



## Introduction à C#

Frédéric CADIER (TELECOM Bretagne)  
Philippe TANGUY (TELECOM Bretagne)

## Présentation de la formation

### ■ Durée : 12 heures (4 x 3 heures)

- Durée courte : l'objectif est de faire une introduction au langage plutôt qu'un cours qui en balaye l'intégralité
- Approche pratique favorisée
  - TP : un sujet unique qui permet de voir les différentes parties abordées dans la formation  
Développement d'un **outil de simulation de crédit immobilier**
  - Quelques points théoriques pour amorcer le travail (≈ 20 % du temps)

### ■ Trois parties (cours/TP)

1. Introduction au langage
  - Parti pris de la présentation : connaissance supposée des bases du langage Java étude des ressemblances/différences entre C# et Java
2. Les interfaces graphiques
3. Connexion à une base de données (Oracle)



## Bibliographie

### ■ Bibliographie « énorme » sur le Web

#### ■ Extrait :

- Introduction au langage C#  
<http://tahe.developpez.com/dotnet/csharp/>
  - Dernière version du langage (C# 3.0) et de la plate-forme .NET (3.5)
  - Support qui a servi de base pour la présentation
- Club d'entraide des développeurs francophones : cours, tutoriaux, documentations, actualités, FAQ, etc.  
<http://www.developpez.com/> / <http://dotnet.developpez.com/>
- A Comparative Overview of C#  
[http://genamics.com/developer/csharp\\_comparative.htm](http://genamics.com/developer/csharp_comparative.htm)
- C# versus Java  
<http://www.dotnetguru.org/articles/CSharpVsJava.htm>
- Microsoft C# cours pour l'ESSI  
<http://alain.vizzini.free.fr/cours/csharp.htm>
  - Attention, la section qui compare C# à Java parle de Java 1.4 !
- Documentation (française) MSDN de Microsoft (*Microsoft Developer Network*)  
<http://msdn.microsoft.com/fr-fr/default.aspx>



## Introduction au langage C#

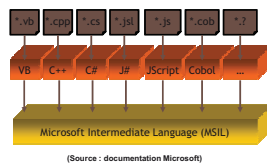
- Architecture de la plate-forme .NET
- Les ressemblances avec le langage Java
- Les principales différences
  - Types de base
  - Propriétés
  - Passage de paramètres
  - Structures
  - Redéfinition d'opérateurs
  - Indexeurs
- Les classes .NET d'usage courant
  - La classe `System.String` (*string*)
  - Les tableaux
  - Les structures de données génériques
  - La manipulation de fichiers texte

## Architecture de la plate-forme .NET

- Avec le langage Java, Sun offre la possibilité de créer des applications qui peuvent fonctionner sur plusieurs plate-formes (Unix(s), Windows, Mac OS, ...) avec un seul langage.

#### ■ Philosophie inverse de la plate-forme .NET : utilisation d'un framework commun à plusieurs langages (C#, Basic, C++, Eiffel, ...)

- Un module créé à l'aide d'un langage peut être utilisé de manière transparente avec un autre langage
- Standardisation par l'ECMA (*European Computer Manufacturers Association*) et l'ISO (*International Organization for Standardization*) des différents aspects de la plate-forme : *Common Language Infrastructure, Common Language Specification*, etc.
- Un langage plusieurs OS :
  - Microsoft ne développe la plate-forme .NET que pour Windows
  - approche alternative : plate-forme Mono (solution *open source*), implémentation partielle de la plate-forme .NET sous Linux (disponible aussi sous *Windows* et *MacOS*)  
<http://www.mono-project.com>



(Source : documentation Microsoft)



## Architecture de la plate-forme .NET

#### ■ Moteur d'exécution virtuel : le CLR (*Common Language Runtime*)

- Comparable à la machine virtuelle Java
  - Les programmes écrits dans les différents langages disponibles avec la plate-forme sont compilés en MSIL (*Microsoft Intermediate Language*) : code intermédiaire équivalent au bytecode Java (fichiers \*.class)
- Responsable du chargement et de l'exécution des programmes
- Fourniture des services nécessaires : ramasse-miettes, threading, débogage, ...



