



# INTRODUCCIÓN MICROSOFT .NET Framework

Departamento de Ciencias de la Computación,  
Universidad de Chile

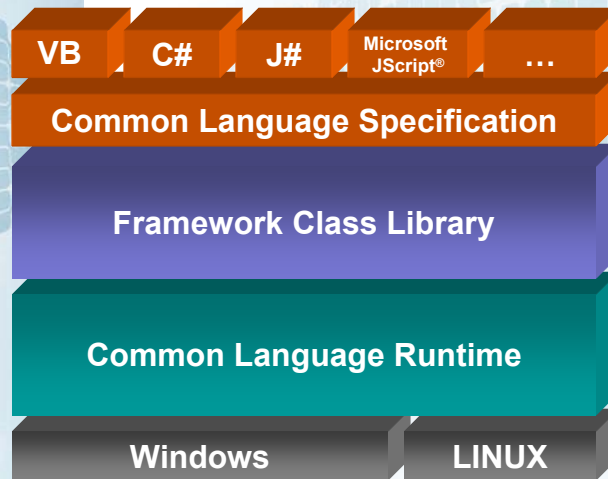
Hugo Andrés Neyem  
[aneyem@dcc.uchile.cl](mailto:aneyem@dcc.uchile.cl)

## Agenda

- Introducción a Microsoft .NET Framework
- Lenguaje: Visual C#
- Ejemplos de Tipos de Desarrollos

2005 | 2

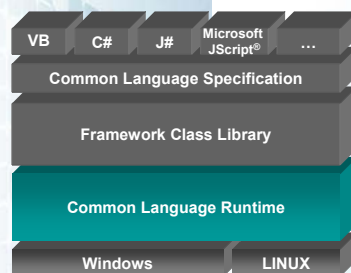
# Arquitectura



Introducción a Microsoft .NET Framework

2005 | 3

# Arquitectura



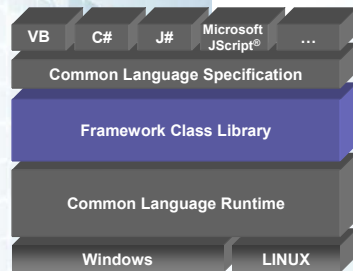
Common Language Runtime (CLR), es el motor de ejecución de las aplicaciones .NET Framework.

Los objetivos principales del CLR son simplificar el desarrollo de aplicaciones, ofrecer un entorno de ejecución robusto y seguro, facilitar la distribución y administración, y ofrecer soporte para múltiples lenguajes.

Introducción a Microsoft .NET Framework

2005 | 4

# Arquitectura

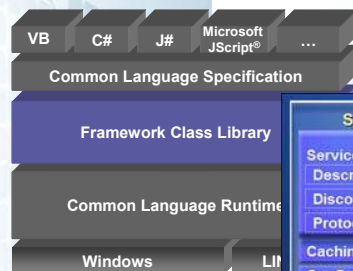


Framework Class Library (FCL), ofrece un conjunto de clases, interfaces y tipos reusables con las funcionalidades más básicas y comunes de la plataforma.

Permite implementar:

- Aplicaciones de consola. (System.Console).
- Aplicaciones basadas en formularios Windows. (System.Windows.Forms)
- Aplicaciones Web ASP.NET y servicios Web (System.Web)
- Acceso a fuentes de datos (System.Data)
- Acceso de ficheros y flujos (System.IO)
- y más...

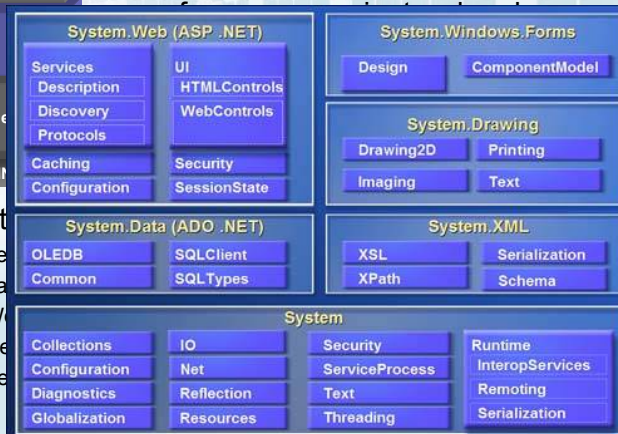
# Arquitectura



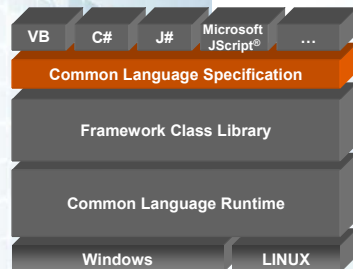
Framework Class Library (FCL),

Permite implementar:

- Aplicaciones de
- Aplicaciones ba
- Aplicaciones W
- Acceso a fuente
- Acceso de fiche
- y más...



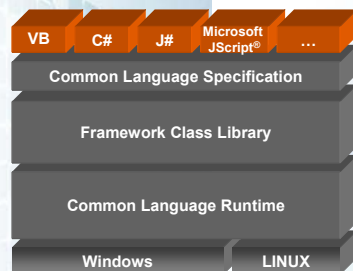
# Arquitectura



Common Language Specification (CLS), se trata de un conjunto de construcciones y restricciones que sirve como guía para los escritores de bibliotecas y compiladores.

Su finalidad no es otra que la de mejorar la comunicación entre programas escritos en otros lenguajes. Cuando un programa declara su compatibilidad con CLS, significa que puede ser utilizado con seguridad en un entorno de múltiples lenguajes.

# Arquitectura

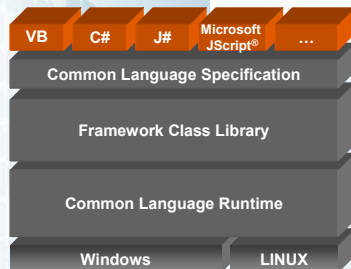


Microsoft provee compiladores CIL (Common Intermediate Language) para C#, J#, C++, VB y JScript.

- Los lenguajes de programación difieren en su sintaxis pero (prácticamente) coinciden en su potencia.
- Todo código fuente escrito para el framework se reduce a CIL.
- Terceros desarrollan compiladores CIL para Eiffel, Pascal, Python, Cobol, Prolog, etc.



