Collections en C#

NORA IZRI

ESILV - DÉPARTEMENT IBO

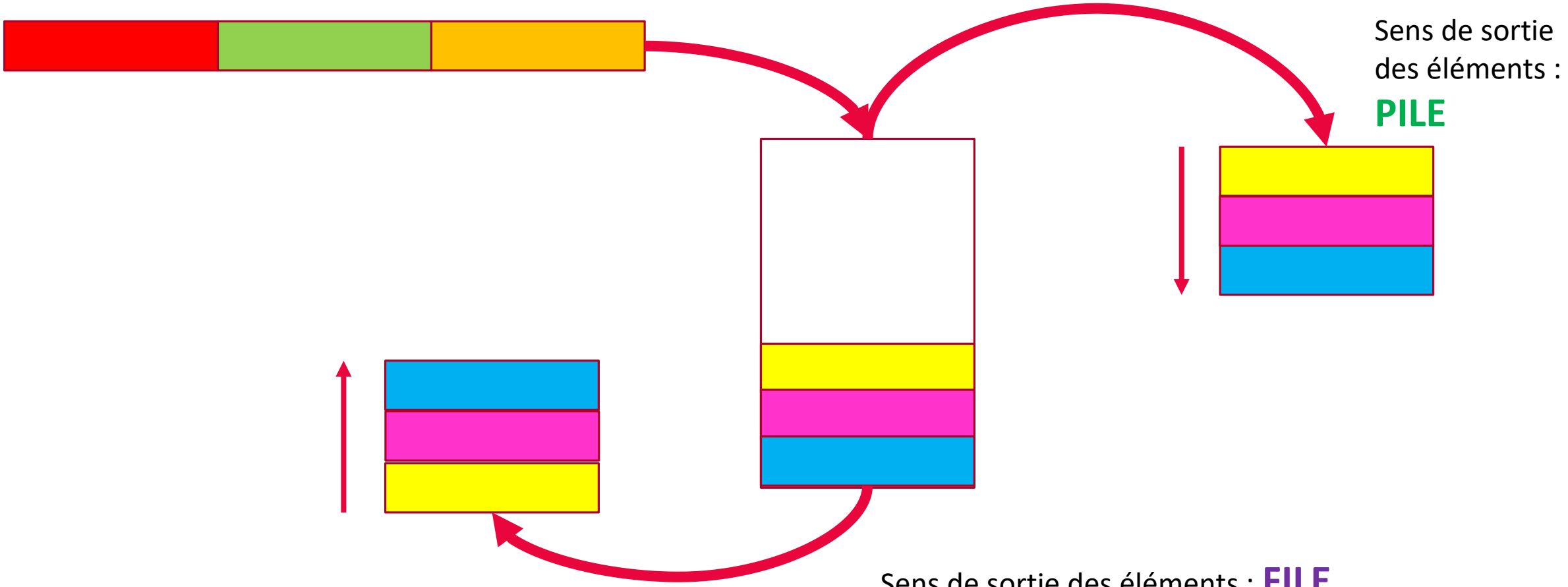
Collections?

- Les collections :
  - sont des objets
  - permettent de regrouper et gérer plusieurs objets de taille et/ou types dynamiques

Exemples de collections

Nom de collections	Définition
BitArray	Tableau statique de booléens
ArrayList	Tableau dynamique
Hashtable	Couple clé/valeur
SortedList	Couple clé/valeur ordonnée selon la clé
Stack	Pile ➔ Principe LIFO (Last In First Out)
Queue	File ➔ Principe FIFO (First In First Out)

Pile / File



Namespace « Collections »

- Inclure le namespace « Collections »

```
using System.Collections;
```

Exemple « BitArray »

- Collection « BitArray » ➔ tableau statique (taille fixe)

```
BitArray b = new BitArray(2);
b.Set(0, true);
b.Set(1, false);
b.Set(2, true); ➔ Impossible car le tableau contient « 2 » éléments
```

Exemple File « Queue » ➔ FIFO

```
Queue q = new Queue();
   q.Enqueue("Bonjour ");
   q.Enqueue("Tout ");
   q.Enqueue(" le monde");
   q.Enqueue(" ca ");
   q.Enqueue(" va? ");

foreach (var o in q)
   Console.Write(o);
```