(Source : documentation Microsoft)

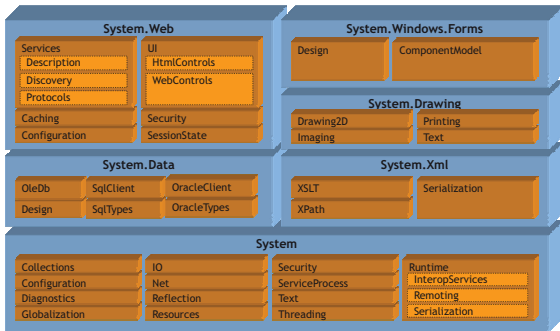
#### ■ Framework .NET : l'ensemble des classes de base qui fournissent les services aux applications .NET : accès aux réseaux, entrées/sorties, sécurité, ...

- CTS (*Common Type System*) : typage identique entre les différents langages (String en VB, C#, ...)

#### ■ Différentes piles de développement

- L'accès aux données (ADO.NET)
- Développement d'applications riches
- Développement d'applications Web

## Le Framework .NET : la librairie de base



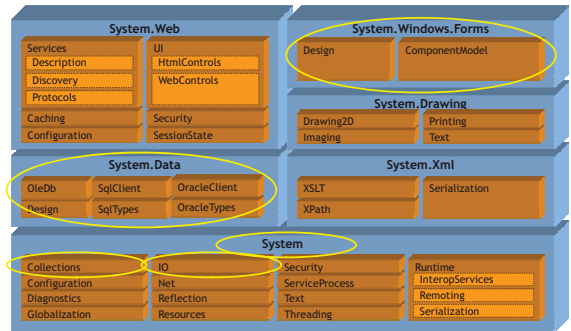
(Source : documentation Microsoft)

7

Mastère IADBA



## Le Framework .NET : la librairie de base



(Source : documentation Microsoft)

8

Mastère IADBA



## Les langages disponibles

- Perl
- Python
- Cobol
- Haskell
- ML
- JScript
- Ada
- APL
- Eiffel
- Pascal
- Fortran
- PHP
- Managed C++
- Visual Basic
- C#
- J#
- SmallTalk
- Oberon
- Scheme
- Mercury
- Oz
- Objective Caml
- Delphi
- ...

9

Mastère IADBA



## Définitions (1)

Source : <http://inico-pyright.developpez.com/tutorial/cv2005/managedworld/>

- **Managé (objet)**
  - Un objet est managé (managed) lorsque son allocation et sa désallocation sont prises en charge par le CLR.
- **Non managé (objet)**
  - Un objet est non managé (unmanaged) lorsque son allocation et sa désallocation sont prises en charge par le CRT (C Run-Time).
  - La compilation d'un programme en mode non managé produit du code machine (x86) directement exécutable mais qui ne bénéficie plus des fonctionnalités de la plate-forme .NET (sécurité, interopérabilité avec les autres langages, garbage collector, etc.).
- **CLS : Common Language Specification**
  - Objectifs : optimiser et assurer l'interopérabilité des langages en définissant un ensemble de fonctionnalités sur lequel les développeurs peuvent compter dans les différents langages.
  - Pour interagir entièrement avec des objets managés quel que soit le langage dans lequel ils ont été implémentés, les objets managés ne doivent exposer aux appelants que les fonctionnalités qui sont communes à tous les langages avec lesquels ils doivent fonctionner.
  - Pour cette raison, un ensemble de fonctionnalités de langage appelé spécification CLS (Common Language Specification), qui comprend les fonctionnalités de langage de base nécessaires à de nombreuses applications, a été défini.

