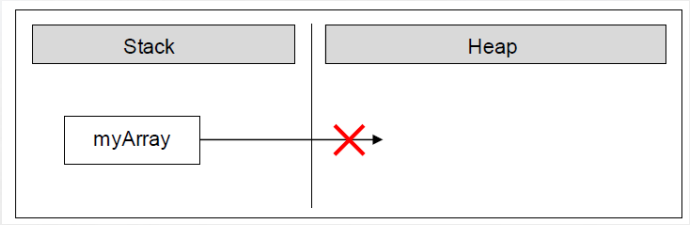
C# Synthèse

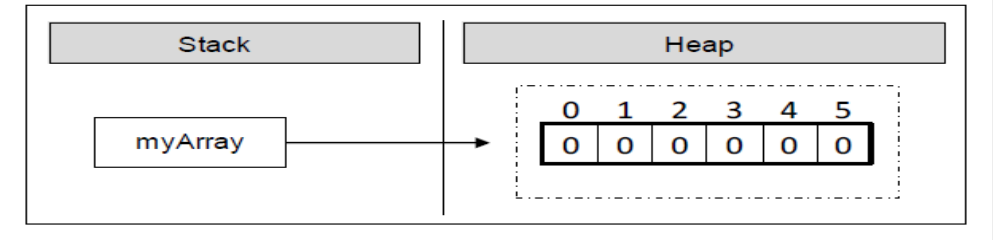
Collections

Tableaux

- Etape 1: création de la référence
  - Exemple:int[] myArray;



- Etape 2: allocation du tableau
  - Exemple: myArray = new int[5];



Écriture alternatives

```
// reference et creation et initialisation
int[] array2 = new int[] { 1, 3, 5, 7, 9 };
// reference et creation et initialisation - Alternative syntax
int[] array3 = { 1, 2, 3, 4, 5, 6 };
```

Tableaux

- Tableaux d'objets complexes

```
8 références
class Mammifere
{
    2 références
    public uint Age { get; set; }
    2 références
    public String Nom { get; set; }

    3 références
    public Mammifere(uint age,string nom)
    {
        Age = age;
        Nom = nom;
    }

    0 références
    public override String ToString()
    {
        return ("mon nom est " + Nom + " et j'ai " + Age);
    }
}
```

```
Mammifere[] zoo= new Mammifere[2]{
                        new Mammifere(2,"oscar"),
                        new Mammifere(20,"zoe")};
Mammifere[] veto=new Mammifere[1];
veto[0]=new Mammifere(3,"peggy");
```

Tableaux

- Parcourir un tableau
  - La boucle for
    - Pour des accès en LECTURE ET/OU ECRITURE

```
for (int i=0; i < zoo.Length; i++)
            {
                Mammifere unMammifere = zoo[i];
                Console.WriteLine(unMammifere);

            }
```

  - La boucle foreach
    - Pour des accès en LECTURE (pas de suppression d'éléments par exemple)

```
foreach (Mammifere unMammifere in zoo)
{
Console.WriteLine(unMammifere);

}
```

Tableaux

```
static void Main(string[] args)
{
 /******************* Array demo ***************************/
 byte[] tab = new byte[10];
 Random rnd1= new Random();
 rnd1.NextBytes(tab);
 foreach (byte ibyte in tab)
         Console.Write(ibyte + " ");
 if (((IList<byte>)tab).Contains(10))
         Console.WriteLine("contient la valeur 10");
 else
         Console.WriteLine("ne contient la valeur 10");
 Array.Sort(tab);
 foreach (byte ibyte in tab)
         Console.Write(ibyte + " ");
 Console.WriteLine(" \n nombre pair");
 byte[] tab_pair = Array.FindAll(tab,  Fct_Recherche_Pair);
 foreach (byte ibyte in tab_pair)
         Console.Write(ibyte + " ");
 Console.WriteLine("\n redimensionne tableau");
 Array.Resize<byte>(ref tab, 20);
 foreach (byte ibyte in tab)
          Console.Write(ibyte) ;
 Console.ReadKey();
}

private static bool Fct_Recherche_Pair(byte val)
{
 if ((val % 2) == 0)
  return (true);
 else
  return (false);
}
```

Methode Contains teste si la valeur est contenue
Il faut caster le tableau avec IList<monType>

La méthode sort est statique

FindAll recherche en se basant sur des critères utlisateurs. On passe en paramètre une référence vers une méthode(et non sur un objet) : concept de délégation (pointeur de fonction en C++)

resize redimmensionne le tableau

Ma méthode pour définir un critère spécifique de recherche

```
221 147 115 135 32 133 201 104 150 94 ne contient la valeur 10
32 94 104 115 133 135 147 150 201 221
 nombre pair
32 94 104 150
 redimensionne tableau
32 94 104 115 133 135 147 150 201 221 0 0 0 0 0 0 0 0 0 0
```

Liste

- Définition
  - ensemble linéaire ordonné d'éléments de même type auxquels on accède séquentiellement

Nœud de départ (*head*): indique le début de la liste chaînée. Ne contient pas de données. Contient une référénce vers le 1er nœud de données

Nœud de données : contient 2 choses : la donnée + une référence vers le noeud de données suivant

Nœud de fin (*tail*): contient un donnée + une référence null indiquant qu'il s'agit du dernioer nœud

