Les conventions de nommage en C#

Dans cet article, nous allons voir comment nommer correctement les éléments d'un programme C# afin d'écrire un code plus clair, plus professionnel et plus facile à maintenir.

Pourquoi des conventions de nommage ?

Quand on programme, on ne choisit pas les noms "au hasard". Un bon nom permet de comprendre rapidement :

- ce que représente une variable ;
- à quoi sert une méthode ;
- quel rôle joue une classe ;
- si un élément est privé, public, constant, local, etc.

Des conventions de nommage communes permettent aussi à plusieurs développeurs de lire le même code sans être perturbés par des styles différents.

En C#, ces conventions sont très importantes. Elles rendent le code plus lisible et plus cohérent.

Principe général

Un bon nom doit être :

- clair ;
- précis ;
- facile à lire ;
- lié au rôle réel de l'élément.

On évite donc les noms vagues comme :

```
1  int x;
2  string truc;
3  bool test;
```

On préfère :

```
1  int age;
2  string nomJoueur;
3  bool estVivant;
```

Deux styles à connaître absolument

En C#, il existe deux grandes conventions d'écriture pour les noms.

PascalCase

Chaque mot commence par une majuscule.

Exemples :

```
NomJoueur
PointsDeVie
AfficherPersonnage
```

camelCase

Le premier mot commence par une minuscule, puis chaque mot suivant commence par une majuscule.

Exemples :

```
nomJoueur
pointsDeVie
afficherPersonnage
```

Règle essentielle en C#

En C#, on utilise généralement :

- **PascalCase** pour les classes, méthodes, propriétés ;
- **camelCase** pour les variables locales et les paramètres.

Nommer une classe

Une classe représente souvent une entité, un objet ou un concept. Son nom doit donc être un nom clair, souvent au singulier.

On utilise **PascalCase**.

Exemples corrects :

```
class Personnage
class CompteBancaire
class CapteurTemperature
class Produit
```

Exemples moins bons :

```
class personnage
class perso
class machin
```

Conseil

Le nom d'une classe doit en général répondre à la question :

"Quel type d'objet est-ce ?"

Exemples :

- `Personnage`
- `Voiture`
- `Eleve`
- `Commande`

Nommer un objet ou une variable

Une variable contient une valeur. Son nom doit décrire cette valeur.

On utilise généralement **camelCase**.

Exemples :

```
string nom;
int pointsDeVie;
bool estVivant;
double temperatureActuelle;
```

Mauvais exemples

```
string a;
int valeur1;
bool b;
```

Ces noms ne disent presque rien.

Nommer une méthode

Une méthode représente une action. Son nom doit donc, en général, commencer par un verbe.

On utilise **PascalCase**.

Exemples :

```
AfficherPersonnage()
Attaquer()
RecevoirDegats()
CalculerTotal()
Demarrer()
```

Idée importante

Le nom d'une méthode doit répondre à la question :

"Que fait cette méthode ?"

On évite donc les noms trop flous comme :

```
Faire()
Traiter()
Go()
Action()
```

sauf si le contexte est vraiment évident.

Nommer une propriété

Une propriété représente souvent une caractéristique d'un objet.

On utilise **PascalCase**.

Exemples :

```
public string Nom { get; set; }
public int PointsDeVie { get; set; }
public bool EstVivant { get; set; }
```

Même si, dans d'autres langages, on voit souvent des attributs en minuscule, en C# une propriété publique se nomme en général comme une classe ou une méthode : avec **PascalCase**.

Nommer un champ privé

Un champ privé est une variable déclarée dans une classe, souvent utilisée en interne.

Il existe plusieurs styles selon les équipes, mais une convention très fréquente en C# est :

- **underscore + camelCase**

Exemples :

```
private string _nom;
private int _pointsDeVie;
private bool _estVivant;
```

Cela permet de repérer immédiatement qu'il s'agit d'un champ privé.

Exemple avec propriété :

```
class Personnage
{
    private int _pointsDeVie;

    public int PointsDeVie
    {
        get { return _pointsDeVie; }
        set
        {
            if (value >= 0)
            {
                _pointsDeVie = value;
            }
        }
    }
}
```

Nommer un paramètre

Un paramètre est une donnée reçue par une méthode.

On utilise généralement **camelCase**.

Exemple :

```
public void RecevoirDegats(int degats)
{
    pointsDeVie -= degats;
}
```

Autres exemples :

```
public void ChangerNom(string nouveauNom)
public int Additionner(int nombre1, int nombre2)
```

Nommer une constante

Une constante est une valeur qui ne change pas.

