UNIVERSIDAD ICESI

Introducción al entorno de desarrollo

Laboratorio de Programación

Lorena Castañeda Bueno

Contenido

- .Net Framework
  - [Plataforma de desarrollo]
- Visual Studio
  - [Herramienta de desarrollo]
- C# (C-Sharp)
  - [Lenguaje de programación]

.Net Framework

[Plataforma de desarrollo]

¿Qué es .Net?

- Una plataforma de software que conecta información, sistemas, personas y dispositivos, así como sus diferentes tecnologías y lenguajes de programación.
- Desarrollada con base en los estándares de servicios Web XML, facilitando la comunicación entre estos sistemas.
- Provee un entorno que permite que diferentes lenguajes de programación y librerías trabajen juntos para crear aplicaciones e integrarlas a otros sistemas ya creados.

- ***Software para conectar información, personas, sistemas y dispositivos***
- ***Construido bajo la base de XML Web services***

Microsoft .Net Framework

- Es un componente de Windows que permite la creación y ejecución de aplicaciones.
  - NOTA: El .NET Framework se instala como un componente aparte en Windows 2000, mientras que Windows XP y versiones posteriores de Windows viene directamente en el sistema operativo.
- Brinda un entorno de programación orientada a objetos, tal que las aplicaciones desarrolladas bien sean: locales, distribuidas en internet ó ejecutadas remotamente.
- Ofrece una experiencia sobre la posibilidad de crear aplicaciones muy diferentes: Cliente, Web, Móviles, etc.