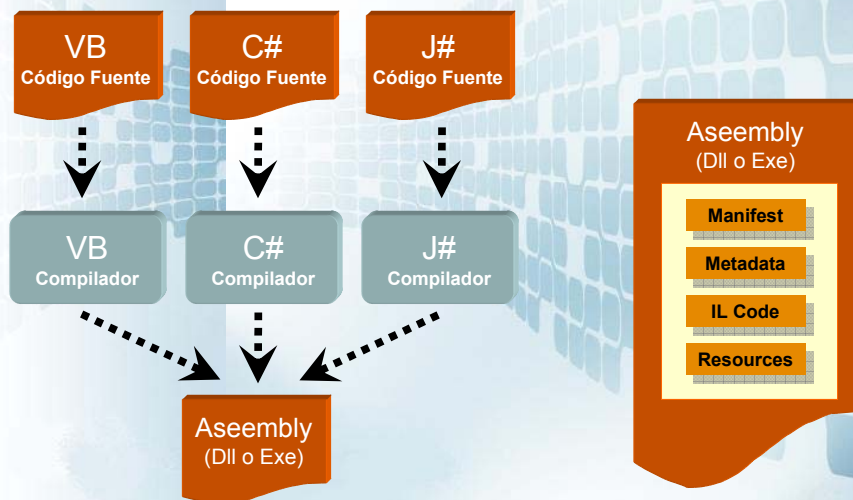
# Arquitectura



Introducción a Microsoft .NET Framework

2005 | 9

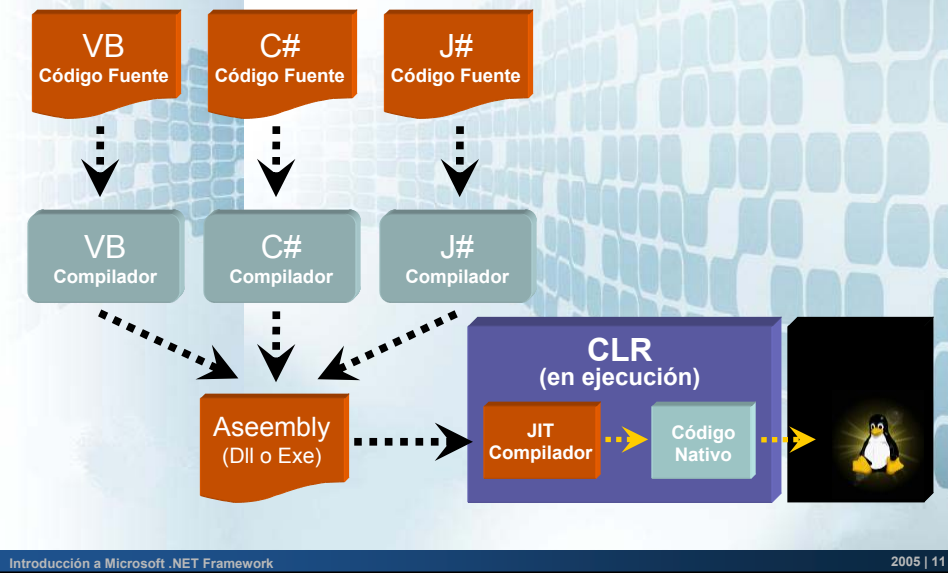
# Arquitectura



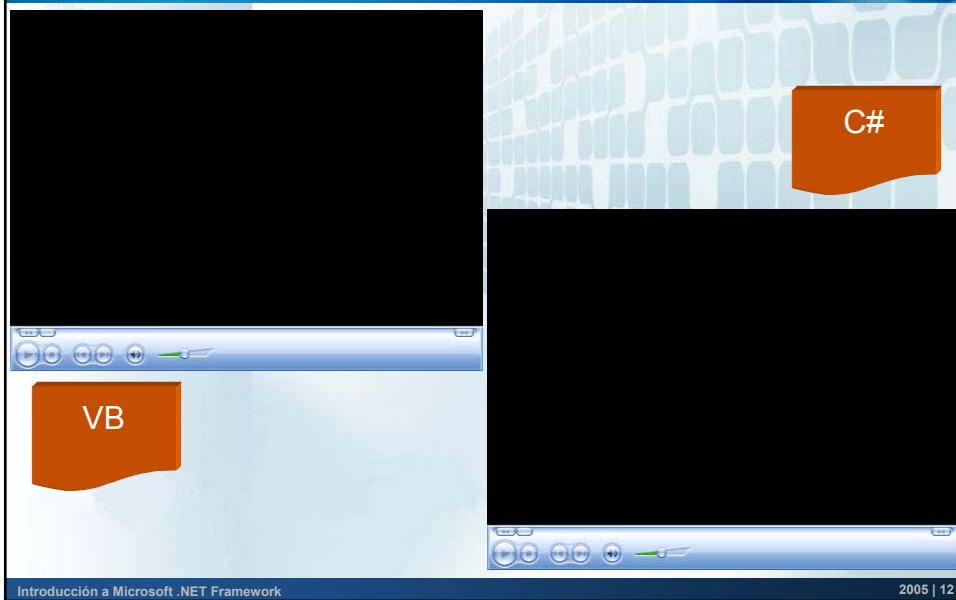
Introducción a Microsoft .NET Framework

2005 | 10

# Arquitectura



# Demo



# Disassembler (ILDASM.exe)

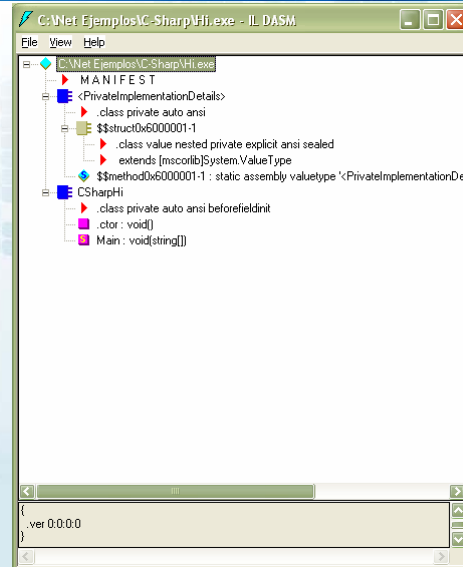
Assembly  
(Hi.exe)

Manifest

Metadata

IL Code

Resources



Introducción a Microsoft .NET Framework

2005 | 13

# Disassembler (ILDASM.exe)

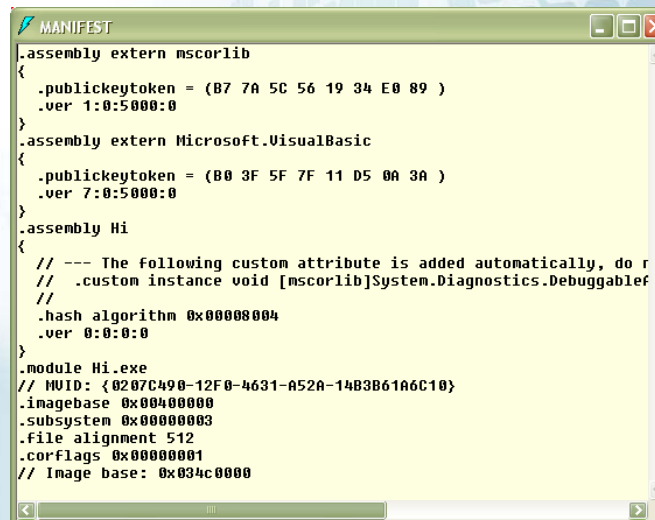
Assembly  
(Hi.exe)

Manifest

Metadata

IL Code

Resources



Introducción a Microsoft .NET Framework

2005 | 14

# Disassembler (ILDASM.exe)

**Assembly (Hi.exe)**

- Manifest
- Metadata
- IL Code
- Resources

```
.method private hidebysig static void Main(string[] args) cil managed
{
    .entrypoint
    // Code size       72 (0x48)
    .maxstack 3
    .locals init (unsigned int8[] V_0,
        unsigned int8 V_1,
        unsigned int8[] V_2,
        int32 V_3)
    IL_0000: ldc.i4.s 31
    IL_0002: newarr [mscorlib]System.Byte
    IL_0007: dup
    IL_0008: ldtoken field valuetype '<PrivateImplementationDetails>'/ '$$s'
    IL_000d: call void [mscorlib]System.Runtime.CompilerServices.RuntimeHelpers::GetFieldOfOffset(int32, IntPtr)
    IL_0012: stloc.0
    IL_0013: ldstr "Mensaje del Primer Ejemplo: "
    IL_0018: call void [mscorlib]System.Console::WriteLine(string)
    IL_001d: ldloc.0
    IL_001e: stloc.2
    IL_001f: ldc.i4.0
    IL_0020: stloc.3
    IL_0021: br.s IL_0036
    IL_0023: ldloc.2
    IL_0024: ldloc.3
    IL_0025: ldlen.u1
    IL_0026: stloc.1
    IL_0027: ldloc.1
    IL_0028: call char [Microsoft.VisualBasic]Microsoft.VisualBasic.StringBuilder::Append(char)
    IL_002d: call void [mscorlib]System.Console::Write(char)
    IL_0032: ldloc.3
    IL_0033: ldc.i4.1
    IL_0034: add
    IL_0035: stloc.3
    IL_0036: ldloc.3
    IL_0037: ldloc.2
    IL_0038: ldlen
    IL_0039: conv.i4
    IL_003a: bit.s IL_0023
    IL_003c: call void [mscorlib]System.Console::WriteLine()
    IL_0041: call string [mscorlib]System.Console::ReadLine()
    IL_0046: pop
    IL_0047: ret
} // end of method CSharpHi::Main
```

