**Les collections C#**

**Introduction**

L’objectif de ce chapitre est :

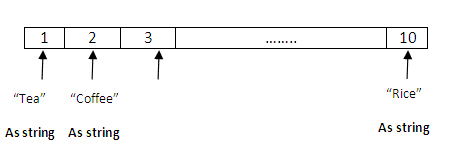
* Présenter différent types de collections
* Maitriser les différences entre une collection et une collection générique

**I - Array**

1. **Longeur fixe** -> la taille n’est pas flexible, déterminé lors de l’instanciation

2. **Strongly Typed** -> le développeur détermine le type lors de l’instanciation  

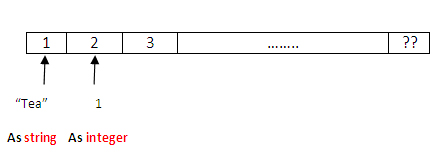
**Longueur fixe et fortement typé** -> Consomme moins de mémoire, par conséquent, l’Array a de bonnes performances.



**II - ArrayList**

1. Arraylist **PAS** de Longeur fixe -> c’est possible que les données augmentent.

2. Arraylist n’est **PAS** fortement typé



**Properties :**

|  |  |
| --- | --- |
| **Property** | **Description** |
| Capacity | Gets or sets the number of elements that the ArrayList can contain. |
| Count | Gets the number of elements actually contained in the ArrayList. |
| IsFixedSize | Gets a value indicating whether the ArrayList has a fixed size. |
| IsReadOnly | Gets a value indicating whether the ArrayList is read-only. |
| Item | Gets or sets the element at the specified index. |

Methods

|  |  |
| --- | --- |
| **Sr.No.** | **Methods** |
| 1 | **public virtual int Add(object value);**  Adds an object to the end of the ArrayList. |
| 2 | **public virtual void AddRange(ICollection c);**  Adds the elements of an ICollection to the end of the ArrayList. |
| 3 | **public virtual void Clear();**  Removes all elements from the ArrayList. |
| 4 | **public virtual bool Contains(object item);**  Determines whether an element is in the ArrayList. |
| 5 | **public virtual ArrayList GetRange(int index, int count);**  Returns an ArrayList which represents a subset of the elements in the source ArrayList. |
| 6 | **public virtual int IndexOf(object);**  Returns the zero-based index of the first occurrence of a value in the ArrayList or in a portion of it. |
| 7 | **public virtual void Insert(int index, object value);**  Inserts an element into the ArrayList at the specified index. |
| 8 | **public virtual void InsertRange(int index, ICollection c);**  Inserts the elements of a collection into the ArrayList at the specified index. |
| 9 | **public virtual void Remove(object obj);**  Removes the first occurrence of a specific object from the ArrayList. |
| 10 | **public virtual void RemoveAt(int index);**  Removes the element at the specified index of the ArrayList. |
| 11 | **public virtual void RemoveRange(int index, int count);**  Removes a range of elements from the ArrayList. |
| 12 | **public virtual void Reverse();**  Reverses the order of the elements in the ArrayList. |
| 13 | **public virtual void SetRange(int index, ICollection c);**  Copies the elements of a collection over a range of elements in the ArrayList. |
| 14 | **public virtual void Sort();**  Sorts the elements in the ArrayList. |
| 15 | **public virtual void TrimToSize();**  Sets the capacity to the actual number of elements in the ArrayList. |

**Exercice 1 :**

1. Créer un projet console
2. Créer ArrayList a1
3. Ajouter les éléments suivant dans a1 : 45 - 78 - 33 - 56 - 12 - 23 - 9
4. Afficher dans la console le nombre d’éléments et la capacité de a1
5. Afficher tous les éléments de a1
6. Trier a1
7. Afficher les éléments triés

**Exercice 2 :**

ArrayList array1 = new ArrayList();

array1.Add(1);

array1.Add("Pony");

int total = 0;

foreach (int num in array1)

{

total += num;

}

1. Exécuter ce code
   1. **Analyser l’erreur :Erreur de compilation ou de Runtime ? pourquoi ne pas utiliser une liste fortement typé ?**