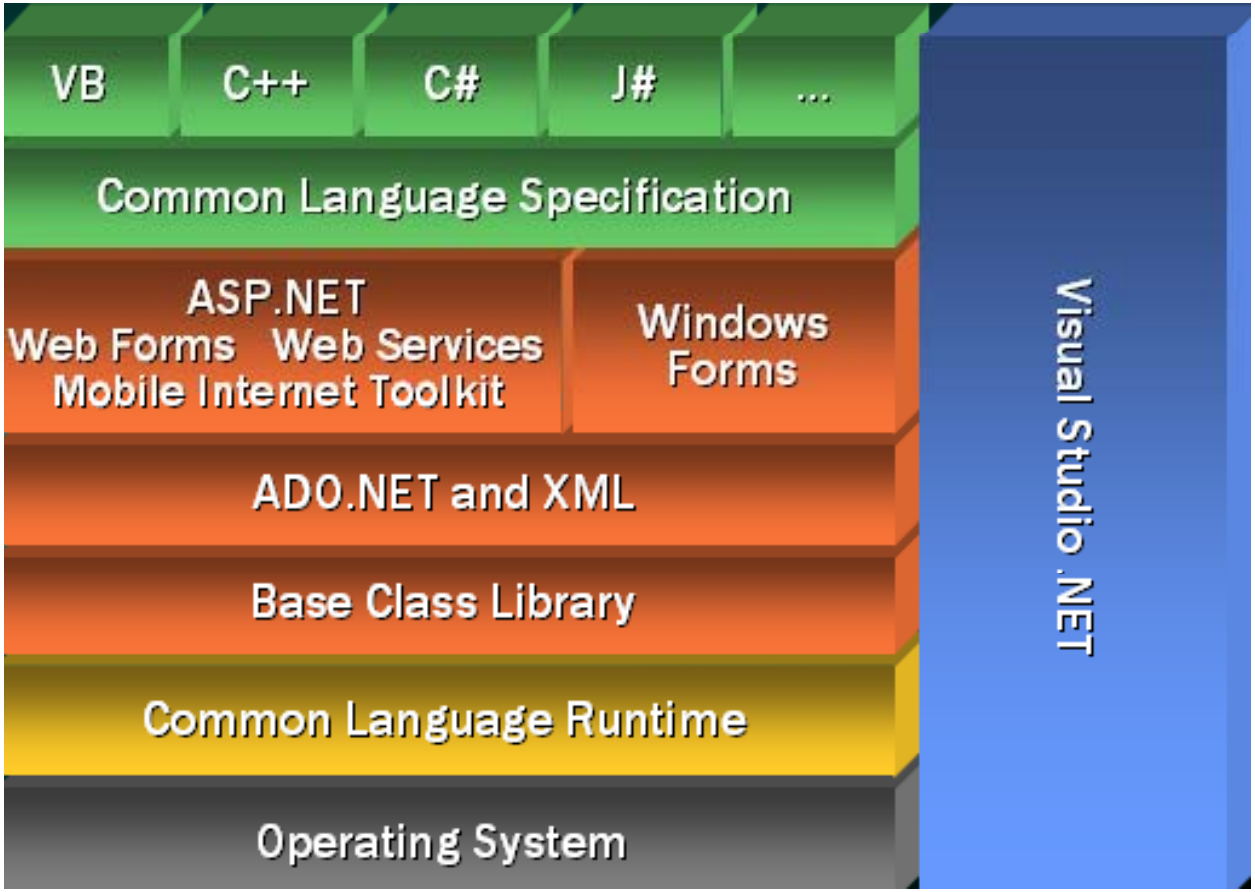
Componentes del Framework

Lenguajes de programación

Naturaleza de la aplicación

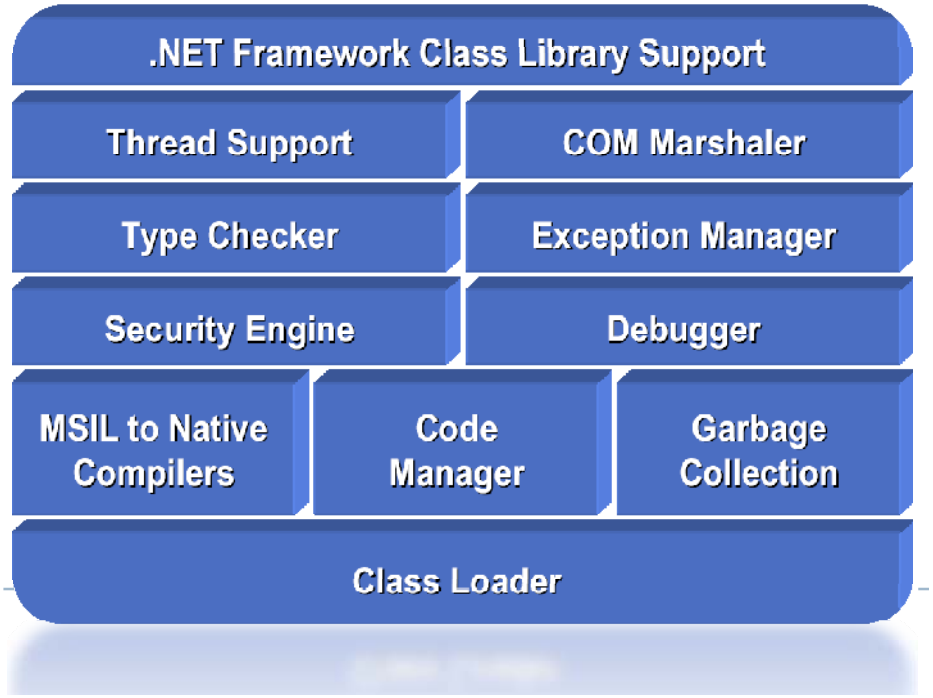
Acceso a datos

Biblioteca de clases

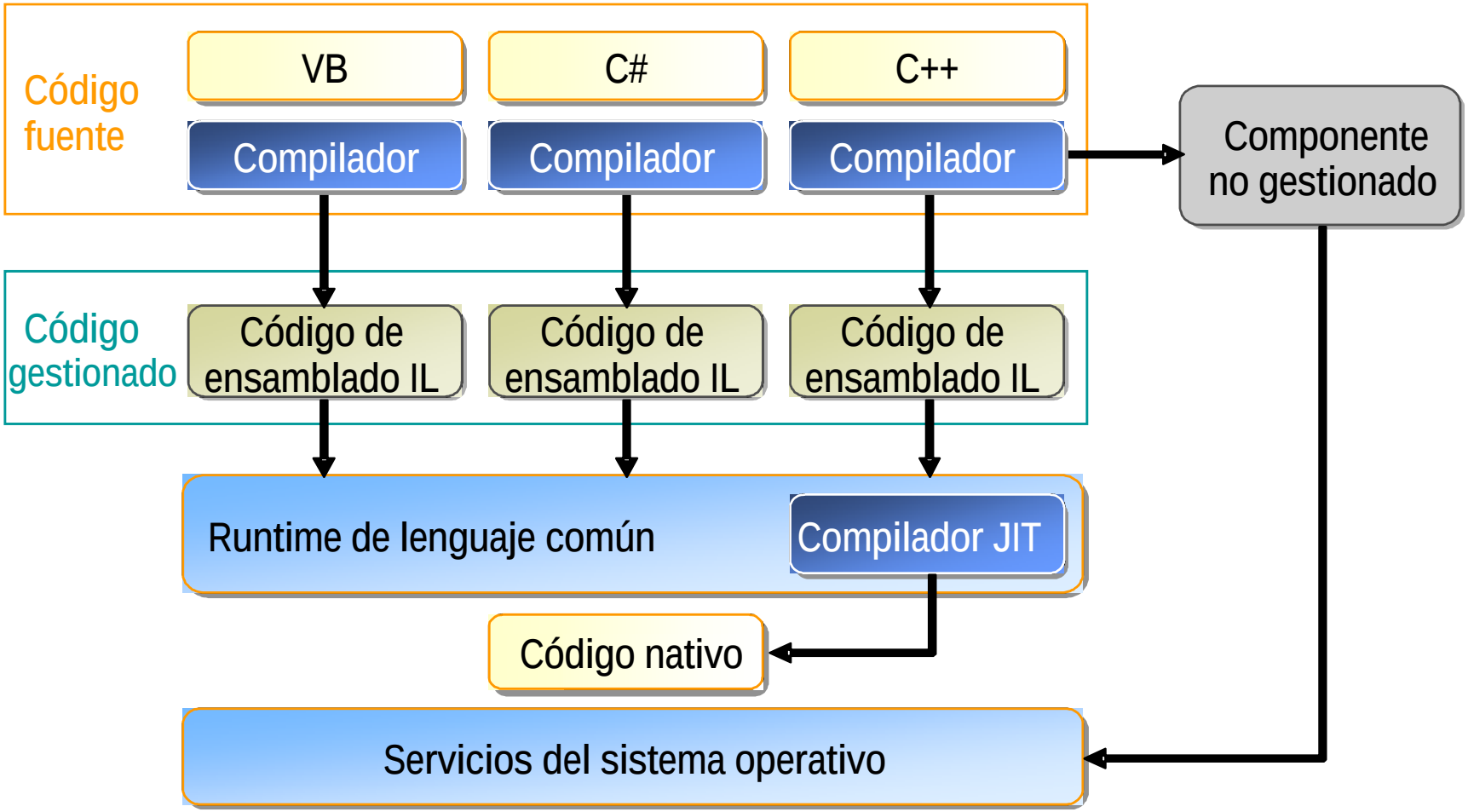


Common Language Runtime (CRL)

- Es el agente que administra el código durante su tiempo de ejecución.
- Administra los recursos físicos y lógicos para que la aplicación ejecute apropiadamente
- Permite ejecutar una aplicación en un sistema operativo

