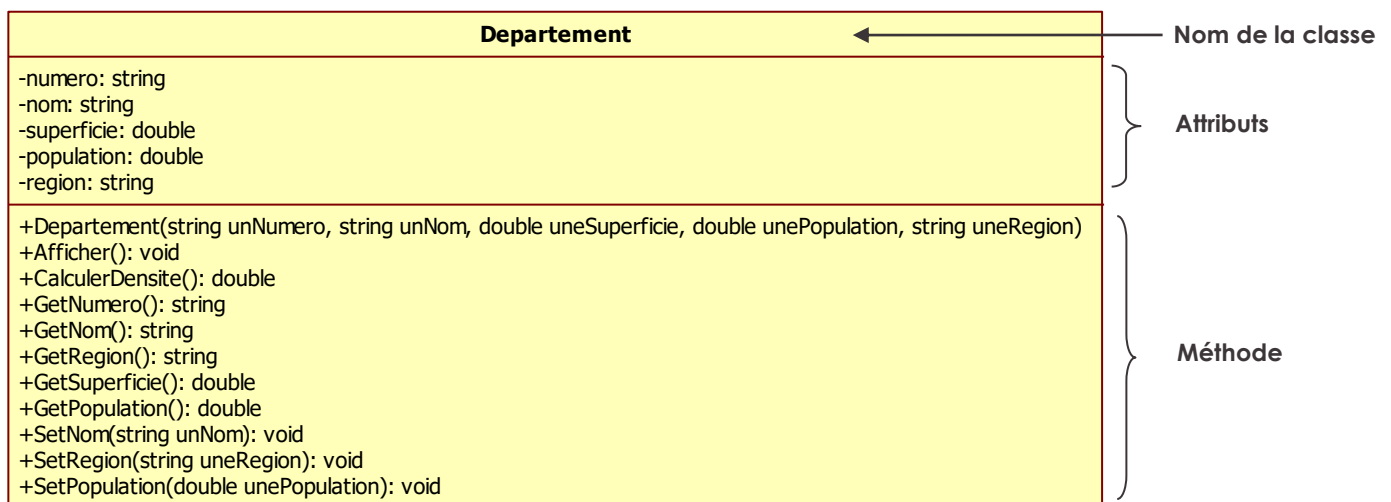


CLASSE Département

Chaque département est caractérisé par un numéro, un nom, une superficie exprimée en km², une population exprimée en milliers d'habitants et par le nom de la région à laquelle le département est rattaché.

Diagramme de classes UML



UML (Unified Modeling Language) : C'est un langage de modélisation objet utilisé par les concepteurs de logiciels pour représenter de manière graphique, sous des angles différents, le contenu et le fonctionnement de l'application.

UML propose 23 différents diagrammes : diagrammes de cas d'utilisation, de classes, de séquences, d'objets, d'états...

Diagramme de classes : Il représente la structure statique d'organisation des classes composant l'application. Les relations entre les classes sont également représentées.

Classe : Chaque classe est représentée sous la forme d'un rectangle divisé en 3 parties :

- le nom de la classe,
- les attributs de la classe (données),
- les méthodes de la classe (traitements sur les données ou comportement)

Accessibilité : Les membres de la classe (attributs et méthodes) sont précédés par un signe précisant leur niveau d'accessibilité :

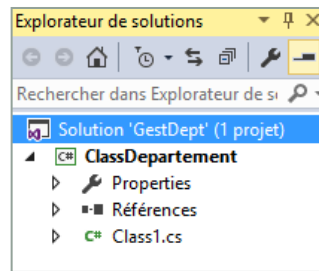
- membre privé, utilisé uniquement par la classe elle-même pour son fonctionnement interne
- + membre public, accessible sans restriction, au sein et en dehors de la classe

CONSTRUCTION PAS A PAS

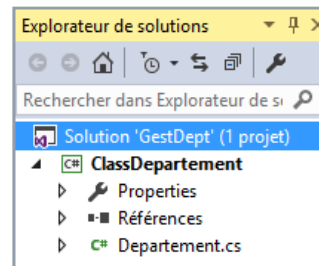
1. Lancer **Visual Studio** et créer un nouveau projet de type **Bibliothèque de classes** nommé ClassDepartement

