

# Chapitres

---

- Révision
- Concept de classe
- Collections
- Héritage/polymorphisme
- Classe abstraite et interface
- La délégation
- Suivi de projet

# Les collection en C#

---

# Exercice

- Créer une classe « Compte » qui contient comme attributs: numeroCompte, nomClient, prénomClient, adresseClient et Solde. Ajouter à la classe un constructeur qui permet d'instancier tous les attributs. Définir les méthodes permettant l'augmentation et la diminution du solde ainsi que la description du solde (méthode ToString())
- Dans le programme principal « main »
  - Définissez un tableau qui va contenir 3 objets compte.
  - Créer 3 objets « compte »
  - Stocker les objets « compte » dans le tableau.
  - Créer un quatrième objet « compte »
  - Ajouter l'objet « compte » dans le tableau.

# Comment faire ?

- Déclarer un tableau plus grand (de taille 4).
- Recopier les 3 objets « compte » du tableau initial
- Insérer le quatrième objet en dernière position

# Comment faire ?

```
class Compte
{
    //attributs
    private int numeroCompte;
    private string nomClient;
    private string prenomClient;
    private string adresseClient;
    private double solde;
    //Constructeur
    public Compte(int numeroCompte, string nomClient, string prenomClient,
        string adresseClient, double solde)
    {
        this.numeroCompte= numeroCompte; this.nomClient= nomClient;
        this.prenomClient= prenomClient; this.adresseClient= adresseClient;
        this.solde= solde;
    }
    //Méthodes
    public void AgmenterSolde(double montant)
    {
        solde += montant;
    }
    public void DiminuerSolde(double montant)
    {
        solde -= montant;
    }
    public string toString()
    {
        return "Numéro compte: " + numeroCompte + "\nNom client: " + nomClient + "\t Prénom client: " + prenomClient + "\n";
    }
}
```

# Comment faire ?

```
static void Main(string[] args)
{
    //Création des 3 objets
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    Compte c2 = new Compte(2,"Michelin","Jean","94100Courbevoie",120);
    Compte c3 = new Compte(3,"Combarel","Bastien","10000Troyes",1620);
    //Création d'un tableau de taille 3 et insertion des objets
    Compte[] tabCompte = new Compte[3];
    tabCompte[0] = c1; tabCompte[1] = c2; tabCompte[2] = c3;
    //Ajout d'un quatrième objet
    Compte c4 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);
    //Insertion du quatrième objet
    Compte[] tabTemp = new Compte[tabCompte.Length+1];
    int i = 0;
    for(i=0;i<tabCompte.Length;i++)
    {
        tabTemp[i] = tabCompte[i];
    }
    tabTemp[i] = c4;
    tabCompte = tabTemp;
    string str = "";
    for (i = 0; i < tabCompte.Length; i++)
    {
        str+= tabCompte[i].ToString();
    }
    Console.WriteLine(str);
    Console.ReadKey();
}
```

E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours.exe

```
Numéro compte: 1
Nom client: Dupont      Prénom client: Maxime
Numéro compte: 2
Nom client: Michelin    Prénom client: Jean
Numéro compte: 3
Nom client: Combarel    Prénom client: Bastien
Numéro compte: 4
Nom client: Renault     Prénom client: Isac
```

# Limites des tableaux

- Dimensions statiques
- Manque de souplesse pour stocker des d'objets de différents types.
- Une organisation des données très limitée (Pas de stockage organisé, etc.).
- Pas de fonctionnalités de gestion de données (tri, ajout, suppression, etc.)

# Les collections d'objets en C#

- Une collection est un outil permettant le stockage et la gestion d'objets homogènes (de même type) ou hétérogènes.
- La différence majeure entre un tableau et une collection est que la taille d'un tableau est fixée alors que celle d'une collection peut varier (on peut ajouter et enlever des éléments).
- Les collections offrent plus de souplesse quand il s'agit de stocker et gérer des groupes d'objets (objets de différents types).
- Une collection est une classe. Pour pouvoir ajouter des éléments à la collection, il faut déclarer une instance.



# Les collections d'objets en C#

- Pour utiliser les collection, il faut ajouter la ligne de code :  
**using System.Collections ;**
- Parmi les collections en C#:
  - ArrayList ()
  - Stack () - (pile, ou tas)
  - Queue () - (file d'attente)
  - List()
  - et autres ..... SortedList() .....

