**Normes de programmation**

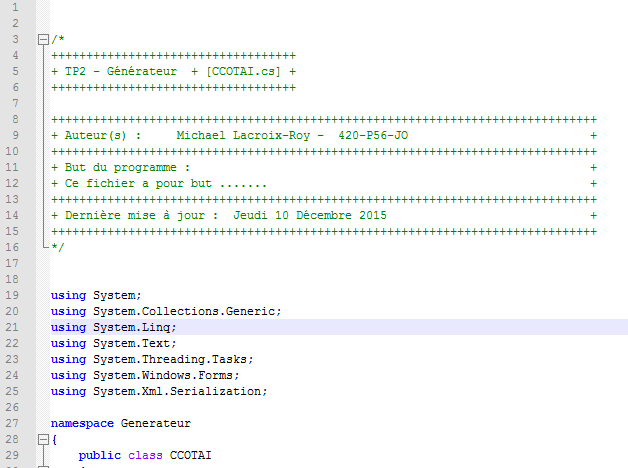
Les normes de programmation visent à augmenter la lisibilité et la compréhension du code en normalisant la façon de coder. De cette façon, les membres de l’équipe et les collaborateurs auront plus de faciliter à s’adapter au projet. Chaque membre de l’équipe à pour tâche de lire ce document et de mettre en pratique les éléments mentionnés dans ce document afin d’assurer la cohérence des normes établies.

1. **Documentation**

Les commentaires sont une façon de laisser de la documentation directement dans le code source. Il est donc important de bien commenter en tout temps chacun des éléments suscitant une compréhension, en évitant de surcharger le code avec des commentaires non pertinent. La langue qui serait utilisé est le français.

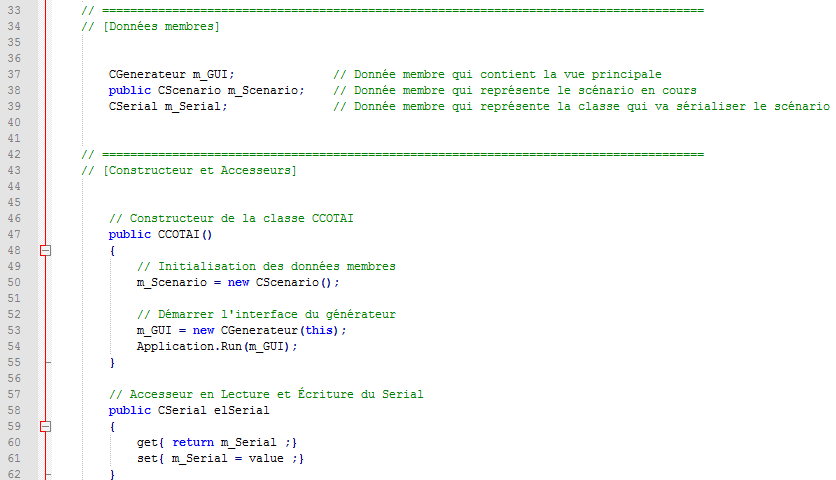
* 1. **Fichiers :**

Tout fichier de code source doit avoir un en-tête indiquant le suivi du développement :



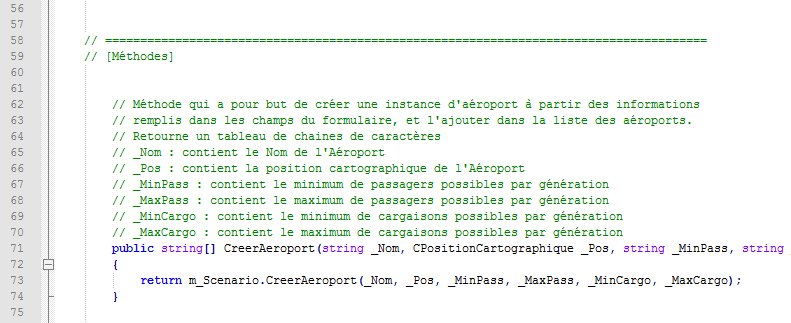
* 1. **Données membres, Constructeur et Accesseurs :**

Toute déclaration de variable ou donnée membre dans du code source devrait être précédée ou suivie d’un commentaire décrivant l’utilité de cette dernière. Les constructeurs et les accesseurs devraient avoir au minimum les descriptions suivantes.