Introducción a Microsoft .NET Framework 2005 | 15

# Assembler (ILASM.exe)

**MANIFEST**

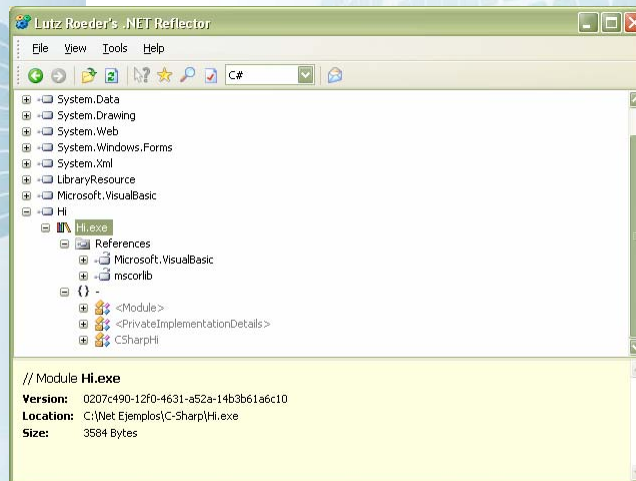
```
.assembly extern mscorlib
{
    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
    .ver 1:0:5:0:0:0
}
.assembly extern Microsoft.VisualBasic
{
    .publickeytoken = (B0 3F 5F 7F 11 D5 0A 3A )
    .ver 7:0:5:0:0:0
}
```

```
.method private hidebysig static void Main(string[] args) cil managed
{
    .entrypoint
    // Code size       72 (0x48)
    .maxstack 3
    .locals init (unsigned int8[] V_0,
        unsigned int8 V_1,
        unsigned int8[] V_2,
        int32 V_3)
    IL_0000: ldc.i4.s 31
    IL_0002: newarr [mscorlib]System.Byte
    IL_0007: dup
    IL_0008: ldtoken field valuetype '<PrivateImplementationDetails>'/ '$$s'
    IL_000d: call void [mscorlib]System.Runtime.CompilerServices.RuntimeHelpers::GetFieldOfOffset(int32, IntPtr)
    IL_0012: stloc.0
    IL_0013: ldstr "Mensaje del Primer Ejemplo: "
    IL_0018: call void [mscorlib]System.Console::WriteLine(string)
    IL_001d: ldloc.0
    IL_001e: stloc.2
    IL_001f: ldc.i4.0
    IL_0020: stloc.3
    IL_0021: br.s IL_0036
    IL_0023: ldloc.2
    IL_0024: ldloc.3
    IL_0025: ldlen.u1
    IL_0026: stloc.1
    IL_0027: ldloc.1
    IL_0028: call char [Microsoft.VisualBasic]Microsoft.VisualBasic.StringBuilder::Append(char)
    IL_002d: call void [mscorlib]System.Console::Write(char)
    IL_0032: ldloc.3
    IL_0033: ldc.i4.1
    IL_0034: add
    IL_0035: stloc.3
    IL_0036: ldloc.3
    IL_0037: ldloc.2
    IL_0038: ldlen
    IL_0039: conv.i4
    IL_003a: bit.s IL_0023
    IL_003c: call void [mscorlib]System.Console::WriteLine()
    IL_0041: call string [mscorlib]System.Console::ReadLine()
    IL_0046: pop
    IL_0047: ret
} // end of method CSharpHi::Main
```

Introducción a Microsoft .NET Framework 2005 | 16



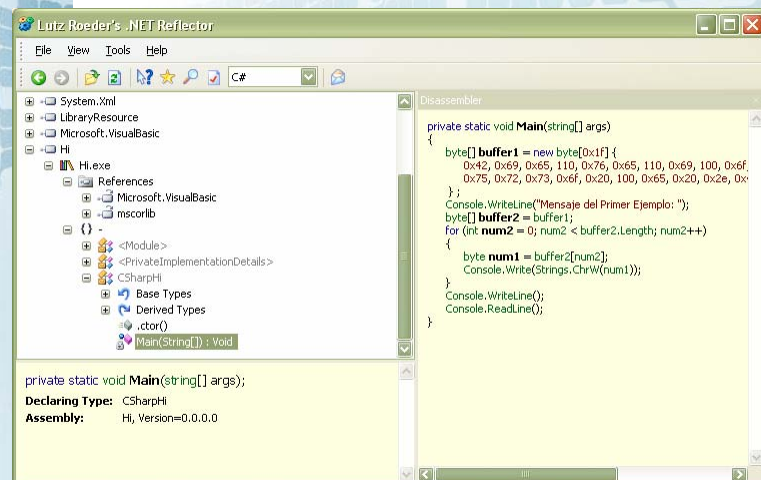
# Class Browser



Introducción a Microsoft .NET Framework

2005 | 17

# Class Browser



Introducción a Microsoft .NET Framework

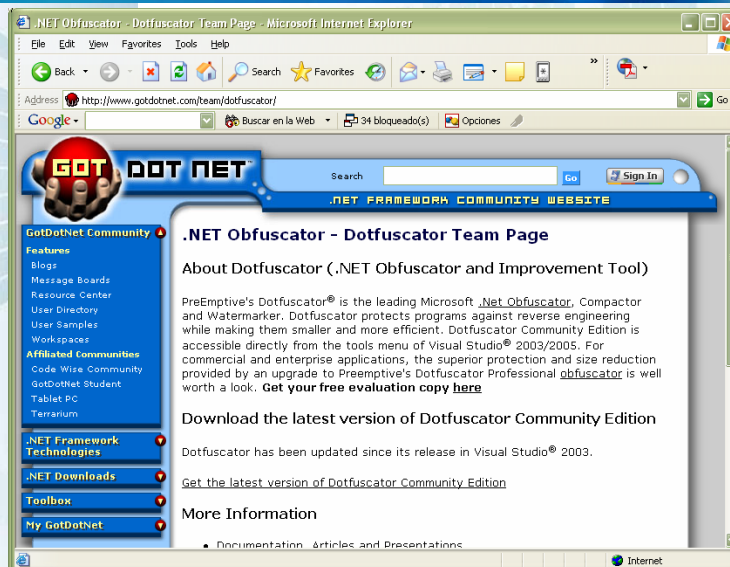
2005 | 18

# Ofuscación (Obfuscator)

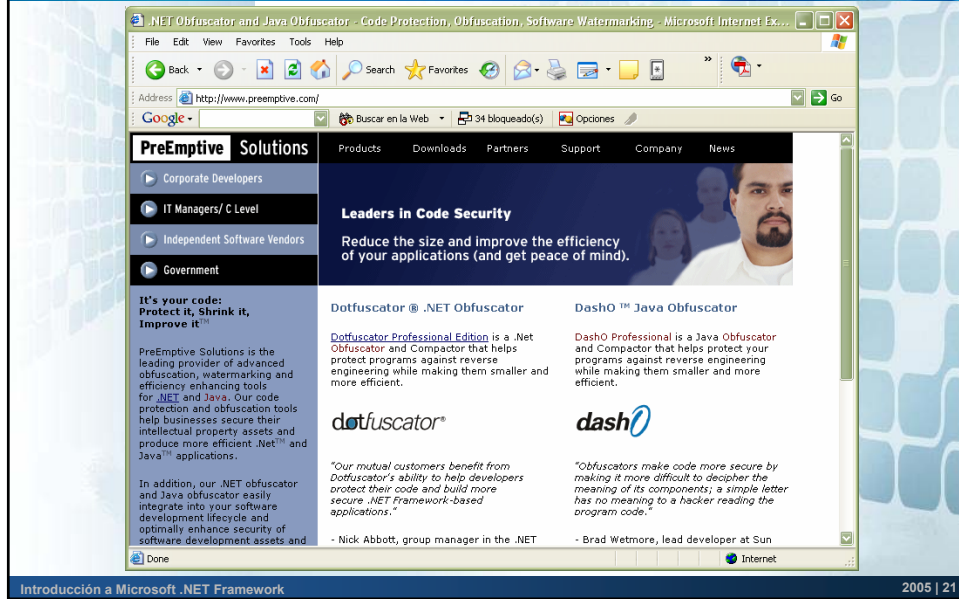
El objetivo de la ofuscación es crear confusión. A medida que crece la confusión, disminuye la capacidad de la mente para comprender conceptos intelectuales de varias facetas.

Observe que no se dice nada acerca de modificar la lógica final (ejecutable), sólo de representarla de manera incomprensible.