10

Mastère IADBA



## Définitions (2)

- **CTS : Common Type System**
  - Afin que des classes définies dans plusieurs langages puissent communiquer entre elles, elles ont besoin d'un ensemble de types de données communs. C'est l'objet du CTS, il définit les types de données que le Runtime .NET comprend et que les applications .NET peuvent utiliser.
- **CLR : Common Language Runtime**
  - On l'appelle aussi le runtime .NET.
  - Le CLR est le moteur d'exécution du noyau .Net pour exécuter des applications. Il apporte des services tels que la sécurité d'accès de code, la gestion de la vie d'un objet, la gestion des ressources, la sûreté des types, etc.
- **CLI : Common Language Infrastructure**
  - Le CLI est l'environnement d'exécution formant le noyau du Framework .NET de Microsoft. La spécification définit un environnement qui permet d'utiliser de nombreux langages de haut niveau sur différentes plateformes sans nécessité de réécrire le code pour des architectures spécifiques.
- **MSIL : Microsoft Intermediate Language**
  - C'est un jeu d'instruction indépendant du CPU généré par les compilateurs .NET, à partir de langages comme le J#, C# ou Visual Basic. Le langage MSIL est compilé avant ou pendant l'exécution du programme par le VES (Virtual Execution System), qui fait partie intégrante du CLR.

11

Mastère IADBA



## Microsoft Visual Studio

- **En plus du Framework d'exécution et de la définition des langages, Microsoft fourni aussi les environnements de développement au sein de la suite Visual Studio**

- Evolution conjointe des environnements de développement et des Frameworks .NET
- Version actuelle : **Visual Studio 2008 (9.0)** fondée sur le Framework .NET 3.5
- Version de travail TELECOM Bretagne : **Visual Studio 2005 (8.0)** fondée sur le Framework .NET 2.0
- Microsoft propose depuis peu des versions gratuites disponibles en téléchargement : les versions **Express**. Contrairement aux versions payantes, il n'existe pas de version complète Express couvrant plusieurs langages (VB, C#, C++,...) :
  - **Visual Studio C# 2008 Express Edition**
  - Visual Studio Basic 2008 Express Edition
  - Visual Studio C++ 2008 Express Edition
  - Visual Studio Web Developer 2008 Express Edition
  - SQL Server 2008 Express Edition



12

Mastère IADBA



## C#, les ressemblances avec Java (1)

### ■ Caractéristiques communes aux deux langages :

- Beaucoup de point communs au niveau de la syntaxe : pour les deux, filiation avec le langage C (même cas de figure pour C++, PHP, Python, etc.)
  - Pour la première version du langage (.NET 1.0 en 2002) outils de conversion de code entre C# et Java. Impossibilité actuellement due à l'évolution disjointe des deux langages
- Le code est compilé dans un langage indépendant de la machine (*MSIL* et *bytecode Java*) destiné à être exécuté au dessus d'une machine virtuelle (*Microsoft CLR* et *JVM Java*)
- Utilisation d'un *ramasse-miettes* (Garbage Collector)
- Contrairement au C/C++, pas de fichier *header (.h)*
- Toutes les classes héritent de la super classe *object* (*Object* en Java) et les deux langages utilisent l'opérateur *new* pour créer de nouvelles instances
- Pas de constantes ou de méthodes globales : les attributs/méthodes appartiennent obligatoirement à une classe
- Contrôle de la longueur pour les tableaux et les chaînes de caractères
- L'opérateur *''* est utilisé, pas d'opérateurs *'->'* ou *'!'*
- null* et *boolean/bool* sont des mot-clés du langage
- Toutes les valeurs doivent être initialisées avant leur utilisation

13

Mastère IADBA



## C#, les ressemblances avec Java (2)

### ■ Instructions de contrôle du déroulement d'un programme :

- if(condition) { instructions conditions vrai } else {instructions conditions fausse }*
- switch(expression) { case v1: instructions; break; case v2: instructions; break; default: instructions; }*
- for (i=id; i<ll; i=i+ip) {instructions; }*
- foreach (Type variable in collection) {instructions; }*  
Equivalent Java (depuis 1.5) : *for (Type variable : collection) {instructions; }*
- while (condition) { instructions; }*
- do { instructions; } while (condition);*
- Break* fait sortir des boucles *for*, *while*, etc.  
*continue* fait passer à l'itération suivante

### ■ Gestion des exceptions

- try { code susceptible de déclencher une exception } catch(Exception e) { traitement de l'exception } finally { code exécuté après try ou catch }*

14

Mastère IADBA



## C#, les ressemblances avec Java (3)

### ■ Déclaration des énumérations

- Enum monEnum { val1, val2, val3};*

### ■ Les tableaux

- Type tableau[] = new Type[n];
- Type tableau [][] = new Type[n][m];