**III - Liste générique List<T>**

Une liste générique List <T> est une liste dynamique fortement typée d'objets accessibles par un index.   
Elle fournit des méthodes de recherche, de tri et de manipulation de listes. La taille de la liste augmente dynamiquement selon les besoins.  
 **Déclaration de liste générique**  
          List<type> nomCollection = new List<type>();

|  |  |
| --- | --- |
| **Constructeur** | **Description** |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png List<>() | Initialise une nouvelle instance de la classe List<T> qui est vide et a la capacité initiale par défaut. |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png List<>(Int32) | Initialise une nouvelle instance de la classe List<T> qui est vide et a la capacité initiale spécifiée. |

|  |  |  |
| --- | --- | --- |
| **Propriété** | **Type** | **Description** |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1412335790755/guicontroles/timer/Propri%C3%A9t%C3%A9.png Capacity | int | Obtient ou définit le nombre total des éléments que la structure de données interne peut contenir sans redimensionnement. |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1412335790755/guicontroles/timer/Propri%C3%A9t%C3%A9.png Count | int | Obtient le nombre total des éléments réellement contenus dans List<T>. |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1412335790755/guicontroles/timer/Propri%C3%A9t%C3%A9.png Item |  | Obtient ou définit l'élément situé à l'index spécifié.         int premierNombre = colNombres[0]; |

Voici quelques méthodes. Consulter la [liste complète](http://msdn.microsoft.com/fr-fr/library/6sh2ey19%28v=VS.95%29.aspx) sur MSDN.

|  |  |
| --- | --- |
| **Méthode** | **Description** |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png Add | Ajoute un objet à la fin de la liste.     colNombres.**Add**(500); |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png Clear | Supprime tous les éléments de la liste.          colNombres.**Clear**(); |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png Contains | Détermine si un élément est dans la liste.            bool estPrésent = colNombres.**Contains**(500); |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png IndexOf | Recherche l'élément spécifié et retourne l'index base 0 de la 1ère occurrence de l'élément.           int indexPremier500 = colNombres.**IndexOf**(500); |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png Insert | Insère l'élément spécifié dans la liste à l'index spécifié.           colNombres.**Insert**(3, 500); |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png LastIndexOf | Recherche l'élément spécifié et retourne l'index base 0 de la dernière occurrence de l'élément.           int indexDernier500 = colNombres.**LastIndexOf**(500); |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png Remove | Supprime la première occurence d'un objet spécifique de la liste.           colNombres.**Remove**(500); |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png RemoveAt | Supprime l'élément à l'index spécifié.           colNombres.**RemoveAt**(5); |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png Sort | Trie les éléments.           colNombres.**Sort**(); |
| https://sites.google.com/site/notionscsharpcem/_/rsrc/1401026160445/instanciation_objet/M%C3%A9thode.png CopyTo | Copie les éléments de la liste vers un tableau.          int[] tabNombres = new int[m\_colNombres.Count];          colNombres.**CopyTo**(tabNombres); |

**Exercice 3**

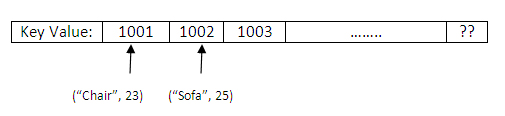
****

1. Créer les classes étudiant et Classe
2. Dans la classe
   1. Ajouter une liste génétique d’étudiant « Etudiants »
3. Dans la classe « Classe » Créer les méthodes suivantes en utilisant les méthodes fournies par la classe List
   1. Ajouter un étudiant
   2. Afficher tous les étudiants
   3. Supprimer un étudiant
   4. Récupérer un étudiant
   5. Récupérer l’index d’un étudiant dans la liste

## IV - HashTable

HashTable est un autre type de structure qui définit un couple clé valeur pour chaque section de donnée. La recherche d’une donnée est facile, elle s’effectue par le pointage sur la clé.

**NB :HashTable n’est pas fortement type et n’as pas de taille fixe**



*//It is NOT obvious that strArrayList is*

*//1. string? int? object? decimal? OR OBJECT?? --> NOT Strongly Type*

*//2. Sized=10? 20? 100? -->NOT Fixed Size*

*// Namespace: System.Collections*

*//Hashtable solve the problem in Arraylist when we are looking for specific item*

*//Hashtable dedicate a key for each item, then finding item is easier and faster*

System.Collections.Hashtable objHashTable = new System.Collections.Hashtable();

objHashTable.Add("1001","Chair");

objHashTable.Add("1002", "Sofa");

objHashTable.Add("1003", "Carpet");

**V - Dictionary**

Dictionary est GENERIC alors que HashTable ne l’est pas : Dictionary<TKey, TValue>, Il ya une petite différence entre les deux : Quand un dictionnaire n’arrive pas à trouver une clé spécifique il lève une exception, alors que HashTable retourne null

*//Dictionary can instantiate from Dictionary , Dictionary is similar to Hashtable,*

*//Dictionary is GENERIC but Hashtable is NON GENERIC*

*//Such Hashtable you can find object by its key*

System.Collections.Generic.Dictionary<int, string=""> objDictionary = new Dictionary<int, string="">();

objDictionary.Add(1001, "Mahsa");

objDictionary.Add(1002, "Hassankashi");

objDictionary.Add(1003, "Cosmicverse");

string str = objDictionary[1002];