En C#, on utilise généralement **PascalCase** pour les constantes, même si dans d'autres langages on voit parfois du MAJUSCULE_AVEC_UNDERSCORES.

Exemple :

```
const int PointsDeVieMaximum = 100;
const double Tva = 0.21;
```

Les noms booléens

Les variables booléennes ne contiennent que deux valeurs : `true` ou `false`.

Leur nom doit donc exprimer clairement un état ou une question.

Exemples :

```
bool estVivant;
bool estConnecte;
bool aGagne;
bool peutAttaquer;
```

On reconnaît souvent un booléen grâce à des débuts comme :

- `est...`
- `a...`
- `peut...`
- `doit...`

Cela rend les conditions beaucoup plus lisibles :

```
if (estVivant)
{
    Console.WriteLine("Le personnage peut continuer.");
}
```

Éviter les abréviations inutiles

Les abréviations rendent souvent le code moins clair.

Exemple moins bon :

```
int nbPv;
string nomPers;
```

Exemple plus clair :

```
int nombreDePointsDeVie;
string nomPersonnage;
```

Cela dit, certaines abréviations très connues peuvent rester acceptables si elles sont comprises par tous, par exemple :

- `id`
- `url`
- `tv`

- `xml`

Mais il faut rester prudent.

Éviter les noms trop longs

Un nom doit être précis, mais pas interminable.

Exemple excessif :

```
int nombreTotalDePointsDeVieRestantsDuPersonnagePrincipal;
```

Exemple plus raisonnable :

```
int pointsDeVieRestants;
```

Le but est de trouver un équilibre entre précision et lisibilité.

Une idée par nom

Un nom doit représenter une seule chose claire.

Exemple confus :

```
string infos;
```

Que contient `infos` ? Un nom ? Un texte complet ? Une liste ? Une description ?

On préfère :

```
string description;
string nomComplet;
string messageErreur;
```

Respecter la langue choisie

Il faut éviter de mélanger plusieurs langues dans le même code.

Exemple à éviter :

```
string playerName;
int pointsDeVie;
bool isAlive;
```

Mieux vaut choisir une convention cohérente :

Tout en français

```
string nomJoueur;
int pointsDeVie;
bool estVivant;
```

Ou tout en anglais

```
string playerName;
int hitPoints;
bool isAlive;
```

Dans un contexte scolaire, surtout au début, tout en français peut être très utile pour comprendre les concepts. Dans un contexte professionnel, l'anglais est très fréquent.

Exemples complets

Exemple peu lisible

```
class perso
{
    public string n;
    public int pv;

    public void atk()
    {
        Console.WriteLine(n);
    }
}
```

Exemple mieux nommé

```
class Personnage
{
    public string Nom;
    public int PointsDeVie;

    public void AfficherNom()
    {
        Console.WriteLine(Nom);
    }
}
```

Exemple avec propriété et champ privé

```csharp
class Personnage
{
    private int _pointsDeVie;

    public string Nom { get; set; }

    public int PointsDeVie
    {
        get { return _pointsDeVie; }
        set
        {
            if (value >= 0)
            {
                _pointsDeVie = value;
            }
        }
    }

    public void RecevoirDegats(int degats)
    {
        PointsDeVie -= degats;
    }
}
```

Tableau récapitulatif

Élément	Convention fréquente	Exemple
Classe	PascalCase	`Personnage`
Méthode	PascalCase	`RecevoirDegats()`
Propriété	PascalCase	`PointsDeVie`
Variable locale	camelCase	`pointsDeVie`
Paramètre	camelCase	`degats`
Champ privé	`_camelCase`	`_pointsDeVie`
Constante	PascalCase	`PointsDeVieMaximum`

Ce qu'il faut retenir

Les conventions de nommage ne sont pas du "décor". Elles servent à écrire un code :

- plus lisible ;
- plus logique ;
- plus professionnel ;
- plus facile à corriger et à faire évoluer.

En C#, retiens surtout ceci :

- **Classes, méthodes, propriétés : PascalCase**
- **Variables locales et paramètres : camelCase**

- **Champs privés : souvent `_camelCase`**
- **Noms clairs, précis, cohérents**

Petit conseil de méthode

Quand tu dois choisir un nom, pose-toi simplement une de ces questions :

- “Qu’est-ce que cette donnée représente ?”
- “Que fait cette méthode ?”
- “Quel type d’objet est-ce ?”
- “Est-ce qu’un autre élève comprendrait ce nom sans explication ?”

Si la réponse est oui, le nom est probablement bon.