Nommer la solution GestDept

Notre bibliothèque de classes va contenir
notre classe métier *Departement*



2. Visual Studio a préparé dans ClassDepartement une première classe vide nommée Class1.cs
Renommer cette **classe** en Departement.cs



Classe : Une classe permet de regrouper les données d'une entité et d'en exposer des services (méthodes). C'est le "moule" qui sera appliqué pour la construction de tous les objets, instances de la classe

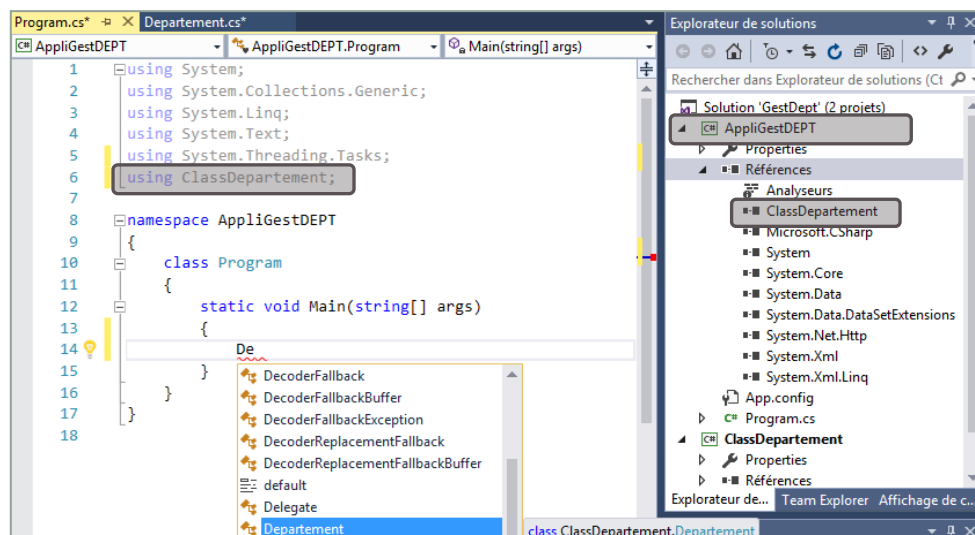
Attributs : Les attributs d'une classe correspondent aux données. Pour respecter le **principe d'encapsulation** de la POO, le niveau d'accessibilité des attributs est restreint (privé)

3. Définir les attributs de la classe Departement

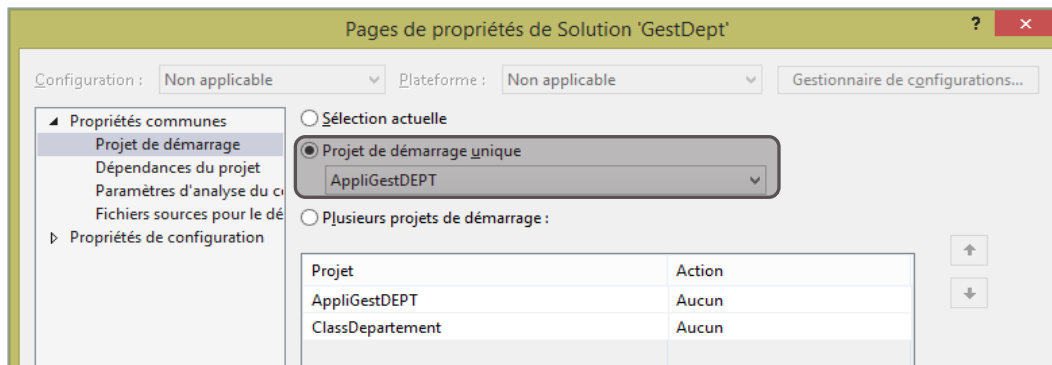
```
// Attributs privés
private string numero;
private string nom;
private double superficie; // en km2
private double population; // en milliers d'habitants
private string region;
```

