Haute École Léonard de Vinci
SCIENCES ET TECHNIQUES

BINV3140-A
.NET Outils et Concepts d'Application d'Entreprise

Semaine 1

Introduction

Java vs C#

Objectifs du cours

- Comprendre les différences .NET et Java
  - Originalités, Avantages
- Etre capable de développer en .NET
  - LINQ , LINQ To Entities
  - WPF & MVVM
  - ASP.NET Web API
- Utiliser une architecture de développement d'entreprise
- Pouvoir trouver des informations

Organisation du cours

- Organisation
  - Solutions de la semaine disponible le vendredi
  - Dépôt Git : https://gitlab.vinci.be/olivier.choquet/net_solutions.git
- Examen
  - Sur machine
  - Ecriture de petites applications respectant une architecture d'entreprise

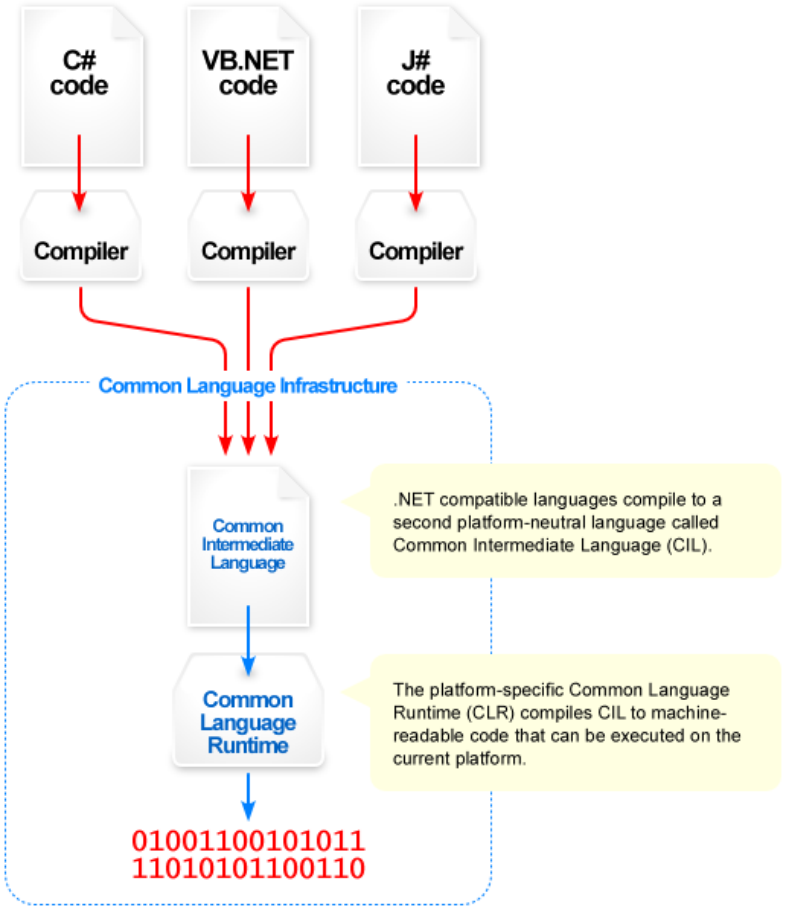
Planning

- S1 : Java vs C#
- S2 : LINQ
- S3 : LINQ To Entities
- S4 : Patterns Repository et UnitOfWork
- S5 : ASP.NET Web APIs
- S6 : async/await
- S7 : ASP.NET MVC
- S8 : WPF Début
- S9 – S10 : WPF - MVVM

.NET c'est quoi ?

- Microsoft .NET
  - Ensemble de produits et de technologies informatiques de l'entreprise Microsoft
- Framework .NET
  - Permet de travailler dans un environnement dit « managé » :
    - Gestion de l'exécution des programmes grâce à une machine virtuelle (cf. dia suivante)
    - Gestion de l'allocation des ressources en mémoire
  - Langage C# fort similaire à Java