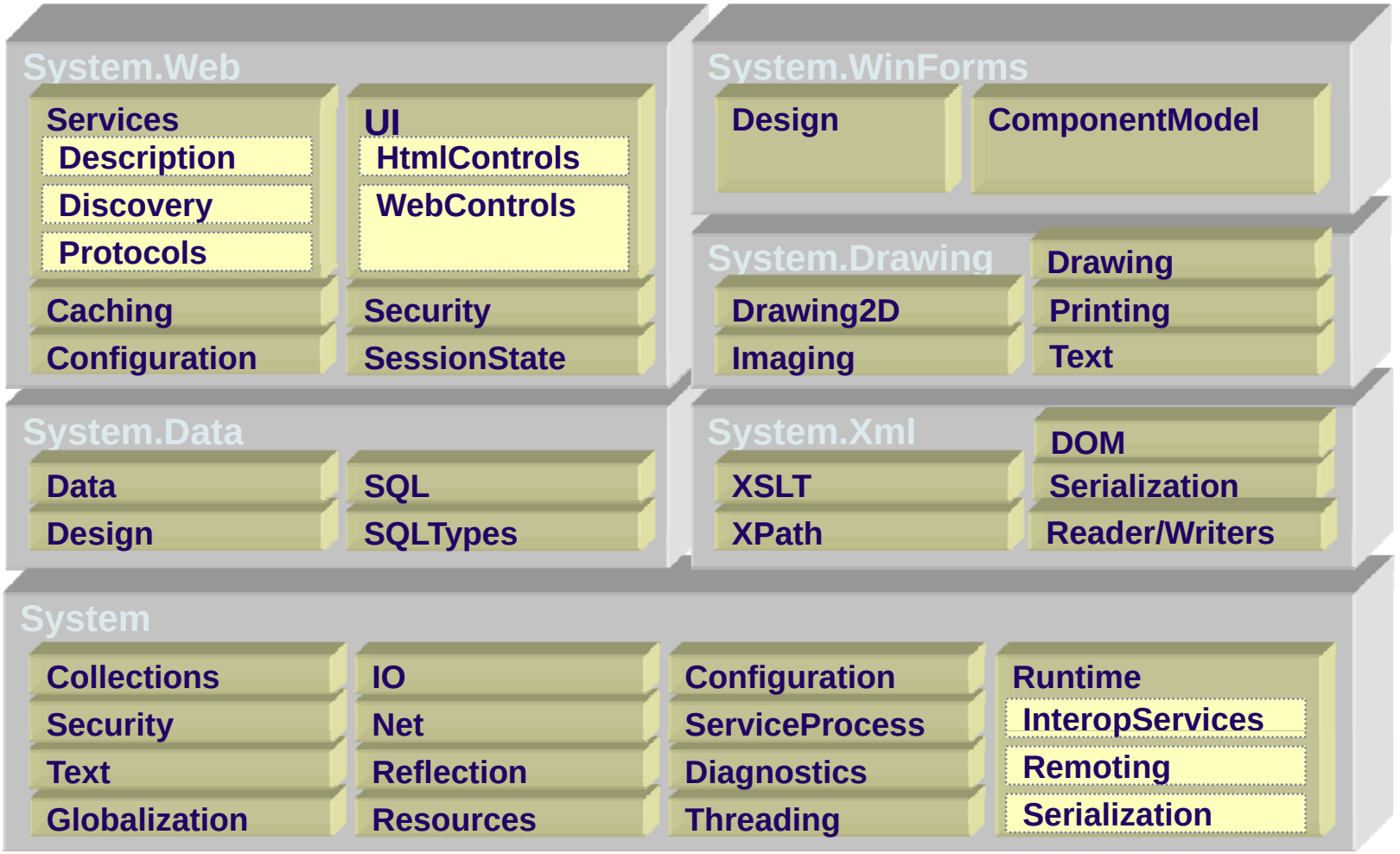


Arquitectura del .NET Framework
Modelo de Ejecución



Biblioteca de clases

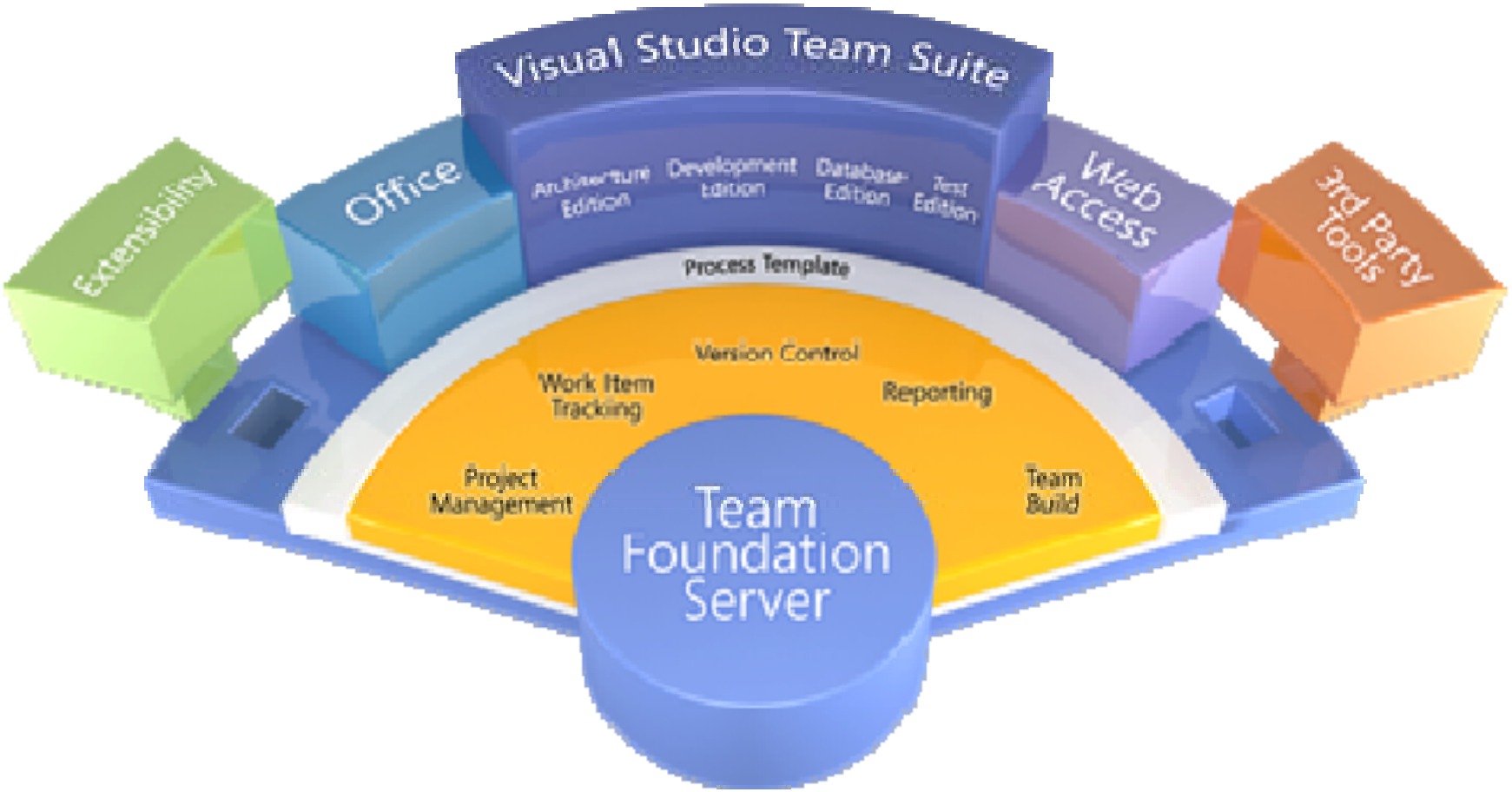
- Colección completa de tipos de datos, reutilizables, orientados a objetos para desarrollar aplicaciones.
- Elementos para el desarrollador, herramientas de interfaz, formularios y servicios Web XML.
- Se organizan en "paquetes" llamados **Namespace**


System.Web
Services
Description
Discovery
Protocols
UI
HtmlControls
WebControls
Caching
Configuration
Security
SessionState
System.WinForms
Design
ComponentModel
System.Drawing
Drawing2D
Imaging
Drawing
Printing
Text
System.Data
Data
Design
SQL
SQLTypes
System.Xml
XSLT
XPath
DOM
Serialization
Reader/Writers
System
Collections
Security
Text
Globalization
IO
Net
Reflection
Resources
Configuration
ServiceProcess
Diagnostics
Threading
Runtime
InteropServices
Remoting
Serialization