### ■ L'héritage :

- Concept identique à celui de Java (héritage simple) mais syntaxiquement différent :
  - Public class SousClasse : SurClasse { . . . }

- Classes abstraites

- Interfaces

### ■ Les packages :

- Concept identique à celui de Java avec une syntaxe différente les espaces de nom (namespace)
  - Namespace MonEspaceDeNom { public class MaClasse { . . . } }

### ■ Visibilité :

- private*, *protected*, *public*
- ≠ Java : *internal* et *protected internal*

15

Mastère IADBA



## C# exemple de code

```
namespace GestionPersonne {
    public class Personne {
        // Les attributs
        private string nom;
        private string prenom;
        // Constructeur sans paramètre
        public Personne() {
            this.nom = "default";
            this.prenom = "default";
        }
        // Constructeur avec paramètre
        public Personne(string nom, string prenom) {
            this.nom = nom;
            this.prenom = prenom;
        }
        // Méthode ToString de object redéfinie
        public override string ToString() {
            return "Personne { "
                + nom + ", " + prenom + " }";
        }
        // Utilisation
        class Program {
            static void Main(string[] args) {
                Personne p1 = new Personne("DUPONT", "Marcel");
                Console.WriteLine(p1);
            }
        }
    }
}
```

### ■ Note, de manière conventionnelle :

- Les méthodes et attributs avec une visibilité *private* ont un nom qui commencent par une minuscule
- Les méthodes et attributs avec une visibilité *public* commencent par une majuscule.

16

Mastère IADBA



## Types de base en C#

### ■ Types de données prédéfinis

| Type C#              | Type .NET                | Donnée représentée   | Suffixe des valeurs littérales | Codage    | Domaine de valeurs  |
|----------------------|--------------------------|----------------------|--------------------------------|-----------|---|
| <code>char</code>    | <code>Char (S)</code>    | caractère            |                                | 2 octets  | caractère Unicode (UTF-16)  |
| <code>string</code>  | <code>String (C)</code>  | chaîne de caractères |                                |           | référence sur une séquence de caractères Unicode  |
| <code>int</code>     | <code>Int32 (S)</code>   | nombre entier        |                                | 4 octets  | $[-2^{31}; 2^{31}-1]$ [-2147483648; 2147483647]   |
| <code>uint</code>    | <code>UInt32 (S)</code>  | -                    | U                              | 4 octets  | $[0; 2^{32}-1]$ [0; 4294967295]   |
| <code>long</code>    | <code>Int64 (S)</code>   | -                    | L                              | 8 octets  | $[-2^{63}; 2^{63}-1]$ [-9223372036854775808; 9223372036854775807]                       |
| <code>ulong</code>   | <code>UInt64 (S)</code>  | -                    | UL                             | 8 octets  | $[0; 2^{64}-1]$ [0; 18446744073709551615]   |
| <code>byte</code>    | <code>Byte (S)</code>    | -                    |                                | 1 octet   | $[0; 2^8-1]$ [0; 255]   |
| <code>sbyte</code>   | <code>Byte (S)</code>    | -                    |                                | 1 octet   | $[-2^7; 2^7-1]$ [-128; 127]   |
| <code>short</code>   | <code>Int16 (S)</code>   | -                    |                                | 2 octets  | $[-2^{15}; 2^{15}-1]$ [-32768; 32767]   |
| <code>ushort</code>  | <code>UInt16 (S)</code>  | -                    |                                | 2 octets  | $[0; 2^{16}-1]$ [0; 65535]  |
| <code>float</code>   | <code>Single (S)</code>  | nombre réel          | F                              | 4 octets  | $[1.5 \cdot 10^{45}; 3.4 \cdot 10^{38}]$ en valeur absolue                              |
| <code>double</code>  | <code>Double (S)</code>  | nombre réel          | D                              | 8 octets  | $[1.7 \cdot 10^{308}; 1.7 \cdot 10^{308}]$ en valeur absolue                            |
| <code>decimal</code> | <code>Decimal (S)</code> | nombre décimal       | M                              | 16 octets | $[1.0 \cdot 10^{28}; 9 \cdot 10^{28}]$ en valeur absolue avec 28 chiffres significatifs |
| <code>bool</code>    | <code>Boolean (S)</code> | booléen              |                                | 1 octet   | true, false   |
| <code>object</code>  | <code>Object (C)</code>  | référence d'objet    |                                |           | référence d'objet   |

G-dessus, on a mis en face des types C#, leur type .NET équivalent avec le commentaire (S) si ce type est une structure et (C) si le type est une classe. On découvre qu'il y a deux types possibles pour un entier sur 32 bits : *int* et *Int32*. Le type *int* est un type C#.