# Dotfuscator



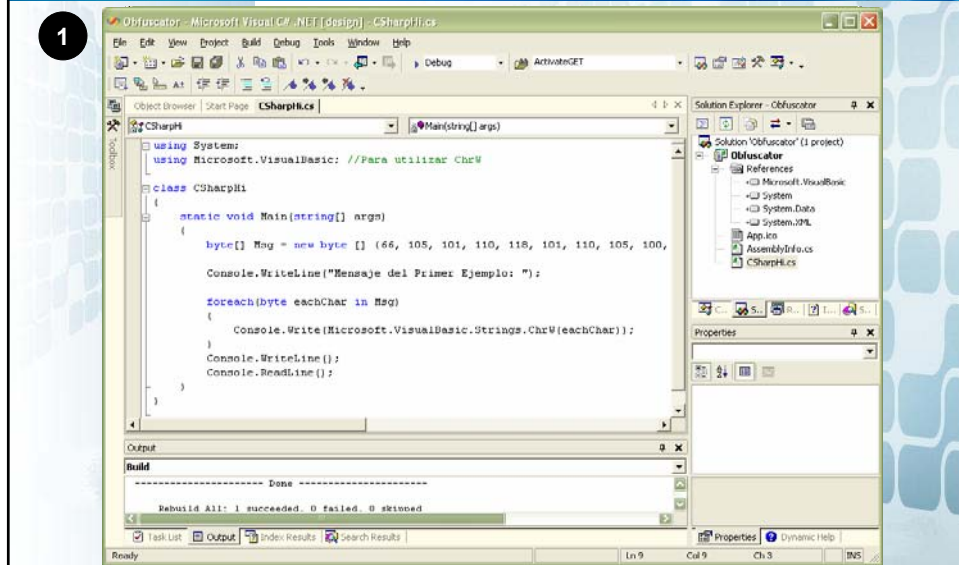
# Dotfuscator



Introducción a Microsoft .NET Framework

2005 | 21

# Dotfuscator

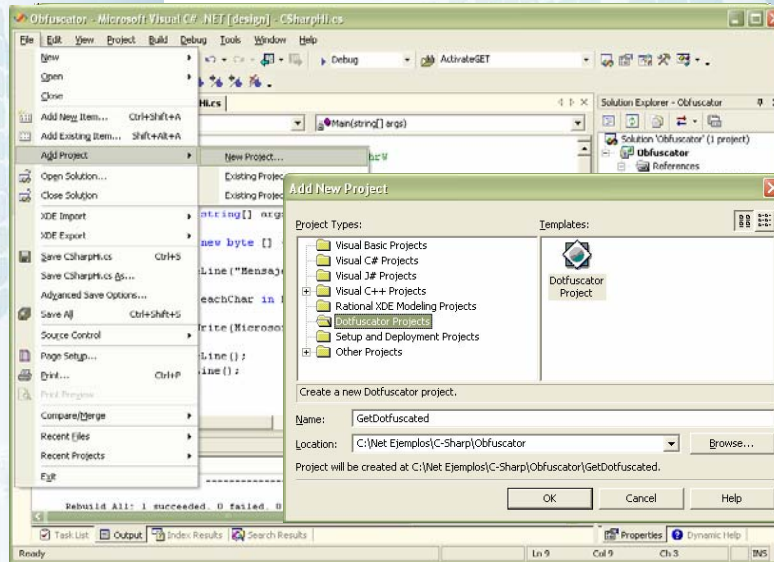


Introducción a Microsoft .NET Framework

2005 | 22

# Dotfuscator

2

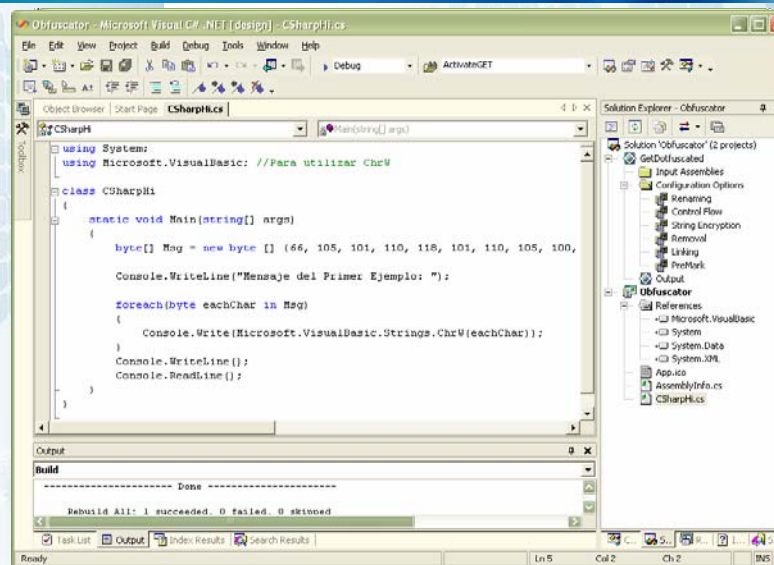


Introducción a Microsoft .NET Framework

2005 | 23

# Dotfuscator

3



Introducción a Microsoft .NET Framework

2005 | 24



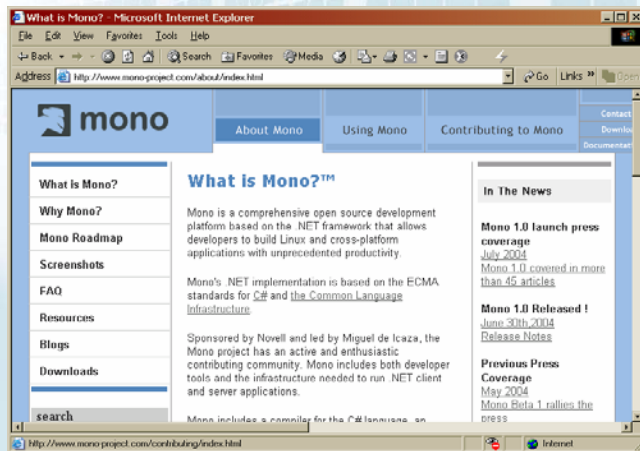




## ¿Cómo se obtiene MS .NET Framework?

Para sistemas operativos Linux visitar la URL:

<http://www.mono-project.com>



Introducción a Microsoft .NET Framework

2005 | 27

## Lenguaje C# (Descripción)

C# es un lenguaje de programación simple y robusto orientado a objetos que combina la simplicidad de Visual Basic con el poder y flexibilidad de C++.

Lenguaje: Visual C#

2005 | 28

## Lenguaje C# (Descripción)

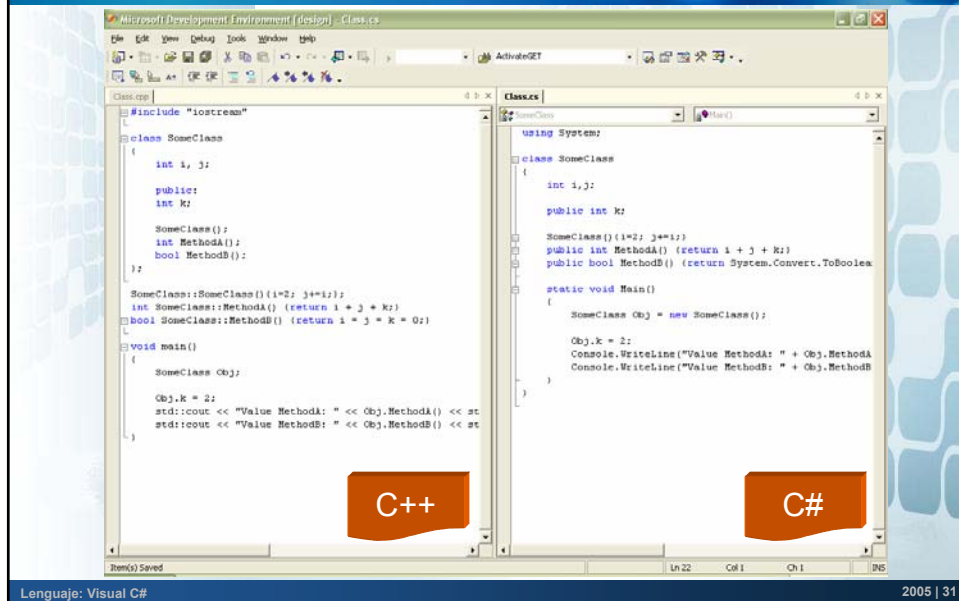
C# es un lenguaje de programación simple y robusto orientado a objetos que combina la simplicidad de Visual Basic con el poder y flexibilidad de C++.

