

# Collections



# Objectifs

- Passer en revue les principales collections : listes, ensembles, dictionnaires
- Les déclarer en C#

# Collections

- But : stocker un ensemble d'objet du même type afin de pouvoir les manipuler
- Les manipulations standards sont :
  - Énumération des éléments de la collection
  - Ajout d'éléments
  - Suppression d'éléments
  - Accès à un élément précis
- Les collections manipulent des types génériques, c'est-à-dire qu'elles peuvent contenir n'importe quel type de données
- Le choix d'un type de collection dépend des algorithmes que nous avons besoin d'appliquer sur les collections en regard des complexités qui seront vues en cours d'algorithmes avancés.

# Listes

- « *List* » est un type qui utilise les génériques : « `List<<UnType>>` »
- Les listes ont une capacité variable. La rapidité des méthodes dépendant de la structure de données utilisées pour les stocker (voir cours d'algorithmes avancés)
- Propriétés « `Count` » :
  - Nombre d'éléments dans la liste
- Opérateur `[]` pour accéder aux différents éléments de la liste à partir d'un indice
- Méthodes :
  - `Add()` : Ajouter un élément à la fin
  - `Insert()` : Insérer à un indice précis.
  - `Remove ()` : Supprimer un élément.
  - `RemoveAt()` : Supprimer un indice précis.
  - `Clear()` : Effacer tous les éléments.
  - `IndexOf()` : Recherche.
  - `Sort()` : Trier.
- Pour les curieux, le code source :
  - <https://github.com/microsoft/referencesource/blob/master/mscorlib/system/collections/generic/list.cs>

# Exemple

```
static void Main(string[] args)
{
    List<double> listeDoubles = new List<double>(20);
    Console.Out.WriteLine(listeDoubles.Count);
    Console.Out.WriteLine(listeDoubles.Capacity);

    listeDoubles.Add(1);
    listeDoubles.Add(1);

    foreach (double valeur in listeDoubles)
    {
        Console.Out.WriteLine(valeur);
    }
    Console.Out.WriteLine();

    listeDoubles.Insert(1, 42);
    listeDoubles.Insert(listeDoubles.Count, 99);
    foreach (double valeur in listeDoubles)
    {
        Console.Out.WriteLine(valeur);
    }
    Console.Out.WriteLine();
}
```

# Exemple avec des entiers

```
int index42 = listeDoubles.IndexOf(42);
if (index42 >= 0)
{
    Console.Out.WriteLine("42 trouvé !");
}
else
{
    Console.Out.WriteLine("42 non trouvé !");
}
```

# Classe AdulTE

```
public class AdulTE
{
    public int Age { get; set; }
    public string Nom { get; set; }

    public AdulTE() : this("", 0)
    {
        ;
    }

    public AdulTE(string p_nom, int p_age)
    {
        this.Nom = p_nom;
        this.Age = p_age;
    }

    public override string ToString()
    {
        return $"AdulTE(Nom : {this.Nom}, Age : {this.Age})";
    }
}
```

# Exemple avec des adultes

```
static void Main(string[] args)
{
    List<Adulte> listeAdultes = new List<Adulte>();

    Adalte mohammed = new Adalte("Mohammed", 23);
    listeAdultes.Add(mohammed);
    listeAdultes.Add(new Adalte("Thierry", 23));
    listeAdultes.Add(new Adalte("Lysane", 23));

    foreach (var adulte in listeAdultes)
    {
        Console.Out.WriteLine(adulte);
    }
}
```

```
Adalte(Nom : Mohammed, Age : 23)
Adalte(Nom : Amrane, Age : 23)
Adalte(Nom : Lysane, Age : 23)
Press any key to continue . . .
```

# Exemple avec des objets

```
static void Main(string[] args)
{
    List<Adulte> listeAdultes = new List<Adulte>();

    Adulte mohammed = new Adulte("Mohammed", 23);
    listeAdultes.Add(mohammed);

    // ...

    int indiceMohammed = listeAdultes.IndexOf(mohammed);
    if (indiceMohammed >= 0)
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }

    Adulte copieMohammed = new Adulte(mohammed);
    int indiceCopieMohammed = listeAdultes.IndexOf(copieMohammed);
    if (indiceCopieMohammed >= 0)
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }
}
```

Adulte trouvé !  
Adulte non trouvé !  
Press any key to continue . . .

