Framework composé de deux éléments:

* Le CLR
* Des bibliothèques de classes

Les bibliothèques de classes :

Le frameword.NET est composé de plusieurs bibliothèques de classes.

—>

System //Espace de nom de racine. Il contient les types de base Framework.NET

System.Collection // Tous les types permettant de gérer les listes et les tableaux.

—>

Les types :

.NET est composé de deux types différents : les types valeur et les types référence.

**Les types valeurs**

—>

int test = 4; //Instruction qui déclare une variable est qui lui affecte une valeur “4”

int test2; //Une déclaration de type valeur sans affectation entraîne la levé

test2 = 4; //Affectation

bool b1 = true; //

bool b2 = b1; // On copie la valeur de “b1” sans aucun autre lien

b1 = false; // L’instruction suivante affecte la variable false a “b1”

—>

**Les types références**

Les données des types références sont stockées à un emplacement mémoire et un pointeur vers ces données et stockées dans la variable.

Les classes et les interfaces sont des types références.

Un type références doit être instancié avec le mot clé “new”.

—>

System.Collections;

ArrayList tab1 = new ArrayList();

ArrayList tab2 = new ArrayList(1);

ArrayList tab2 = new(); //On peut omettre le type lorsqu’il n’y a pas de paramètres.

—>

Pour illustrer le fonctionnement des types références, prenons un exemple :

—>

class Test

{

public bool booleen;

}

Class C1 = new Test();

C1.booleen = true;

Test C2 = C1;

C1.booleen = false;

—>

Les identifiants et mots-clés

—>

string monIdentifiant;

string @monIdentifiant;

—>

“@” sert à transformer un mot clé de programmation en nom de variable.

ex :

private bool @string; → ici string devient un nom de variable.

**Les opérateurs d’assignation :**

—>

int i = 4;

i += 2; // 6

i -= 2; // 2

i \*= 2; // 8

i /= 2; // 2

i %= 2; // 0

int x =4;

x is int; // Renvoie true si x est de type “int”

—>

!!! Aucun “===” en C# !!!

—-----------------------

enum JourDeLaSemaine

{

Lundi,

Mardi,

Mercredi,

Jeudi,

Vendredi,

Samedi,

Dimanche,

}

—----------------------

Un type d’énumération affecte une valeur automatiquement à tous les “objets” qui sont instanciés dedans. On peut aussi leur donner une valeur manuelle. |

<–

enum JourDeLaSemaine

{

Lundi = 0,

Mardi = 1,

Mercredi = 100,

Jeudi = 101,

Vendredi = 500,

Samedi = 501,

Dimanche = 502,

}

—-------------------------

short s = 300;

byte b = (byte)s;

Console.Writeline(b);

Console.ReadLine();

**Les niveaux d’accès :**

* public : autorise l’accès pour les types de l’assemblage (assembly)
* private : autorise uniquement l’accès pour les autres membres du type (de la classe par exemple)
* internal : autorise l’accès pour les types d’assembly
* protected : autorise l’accès uniquement pour les autres membres du type et pour les types héritant de celui-ci (dans l’assembly et en dehors de l’assembly)
* protected internal : autorise l’accès uniquement pour les autres membres du type et pour les types héritant de celui-ci (dans l’assembly uniquement)

Exercice :

* Que signifie le terme “assembly” ?
* Citez un exemple réel d’un usage pertinent du mot clé “private”

Si aucune portée n’est précisée sur un membre, il est considéré comme private. Une classe sans modificateur sera quant à elle considérée comme public.