# Les collections d'objets en C#

```
class Employee
{
    //attributs
    private int matricule;
    private string nomEmploye;
    private string prenomEmploye;
    private string adresseEmploye;
    //Constructeur
    public Employee(int matricule, string nomEmploye, string prenomEmploye,
        string adresseEmploye)
    {
        this.nomEmploye = nomEmploye;
        this.prenomEmploye = prenomEmploye; this.adresseEmploye = adresseEmploye;
        this.matricule= matricule;
    }
    public string toString()
    {
        return "Matricule employé: " + matricule + "\nNom employé: " + nomEmploye + "\t Prénom employé: " + prenomEmploye + "\n";
    }
}
```

# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

- un ArrayList est un tableau dynamique **d'objets**
- Un ArrayList grandit automatiquement selon les besoins pendant l'exécution
- Un ArrayList peut contenir n'importe quel type d'objet
- Un ArrayList est homogène si tous ses objets sont de même type. Il devient hétérogène si ses objets sont type différents.

## □ Déclaration

```
//Déclaration d'un ArrayList  
ArrayList arrayList = new ArrayList();
```

# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

### □ Déclaration

```
//Déclaration d'un ArrayList
ArrayList arrayList = new ArrayList();
```

### □ Utilisation

#### ➤ Insertion d'élément à la fin du ArrayList ( méthode Add)

```
static void Main(string[] args)
{
    //Création de 1 objet Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    //Création d'un objet Employé
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");
    //Déclaration d'un ArrayList
    ArrayList arrayList = new ArrayList();
    arrayList.Add(c1);
    arrayList.Add(e1);

    Console.ReadKey();
}
```

# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

### □ Déclaration

```
//Déclaration d'un ArrayList
```

### □ Utilisation

```
ArrayList arrayList = new ArrayList();
```

- Accès à un élément (comme les tableau [index ] mais il faut utiliser le caste)

```
static void Main(string[] args)
{
    //Création de 1 objet Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    //Création d'un objet Employé
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");
    //Déclaration d'un ArrayList
    ArrayList arrayList = new ArrayList();
    //Insertion d'objets hétérogène
    arrayList.Add(c1);
    arrayList.Add(e1);
    //Accès à un objet
    Console.WriteLine(((Compte)arrayList[0]).ToString());
    Console.WriteLine(((Employe)arrayList[1]).ToString());
    Console.ReadKey();
}
```

E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours.exe

```
Numéro compte: 1
Nom client: Dupont      Prénom client: Maxime

Matricule employé: 1
Nom employé: Michelin   Prénom employé: Guillaume
```

# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

### □ Déclaration

```
//Déclaration d'un ArrayList
```

### □ Utilisation

```
ArrayList arrayList = new ArrayList();
```

#### ➤ Insertion d'un élément à une position (méthode Insert)

```
static void Main(string[] args)
{
    //Création de 1 objet Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    //Création d'un objet Employé
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");
    //Déclaration d'un ArrayList
    ArrayList arrayList = new ArrayList();
    //Insertion d'objet à une position
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);
    arrayList.Insert(0,c2);
    Console.WriteLine(((Compte)arrayList[0]).ToString());
    Console.ReadKey();
}
```

E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours.exe

Numéro compte: 4  
Nom client: Renault      Prénom client: Isac

# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

### □ Déclaration

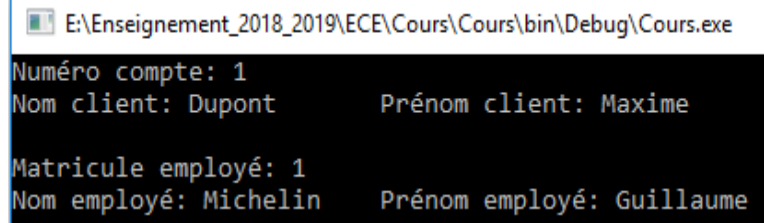
```
//Déclaration d'un ArrayList
```

### □ Utilisation

```
ArrayList arrayList = new ArrayList();
```

#### ➤ Suppression de la première occurrence d'un objet (Remove)

```
static void Main(string[] args)
{
    //Création de 2 objets Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);
    //Création d'un objet Employé
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");
    //Déclaration d'un ArrayList
    ArrayList arrayList = new ArrayList();
    //Insertion des objets
    arrayList.Add(c1);arrayList.Add(c2);arrayList.Add(e1);
    //Suppression de la première occurrence de l'objet c2
    arrayList.Remove(c2);
    //Affichage du contenu de la liste
    Console.WriteLine(((Compte)arrayList[0]).ToString());
    Console.WriteLine(((Employe)arrayList[1]).ToString());
    Console.ReadKey();
}
```