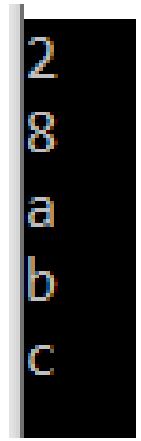
```
Bonjour Tout  le monde ca  va?
```

```
Queue q = new Queue();
   q.Enqueue("Bonjour ");
   q.Enqueue("Tout ");
   q.Enqueue(" le monde");
   q.Enqueue(" ca ");
   q.Enqueue(" va? ");
   q.Dequeue();
foreach (var o in q)
   Console.Write(o);
```

```
Tout  le monde ca  va?
```

Exemple « ArrayList »

```
ArrayList arrayL = new ArrayList();
    arrayL.Add(2);
    arrayL.Add("Ola");
    arrayL.Add('c');
    arrayL.Add(8);
    arrayL.Add('a');
    arrayL.Add('b');
    arrayL.Add('c');
    arrayL.Remove('c');//supprime la première occurrence de la lettre 'c'
    arrayL.RemoveAt(1);//supprime l'élément se trouvant à la position 1
foreach (var o in arrayL)
     Console.WriteLine(o);
```



- Tableau dynamique d'objets
- Taille dynamique
- Différents types possibles

arrayL.**Count** ➔ retourne la taille de la liste

Collections générique?

Collections génériques?

- Les collections :
  - sont des collections
  - permettent de regrouper et gérer plusieurs objets en précisant le **type** mais de taille dynamique

➔ Evite d'avoir des exceptions si le type attendu ne correspond pas au type d'un élément

➔ Evite d'effectuer des conversions

➔ Impose un type pour tous les éléments

Namespace « Collections Génériques »

- Inclure le namespace :

```
using System.Collections.Generic;
```

Les listes

Déclaration « List »

- Déclaration

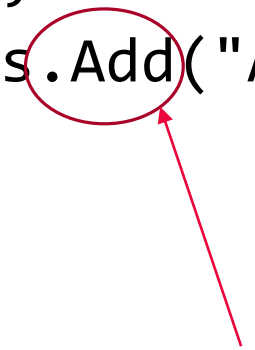
List<type> maListe **= new List<type>();**

- Exemples :
  - `List<string> listeMots = new List<string>();` ➔ “listeMots” est vide.
  - `List<string> listeMots = new List<string>{"Bonjour", "Ok", "Ko", "Salut"};` ➔ déclaration et initialisation

Ajout d'un élément dans « List »

- Ajouter un élément dans une liste « **Add** »

- Exemples :
  - `List<string> listeMots = new List<string>{"Bonjour", "Ok", "Ko", "Salut"};`
  - `listeMots.Add("Au revoir");`



Ajout à la fin de la liste

Accès à un élément dans « List »

- L'accès à un élément dans une liste se fait avec des [] (comme pour les tableaux)

  ➔ Lecture et écriture avec index

- Exemples :
  - `Console.WriteLine(listeMots[0]);` ➔ Bonjour
  - `listeMots[2]="Oui";` ➔ on écrase le "Ko"
  - `listeMots[5]="Non";` ➔ Impossible car il n'existe aucun élément à cette position

Parcours d'une « List »

- Boucle « For »

Taille de la List ➔ Nombre d'éléments

```
for (int i = 0; i < listeMots.Count; i++)
    Console.WriteLine(listeMots[i]);
```

- Boucle « Foreach »

```
foreach (var mot in listeMots){
    Console.Write(mot);
 }
```

- Expression Lambda

```
listeMots.ForEach(elt => Console.Write(elt));
```

Méthodes utiles sur les « List »

Méthodes	Définition
RemoveAt(n)	Supprime d'une liste l'élément d'indice « n ». Il existe aussi ***Remove*** (supprime la première occurrence) et ***RemoveAll***
IndexOf(n)	Retourne l'indice de la 1ère occurrence de la valeur « n » dans une liste
Contains(n)	Retourne true si « n » existe dans la liste
Find(elt => condition)	Retourne le 1er élément de la liste qui respecte condition (Exists fonction d'une manière similaire et retourne un booléen)
Sort()	Trie une liste d'entiers
ToArray()	Retourne un tableau statique résultat de la conversion de la liste

Tableaux / Listes ???