# Ajout de la méthode equals

```
public class Adulte
{
    // ...

    public override bool Equals(object p_obj)
    {
        bool egaux = false;
        Adulte conversionAdulte = p_obj as Adulte;
        if (conversionAdulte != null)
        {
            egaux = this.Nom.CompareTo(conversionAdulte.Nom) == 0
                    && this.Age == conversionAdulte.Age;
        }
        return egaux;
    }
}
```

# Exemple avec des objets avec méthode equals

```
static void Main(string[] args)
{
    List<Adulte> listeAdultes = new List<Adulte>();

    Adulte mohammed = new Adulte("Mohammed", 23);
    listeAdultes.Add(mohammed);

    // ...

    int indiceMohammed = listeAdultes.IndexOf(mohammed);
    if (indiceMohammed >= 0)
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }

    Adulte copieMohammed = new Adulte(mohammed);
    int indiceCopieMohammed = listeAdultes.IndexOf(copieMohammed);
    if (indiceCopieMohammed >= 0)
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }
}
```

Adulte trouvé !  
Adulte trouvé !  
Press any key to continue . . .

# Ensemble

- « *HashSet* » est un type qui utilise les génériques : « `HashSet<<UnType>>` »
- Optimisé pour la recherche et particulièrement pour déterminer si une valeur appartient à l'ensemble ou non
- Propriétés « `Count` » :
  - Nombre d'éléments dans le tableau
- Méthodes :
  - `Add()` : Ajouter un élément à la fin
  - `Remove ()` : Supprimer un élément.
  - `Contains()` : Vérifier si un élément appartient à l'ensemble
- Pour les curieux, le code source :
  - <https://github.com/microsoft/referencesource/blob/master/System.Core/System/Collections/Generic/HashSet.cs>

# Exemple avec des objets

```
public class Adulte
{
    public int Age { get; set; }
    public string Nom { get; set; }

    public Adulte() : this("", 0) { ; }

    public Adulte(string p_nom, int p_age)
    {
        this.Nom = p_nom;
        this.Age = p_age;
    }

    public override string ToString()
    {
        return $"Adulte(Nom : {this.Nom}, Age : {this.Age})";
    }
}
```

# Exemple avec des adultes

```
public static void Main(string[] args)
{
    HashSet<Adulte> ensembleAdultes = new HashSet<Adulte>();
    Adulte mohammed = new Adulte("Mohammed", 23);
    ensembleAdultes.Add(mohammed);

    // ...

    if (ensembleAdultes.Contains(mohammed))
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }

    Adulte copieMohammed = new Adulte(mohammed.Nom, mohammed.Age);
    if (ensembleAdultes.Contains(copieMohammed))
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }
}
```

Adulte trouvé !  
Adulte non trouvé !  
Press any key to continue . . .

# Ajout des méthodes GetHashCode et Equals

```
public class Adulte
{
    // ...
    public override int GetHashCode()
    {
        return HashCode.Combine(this.Age, this.Nom);
    }

    public override bool Equals(object p_obj)
    {
        bool egaux = false;
        Adulte conversionAdulte = p_obj as Adulte;
        if (conversionAdulte != null)
        {
            egaux = this.Nom.CompareTo(conversionAdulte.Nom) == 0
                    && this.Age == conversionAdulte.Age;
        }
        return egaux;
    }
}
```

# Exemple avec des adultes

```
public static void Main(string[] args)
{
    HashSet<Adulte> ensembleAdultes = new HashSet<Adulte>();
    Adulte mohammed = new Adulte("Mohammed", 23);
    ensembleAdultes.Add(mohammed);

    // ...

    if (ensembleAdultes.Contains(mohammed))
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }

    Adulte copieMohammed = new Adulte(mohammed.Nom, mohammed.Age);
    if (ensembleAdultes.Contains(copieMohammed))
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }
}
```

Adulte trouvé !  
Adulte trouvé !  
Press any key to continue . . .

