if (laCondition)
{
   AfficherMessage();
}

// Ne pas faire
if (laCondition)
   AfficherMessage();
```

Exception : Dans une instruction `switch`, il n'est pas nécessaire d'ajouter des accolades à chaque bloc d'instructions d'un `case`.

- Une seule déclaration de variable est présente sur une ligne.

```
// Faire
int dividende;
int diviseur;

// Ne pas faire
int dividende, diviseur;
```

- Une seule expression est présente sur une ligne.

Exception : Une instruction `else` suivie d'une instruction `if` est considérée comme une seule expression.

```
// Préférer
if (laCondition)
{
   AfficherMessage();
}
else if (autreCondition)
{
   AfficherAutreMessage();
}

// À
if (laCondition)
{
   AfficherMessage();
}
else
{
   if (autreCondition)
   {
      AfficherAutreMessage();
   }
}
```

Exception : La définition d'une propriété automatique, ou la méthode `get` d'une propriété qui ne contient que l'instruction `return`, peuvent s'écrire sur une seule ligne

```
public int UnePropriete { get; private set; }

public int UneValeur
{
   get { return _valeur; }
}
```

- L'appel à un autre constructeur se fait sur sa propre ligne. Les deux points (:) sont placés à la fin de la déclaration du constructeur.

```
public MaClasse() :
   base()
{
}
```

Commentaires

- Chaque classe a un commentaire d'en-tête au format *CodeDoc* décrivant la classe. Il contient les balises :
  - `<summary>`,
  - `<remarks>`, optionnelle, seulement si nécessaire,
  - `<typeparam>`, pour une classe générique.
- Les commentaires ne dupliquent pas l'information fournie par le gestionnaire de code source (auteur, date, etc.).
- Chaque déclaration de méthode publique est accompagnée d'un commentaire en format *CodeDoc* décrivant la tâche effectuée. Il contient les balises :
  - `<summary>`,
  - `<remarks>`, optionnelle, seulement si nécessaire,
  - `<typeparam>`, pour une méthode générique,
  - `<param>`, une balise par paramètre de la méthode,
  - `<returns>`, pour une méthode avec une valeur de retour,
  - `<exception>`, une balise par type d'exceptions que la méthode peut lancer.

  ou
  - `<inheritdoc>`, dans une classe dérivée, pour une méthode abstraite ou virtuelle redéfinie.
- Chaque déclaration de propriété publique est accompagnée d'un commentaire en format *CodeDoc* décrivant la valeur. Il contient l'étiquette `<summary>`.
- Chaque déclaration de méthode et propriété non publique est accompagnée d'un commentaire indiquant la même information qu'indiqué ci-dessus, mais dans un format libre.
- À l'intérieur du code, les commentaires facilitent la compréhension. De ce fait, ils sont pertinents et expliquent des choses qui ne sont pas apparentes à la lecture du code.

Exemples à ne pas faire :

```
// Déclaration de la variable nombre
int nombre = 1;

// Incrémente le nombre de 1
++nombre;

// Affiche un message à l'écran
Console.WriteLine("Bonjour");

// Appel de la méthode AfficherMessage
AfficherMessage();

// Retourne le nombre
return nombre;
```