

RESUME DE SYNTAXE DU LANGAGE C#
L. BUZER – B. PERRET – ESIEE 2011

ESPACE DE NOMMAGE

```
using System;
namespace Projet { ... }
```

TYPES & CONVERSION

```
int i = 1;
float s = 1.0f;
double d = 1.0;
int k = (int) (s);
```

CHAINES DE CARACTERES

```
string g = "Hello !\n";
string s = "Bon" + "jour";
```

TABLEAUX

```
int[] T = new int[100];
int[] T = new int[] { 5, 6, 7 };
for(int i=0;i<T.length;i++) Aff(T[i]);
ou foreach (int v in T) Aff(v);
```

```
int[,] K = new int[2,2];
K = new int[,] {{4, 5},{6,7}};
Aff(K[1,2]);
```

TABLEAUX EN ESCALIER

```
int[][] T = new int[2][];
T[0] = new int[] {4,5};
T[1] = new int[] { 1, 2, 3 };
Aff(K[1][2]);
```

CLONAGE

```
Choix 1 :
int [] T2 = new int[T1.Length];
T1.CopyTo(T2,0);
Choix 2 :
int [] T2 = (int []) T1.Clone();
```

PROPRIETES

```
class P
{
    private string _Nom;
    public string Nom
    {
        get { return _Nom; }
        set { _Nom = value; }
    }
    // ou
    public int Age { get; set; }
}
```

PASSAGE PAR REFERENCE DES VALEURS

```
void fnt1()
{
    int a = 7 ;
    fnt2(ref a);
    aff(a); // 8
}
void fnt2(ref int b)
{ b++; }
```

HERITAGE

```
class A
{
    public A() {...}
}
```

```
class B : A
{
    public B() : base() {...}
}
```

CONSTRUCTEURS

```
class Cercle
{
    // CTR par défaut
    public int x = 0; // init explicite
    public int y; // init implicite

    public Cercle() // CTR sans argument
    {
        x = 10 ; y = 10 ;
    }

    public Cercle(int _x, int _y)
    {
        x = _x ; y = _y ;
    }
}
```

```
Cercle c1 = new Cercle();
Cercle c2 = new Cercle(4,5);
Cercle [] v = new Cercle[5];
Cercle c3 = new Cercle;
```

POLYMORPHISME

```
class A
{
    public virtual void Aff() {...}
}
class B : A
{
    public override void Aff() {...}
}
```

INTERFACE

```
interface IPoli
{ void Bonjour();
  void Aurevoir(); }

class Enfant : IPoli
{
    public void Bonjour() {...}
    public void Aurevoir() {...}
}
```

SURCHARGE D'OPERATEUR

```
struct P2
{
    public int x;
    public int y;

    public static P2 operator * (int a, P2 b)
    {
        P2 t;
        t.x = b.x*a;
        t.y = b.y*a;
        return t;
    }
}
```

INDEXEUR

```
class Dictionnaire
{
    string[] Mots =
        new string[] {"MA","LE","FA"};
    public string this[int pos]
    {
        get { return Mots[pos]; }
        set { Mots[pos] = value; }
    }
}
```

CLASSE GENERIQUE

```
public class Stack<T>
{
    T[] m_Items;
    public void Push(T item) {...}
    public T Pop() {...}
}
Stack<int> stack = new Stack<int>();
stack.Push(1);
int number = stack.Pop();
```

FONCTIONS GENERIQUES

```
public class MyClass
{
    public void MyMethod<T>(T t)
    {...}
}
```

CONTENEURS GENERIQUES

```
using System.Collections.Generic;

List<T> : Tableau dynamique
Stack<T> : Pile
LinkedList<T> : Liste doublement chaînée
Dictionary<K,V> : Couples clé/valeur
```