- Comment/Quoi choisir?
  - Un tableau est de taille fixe ➔ peut être multidimensionnel
  - Une liste est unidimensionnel ➔ Taille variable
  - Suppression impossible dans un tableau

Exemple

```
object[] obj = new object[2];
    obj[0] = "chaine";
    obj[1] = 2;
    foreach (object elt in obj){
       Console.WriteLine(elt);
    }
```

Tableau d'objets

Liste

```
List<object> list = new List<object>();
     list.Add("chaine");
     list.Add(2);
     foreach (object elt in list){
         Console.WriteLine(elt);
     }
```

Les dictionnaires

Qu’est ce qu’un dictionnaire?

- une collection manipulant deux éléments : une clé et une valeur
- une clé est un indice unique
- Déclaration

```
Dictionary<type1, type2> monDictionnaire = new Dictionary<type1,type2>();
```

- Exemple

Dictionary<int, string> dico = new Dictionary<int, string>();

Ajout d’un élément dans « dictionary »

- Ajouter un élément dans un dictionnaire « Add »

- Exemples :

```
Dictionary<int, string> dico = new Dictionary<int, string>();
    dico.Add(10, "Bon");
    dico.Add(23, "Ok");
    dico.Add(4, "Non");
    dico.Add(9, "Oui");
```

Existence dans un « dictionary »

- Vérifier l'existence d'une valeur et/ou d'une clé dans un dictionnaire ➔ « Contains »
- Exemple

```
Dictionary<int, string> dico = new Dictionary<int, string>();
        dico.Add(10, "Bon");
        dico.Add(23, "Ok");
        dico.Add(4, "Non");
        dico.Add(9, "Oui"); "
        Console.WriteLine(dico.ContainsValue("Ok"));
        Console.WriteLine(dico.ContainsKey(10));
```

Parcourir un « dictionary »

- Parcourir et afficher les valeurs

```
foreach (string elt in dico.Values)
        Console.WriteLine(elt);
```

- Parcourir et afficher les clés

```
Dictionary<int, string>.KeyCollection clefs = dico.Keys;
foreach (int elt in clefs)
    Console.WriteLine(elt);
```

- Parcourir et afficher les clés et les valeurs (clé, valeur)

```
foreach (KeyValuePair<int, string> elt in dico)
        Console.WriteLine(elt.Key + elt.Value);
```

Accès aux éléments d'un « dictionary »

- Récupérer la liste des valeurs

```
Dictionary<int, string>.ValueCollection vals = dico.Values;
        foreach (string elt in vals)
            Console.WriteLine(elt);
```

- Modifier la valeur d'un éléments en fonction de la clé

```
dico[10] = "Bof bof";
        foreach (int elt in clefs)
            Console.WriteLine(elt);
```

Les piles

Déclaration

- Stratégie LIFO
  - ➔ Impossible de supprimer/modifier/récupérer un élément au milieu
  - ➔ Retrait du dernier élément inséré dans la pile
- Déclaration

Stack<type> maPile **=** **new** **Stack<type>();**

- Exemple

```
Stack<string> pile1 = new Stack< string >();
Stack<int> pile2 = new Stack<int>();
```

Accès/Ajout élément

```
Stack<int> maPile = new Stack<int>();
```

- Ajouter un élément au sommet de pile ➔ **EMPILER**

```
maPile.Push(27);
maPile.Push(8);
maPile.Push(7);
maPile.Push(1);
```

- Parcourir une pile

```
foreach ( int element in  maPile) {
        Console.Write(element + "  ;  "); }
```

➔ 1 ; 7 ; 8 ; 27 ;

- Consulter l'élément de sommet de pile sans l'enlever (dernier élément ajouté à la pile)

```
maPile.Peek();
```