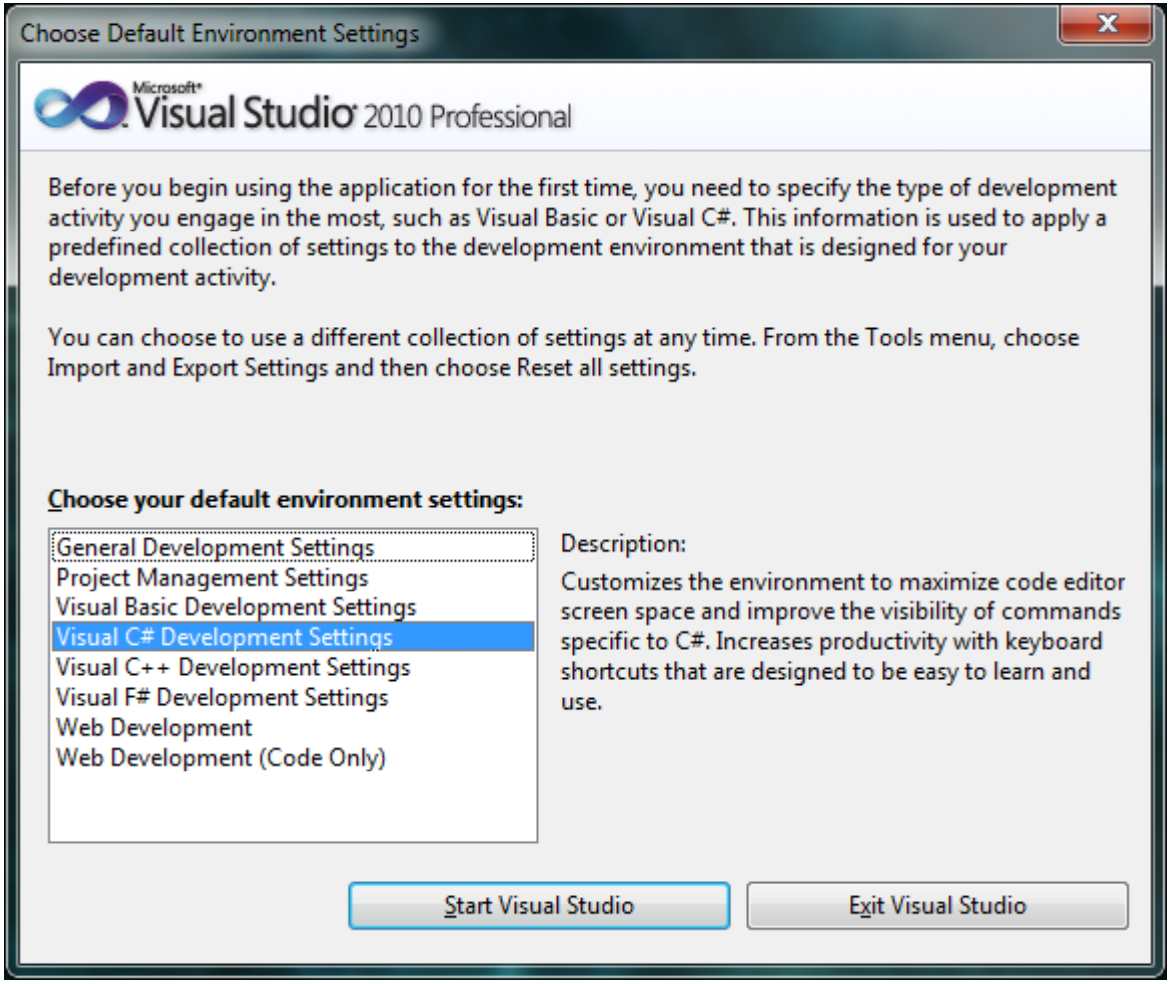
Visual Studio

[Herramienta de Desarrollo]

Visual Studio Team Suite



El entorno de usuario



C# (C-Sharp)

[Lenguaje de Programación]

C# (C Sharp)

- Lenguaje de Programación Orientado a Objetos (POO)
- Desarrollo y estándar de Microsoft
- Desciende de C/C++, sin embargo es altamente parecido a Java

Estructura básica de un programa en C#

```
using System;

class Hola
{
  public static void Main()
  {
    Console.WriteLine("Hola, mundo");
  }
}
```

La Clase

- Una aplicación C# es una colección de clases, estructuras y tipos
- Por ahora: una clase es un conjunto de datos y métodos
- Sintaxis

```
class nombre
{
      ...
}
```

- Una aplicación C# puede incluir muchos archivos
- Una clase no puede abarcar más de un archivo

El método Main

- Al escribir Main hay que:
  - Utilizar una "M" mayúscula, como en "Main"
  - Designar un Main como el punto de entrada al programa
  - Declarar Main como public static void Main
- Un Main puede pertenecer a múltiple clases
- La aplicación termina cuando Main acaba o ejecuta un return

El método Main

- .NET Framework ofrece muchas clases de utilidad
  - Organizadas en espacios de nombres
- System es el espacio de nombres más utilizado
- Se hace referencia a clases por su espacio de nombres

```
System.Console.WriteLine("Hola, mundo");
```

- La sentencia using

```
using System;
…
Console.WriteLine("Hola, mundo");
```

La Clase Console

- Funcionalidad básica utilizada en las primeras aproximaciones a la herramienta.
- Métodos Write y WriteLine

```
System.Console.WriteLine("Hola, mundo");
```