E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours.exe

Numéro compte: 1	
Nom client: Dupont	Prénom client: Maxime
Matricule employé: 1	
Nom employé: Michelin	Prénom employé: Guillaume

# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

### □ Déclaration

```
//Déclaration d'un ArrayList
```

### □ Utilisation

```
ArrayList arrayList = new ArrayList();
```

#### ➤ Suppression d'un objet à une position (RemoveAt())

```
static void Main(string[] args)
{
    //Création de 2 objets Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);
    //Création d'un objet Employé
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");
    //Déclaration d'un ArrayList
    ArrayList arrayList = new ArrayList();
    //Insertion des objets
    arrayList.Add(c1);arrayList.Add(c2);arrayList.Add(e1);
    //Suppression de l'objet en position 2
    arrayList.RemoveAt(2);
    //Affichage du contenu de la liste
    Console.WriteLine(((Compte)arrayList[0]).ToString());
    Console.WriteLine(((Compte)arrayList[1]).ToString());
    Console.ReadKey();
}
```

E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours.exe

```
Numéro compte: 1
Nom client: Dupont      Prénom client: Maxime

Numéro compte: 4
Nom client: Renault     Prénom client: Isac
```



# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

### □ Déclaration

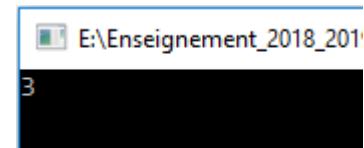
```
//Déclaration d'un ArrayList
```

### □ Utilisation

```
ArrayList arrayList = new ArrayList();
```

#### ➤ Déterminer la taille de la liste (la propriété: Count)

```
static void Main(string[] args)
{
    //Création de 2 objets Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);
    //Création d'un objet Employé
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");
    //Déclaration d'un ArrayList
    ArrayList arrayList = new ArrayList();
    //Insertion des objets
    arrayList.Add(c1);arrayList.Add(c2);arrayList.Add(e1);
    //Affichage de la taille de la liste
    Console.WriteLine(arrayList.Count);
    Console.ReadKey();
}
```



# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

### □ Déclaration

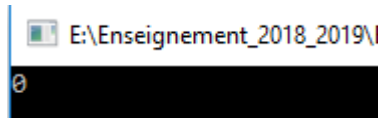
```
//Déclaration d'un ArrayList
```

### □ Utilisation

```
ArrayList arrayList = new ArrayList();
```

#### ➤ Supprimer tous les objets de la liste (Clear())

```
static void Main(string[] args)
{
    //Création de 2 objets Compte
    Compte c1 = new Compte(1, "Dupont", "Maxime", "75000Paris", 1500);
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);
    //Création d'un objet Employé
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");
    //Déclaration d'un ArrayList
    ArrayList arrayList = new ArrayList();
    //Insertion des objets
    arrayList.Add(c1);arrayList.Add(c2);arrayList.Add(e1);
    //Supprimer tous les objets
    arrayList.Clear();
    //Affichage de la taille de la liste
    Console.WriteLine(arrayList.Count);
    Console.ReadKey();
}
```



# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

### □ Déclaration

```
//Déclaration d'un ArrayList
```

### □ Utilisation

```
ArrayList arrayList = new ArrayList();
```

➤ et d'autres

# Les collections d'objets en C#

## ■ ArrayList(): (tableau unidimensionnel dynamique)

- Un ArrayList peut contenir des objets de différents types
- Quand on extrait un objet d'un ArrayList, le programme ne connaît pas le type de l'objet extrait et ne peut donc pas dimensionner d'espace mémoire pour l'objet lu.
- **il faut typer l'objet lu par un cast.**
- Ce principe se répète pour toutes les collections (queue, piles, etc....)