4. Ajouter à la solution, un projet de type **Application Console** nommé AppliGestDEPT
Ajouter à ce projet une référence vers ClassDepartement
puis ajouter la directive `using ClassDepartement;`

Vous obtenez :



Dans les propriétés de la solution `GestionDept`, définir le projet de démarrage



Constructeur : Méthode particulière d'une classe, qui porte le nom de la classe et a pour rôle d'initialiser les attributs de la classe. Le constructeur est appelé pour construire (instancier) un objet d'une classe

Instanciation : Opération qui consiste à créer un objet (instance) d'une classe par appel du constructeur avec le mot clé **new**

5. Dans la méthode `Main()` de l'application `AppliGestDEPT`, instancier (construire) un objet de la classe `Departement`

```
Departement d = new Departement();
```

Exécuter le programme en mode *Pas à Pas détaillé (F11)* et observer la valeur des attributs de votre objet (*Fenêtre Variables locales*). Que peut-on en conclure ?

6. Implémenter le constructeur de la classe `Departement` conformément au diagramme de classe UML

```
public Departement(string unNumero, string unNom, double uneSuperficie,
                  double unePopulation, string uneRegion)
{
    this.numero = unNumero;
    this.nom = unNom;
    this.superficie = uneSuperficie;
    this.population = unePopulation;
    this.region = uneRegion;
}
```

this : Mot clé qui permet d'accéder à un membre (attribut ou méthode) de l'instance courante de la classe

7. Dans l'application `AppliGestDEPT`, créer un objet de la classe `Departement` à l'aide du constructeur que vous venez d'implémenter.

```
Departement d = new Departement("40", "Landes", 9243, 327, "Aquitaine");
```

Exécuter le programme en mode *Pas à Pas détaillé* et observer la valeur des attributs de votre objet

8. Implémenter les méthodes `Afficher()` et `CalculerDensite()` de la classe `Departement`

- La méthode `Afficher()` permet d'afficher le détail des informations d'un département sous la forme suivante :

```
40      Landes  9243  327  Aquitaine
```

```
public void Afficher()
{
    Console.WriteLine("{0}\t{1}\t{2}\t{3}\t{4}", this.numero, this.nom,
                  this.superficie, this.population, this.region);
}
```

- La méthode `CalculerDensite()` retourne la densité du département (en nombre d'habitants au km²)

```
public double CalculerDensite()
{
    double densite;
    densite = this.population * 1000 / this.superficie;
    return densite;
}
```

9. Dans l'application `AppliGestDEPT`, tester le comportement des deux méthodes créées précédemment
10. Dans l'application `AppliGestDEPT`, on souhaite :
 - a. afficher le nom du département crée. Est-ce possible ? Si non, pourquoi ?
 - b. modifier le nom du département crée. Est-ce possible ? Si non, pourquoi ?

Accesseur / Mutateur : Ces méthodes permettent d'obtenir et définir la valeur d'un attribut privé de la classe :

- méthode pour la lecture : **accesseur** ou **getter** (*get = obtenir*)
- méthode pour l'écriture : **mutateur** ou **setter** (*set = définir*)

Les accesseurs/mutateurs rendent ainsi possible(s) la lecture et/ou la modification de la valeur d'un attribut privé en dehors de la classe dans laquelle il a été défini

11. Implémenter les accesseurs (méthode `Get...()`) pour chaque attribut de la classe `Departement`

```
public string GetNumero()
{
    return this.numero;
}
```

12. Implémenter les mutateurs (méthode `Set...()`) sur les attributs `nom`, `region` et `population` de la classe `Departement`

```
public void SetNom(string unNom)
{
    this.nom = unNom;
}
```

13. Dans l'application `AppliGestRH`, utiliser les accesseurs/mutateurs que vous venez de définir
14. Organiser en régions le code de la classe `Departement`

```
public class Departement
{
    #region attributs
    private string numero;
    private string nom;
    private double superficie; // en km2
    private double population; // en milliers d'habitants
    private string region;
    #endregion

    constructeur

    méthodes
}
```