(Source : Apprentissage du langage C# - Serge Tahé - page 8)

17

Mastère IADBA



## Les propriétés

```
namespace GestionPersonne {
    public class Personne {
        // Les attributs
        private string nom;
        // Constructeur par défaut
        public Personne() {}
        // getters
        public string GetNom() {
            return nom;
        }
        // setters
        public string SetNom(string nom) {
            this.nom=nom;
        }
        // Utilisation
        class Program {
            static void Main(string[] args) {
                Personne p1 = new Personne();
                p1.SetNom("DUPONT");
                Console.WriteLine(p1.GetNom());
            }
        }
    }
}
```

### ■ Méthode de lecture et d'écriture des attributs privés

- Comme en Java, écriture possible d'accessieurs
  - Lecture : `getters`    `getXXX()`
  - Écriture : `setters`    `setXXX(Type XXX)`

18

Mastère IADBA



## Les propriétés

```
namespace GestionPersonne {
    public class Personne {
        // Les attributs
        private string nom;

        // Constructeur par défaut
        public Personne() {}

        // propriétés
        public string Nom {
            get { return nom; }
            set {
                if (nom != null)
                    nom = value;
                else
                    throw new Exception("Le nom ne peut pas être nul");
            }
        }

        public string Prenom { get; set; }

        // Utilisation
        class Program {
            static void Main(string[] args) {
                Personne p1 = new Personne();
                p1.Nom = "DUPONT";
                Console.WriteLine(p1.Nom);
            }
        }
    }
}
```

### ■ Méthode de lecture et d'écriture des attributs privés

- Autre façon : écriture possible de propriétés
- Une propriété permet de lire (*get*) et d'écrire (*set*)
  - Intérêt : faire un traitement (test de non nullité, ...) dans les accesseurs.
- Quand il n'y a pas de traitement particulier :
  - Le champ privé associé à la propriété n'est pas déclaré, le compilateur le crée automatiquement
  - L'accès à la valeur ne peut se faire que via la propriété (même au sein de la classe)
  - Equivalent à `public String Prenom;`
  - Intérêt : redéfinition dans le cas d'une propriété déclarée *virtual*

19

Mastère IADBA



## Passage de paramètres à une fonction

```
namespace XYX {
    public class ManipuleEntiers {
        public void DoubleEntier(int i) { // Valeur
            i = i * 2;
        }
        public void DoubleEntier(ref int i) { // Référence
            i = i * 2;
        }
    }

    // Utilisation
    class Program {
        static void Main(string[] args) {
            int i = 7;
            Console.WriteLine("i après initialisation = " + i);
            ManipuleEntiers.DoubleEntier(i);
            Console.WriteLine("i passage par valeur = " + i);
            ManipuleEntiers.DoubleEntier(ref i);
            Console.WriteLine("i passage par ref = " + i);
        }
    }
}
```

```
Sortie
*****
i après initialisation = 7
i passage par valeur = 7
i passage par ref = 14
```

### ■ Le passage par valeur

- Mode par défaut pour les types 'structure' (*struct*), 'énumération' (*enum*), numériques, caractère et booléen (ces 3 derniers sont en fait des structures, cf. *types de base*).
  - Le paramètre formel (interne) est une copie du paramètre effectif : on a deux entités distinctes et la valeur d'origine n'est pas modifiée.
- ### ■ Le passage par référence
- Le paramètre effectif et paramètre formel forment une seule entité.
  - Utilisation du mot-clé : *ref*
  - Mode par défaut pour les types *classes*

20

Mastère IADBA



## Les structures

### ■ Structure C# analogue à la structure du langage C mais reste très proche de la notion de classe

#### ■ Définition :

```
struct NomStructure {
    // attributs
    ...
    // propriétés
    ...
    // constructeurs
    ...
    // méthodes
    ...
}
```