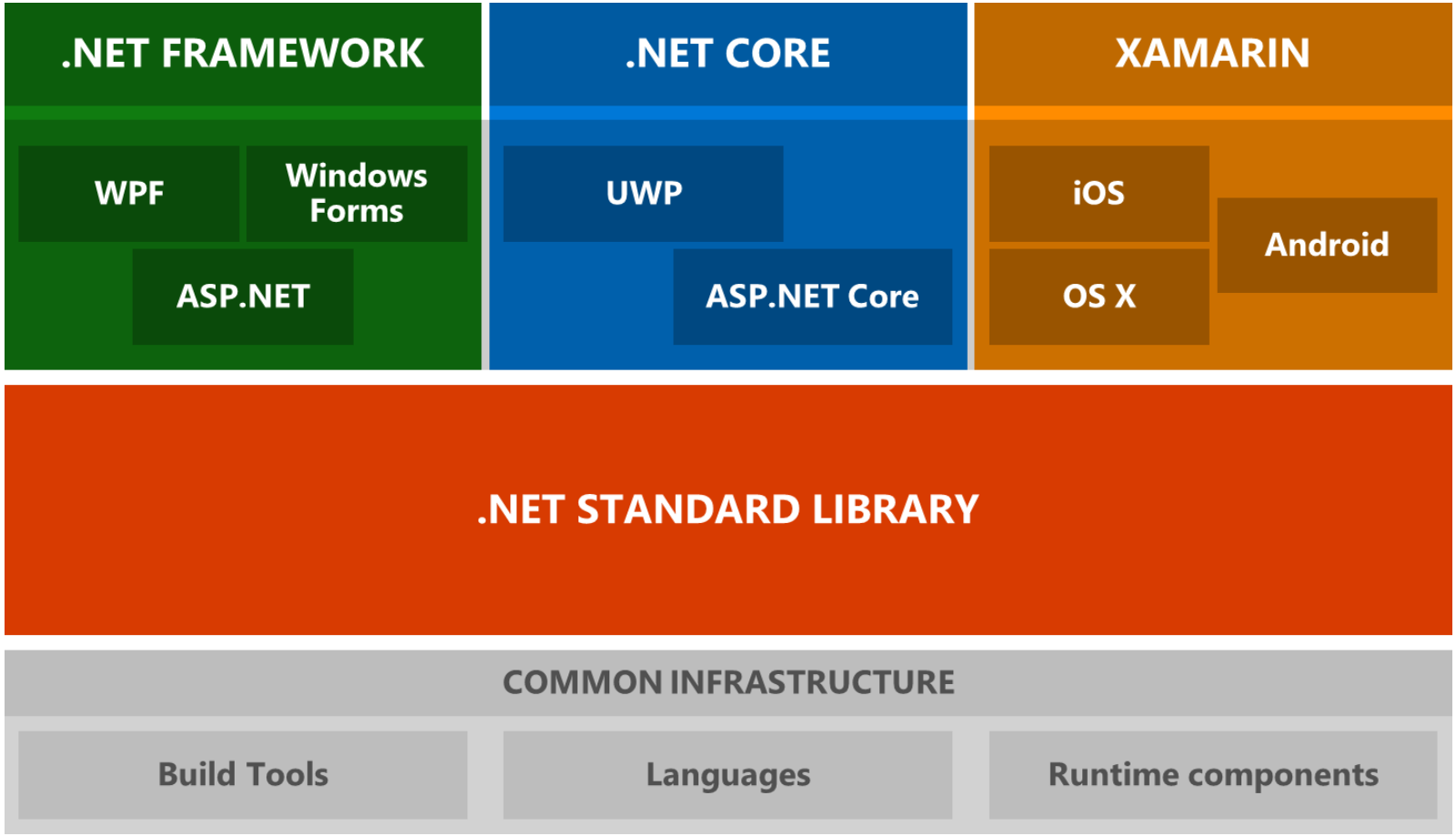
Machine virtuelle compatible CLI



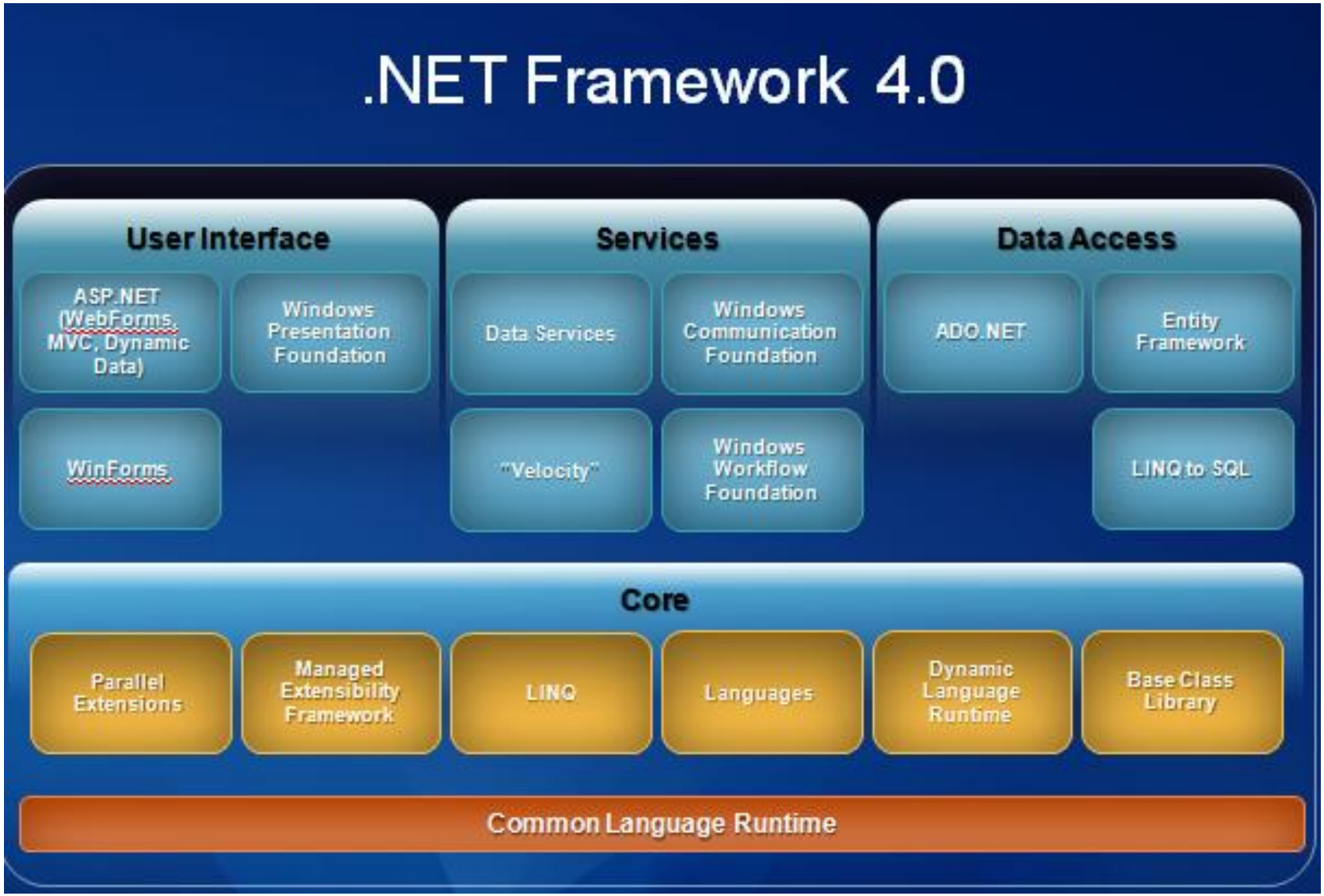
.NET en pratique

- Plateforme professionnelle pour les applications Windows ou Web
  - Windows 10, 11, ...
  - Suite Office
- Xbox 360
- Mais pas seulement :
  - ouvre une porte vers le multiplateforme
- Environnement très productif
  - Outils RAD intégrés dans Visual Studio 2022

.NET (Petit historique)



.NET Framework (Petit historique)



.NET 6.0 C# 10 (Actuellement)

Image issue de : https://www.onmsft.com/news/announcing-net-6-preview-1-whats-new?preview_id=219592

.NET & C# (Actuellement)

C# Version History

Version	.NET Framework	Visual Studio	Features Added
C# 1.0	.NET Framework 1.0/1.1	Visual Studio .NET 2002	The first release of C#
C# 2.0	.NET Framework 2.0	Visual Studio 2005	Generics Partial types, Anonymous methods, Nullable types
C# 3.0	.NET Framework 3.0\3.5	Visual Studio 2008	Auto-implemented properties, Anonymous types, Lambda expression
C# 4.0	.NET Framework 4.0	Visual Studio 2010	Dynamic binding, Named/optional arguments, Generic covariant
C# 5.0	.NET Framework 4.5	Visual Studio 2012/2013	Asynchronous members, Caller info attributes
C# 6.0	.NET Framework 4.6	Visual Studio 2013/2015	Static imports, Exception filters, Property initializers, Dictionary initializer
C# 7.0	.NET Core	Visual Studio 2017	Improved performance and productivity Azure Support, AI Support, Game development, Cross-platform, Mobile App Development
C# 8.0	.NET Core	Visual Studio 2019	Default interface methods, Static local function, Pattern Matching enhancements.
C# 9.0	.NET 5	Latest Visual Studio 2019	Top-level statements, Init only setters, Records
C# 10	.NET 6	Visual Studio 2022	Record Struct, nested property, Global using directives

.NET (Actuellement)

.NET and .NET Core release lifecycle

Supported versions

The following table tracks release dates and end of support dates for .NET and .NET Core versions.