# Les collections d'objets en C#

## ■ Les piles (ou stack) LIFO

- Les piles sont des collections permettant le stockage d'objets de différents types.
- on retire les objets de la pile suivant en mode LIFO (Last in - First out)
- quand on retire les objets de la pile, il faudra les "caster" ( (int) pile.Pop() )

## □ Déclaration

```
//Déclaration d'un stack  
Stack tas = new Stack();
```

# Les collections d'objets en C#

## ■ Les piles (ou stack) LIFO

### □ Déclaration

```
//Déclaration d'un stack  
Stack tas = new Stack();
```

### □ Utilisation

#### ➤ Empiler des objets (push())

```
static void Main(string[] args)  
{  
    //Création de 2 objets Compte  
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);  
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);  
    //Création d'un objet Employé  
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");  
    //Déclaration d'un stack  
    Stack tas = new Stack();  
    //Empiler des objets  
    tas.Push(c1); tas.Push(c2); tas.Push(e1);  
}
```

# Les collections d'objets en C#

## ■ Les piles (ou stack) LIFO

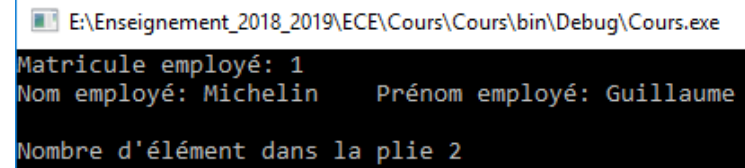
### □ Déclaration

```
//Déclaration d'un stack  
Stack tas = new Stack();
```

### □ Utilisation

#### ➤ Supprime et renvoie le dernier objet stocké dans la pile (Pop())

```
static void Main(string[] args)  
{  
    //Création de 2 objets Compte  
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);  
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);  
    //Création d'un objet Employé  
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");  
    //Déclaration d'un stack  
    Stack tas = new Stack();  
    //Empiler des objets  
    tas.Push(c1); tas.Push(c2); tas.Push(e1);  
    //Supprimer et renvoyer le dernier objet stocker dans la pile  
    Console.WriteLine(((Employe)tas.Pop()).ToString());  
    Console.WriteLine("Nombre d'élément dans la plie " + tas.Count);  
    Console.ReadKey();  
}
```



```
E:\Enseignement_2018_2019\ECE\Cours\Cours\bin\Debug\Cours.exe  
Matricule employé: 1  
Nom employé: Michelin    Prénom employé: Guillaume  
Nombre d'élément dans la plie 2
```

# Les collections d'objets en C#

## ■ Les piles (ou stack) LIFO

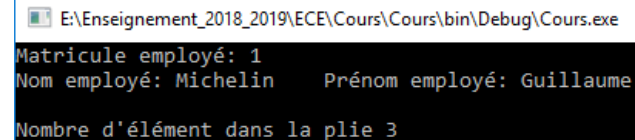
### □ Déclaration

```
//Déclaration d'un stack  
Stack tas = new Stack();
```

### □ Utilisation

#### ➤ Renvoyer le dernier objet stocké dans la pile sans le supprimer (Peek())

```
static void Main(string[] args)  
{  
    //Création de 2 objets Compte  
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);  
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);  
    //Création d'un objet Employé  
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");  
    //Déclaration d'un stack  
    Stack tas = new Stack();  
    //Empiler des objets  
    tas.Push(c1); tas.Push(c2); tas.Push(e1);  
    //renvoyer le dernier objet stocker dans la pile sans le supprimer  
    Console.WriteLine(((Employe)tas.Peek()).ToString());  
    Console.WriteLine("Nombre d'élément dans la plie " + tas.Count);  
    Console.ReadKey();  
}
```



E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours.exe  
Matricule employé: 1  
Nom employé: Michelin Prénom employé: Guillaume  
Nombre d'élément dans la plie 3



# Les collections d'objets en C#

## ■ Les files (ou Queue) FIFO

- Les queues sont des collections permettant le stockage d'objets de différents types.
- On retire les objets de la queue suivant en mode FIFO (First in -First out)
- Quand on retire les objets de la file, il faudra les "caster" ( (int) pile.Pop() )

## □ Déclaration

```
//Déclaration d'une queue  
Queue file = new Queue();
```

# Les collections d'objets en C#

## ■ Les piles (ou stack) LIFO

### □ Déclaration

```
//Déclaration d'une queue  
Queue file = new Queue();
```

### □ Utilisation

#### ➤ Ajouter un objet à la fin de la queue (Enqueue)

```
static void Main(string[] args)  
{  
    //Création de 2 objets Compte  
    Compte c1 = new Compte(1, "Dupont", "Maxime", "75000Paris", 1500);  
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);  
    //Création d'un objet Employé  
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");  
    //Déclaration d'une queue  
    Queue file = new Queue();  
    //Stocker des objets à la fin de la queue  
    file.Enqueue(c1); file.Enqueue(c2); file.Enqueue(e1);  
    Console.ReadKey();  
}
```