### ■ Contrairement aux classes, les structures ne supportent pas l'héritage (hors interface)

### ■ Le passage d'une structure en tant que paramètre se fait par défaut par valeur

21

Mastère IADBA



## Redéfinition d'opérateurs

```
namespace GestionPersonne {
    public struct Personne {
        // Les attributs
        public string Nom;
    }

    class ListePersonnes : ArrayList {
        public static ListePersonnes operator +(ListePersonnes l,
            Personne p) {
            l.Add(p);
            return l;
        }
    }

    // Utilisation
    class Program {
        static void Main(string[] args) {
            Personne p1 = new Personne();
            p1.Nom = "DUPONT";
            Personne p2 = new Personne();
            p2.Nom = "DURAND";
            ListePersonnes liste = new ListePersonne(); // Vide !
            liste = liste + p1; // [p1]
            liste += p2; // [p1, p2]
        }
    }
}
```

### ■ Redéfinition de la signification d'un opérateur

- *opérande1 + opérande2*  
Il faut redéfinir l'opérateur '+' pour la classe de 'opérande1'
- La méthode doit être *public* et *static*
- Le compilateur remplace alors :
  - $x = op1 + op2$   
par  
 $x = ClasseOp1.operator+(op1, op2)$
- Possibilité de redéfinir aussi les opérateurs unaires (*++*, par exemple)
- Certains opérateurs doivent être redéfinis en même temps :
  - $==$  et  $!=$ ,  $<$  et  $>$ ,  $<=$  et  $>=$
- Redéfinition impossible pour :
  - $&&$  et  $||$  et  $[] = . ? :$

22

Mastère IADBA



## Définir un indexeur pour une classe

```
namespace GestionPersonne {
    public struct Personne {
        public string Nom;
    }

    class ListePersonnes {
        private Personne[] liste = new Personne[10];
        public string this[int index] {
            get {
                if (index < liste.Length)
                    return liste[index];
                else
                    return string.Empty;
            }
            set {
                if (index < liste.Length)
                    liste[index] = value;
            }
        }
    }

    // Utilisation
    class Program {
        static void Main(string[] args) {
            Personne p1 = new Personne();
            p1.Nom = "DUPONT";
            Personne p2 = new Personne();
            p2.Nom = "DURAND";
            ListePersonnes liste = new ListePersonne(); // Vide !
            liste[0] = p1; // [p1]
            liste[1] = p2; // [p1, p2]
        }
    }
}
```

### ■ La redéfinition de [] est impossible mais il est possible de définir un indexeur

- Permet d'accéder à un objet à la manière d'un tableau.
- La déclaration est similaire à celle d'une propriété dont le nom est *this*
- Comme pour une propriété, présence des accesseurs *get* et *set*.

23

Mastère IADBA



## La classe System.String

### ■ System.String est la classe du Framework .NET, string est le nom de la classe en C# (raccourci).

### ■ Comme en Java, cette classe possède un certain de fonctionnalités (extrait) :

- `int Length`
- `bool Equals(string s)`
- `string Insert(int index, string s)`
- `string Replace(char c1, char c2)`
- `string ToLower(string s)`
- `string ToUpper(string s)`
- `string Trim()`
- `uneChaine[]`

nb de caractères de la chaîne  
rend vrai si la chaîne est égale à s  
insère la chaîne s dans la chaîne à la position *index*  
renvoie une chaîne où le caractère *c1* est remplacé par le caractère *c2*  
renvoie la chaîne mise en minuscule  
renvoie la chaîne mise en majuscule  
renvoie la chaîne débarrassée des espaces de début et de fin  
i<sup>ème</sup> caractère de la chaîne *uneChaine*