Version	Original Release Date	Latest Patch Version	Patch Release Date	Support Level	End of Support
.NET 6	November 8, 2021	6.0.3	March 8, 2022	LTS	November 8, 2024
.NET 5	November 10, 2020	5.0.15	March 8, 2022	Current	May 8, 2022
.NET Core 3.1	December 3, 2019	3.1.23	March 8, 2021	LTS	December 3, 2022

Architecture d'Entreprise

Layers	.NET Framework Components
Frontend	Winforms, WPF MVVM, ASP.NET MVC, ConsoleApplication
Backend – Service (UCC)	ASP.NET Web APIs, WCF
Backend – Business Logic	C# Classes
Backend – DAL – Repository / UnitOfWork	Pattern Repository / UnitOfWork
Backend – DAL	LINQ To Entities – Entity Framework
Database	SQL Server

C# vs Java

- "Same but different"
- Présentation des différences dans les slides suivants

Java vs C#

Concept	Java	C#
Niveaux accessibilité	public /protected /private	https://docs.microsoft.com/fr-fr/dotnet/csharp/language-reference/keywords/accessibility-levels
Non redéfinissable	final	readonly, sealed, const
Annotation	@ (Ex: @Override)	[] (Ex: [Serializable]
Convention nommage	minuscule (Ex: main)	Majuscule (Ex: Main)
Propriétés (getter/setter)	Méthodes (Ex : obj.getTitle(), obj.setTitle())	Propriétés (Ex: obj.Title , obj.Title = "..") Voir détails slides suivants
Packaging	Organisation physique et logique (Ex: package be.ipl;)	Organisation logique (Ex: namespace be.ipl;)
Appel constructeur parent	super()	base() Attention à appeler avant le code {}
Date	Calendar (assez pourri)	Datetime
Collections	List -> ArrayList & Map -> hashMap	IList -> List & IDictionnary -> Dictionnary
Héritage	extends	:
Héritage	Simplement redéfinir une méthode dans la classe enfant	virtual dans la classe parent override dans la classe enfant
Implémentation interface	implements	:

Java vs C#

Concept	Java	C#
Itérateurs	Iterator	IEnumerator
Itérateurs	hasNext(), next()	MoveNext(), Current
Affichage	System.out.println()	Console.WriteLine()
Lecture au clavier	Scanner.nextLine()	Console.ReadLine()
String	Classe String	Classe String et alias string Utilisez plutôt l'alias (recommandation M$)
Verbatim String	Escape string	string myFileName = @"C:\myfolder\myfile.txt";
Interpolated String		string name = "olivier"; string str = $"bonjour{name}";
Nullable field	/	?bool : true/false or null
Commentaires	// /* …. */ /** Javadoc */	// /* ….*/ /// C# Documentation

Java vs C#

Concept	Java	C#
Déploiement	Création d'un exécutable à partir de .class ou .jar	Assembly .exe ou .dll (librairie de classes sans main)
Importation	import	using
Importation static		using static System.Console; -> ne plus devoir écrire Console.Writeline() mais directement Writeline()

Héritage

- Java
  - toutes les méthodes sont « virtual » c'est-à-dire qu'elles peuvent être redéfinies dans les classes enfants et que le type de l'objet est recherché à l'exécution
- C#
  - par défaut les méthodes ne sont pas « virtual » c'est-à-dire que le type de l'objet à l'exécution sera la classe la plus haute dans la hiérarchie.

- Pour obtenir le même comportement qu'en Java
  - virtual(parent) et override (enfant)

Concept de Propriété

- En Java
  - Getter/Setter
  - Ex :

```
public String getName() {
   return name;
}
```

- En C#
  - Propriétés
  - Ex :

```
public string Name {
   get { return _name; }
   set {_name = value;}
}
```

Concepts de Propriété

- Autre notation possible
- Ex :

```
public string Name {
    get { return _name; }
    set {_name = value;}
}

public string Name {
    get => _name;
    set => _name = value;
}
```

Concepts de Propriété

- Propriété implémentée automatiquement
  - Dans certains cas, les accesseurs de propriété `get` et `set` ne font qu'assigner une valeur à un champ de stockage, ou récupérer une valeur d'un champ de stockage, sans inclure de logique supplémentaire. En utilisant des propriétés implémentées automatiquement, vous simplifiez votre code, tout en laissant le compilateur C# fournir le champ de stockage de manière transparente.
  - Très utilisé les propriétés automatiques !
- Ex :

```
public string Name { get ; set; }
```

- Propriété en lecture seule

```
public string Name { get ; private set; }
```

- Propriété en écriture seule

```
public string Name { private get ; set; }
```

Delegate

- C# propose une originalité par rapport à Java
- Un délégué est un type de fonction
  - Ex : `delegate double funcDouble(double n);`
- Un délégué peut être passé en paramètre à une fonction
- Dès lors le nom d'un champ et d'une fonction ne peut pas être identique !!!
- Nous reviendrons et utiliserons les délégués en WPF !

C# vs Java

- "Same but different"
- Découverte via l'exercice … consultez les slides suivants
- Petite Démo Visual Studio
  - Notion Projet / Solution
    - .sln == solution /workspace
  - Ajouter une classe
  - Propriété
    - Propfull snippet
  - Déploiement Projet (Assembly)
    - DLL ou EXE
  - Intégration Git

Référence M$

- https://docs.microsoft.com/fr-fr/dotnet/csharp/programming-guide/