**1. Classe - Objets**

**1.1 Un objet est une instance particulière d'un modèle de classe.**

Le type d'un objet est sa classe d'appartenance, accessible avec la méthode **GetType**() : **ObjetInstancié.GetType()**

La classe est caractérisé par

* ses **propriétés** : "private", ont un nom, un type
* son **constructeur** : méthode de **même nom que la classe**, qui spécifie les modalités d'instanciation de l'objet
* ses **accesseurs** : **set** = affectation de valeur, **get** = retour de valeur
* ses **méthodes** : d’entrée/sortie, de calcul, de comparaison, ...

**1.2 L'encapsulation**

**public class NomDeLaClasse**

{

*// Déclaration des propriétés privées*

*// Déclaration des méthodes de l'objet*

}

**E**st le principe qui permet de rassembler les propriétés et les méthodes d'un objet pour les manipuler en toute sécurité à partir de sa classe d'appartenance Les propriétés ne sont accessibles depuis ou vers l'extérieur de la classe qu'à partir des accesseurs.

Déclaration d'une classe en C#

**namespace libEtudiant** *// sera déclaré en clause using*

{

**public class Etudiant** *// sera instancié pour créer un objet de type Etudiant*

*// Etudiant Durand = new Etudiant(15)*

{

*// propriétés* ***privées*** *de la classe toujours déclarées en premier*

private int **n**umEtudiant ;

private string **n**omEtudiant ;

private string **p**renomEtudiant ;

*//* ***constructeur******public*** *de la classe Etudiant peut être surchargé*

public **Etudiant** (int **N**umEtudiant)

{

**this.numEtudiant = NumEtudiant ;** *// au moment de la création de l'objet on doit impérativement*

*valoriser numEtudiant*

}

public Etudiant()

{

}

*//* ***accesseurs*** *de la classe*

public string **N**umEtudiant *// variable visible et utilisable à l'extérieure de la classe*

{

**get**  { **return** NumEtudiant ; } *//* ***retour*** *de valeur vers programme externe*

}

public string **N**omEtudiant *// variable visible et utilisable à l'extérieure de la classe*

{

**get**  { **return** NomEtudiant ; } *//* ***retour*** *de valeur vers programme externe*

**set** { NomEtudiant = **value** ; } *//* ***affectation*** *d'une valeur à la propriété*

}

*// Méthode privée de la classe*

public void AfficherNomPrenom()

Console.WriteLine( ...................)

Quand on considère une méthode par rapport à l'objet à laquelle elle est appliquée, il faut voir l'objet comme étant sollicité de l'extérieur par un **message**. **Ce message peut comporter des paramètres**.

L'objet doit alors réagir à ce message en exécutant cette fonction ou méthode diagramme de cas d'utilisation UML

**2. Manipulation des objets**

**2.1 Une classe n'est pas exécutable.**

Il faut donc créer un programme, externe à la classe et qui permet :

* d'instancier les objets
* de fournir des valeurs aux propriétés de l'objet
* d'utiliser les méthodes de gestion des objets instanciés

**2.2 Seuls les objets de la classe ont une "existence"**

Pour qu'un **objet** ait une existence, il faut qu'il soit **instancié**

Une même classe peut être instanciée plusieurs fois, permettant la création de n objets, ou instances, ayant des valeurs de propriétés spécifiques.

public static void Main(String[] args)

{

*// On crée une instance de type nomClasse*

**nomClasse NomObjet = new nomClasse(valeurs)**

...........

public static void Main(String[] args)

{

*// On définit le type de l'objet*

*// Puis on instancie une nouvel objet de ce type*

**nomClasse NomObjet ;**

**NomObjet = new nomClasse(valeurs)**

**...........**

En C#, il n'existe qu'une seule manière de créer une instance d'une classe avec l'ordre new()

La création d'un objet peut se faire en deux temps

* Déclaration d'une variable du type de la classe de l'objet,
* Instanciation de cette variable par l'instruction new

OU la déclaration du type et la création de l'objet s'effectuent en une seule instruction

Les valeurs passées au moment de la création de l'objet dépendent du, des **constructeurs** déclarés dans la classe

La **surcharge de constructeurs** permet de créer un objet :

* sans affecter de valeurs à ses propriétés au moment de l'instanciation
* en affectant à toutes les propriétés de l'objet une valeur
* en affectant une ou plusieurs propriétés d'une valeur

**2.3 Accès aux propriétés**

L'accès aux propriétés et aux méthodes d'une classe se fait par l'intermédiaire d'opérateurs.

* Etudiant Dupont = new Etudiant(15); *// instanciation de Dupont avec le numéro = 15*
* Dupont.**NomEtudiant**  = "Dupont"; *// affectation de son nom*
* Dupont.**PrenomEtudiant**  = "Alicia"; *// affectation de son prénom*
* Dupont. **AfficherNomPrenom()**; *// Appel de la méthode AfficherNomPrenom()*

Toute méthode appelée - message - est suivie des valeurs des paramètres de même type que ceux déclarés dans la classe.

**3. Conventions d'écriture**

Ces conventions ne sont pas normalisées et peuvent différer en fonction des ouvrages consultés.

Classes Les identificateurs de classe commencent par une Majuscule

Propriétés Les identificateurs public de propriétés commencent par une majuscule

Les propriétés private ou protected par une minuscule

Méthodes Commencent par une lettre en majuscule : AjouterVehicule(), Demarrer(), CalculerImpot()

Le premier mot est un verbe.

Nom des fichiers Une classe d'objet a pour extension .cs

Le nom de fichier associé à une classe est identique au nom de la classe : Impot.cs

L'espace de nom sera différent : libImpot

Variables de travail Commencent par une minuscule