24

Mastère IADBA



## Les tableaux

- Les tableaux héritent de la classe *Array*
- Cette classe possède différentes fonctionnalités pour :
  - Trier les éléments du tableau
    - static void **Sort**<T>(T[] tab, IComparer<T> compareur)  
Tri du tableau *tab* selon l'ordre défini par *compareur*
  - Rechercher un élément dans le tableau
    - static int **BinarySearch**<T>(T[] tab, T value)  
Renvoie la position de la valeur *value* dans le tableau *tab*
  - Redimensionner le tableau
    - static void **Resize**<T>(T[] tab, T newSize)  
Redimensionne le tableau *tab* à la nouvelle taille *newSize*
  - Divers
    - static void **Clear**(Array tab, int index, int length)  
Met les éléments du tableau *tab* (de la position *index* sur *length* éléments) à 0 si numérique, false si booléen, null si référence
    - static void **Copy**(Array tabSource, Array tabDestination, int length)  
Copie *length* éléments de *tabSource* dans *tabDestination*

25

Mastère IADBA



## Les collections génériques

- Différentes classes conteneurs pour stocker des collections d'éléments
  - Versions non génériques dans *System.Collections*
    - System.Collections.ArrayList* : collection d'objets dont la taille peut varier au cours du temps

```
System.Collections.ArrayList list = new System.Collections.ArrayList();
list.Add(3); // Ajout d'un entier
list.Add("Cocou c'est moi !"); // Ajout d'une chaîne de caractères

int t = 0;
foreach (int x in list) {
    t += x; // Erreur qui ne peut être détectée qu'à l'exécution
}
```
  - Versions génériques dans *System.Collections.Generic*
    - Structure de données pour laquelle il est possible de fixer le type d'éléments contenus
      - System.Collections.Generic.List*<T> *List<string>, List<int>*, etc.
      - System.Collections.Dictionary*<Tkey,Tvalue> *Dictionary<int,string>, Dictionary<string,int>*, etc.

26

Mastère IADBA



## System.Collections.Generic.List<T>

```
namespace GestionPersonne {
    struct Pers {
        public string Nom { get; set; }
        public string Prenom { get; set; }
        public override string ToString() { ... }
        // Comparateur pour un tri croissant par nom
        class TriCroissantNomHelper : IComparer<Pers> {
            int IComparer<Pers>.Compare(Pers p1, Pers p2) {
                return p1.Nom.CompareTo(p2.Nom);
            }
            // p1 < p2 : -1
            // p1 = p2 : 0
            // p1 > p2 : 1
        }
        // Obtention de l'instance du comparateur
        public static IComparer<Pers> TriCroissantNom () {
            return (IComparer<Pers>)new TriCroissantNomHelper();
        }
    }
    // Utilisation
    class Program {
        static void Main(string[] args) {
            List<Pers> liste = new List<Pers>();
            // Impossibilité de stocker autre chose que des 'Pers'
            liste.Add(new Pers { Nom = "DUBOIS", Prenom = "JP" });
            liste.Add(new Pers { Nom = "CADIER", Prenom = "JP" });

            liste.Sort(Pers.TriCroissantNom());
            Console.WriteLine("Après le tri");
            foreach (Pers p in liste)
                Console.WriteLine(p);
        }
    }
}
```

- Collection d'objets dont la taille peut varier au cours du temps
  - *ArrayList* est équivalent à *List<Object>*
- Principales méthodes
  - public void **Add**(T item)
  - public void **Clear**()
  - public bool **Contains**(T item)
  - public void **CopyTo**(T[] tableau)
  - public int **IndexOf**(T item)
  - public void **Insert**(T item, int index)
  - public bool **Remove**(T item)
  - public void **RemoveAt**(int index)
  - public int **BinarySearch**<T>(T item)
  - public int **BinarySearch**<T>(T item, IComparer<T> compareur)
  - public void **Sort**()
  - public void **Sort**<T>(compareur)
  - public T[] **ToArray**()

27

Mastère IADBA



## System.Collections.Generic.Dictionary<Tkey,TValue>

```
namespace GestionPersonne {
    struct Pers { ... }
    // Utilisation
    class Program {
        static void Main(string[] args) {
            List<Pers> liste = new List<Pers>();
            liste.Add(new Pers { Nom = "DUBOIS", Prenom = "JP" });
            liste.Add(new Pers { Nom = "CADIER", Prenom = "JP" });

            List<Pers> liste2 = new List<Pers>();
            liste2.Add(new Pers { Nom = "DUBOIS", Prenom = "A" });
            liste2.Add(new Pers { Nom = "DUBOIS", Prenom = "B" });

            // Le dictionnaire sera indexé par des chaînes
            // de caractères et les valeurs des instances
            // de "List<Pers>"
            Dictionary<string,List<Pers>> dict =
                new Dictionary<string,List<Pers>>();
            dict.Add("Equipe 1", liste);
            dict.Add("Equipe 2", liste2);

            foreach (P2 p in dict["Equipe 1"])
                Console.WriteLine(p);
        }
    }
}
```