# Dictionnaire

- « *Dictionary* » est un type qui utilise les génériques : « `Dictionary<<TypeClef>, <TypeValeur>>` »
- Un dictionnaire associe une clef à une valeur. La clef et la valeur peuvent être de types différents
- Optimisé pour la recherche à partir d'une clef
- Opérateur `[]` pour accéder aux différents éléments d'un dictionnaire à partir de la clef
- Propriétés « `Count` » :
  - Nombre d'éléments dans le tableau
- Méthodes :
  - `Add()` : Ajouter un élément à la fin
  - `Remove ()` : Supprimer un élément.
  - `ContainsKey()` : Vérifier si un élément appartient à l'ensemble
  - `ContainsValue()` : Vérifier si un élément appartient à l'ensemble
- Pour les curieux, le code source :
  - <https://github.com/microsoft/referencesource/blob/master/System.Core/System/Collections/Generic/HashTable.cs>

# Exemple avec des objets

```
public class Adulte
{
    public int Id { get; set; }
    public int Age { get; set; }
    public string Nom { get; set; }

    public Adulte(int p_id, string p_nom, int p_age)
    {
        this.Id = p_id;
        this.Nom = p_nom;
        this.Age = p_age;
    }

    public override string ToString()
    {
        return $"Adulte(Id : {this.Id}, Nom : {this.Nom}, Age : {this.Age})";
    }
}
```

# Exemple avec des adultes

```
public static void Main(string[] args)
{
    Adulte mohammed = new Adulte(1, "Mohammed", 23);
    Adulte lysane = new Adulte(2, "Lysane", 23);
    Adulte thierry = new Adulte(3, "Thierry", 23);

    Dictionary<int, Adulte> dictionnaireAdultes = new Dictionary<int, Adulte>();
    dictionnaireAdultes.Add(mohammed.Id, mohammed);
    dictionnaireAdultes[lysane.Id] = lysane;

    if (dictionnaireAdultes.ContainsKey(1))
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }

    Adulte copieMohammed = new Adulte(1, mohammed.Nom, mohammed.Age);
    if (dictionnaireAdultes.ContainsValue(copieMohammed))
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }
}
```

Adulte trouvé !  
Adulte non trouvé !  
Press any key to continue . . .

# Ajout de la méthode equals

```
public class Adulte
{
    // ...

    public override bool Equals(object p_obj)
    {
        bool egaux = false;
        Adulte conversionAdulte = p_obj as Adulte;
        if (conversionAdulte != null)
        {
            egaux = this.Id == conversionAdulte.Id
                    && this.Nom.CompareTo(conversionAdulte.Nom) == 0
                    && this.Age == conversionAdulte.Age;
        }
        return egaux;
    }
}
```

# Exemple avec des adultes

```
public static void Main(string[] args)
{
    Adulte mohammed = new Adulte(1, "Mohammed", 23);
    Adulte lysane = new Adulte(2, "Lysane", 23);
    Adulte thierry = new Adulte(3, "Thierry", 23);

    Dictionary<int, Adulte> dictionnaireAdultes = new Dictionary<int, Adulte>();
    dictionnaireAdultes.Add(mohammed.Id, mohammed);
    dictionnaireAdultes[lysane.Id] = lysane;

    if (dictionnaireAdultes.ContainsKey(1))
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }

    Adulte copieMohammed = new Adulte(1, mohammed.Nom, mohammed.Age);
    if (dictionnaireAdultes.ContainsValue(copieMohammed))
    {
        Console.Out.WriteLine("Adulte trouvé !");
    }
    else
    {
        Console.Out.WriteLine("Adulte non trouvé !");
    }
}
```

Adulte trouvé !  
Adulte trouvé !  
Press any key to continue . . .