C# ha sido diseñado específicamente por Microsoft para ser la opción de lenguaje para escribir las aplicaciones para su nueva plataforma .NET (plataforma de desarrollo diseñada pensando en Internet)

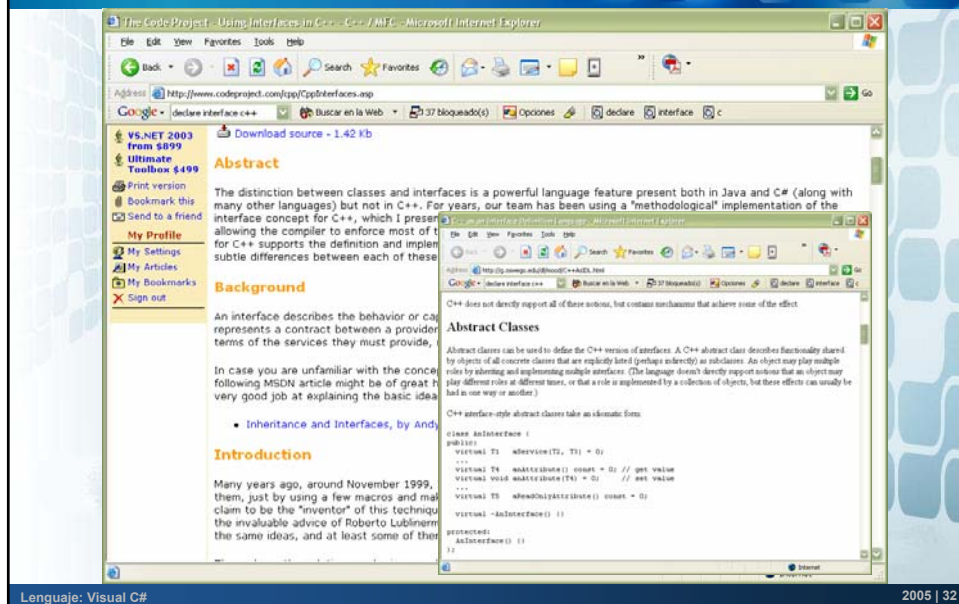
## C# class ≠ C++ class

La primero por comprender cuando se codifica en C#, es que cuando uno define una clase y sus métodos, la definición de éstos métodos deben estar dentro del propio cuerpo de la clase.

# C# class ≠ C++ class

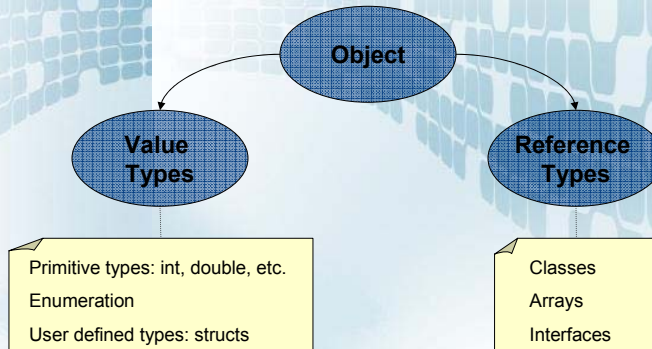


# C# class ≠ C++ class



# Tipos de Datos

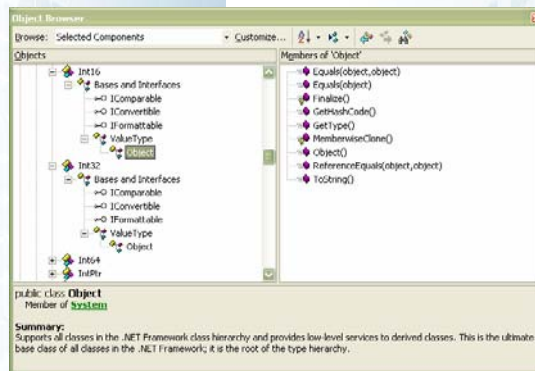
TODO ES UN OBJETO, ya que tiene herencia implícita de System.Object



Lenguaje: Visual C#

2005 | 33

# Tipos de Datos



Espacio	C#	VB .NET	.NET Framework
2 Bytes	short	Short	System.Int16
4 Bytes	int	Integer	System.Int32
8 Bytes	long	Long	System.Int64
...	...	...	...

Lenguaje: Visual C#

2005 | 34



# Tipos de Datos

Posibles Valores	Espacio	C#	VB .NET	.NET Framework
-32768 / 32767	2 Bytes	short	Short	System.Int16
-2147483648 / 2147483647	4 Bytes	int	Integer	System.Int32
-9223372036854775808 / 9223372036854775807	8 Bytes	long	Long	System.Int64
0 / 255	1 Byte	byte	Byte	System.Byte
-128 / 127	1 Byte	sbyte	---	System.Sbyte
0 / 65535	2 Byte	ushort	---	System.UInt16
0 / 4294967295	4 Byte	uint	---	System.UInt32
0 / 18446744073709551615	8 Byte	ulong	---	System.UInt64
$\pm 1.5 \times 10^{-45}$ / $\pm 3.4 \times 10^{38}$	4 Byte	float	Single	System.Single
$\pm 5.0 \times 10^{-324}$ / $\pm 1.7 \times 10^{308}$	8 Byte	double	Double	System.Double
$\pm 1.0 \times 10^{-28}$ / $\pm 7.9 \times 10^{28}$	12 Byte	decimal	Decimal	System.Decimal
true / false	2 Byte	bool	Boolean	System.Boolean
Cualquier Caracter Unicode 16-bit	2 Byte	char	---	System.Char
Cualquier cadena de caracteres	variable	string	String	System.String

Lenguaje: Visual C#

2005 | 35

# Tipos de Datos

C# incluye la construcción enum con la que se definen tipos por valor.

```

// In this example, an enumeration, Days, is declared.
// The enumerators are explicitly converted to int
// and assigned to int variables.
//
using System;
public class EnumTest
{
    enum Days {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};

    public static void Main()
    {
        int x = (int) Days.Sat;
        int y = (int) Days.Fri;
        Console.WriteLine("Sat = {0}", x);
        Console.WriteLine("Fri = {0}", y);
    }
}
/*
Output:
Sat = 1
Fri = 5
Notice that if you remove the initializer
from Sat=1, the result will be:
Sat = 0
Fri = 6
*/
//

```

```

// In this example, the long-type option is used to declare
// an enum whose members are of the type long.
// Notice that even though the underlying
// type of the enumeration is long,
// the enumeration members must still be explicitly
// converted to type long using a cast.
//
using System;
public class EnumTest
{
    enum Range {long {Max = 2147483648L, Min = 255L};

    public static void Main()
    {
        long x = (long) Range.Max;
        long y = (long) Range.Min;
        Console.WriteLine("Max = {0}", x);
        Console.WriteLine("Min = {0}", y);
    }
}
/*
Output:
Max = 2147483648
Min = 255
*/
//

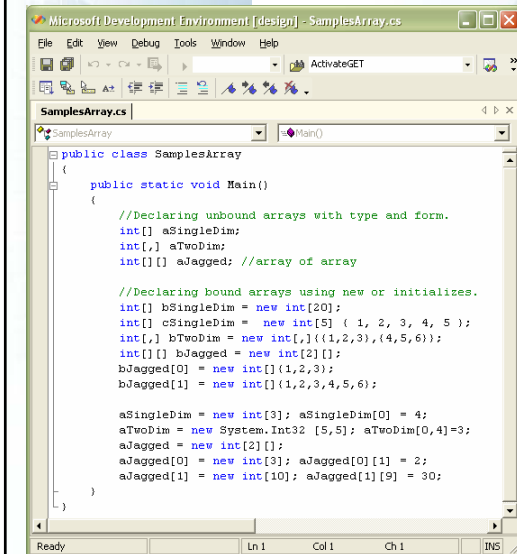
```

Lenguaje: Visual C#

2005 | 36



## Tipos de Datos



```
public class SamplesArray
{
    public static void Main()
    {
        //Declaring unbound arrays with type and form.
        int[] aSingleDim;
        int[,] aTwoDim;
        int[][] aJagged; //array of array

        //Declaring bound arrays using new or initializes.
        int[] bSingleDim = new int[20];
        int[] cSingleDim = new int[5] { 1, 2, 3, 4, 5 };
        int[,] bTwoDim = new int[,]{ {1,2,3}, {4,5,6} };
        int[][] bJagged = new int[2][];
        bJagged[0] = new int[] {1,2,3};
        bJagged[1] = new int[] {1,2,3,4,5,6};

        aSingleDim = new int[3]; aSingleDim[0] = 4;
        aTwoDim = new System.Int32 [5,5]; aTwoDim[0,4]=3;
        aJagged = new int[2][];
        aJagged[0] = new int[3]; aJagged[0][1] = 2;
        aJagged[1] = new int[10]; aJagged[1][9] = 30;
    }
}
```

Los arreglos están basados en la clase `System.Array` de .NET Framework y sus índices comienza en cero.