# Les collections d'objets en C#

## ■ Les piles (ou stack) LIFO

### □ Déclaration

```
//Déclaration d'un stack  
Stack tas = new Stack();
```

### □ Utilisation

## ➤ Supprime et renvoie le premier objet stocker dans la file (Dequeue)

```
static void Main(string[] args)  
{  
    //Création de 2 objets Compte  
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);  
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);  
    //Création d'un objet Employé  
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");  
    //Déclaration d'une queue  
    Queue file = new Queue();  
    //Stocker des objets à la fin de la queue  
    file.Enqueue(c1); file.Enqueue(c2); file.Enqueue(e1);  
    //Supprimer et Renvoyer le premier objet stocké  
    Console.WriteLine(((Compte)file.Dequeue()).toString());  
    Console.WriteLine("Nombre d'élément dans la file " + file.Count);  
    Console.ReadKey();  
}
```

E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours.exe

```
Numéro compte: 1  
Nom client: Dupont      Prénom client: Maxime  
Nombre d'élément dans la file 2
```

# Les collections d'objets en C#

## ■ Les piles (ou stack) LIFO

### □ Déclaration

```
//Déclaration d'un stack  
Stack tas = new Stack();
```

### □ Utilisation

#### ➤ Renvoyer le premier objet stocké dans la file sans le supprimer (Peek())

```
static void Main(string[] args)  
{  
    //Création de 2 objets Compte  
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);  
    Compte c2 = new Compte(4, "Renault", "Isac", "10450Breviandes", 1130);  
    //Création d'un objet Employé  
    Employe e1 = new Employe(1, "Michelin", "Guillaume", "10800Buchère");  
    //Déclaration d'une queue  
    Queue file = new Queue();  
    //Stocker des objets à la fin de la queue  
    file.Enqueue(c1); file.Enqueue(c2); file.Enqueue(e1);  
    //Renvoyer le premier objet stocké sans le supprimer  
    Console.WriteLine(((Compte)file.Peek()).ToString());  
    Console.WriteLine("Nombre d'élément dans la file " + file.Count);  
    Console.ReadKey();  
}
```

E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours.exe

```
Numéro compte: 1  
Nom client: Dupont      Prénom client: Maxime  
Nombre d'élément dans la file 3
```

# Les collections génériques en C#

- Les collections génériques sont des outils permettant le stockage et la gestion d'objets homogènes (de même type).
- Elles sont très efficaces dans un environnement typé (C#).
- Elles sont plus modernes que les collections d'objets.
- Elles se différencient des collections d'objets en ce que le type des éléments composants la collection est précisé lors de la création de la collection.
- il n'est plus nécessaire de caster les lectures car le type des éléments composants la collection est précisé lors de la création de la collection.
- leur emploi est similaire à celui des collections d'objets correspondantes.

# Les collections génériques en C#

## ■ List<T>: (tableau unidimensionnel dynamique)

### □ Déclaration

```
//Déclaration d'une liste de comptes  
List<Compte> listComptes = new List<Compte>();
```

### □ Utilisation

#### ➤ Ajouter un objet (Add)

```
static void Main(string[] args)  
{  
    //Création de 3 objets Compte  
    Compte c1 = new Compte(1, "Dupont", "Maxime", "75000Paris", 1500);  
    Compte c2 = new Compte(2, "Renault", "Isac", "10450Breviandes", 1130);  
    Compte c3 = new Compte(3, "Durant", "Alice", "92100Coubevoie", 1100);  
    //Déclaration d'une liste de comptes  
    List<Compte> listComptes = new List<Compte>();  
    //Ajout des 3 objets compte  
    listComptes.Add(c1); listComptes.Add(c2); listComptes.Add(c3);  
    Console.ReadKey();  
}
```

# Les collections génériques en C#

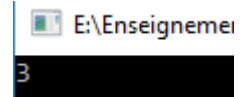
## ■ List<T>: (tableau unidimensionnel dynamique)