Accès à un élément

- Supprimer l'élément au sommet de pile ➔ **DEPILER**

```
maPile.Pop();
```

- Suppression de tous les éléments de la pile

```
maPile.Clear();
```

- Vérification existence d'un élément dans la pile

```
bool test = nomPile.Contains(type element);
```

```
bool test = maPile.Contains(8);
```

Exemple : Queue / Stack

```
Console.WriteLine("****  Queue ****\n\n");
Queue q = new Queue();
   q.Enqueue("Bleu ");
   q.Enqueue("Blanc ");
   q.Enqueue("Rouge ");
foreach (var o in q){
   Console.Write(o);}
```

```
Console.WriteLine("\n\n**** Stack ****\n \n");
Stack s = new Stack();
   s.Push("Bleu ");
   s.Push("Blanc ");
   s.Push("Rouge ");
foreach (var o in s){
   Console.Write(o);}
```

```
Console.WriteLine("\n\n --Retrait du premier élément de chacune des
collections\n\n");
Console.WriteLine("1er élement de la Queue : " + q.Dequeue());
Console.WriteLine("1er élement de la Stack : " + s.Pop());
```

Exemple : Queue / Stack

```
Queue q = new Queue();
   q.Enqueue("Bleu ");
   q.Enqueue("Blanc ");
   q.Enqueue("Rouge ");
Console.WriteLine("****   Queue **** \n \n");
foreach (var o in q)
   Console.Write(o);
Console.WriteLine("\n \n ****   Stack **** \n \n");
Stack s = new Stack();
   s.Push("Bleu ");
   s.Push("Blanc ");
   s.Push("Rouge ");
foreach (var o in s)
   Console.Write(o);

Console.WriteLine("\n \n \n \n---  Retrait du premier élément de
chacune des collections \n \n");
Console.WriteLine("1er élement de la Queue : " + q.Dequeue());
Console.WriteLine("1er élement de la Stack : " + s.Pop());
```

```
****   Queue ****


Bleu Blanc Rouge

 ****   Stack ****


Rouge Blanc Bleu



---  Retrait du premier élément de chacune des collections


1er élement de la Queue : Bleu
1er élement de la Stack : Rouge
```

Exercices

Exercice 1 : Notation polonaise inversée

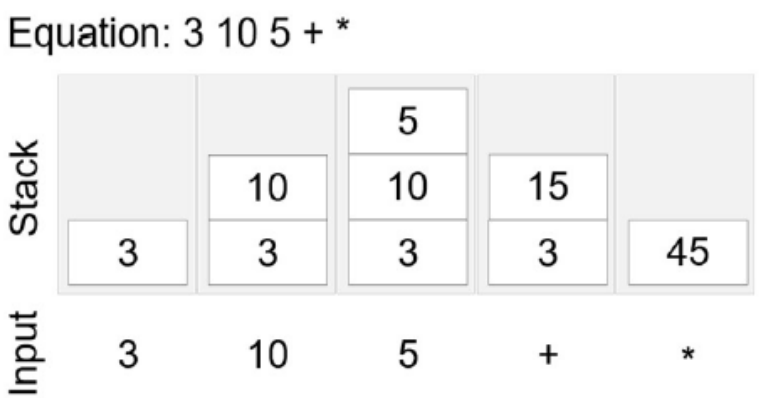
https://fr.wikipedia.org/wiki/Notation_polonaise_inverse

Ecrire une méthode qui permettrait de faire les opérations élémentaires (*,/,+,-) d'une expression arithmétique stockée dans une chaine de caractères en suivant la notation polonaise inversée.

Dans notre implémentation, nous simplifierons l'écriture en mettant l'opérateur après les 2 opérandes et ne considérons que 2 opérandes maximum

string line0 = "3,3,+,4,*,2,/" ; qui équivaut à ((3 +3) * 4)/2

Utiliser la collection adéquate.



Exercice 2 :

Ecrire une fonction Melange qui prend en arguments 2 piles et qui mélange leurs éléments dans une 3ème pile de la façon suivante : tant qu'une pile au moins n'est pas vide, on retire aléatoirement un élément au somment d'une des 2 piles et on l'empile sur la pile résultat.

Exemple un mélange possible des piles [1,2,3] et [5,4] est [3,2,4,1,5]

A la fin, les 2 piles doivent être vides.