- Métodos Read y ReadLine

```
string cadena;
cadena = System.Console.ReadLine("Hola, mundo");
```

Comentarios en C#

```
// Obtener el nombre del usuario
Console.WriteLine(“¿Cómo se llama? ");
name = Console.ReadLine( );
```

```
/* Encontrar la  mayor raíz
de la ecuación cuadrática */
x = (…);
```

Definición y Asignación de Variables

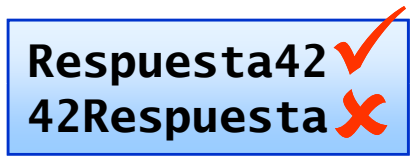
```
int edad;
edad = 22;
```

```
int edad = 22;
```

```
char letra = 'J';
```

Reglas y Recomendaciones Para Identificadores

- Reglas
  - Use letras, el signo de subrayado y dígitos
- Recomendaciones
  - Evite poner todas las letras en mayúsculas
  - Evite empezar con un signo de subrayado
  - Evite el uso de abreviaturas
  - Use PascalCasing para nombres con varias palab



Tipos Básicos Definidos por el Usuario

- Enumeraciones

```
enum Día { Lunes, Martes, Miércoles, Jueves, Viernes }

Día díaHoy = día.Miércoles;
```

```
public struct Persona
{
    public string nombre;
    public int edad;
}
```

```
Persona unaPersona;
unaPersona.nombre = “Diana";
unaPersona.edad = 23;
```

Conversiones Entre Tipos de Datos

- Conversión implícita

```
using System;
class Test
{
      static void Main( )
   {
            int intValor = 123;
            long longValor = intValor;
            Console.WriteLine("(long) {0} = {1}", intValor,
     ➥longValor);
      }
}
```

- Conversión explícita (cast)

```
using System;
class Test
{
      static void Main( )
   {
            long longValor = Int64.MaxValor;
            int intValor = (int) longValor;
            Console.WriteLine("(int) {0} = {1}", longValor,
     ➥intValor);
      }
}
```

Operadores Comunes

• Operadores de igualdad	== !=
• Operadores relacionales	< > <= >= is
• Operadores condicionales	&& || ?:
• Operador de incremento	++
• Operador de decremento	- -
• Operadores aritméticos	+ - * / %
• Operadores de asignación	= *= /= %= += -= <<= >>= &= ^= |=

Ejemplos de Uso de Operadores

```
número = número + 17;
número += 17;
número -= 7;
número += 1;
número -= 1;
número++;
número--;
++número;
--número;
```

¿Con qué valor termina la variable número?

¿Cuál es la diferencia entre los dos últimos pares de operaciones?

Instrucciones Condicionales

```
if ( expresión-booleana )
  primera-instrucción-incrustada
else
  segunda-instrucción-incrustada
```

```
enum Palo { Treboles, Corazones, Diamantes, Picas}
Palo cartas = Palo.Corazones;
if (cartas == Palo.Treboles)
    color = “Negro”;
else if (cartas == Palo.Corazones)
    color = “Rojo";
else if (palo == Palo.Diamantes)
    color = "Rojo";
else
    color = “Negro";
```

Instrucción Selectiva Múltiple

```
switch (palo) {
case Palo.Treboles :
case Palo.Picas :
    color = "Negro"; break;
case Palo.Corazones :
case Palo.Diamantes :
    color = "Rojo"; break;
default:
    color = "ERROR"; break;
}
```

Instrucción Iterativas

```
int i = 0;
while (i < 10) {
    Console.WriteLine(i);
    i++;
}
```

```
int i = 0;
do {
    Console.WriteLine(i);
    i++;
} while (i < 10);
```

```
for (int i = 0; i < 10; i++) {
    Console.WriteLine(i);
}
```

Instrucción Iterativas
La Instrucción foreach

```
ArrayList numeros = new ArrayList( );
for (int i = 0; i < 10; i++ ) {
    numeros.Add(i);
}

foreach (int number in numeros) {
    Console.WriteLine(numero);
}
```

Referencias

- Sitio Web oficial de Microsoft para desarrolladores de .Net
- http://msdn.microsoft.com/es-es/default.aspx