Lenguaje: Visual C#

2005 | 37

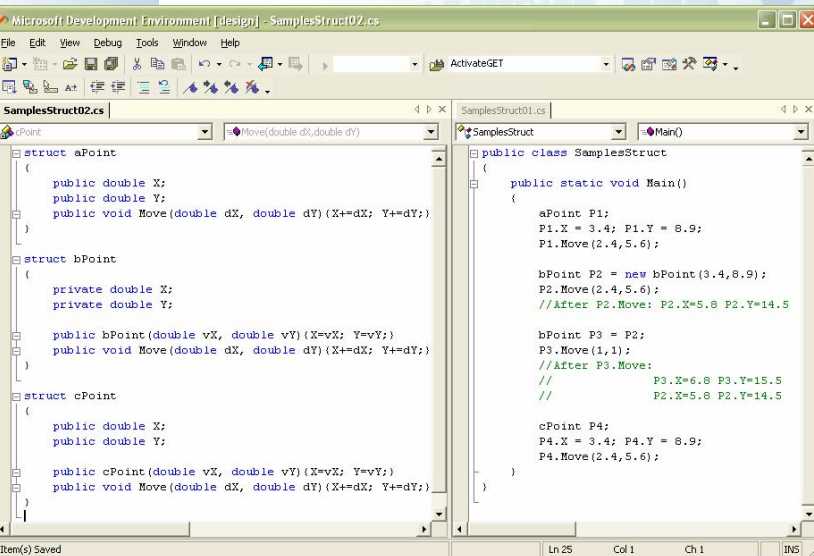
## Tipos de Datos

Las estructuras (structs) permiten agrupar código y datos pero, a diferencia de las clases, no permiten herencia, eventos y siempre son pasadas por valor.

Lenguaje: Visual C#

2005 | 38

# Tipos de Datos



The screenshot displays the Microsoft Development Environment (MDE) interface. The title bar reads "Microsoft Development Environment [design] - SamplesStruct02.cs". The menu bar includes File, Edit, View, Debug, Tools, Window, and Help. The toolbar contains various icons for file operations and development tools. The main workspace is divided into two panes. The left pane, titled "SamplesStruct02.cs", shows the definition of a `Point` struct and a `bPoint` struct. The `Point` struct has public double fields `X` and `Y`, and a public void method `Move(double dX, double dY)`. The `bPoint` struct has private double fields `X` and `Y`, and a public void method `Move(double dX, double dY)`. The right pane, titled "SamplesStruct", shows the definition of a `SamplesStruct` class with a public static void `Main()` method. The `Main` method contains several lines of code that create and manipulate `Point` and `bPoint` objects. The status bar at the bottom shows "Item(s) Saved", "Ln 25", "Col 1", "Ch 1", and "INS".

```
struct aPoint
{
    public double X;
    public double Y;
    public void Move(double dX, double dY) (X+=dX; Y+=dY;)
}

struct bPoint
{
    private double X;
    private double Y;

    public bPoint(double vX, double vY) (X=vX; Y=vY;)
    public void Move(double dX, double dY) (X+=dX; Y+=dY;)
}

struct cPoint
{
    public double X;
    public double Y;

    public cPoint(double vX, double vY) (X=vX; Y=vY;)
    public void Move(double dX, double dY) (X+=dX; Y+=dY;)
}

public class SamplesStruct
{
    public static void Main()
    {
        aPoint P1;
        P1.X = 3.4; P1.Y = 8.9;
        P1.Move(2.4,5.6);

        bPoint P2 = new bPoint(3.4,8.9);
        P2.Move(2.4,5.6);
        //After P2.Move: P2.X=5.8 P2.Y=14.5

        bPoint P3 = P2;
        P3.Move(1,1);
        //After P3.Move:
        //                P3.X=6.8 P3.Y=15.5
        //                P2.X=5.8 P2.Y=14.5

        cPoint P4;
        P4.X = 3.4; P4.Y = 8.9;
        P4.Move(2.4,5.6);
    }
}
```

# Tipos de Datos

Una clase es como una plantilla que describe cómo deben ser las instancias de dicha clase (objetos), de forma que cuando se crea una instancia, ésta tendrá exactamente los mismos métodos y variables que tiene la clase.

Lenguaje: Visual C#

2005 | 40

# Tipos de Datos

```
public class Employee
{
    public string FirstName, LastName;
    private DateTime _BirthDate;

    public Employee() { _BirthDate = DateTime.Now; }
    public Employee(string FirstName)
    { this.FirstName = FirstName; }
    public Employee(string FirstName, string LastName)
    { this.FirstName = FirstName;
      this.LastName = LastName; }
    public Employee(string FirstName, string LastName, DateTime BirthDate)
    { this.FirstName = FirstName;
      this.LastName = LastName;
      _BirthDate = BirthDate; }

    public DateTime BirthDate
    {
        get { return _BirthDate; }
        set { _BirthDate = value; }
    }

    public byte Age
    {
        get { return (byte) (DateTime.Now.Year - _BirthDate.Year); }
    }

    public void ConsolePrint()
    {
        Console.WriteLine("Employee: " + LastName + ", " +
                          Console.WriteLine("Age: " + Age.ToString());
    }
}
```

La sintaxis usada para definir clases en C# es simple  
si usualmente uno ha programado en C++ o Java.

[Attribute] [modifiers] class <className> [:baseClass]  
{  
 [class-body]  
}

[Attribute]: public, internal (limita el alcance al  
assembly contenedor)  
[modifiers]: abstract, sealed (prohibe la herencia)

Item(s) Saved

Ln 17 Col 1 Ch 1

Lenguaje: Visual C#

2005 | 41

# Tipos de Datos

```
public class Employee
{
    public string FirstName, LastName;
    private DateTime _BirthDate;

    public Employee() { _BirthDate = DateTime.Now; }
    public Employee(string FirstName)
    { this.FirstName = FirstName; }
    public Employee(string FirstName, string LastName)
    { this.FirstName = FirstName;
      this.LastName = LastName; }
    public Employee(string FirstName, string LastName, DateTime BirthDate)
    { this.FirstName = FirstName;
      this.LastName = LastName;
      _BirthDate = BirthDate; }

    public DateTime BirthDate
    {
        get { return _BirthDate; }
        set { _BirthDate = value; }
    }

    public byte Age
    {
        get { return (byte) (DateTime.Now.Year - _BirthDate.Year); }
    }

    public void ConsolePrint()
    {
        Console.WriteLine("Employee: " + LastName + ", " + FirstName);
        Console.WriteLine("Age: " + Age.ToString());
    }
}
```

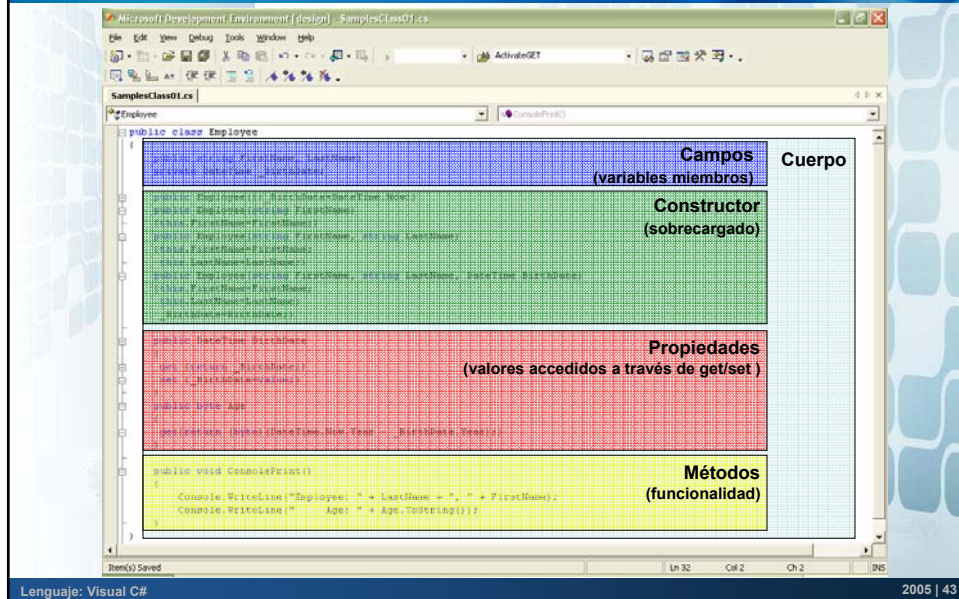
Item(s) Saved

Ln 32 Col 2 Ch 2

Lenguaje: Visual C#

2005 | 42

## Tipos de Datos



## Tipos de Datos

La herencia es un mecanismo que permite que una clase pueda adquirir las cualidades de otra ya existente. C# posee herencia simple y puede implementar más de una Interface (igual que Java).