* 1. **Fonctions et Méthodes :**

Toute fonction ou méthode dans du code source devrait avoir un en-tête la décrivant :

**

* 1. **Algorithmes :**

Il n’y a pas de règles précises sur les commentaires dans les algorithmes. Ceux-ci sont laissés au bon jugement des programmeurs.

**2. Nomenclature**

La nomenclature consiste à la façon de nommer et classer les termes que nous utilisés dans le domaine de la programmation.

**2.1 Constantes :**

Le nom des constantes doit être constitué que de lettres majuscules et de caractères de soulignement :



**2.2 Variables et Données membres :**

Le nom des variables devrait être significatif et avoir comme préfixe une abréviation de son type. De plus, toujours commencé avec une minuscule pour les noms de variables.

|  |  |  |
| --- | --- | --- |
| **Préfixe** | **Type** | **Exemple** |
| bo | bool | bool boSiValide = true; |
| i | int | int iNombre = 5; |
| f | float | float fMoyenne = 10.0; |
| d | double | double dMoyenne = 10.0; |
| str | string | string strMessage = “Coucou” |

D’autres préfixes devraient pouvoir s’ajouter :

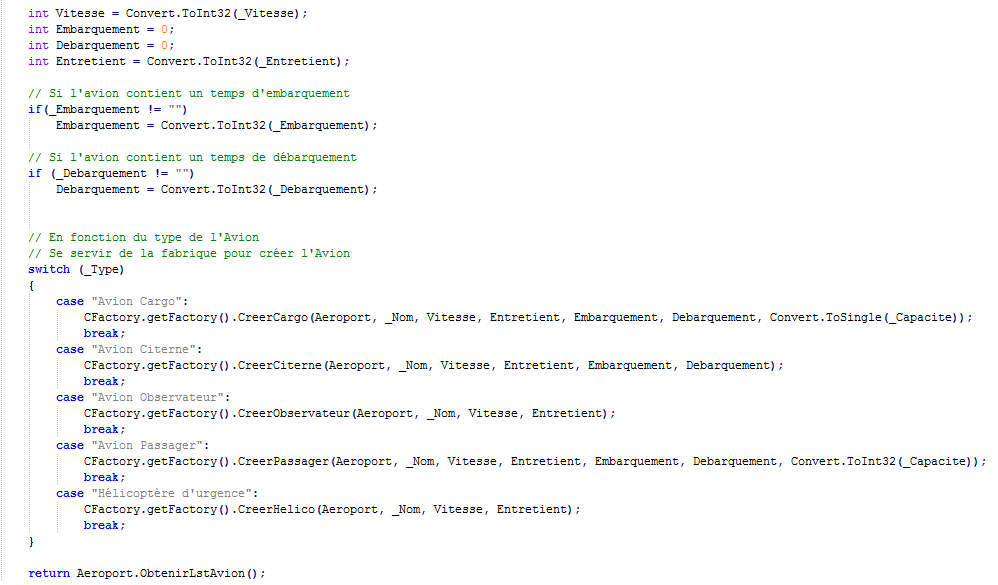
|  |  |  |
| --- | --- | --- |
| **Préfixe** | **Description** | **Exemple** |
| C | Classe | Public Class CBanane |
| m\_ | Donnée membre | CBanane m\_Banane = new CBanane(); |
| tab | Tableau | CBanane [ ] tabBanane = new CBanane[10]; |
| \_ | Paramètre | Public CBanane (CBanane \_Banane){} |

1. **Code source**

Les normes pour l’écriture du code source ont comme objectif de le rendre le mieux lisible possible.

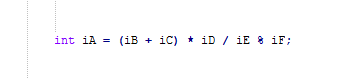
* 1. **Indentation**

L’indentation consiste à décaler, par des tabulations, certains blocs de code afin de représenter une certaine hiérarchie. Le code sous les termes « { », « if », « else », « case: », « private: », « public: » devrait être indenté. De plus, il ne faut en aucun cas dépasser la colonne 80 pour éviter le dépassement.



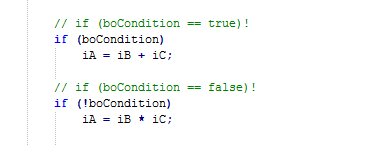
* 1. **Opérateurs**

Tous les opérateurs devraient être précédés et suivis d’un espace :



* 1. **Pléonasme**

Il est préférable d’éviter les pléonasmes de programmation :



* 1. **Redondance**

Il est préférable d’éviter la redondance de code inutile :