15. Documenter la classe Département

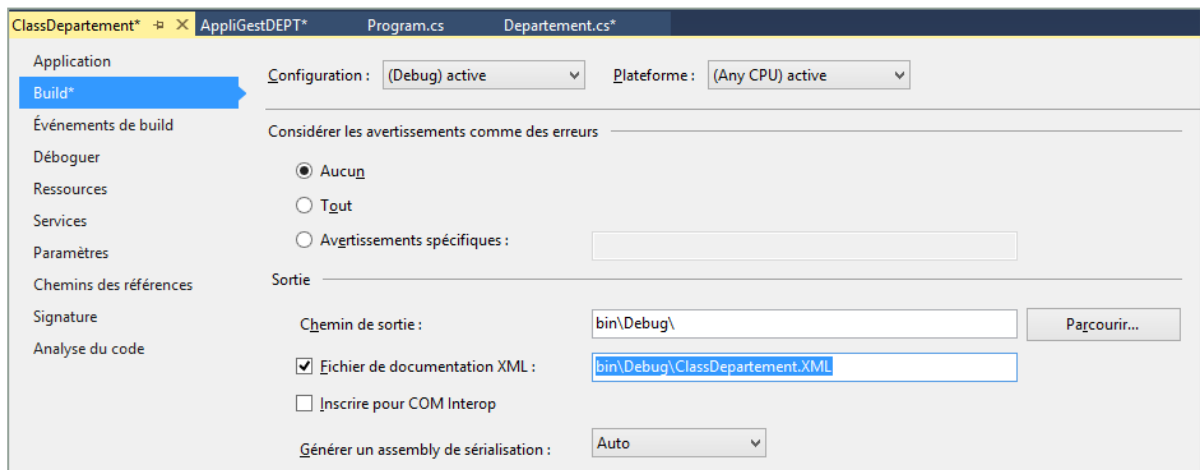
Compléter les tags de documentation pour la classe elle-même et chaque méthode de la classe

Exemple :

```
/// <summary>
/// Retourne la densité du département
/// </summary>
/// <returns>densité du département en nombre d'habitants / km2</returns>
public double calculerDensite()
```




16. Générer le composant logiciel (assembly ou bibliothèque de liens dynamiques - DLL) documenté correspondant à votre bibliothèque de classe ClassDépartement

- Au préalable, dans les propriétés du projet ClassDépartement, aller dans l'onglet Build pour définir les paramètres de génération du composant (cocher notamment l'option Fichier de documentation XML)



- Dans le menu Generer, choisir l'option Generer ClassDépartement

Vous obtenez dans le dossier bin/debug de ClassDépartement :

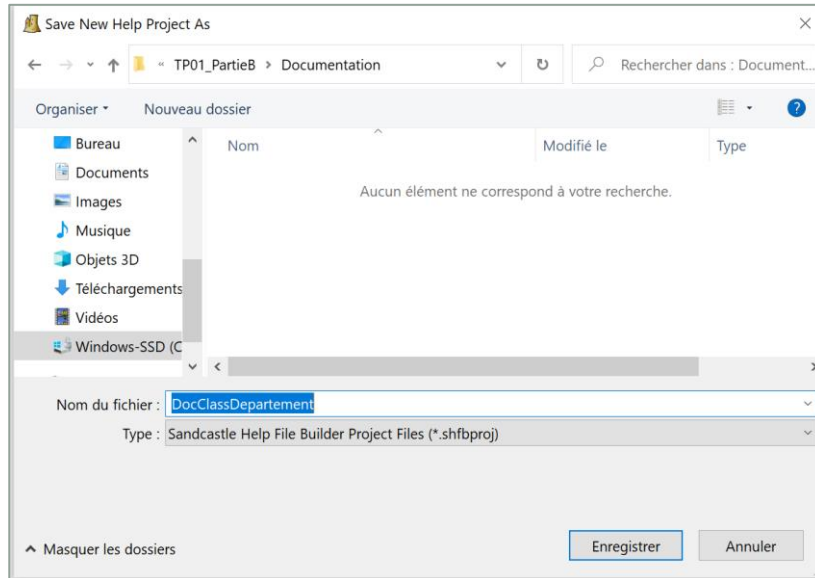
	ClassDépartement.dll	15/01/2023 17:37	Extension de l'application
	ClassDépartement.pdb	15/01/2023 17:37	Program Debug Database
	ClassDépartement.XML	15/01/2023 17:37	Document XML