## Tipos de Datos

La herencia es un mecanismo que permite que una clase pueda adquirir las cualidades de otra ya existente. C# posee herencia simple y puede implementar más de una Interface (igual que Java).

Herencia Multiple de Clases como C++



Auto



Anfibio



Lancha

Lenguaje: Visual C#

2005 | 45

## Tipos de Datos

```
Microsoft Development Environment [design] - SamplesInheritance01.cs
File Edit View Debug Tools Window Help
SamplesInheritance01.cs
SalariedEmployee

abstract class Person
{
    protected string _FirstName, _LastName;
}

class Employee : Person
{
    private float _Salary=0;
    public float Salary
    {
        get{return _Salary;}
        set{_Salary=value;}
    }
}

class ContractEmployee : Employee
{
}

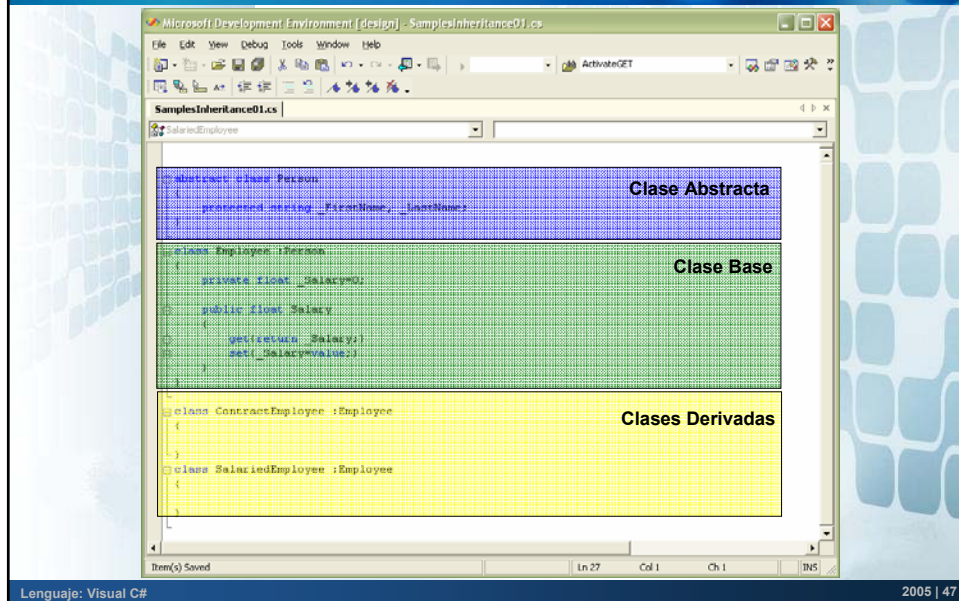
class SalariedEmployee : Employee
{
}
```

Lenguaje: Visual C#

2005 | 46



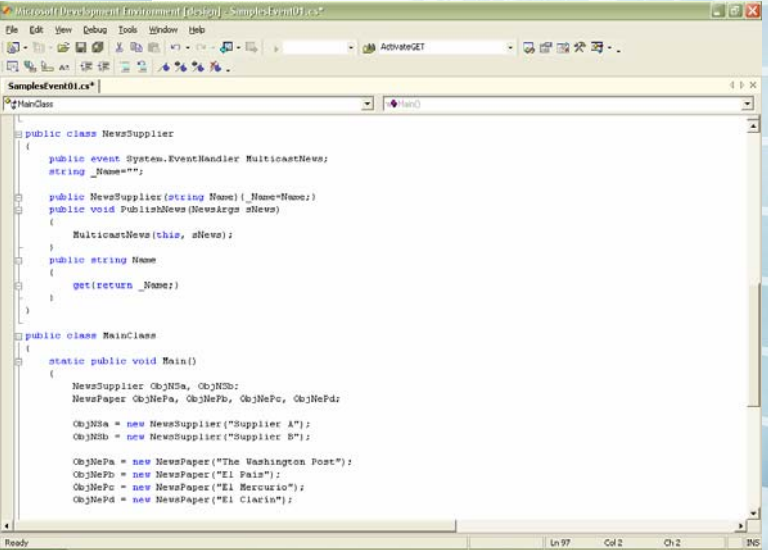
## Tipos de Datos



## Evento

Un evento (event) proporciona a una clase una manera de enviar notificaciones a los clientes de esa clase cuando cierta cosa interesante le sucede a una instancia de la clase (objeto).

## Demo



```
Microsoft Development Environment [Design] - SampleEvent01.cs*
File Edit View Debug Tools Window Help
Solution Explorer
SampleEvent01.cs*
MainClass
public class NewsSupplier
{
    public event System.EventHandler MulticastNews;
    string _Name="";

    public NewsSupplier(string Name) { _Name=Name; }
    public void PublishNews(NewEventArgs eNews)
    {
        MulticastNews(this, eNews);
    }
    public string Name
    {
        get{return _Name;}
    }
}

public class MainClass
{
    static public void Main()
    {
        NewsSupplier ObjNSa, ObjNSb;
        Newspaper ObjNePa, ObjNePb, ObjNePc, ObjNePd;

        ObjNSa = new NewsSupplier("Supplier A");
        ObjNSb = new NewsSupplier("Supplier B");

        ObjNePa = new Newspaper("The Washington Post");
        ObjNePb = new Newspaper("El Pais");
        ObjNePc = new Newspaper("El Mercurio");
        ObjNePd = new Newspaper("El Ciacin");
    }
}
```

Lenguaje: Visual C#

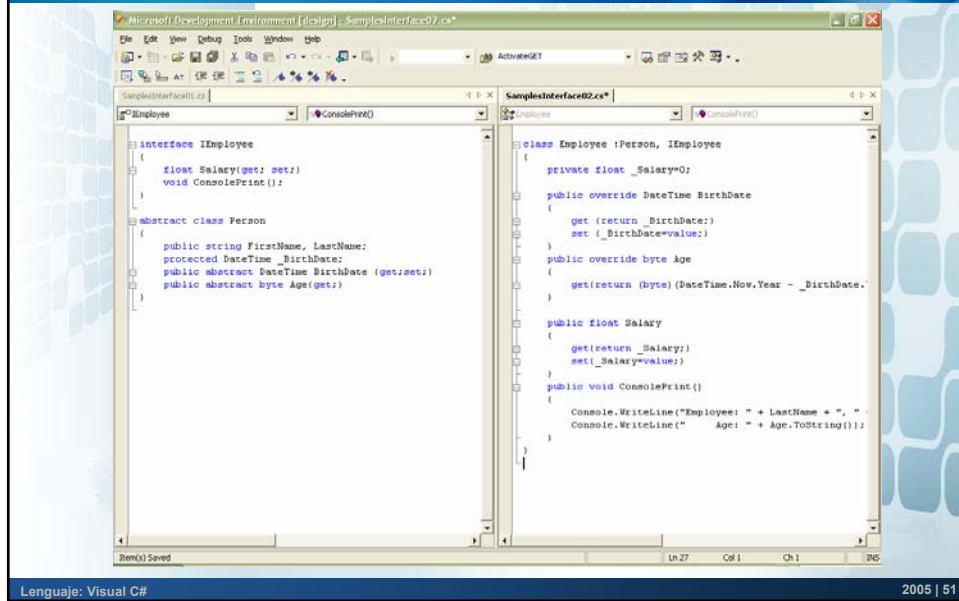
2005 | 49

## Interfaces

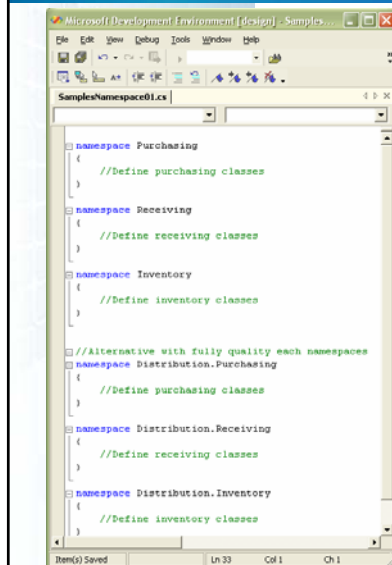
Una interface es un contrato entre dos piezas de código. Permite definir métodos y propiedades “abstractas”, ya que no requieren definir código de ejecución interno.