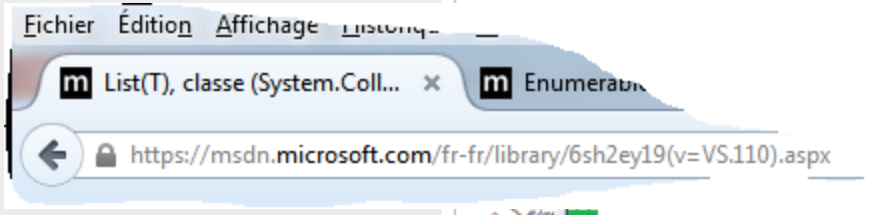
structures évoluant dynamiquement =>
Utile lorsque la taille n'est pas connue à la compilation)

Liste

- Classe générique List<T>
  - T répresente le type de la liste qui est générique

List<int> maListeInt=new List<int>();
List<Mammifere> monZoo=new List<Mammifere>();

- Méthodes
  - Méthodes d'instances de la classe List
    - Exemple: Ajout d'un élément
  - Méthodes d'extensions
    - Utilise expression Lamba (+ tard)



	Nom	Description
	Add	Ajoute un objet à la fin de List<T>.
	AddRange	Ajoute les éléments de la collection spécifiée à la fin du List<T>.
	AsReadOnly	Retourne un wrapper IList<T> en lecture seule pour la collection actuelle.
	BinarySearch(T)	Recherche un élément utilisant le

Liste

- Exemple1

```
static void Main(string[] args)
{
    Random rnd2 = new Random();
    List<int> listTab = new List<int>(10); // capacité de la liste:  10 elts
    for (int i = 0; i < 10; i++)  // ATTENTION: LA LISTE EST VIDE ,CAPACITE différent de TAILLE
        // listTab[i] = 100; COMPILATEUR KO car la liste est vide
        listTab.Add(rnd2.Next(1000)); // rajout nbre random <1000
    foreach (int val in listTab) Console.Write(val + " "); // parcours de la liste
    listTab[0] = 100; // acces à un elt EXISTANT
    listTab.Insert(1, 101); // insertion elt 101 à l'index 1
    Console.WriteLine("\ncapacité de la liste "+listTab.Capacity); //la capacité a été redimmensionné!
    listTab.RemoveAt(2); // suppression elt à l'index 3
    Console.WriteLine("\n Liste mofidifiée");
```

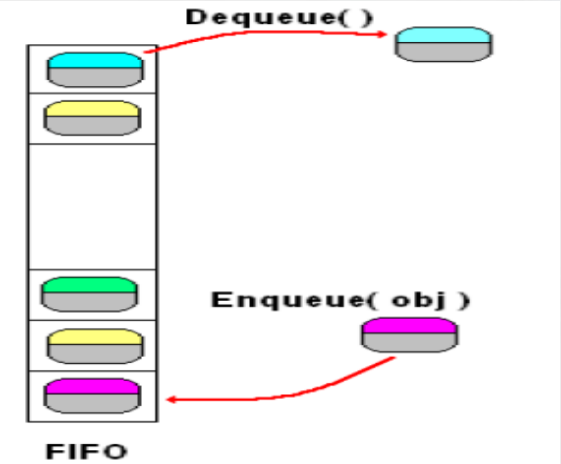
Liste

- exemple2

```
0 références
static void Main(string[] args)
{
    List<string> words = new List<string>(); // New string-typed list
    words.Add("melon");
    words.Add("avocado");
    words.AddRange(new[] { "banana", "plum" });
    words.Insert(0, "lemon"); // Insert at start
    words.InsertRange(0, new[] { "peach", "nashi" }); // Insert at start
    words.Remove("melon");
    words.RemoveAt(3); // Remove the 4th element
    words.RemoveRange(0, 2); // Remove first 2 elements
    // Remove all strings starting in 'n':
    words.RemoveAll(s => s.StartsWith("n"));
    Console.WriteLine(words[0]); // first word
    Console.WriteLine(words[words.Count - 1]); // last word
    foreach (string s in words) Console.WriteLine(s); // all words
```

PILE

- FIFO
  - Les premiers seront les derniers!




```
static void Main(string[] args)
        {
            // Creates and initializes a new Queue.
            Queue<string> myQ = new Queue<string>();
            myQ.Enqueue("Hello");
            myQ.Enqueue("World");
            myQ.Enqueue("!");
            // Displays the properties and values of the Queue.
            Console.WriteLine("myQ");
            Console.WriteLine("\tCount:    {0}", myQ.Count);
            Console.Write("\tValues:");
            PrintValues(myQ);
            Console.ReadKey();
        }


        public static void PrintValues(IEnumerable<string> myCollection)
        {
            foreach (string obj in myCollection)
                Console.Write("    {0}", obj);
            Console.WriteLine();
        }
```

DICTIONNAIRE

- Principe
  - Un dictionnaire est une collection d'objets de type clé/ Valeur
  - Chaque clé doit être unique
- Classe dictionnary
  - exemple

```
Dictionary<UInt32,Etudiant> dicLPSE= newDictionary<UInt32,Etudiant> ();
```

Ajout :

```
dicLPSE.Add(100214,new Etudiant("Copter","Elie",20,100214));
```

Accès par clé (sans protection : clé inexistante)

```
if (!dicLPSE.TryGetValue(100214, out unEtudiant))
Console.WriteLine("No val"); // "No val"
else
Console.WriteLine(unEtudiant);
```

Itérer (parcourir) la collection

```
foreach (KeyValuePair<UInt32, Etudiant> kv in dicLPSE)
    Console.WriteLine(kv.Key + "; " + kv.Value);
```