### □ Déclaration

### □ Utilisation

#### ➤ Nombre d'éléments de la liste (Count)

```
static void Main(string[] args)
{
    //Création de 3 objets Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    Compte c2 = new Compte(2, "Renault", "Isac", "10450Breviandes", 1130);
    Compte c3 = new Compte(3, "Durant", "Alice", "92100Coubevoie", 1100);
    //Déclaration d'une liste de comptes
    List<Compte> listComptes = new List<Compte>();
    //Ajout des 3 objets compte
    listComptes.Add(c1); listComptes.Add(c2); listComptes.Add(c3);
    //Affichage le nombre d'élément
    Console.WriteLine(listComptes.Count);
    Console.ReadKey();
}
```



E:\Enseignement  
3

# Les collections génériques en C#

## ■ List<T>: (tableau unidimensionnel dynamique)

### □ Déclaration

### □ Utilisation

### ➤ Accès à un élément de la liste (avec les index [ index])

```
static void Main(string[] args)
{
    //Création de 3 objets Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    Compte c2 = new Compte(2, "Renault", "Isac", "10450Breviandes", 1130);
    Compte c3 = new Compte(3, "Durant", "Alice", "92100Coubevoie", 1100);
    //Déclaration d'une liste de comptes
    List<Compte> listComptes = new List<Compte>();
    //Ajout des 3 objets compte
    listComptes.Add(c1); listComptes.Add(c2); listComptes.Add(c3);
    //Affichage de la liste en accédant à chaque élément
    string str = "";
    for (int i = 0; i < listComptes.Count; i++)
    {
        str += listComptes[i].toString();
    }
    Console.WriteLine(str);
    Console.ReadKey();
}
```

E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours.exe

```
Numéro compte: 1
Nom client: Dupont      Prénom client: Maxime
Numéro compte: 2
Nom client: Renault     Prénom client: Isac
Numéro compte: 3
Nom client: Durant      Prénom client: Alice
```



# Les collections génériques en C#

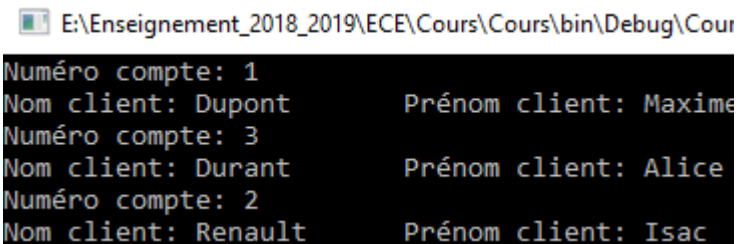
## ■ List<T>: (tableau unidimensionnel dynamique)

### ❑ Déclaration

### ❑ Utilisation

### ➤ Supprimer la première occurrence d'un objet (Remove)

```
static void Main(string[] args)
{
    //Création de 3 objets Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    Compte c2 = new Compte(2, "Renault", "Isac", "10450Breviandes", 1130);
    Compte c3 = new Compte(3, "Durant", "Alice", "92100Coubevoie", 1100);
    //Déclaration d'une liste de comptes
    List<Compte> listComptes = new List<Compte>();
    //Ajout des 3 objets compte
    listComptes.Add(c1); listComptes.Add(c2); listComptes.Add(c3); listComptes.Add(c2);
    //Suppression de la première occurrence
    listComptes.Remove(c2);
    //Affichage de la liste en accédant à chaque élément
    string str = "";
    for (int i = 0; i < listComptes.Count; i++)
    {
        str += listComptes[i].toString();
    }
    Console.WriteLine(str);
    Console.ReadKey();
}
```



```
E:\Enseignement_2018_2019\ECE\Cours\Cours\bin\Debug\Cour
Numéro compte: 1
Nom client: Dupont      Prénom client: Maxime
Numéro compte: 3
Nom client: Durant      Prénom client: Alice
Numéro compte: 2
Nom client: Renault      Prénom client: Isac
```