Una interface siempre es implementada por una clase.

# Interfaces



# Namespaces



Forma de agrupar LÓGICAMENTE clases. Un namespace puede contener a clases y a otros namespace.

Referenciados con using

# Operadores

Category of Operator	Operators
Primary	new, typeof, sizeof
Unary	+, -, !, ~, ++X, --X
Multiplicative	*, /, %
Additive	+, -
Shift	<<, >>
Relational	<, >, <=, >=, is (is used to check whether the run-time type of an object is compatible with a given type)
Equality	==
Logical AND	&
Logical XOR	^
Logical OR	
Conditional AND	&&
Conditional OR	
Conditional	?:
Assignment	=, *=, /=, %=, -=, <<=, >>=, &=, ^=,  =

Lenguaje: Visual C#

2005 | 53

# Operadores

Category of Operator	Operators
Primary	new, typeof, sizeof
Unary	+, -, !, ~, ++X, --X <b>Admiten sobrecarga</b>
Multiplicative	*, /, %
Additive	+, -
Shift	<<, >>
Relational	<, >, <=, >=, is (is used to check whether the run-time type of an object is compatible with a given type)
Equality	==
Logical AND	&
Logical XOR	^
Logical OR	
Conditional AND	&&
Conditional OR	
Conditional	?:
Assignment	=, *=, /=, %=, -=, <<=, >>=, &=, ^=,  =

Lenguaje: Visual C#

2005 | 54



# Control del Flujo del Programa

Statements			
Selection Statements	<pre>if (expression)     statement1 [else     statement2]</pre>	<pre>if (expression) {     statement1     statement2 }</pre>	<pre>switch (switch_expression) {     case constant-expression:         statement         jump-statement     case constant-expressionN:         statementN     [default]}</pre>
	<pre>while (Boolean-expression)     embedded-statement  do     embedded-statement while(Boolean-expression)</pre>	<pre>for (initialization; Boolean-expression; step)     embedded-statement</pre>	<pre>foreach (type in expression)     embedded-statement</pre>
Branching with Jump Statements	<pre>break; continue; goto;</pre>		

Lenguaje: Visual C#

2005 | 55

# Manejo de Excepciones

Los programas deben poder controlar los errores que se producen durante la ejecución de manera uniforme.

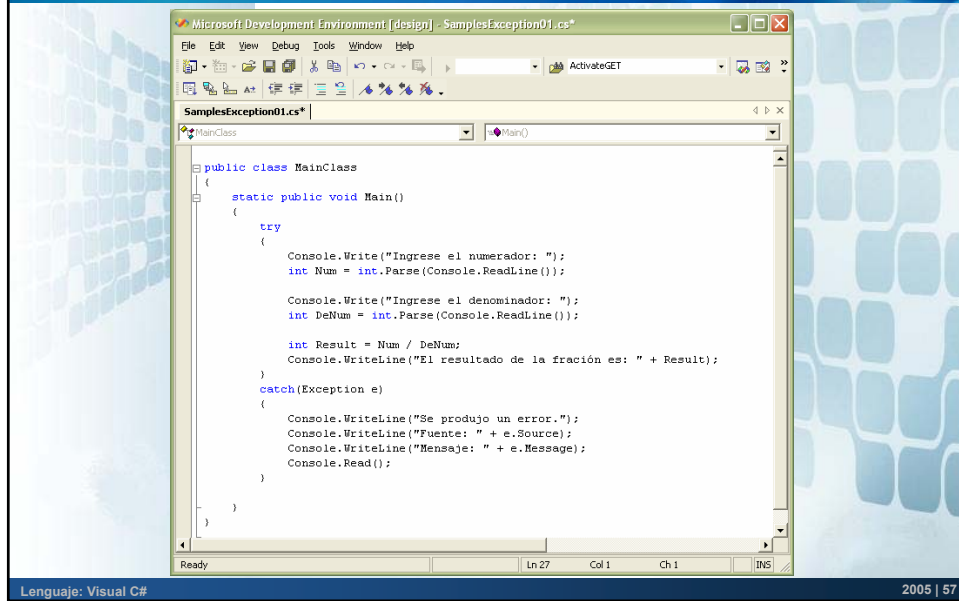
Todas las operaciones de .NET Framework informan de un error iniciando excepciones y son controladas mediante las sentencias Try/Catch/Finally

Lenguaje: Visual C#

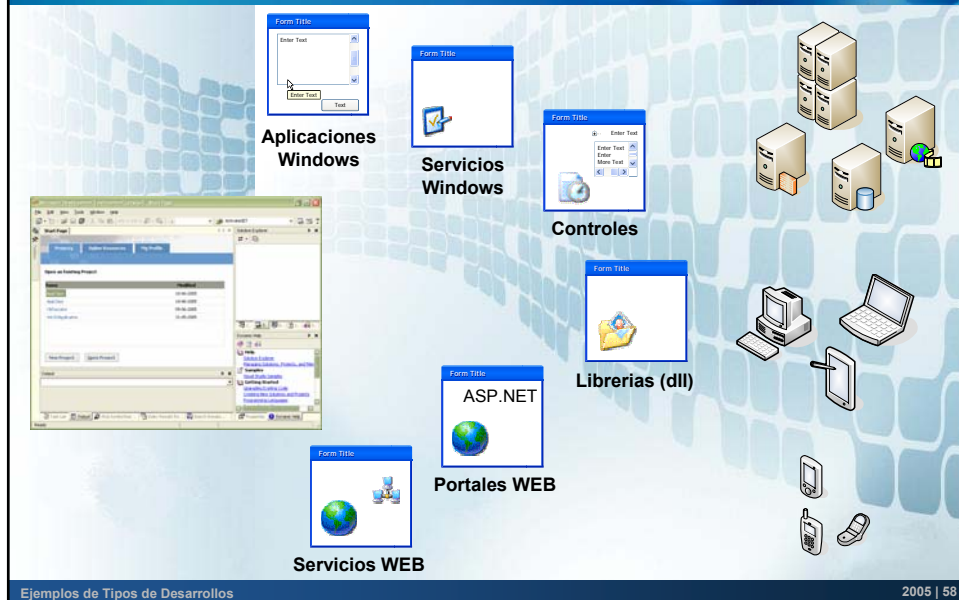
2005 | 56



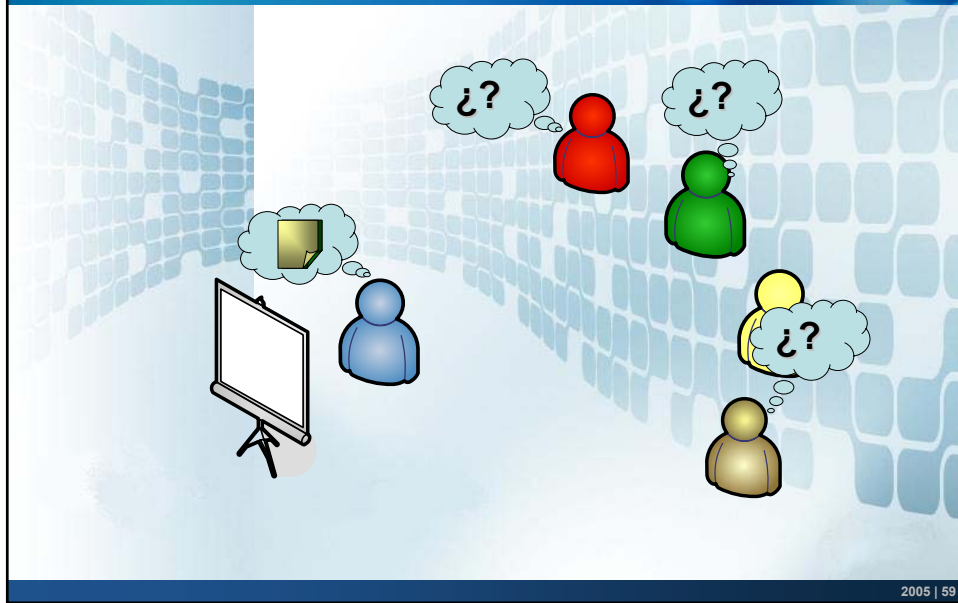
# Manejo de Excepciones



# ¿Qué se puede desarrollar?



## Sesión de Preguntas



2005 | 59