- Implémentation d'un dictionnaire
  - Stockage de couples (clé, valeur)
  - Chaque clé est unique dans l'ensemble des clés
- Principales méthodes
  - public void **Add**(Tkey key, TValue value)
  - public void **Clear**()
  - public bool **ContainsKey**(Tkey key)
  - public bool **ContainsValue**(TValue value)
  - public void **CopyTo**(T[] tableau)
  - public bool **Remove**(Tkey key)
  - public bool **TryGetValue**(Tkey key,out TValue value)

28

Mastère IADBA



## Lecture à partir d'un fichier texte

```
namespace GestionPersonne {
    // Utilisation
    class Program {
        static void Main(string[] args) {
            try {
                using (StreamReader flux
                    = new StreamReader("C:/Test.txt"))
                {
                    // ligne non null pour entrer dans la boucle
                    string ligne = "";
                    while (ligne != null) {
                        ligne = flux.ReadLine();
                        Console.WriteLine(ligne);
                    }
                }
            } catch (IOException e) {
                Console.Error.WriteLine("Erreur : " + e.Message);
            }
        }
    }
}
```

- Utilisation de la classe *System.IO.StreamReader*
  - La classe est aussi capable de manipuler des flux qui ne sont pas des fichiers
- Principales méthodes
  - public void **Close**()
    - » ferme le flux et libère les ressources allouées pour sa gestion. A faire obligatoirement après exploitation du flux (alternative : utilisation de `using` [...] qui ferme le flux automatiquement).
  - public override int **Peek**()
    - » rend le caractère suivant du flux sans le consommer.
  - public override int **Read**()
    - » rend le caractère suivant du flux et avance d'un caractère dans le flux.
  - public override int **Read(char[] buffer, int index, int count)**
    - » il count caractères dans le flux et les met dans *buffer* à partir de la position *index*.
  - public override string **ReadLine**()
    - » rend la ligne suivante du flux ou *null* à la fin du flux.
  - public override string **ReadToEnd**()
    - » rend la fin du flux.

29

Mastère IADBA



## Ecriture dans un fichier texte

```
namespace GestionPersonne {
    // Utilisation
    class Program {
        static void Main(string[] args) {
            try {
                using (StreamWriter flux
                    = new StreamWriter("C:/Test2.txt")) {
                    flux.AutoFlush = true;
                    Console.WriteLine("Tapez une ligne vide pour arrêter la saisie.");
                    Console.WriteLine(" ");
                    string ligne = Console.ReadLine().Trim();
                    while (ligne != "") {
                        flux.WriteLine(ligne);
                        Console.WriteLine(" ");
                        ligne = Console.ReadLine().Trim();
                    }
                }
            } catch (IOException e) {
                Console.Error.WriteLine("Erreur : " + e.Message);
            }
        }
    }
}
```

- Utilisation de la classe *System.IO.StreamWriter*
  - La classe est aussi capable de manipuler des flux qui ne sont pas des fichiers
- Principales méthodes
  - public void **Close**()
    - » ferme le flux et libère les ressources allouées pour sa gestion.
  - public override void **Flush**()
    - » écrit dans le fichier, le buffer du flux, sans attendre qu'il soit plein.
  - public virtual void **Write**(T value)
    - » écrit *value* dans le fichier associé au flux. Ici *T* n'est pas un type générique mais symbolise le fait que la méthode `Write` accepte différents types de paramètres (string, int, object, ...). La méthode `value.ToString` est utilisée pour produire la chaîne écrite dans le fichier.
  - public virtual void **WriteLine**(T value)
    - » même chose que `Write` mais avec la marque de fin de ligne (`NewLine`) en plus.

30

Mastère IADBA