# Les collections génériques en C#

## ■ List<T>: (tableau unidimensionnel dynamique)

### ❑ Déclaration

### ❑ Utilisation

### ➤ Supprimer un objet à une position (RemoveAt(position))

```
static void Main(string[] args)
{
    //Création de 3 objets Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    Compte c2 = new Compte(2, "Renault", "Isac", "10450Breviandes", 1130);
    Compte c3 = new Compte(3, "Durant", "Alice", "92100Coubevoie", 1100);
    //Déclaration d'une liste de comptes
    List<Compte> listComptes = new List<Compte>();
    //Ajout des 3 objets compte
    listComptes.Add(c1); listComptes.Add(c2); listComptes.Add(c3); listComptes.Add(c2);
    //Suppression de l'objet en position 3
    listComptes.RemoveAt(3);
    //Affichage de la liste en accédant à chaque élément
    string str = "";
    for (int i = 0; i < listComptes.Count; i++)
    {
        str += listComptes[i].toString();
    }
    Console.WriteLine(str);
    Console.ReadKey();
}
```

E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cou

```
Numéro compte: 1
Nom client: Dupont      Prénom client: Maxim
Numéro compte: 2
Nom client: Renault    Prénom client: Isac
Numéro compte: 3
Nom client: Durant     Prénom client: Alice
```

# Les collections génériques en C#

## ■ List<T>: (tableau unidimensionnel dynamique)

### □ Déclaration

### □ Utilisation

### ➤ Supprimer tous les objets de la liste (Clear())

```
static void Main(string[] args)
{
    //Création de 3 objets Compte
    Compte c1 = new Compte(1, "Dupont", "Maxime", "75000Paris", 1500);
    Compte c2 = new Compte(2, "Renault", "Isac", "10450Breviandes", 1130);
    Compte c3 = new Compte(3, "Durant", "Alice", "92100Coubevoie", 1100);
    //Déclaration d'une liste de comptes
    List<Compte> listComptes = new List<Compte>();
    //Ajout des 3 objets compte
    listComptes.Add(c1); listComptes.Add(c2); listComptes.Add(c3);
    //Suppression de tous les objets
    listComptes.Clear();
    Console.WriteLine(listComptes.Count);
    Console.ReadKey();
}
```

E:\Enseignement\_2018\_2019\

0

# Les collections génériques en C#

- **List<T>: (tableau unidimensionnel dynamique)**
- **Déclaration**
- **Utilisation**
- **D'autres méthodes à découvrir...**

# Les collections génériques en C#

## ■ List<T>: (tableau unidimensionnel dynamique)

### □ Déclaration

### □ Utilisation

#### ➤ Parcourir une collection avec « foreach »

#### ➤ foreach permet de parcourir les objets des collections génériques

```
static void Main(string[] args)
{
    //Création de 3 objets Compte
    Compte c1 = new Compte(1,"Dupont","Maxime","75000Paris",1500);
    Compte c2 = new Compte(2, "Renault", "Isac", "10450Breviandes", 1130);
    Compte c3 = new Compte(3, "Durant", "Alice", "92100Coubevoie", 1100);
    //Déclaration d'une liste de comptes
    List<Compte> listComptes = new List<Compte>();
    //Ajout des 3 objets compte
    listComptes.Add(c1); listComptes.Add(c2); listComptes.Add(c3);
    //Parcourir les objets
    foreach(Compte c in listComptes)
    {
        Console.WriteLine(c.toString());
    }
    Console.ReadKey();
}
```

E:\Enseignement\_2018\_2019\ECE\Cours\Cours\bin\Debug\Cours

```
Numéro compte: 1
Nom client: Dupont      Prénom client: Maxime

Numéro compte: 2
Nom client: Renault     Prénom client: Isac

Numéro compte: 3
Nom client: Durant      Prénom client: Alice
```

# Les collections génériques en C#

## ■ Stack()

### □ Déclaration

```
Stack<char> tas = new Stack<char> ( ) ;
```

### □ Utilisation

- le nombre d'éléments dans la pile: **tas.Count**
- empiler un élément : **void tas.Push** (char item)
- dépiler et renvoyer l'élément du dessus de la pile : **char tas.Pop** ( )
- renvoie l'élément du dessus sans l'enlever : **char tas.Peek** ( )
- true ou false si item est dans tas : **bool tas.Contains** (char item)

# Les collections génériques en C#

## ■ Queue

### □ Déclaration

```
Queue<int> file = new Queue<int> ( ) ;
```

### □ Utilisation

- le nombre d'éléments contenus: file.**Count**
- placer un élément à la fin : void file.**Enqueue** (int item)
- retirer et renvoyer l'élément au début : int file.**Dequeue** ( )
- renvoie l'élément du debut sans l'enlever : int file.**Peek** ( )
- true ou false si item est dans tas : bool tas.**Contains** (char item)
- et d'autres ...