Pour tester votre composant : créer un nouveau projet console, ajouter une référence à ClassDépartement.dll, ajouter la directive using ...

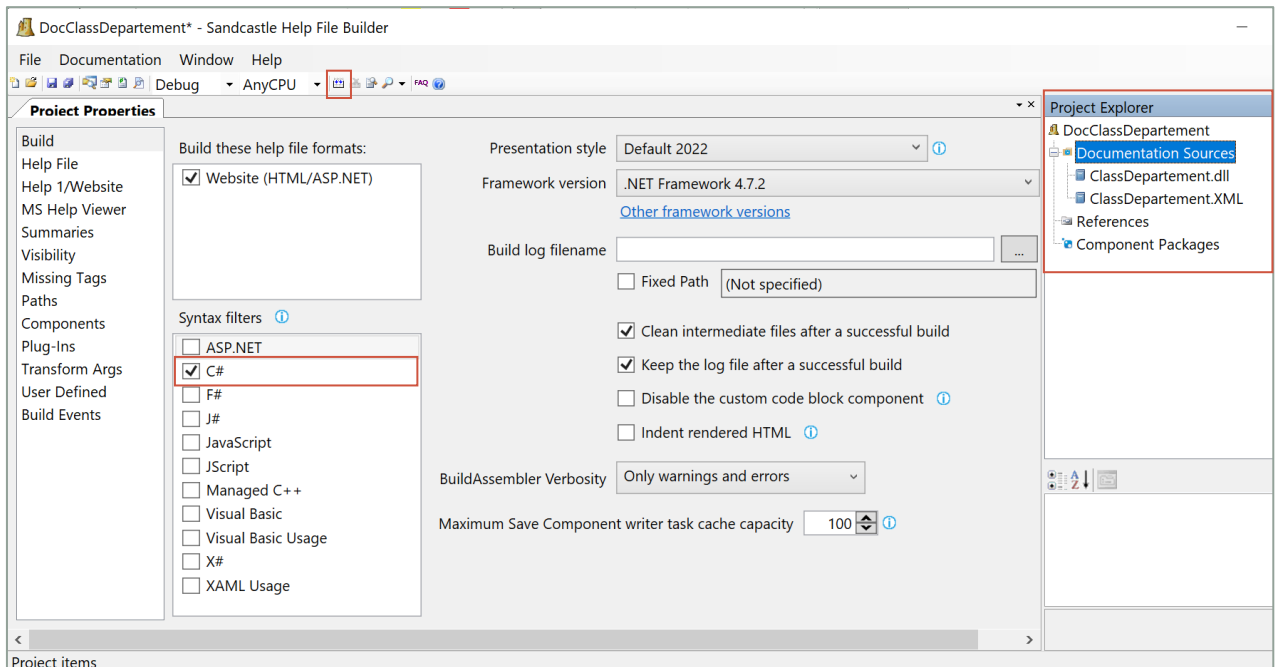
17. Générer la documentation technique avec **Sandcastle Help File Builder**

- Télécharger et installer SHFB : <https://github.com/EWSoftware/SHFB/releases>
- Lancer SHFB et créer un nouveau projet (File / New project)

Créer un dossier Documentation et nommer le fichier DocClassDepartement



- Choisir le langage C# et ajouter le code source (dans Project Explorer, click droit sur Documentation Sources / Add Documentation Source)



- Cliquer sur l'icône Build the help file... la documentation a été générée (dossier Documentation\Help)
Ouvrir le fichier index.html dans le navigateur