



Taller de Programación Visual III

Daniel Alejandro Nuño
Ramirez

Presentación

- Daniel Alejandro Nuño Ramirez
- Ingeniero en Computación, Universidad de Guadalajara
- Maestro en Informática Aplicada*, ITESO
- Gerente de Desarrollo de Software, Oracle
- TCS, Intel, Bank of America, Flextronics, Estratel, UdeG
- Correr, Cine, Series, Música
- Email: daniel_nuno@hotmail.com

Contenido

Unidad I, Introducción a Microsoft Visual Studio .NET

Unidad II, Introducción a Microsoft Visual C#

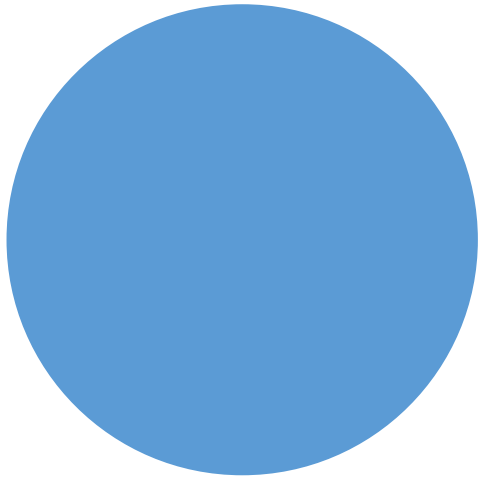
Unidad III, C# Avanzado

Unidad IV, Exploración de la biblioteca C#

Unidad V, Programación multiproceso

Unidad VI, Creación de aplicaciones Windows

Unidad VII, El modelo de acceso a datos ADO.NET



Unidad I, Introducción Microsoft Visual Studio .NET



Unidad I, Introducción Microsoft Visual studio .NET

1. El entorno de desarrollo de Visual Studio .NET
2. Tipos de proyectos en Visual Studio .NET
3. Elementos esenciales de la plataforma .NET
4. Componentes y objetos de la plataforma .NET
5. Comprensión del modelo de desarrollo .NET
6. La estructura básica de .NET Framework.

Unidad I, Introducción Microsoft Visual Studio .NET

- C# vs .NET
 - C# es un lenguaje de programación
 - .NET es un plataforma para construir aplicaciones con Windows
 - No se limita a C#, se puede usar F# o VB.NET
- .NET se conforma por
 - CLR (Common Language Runtime)
 - Class Library

Unidad I, Introducción Microsoft Visual Studio .NET

- Visual Studio (<https://visualstudio.microsoft.com/>)
 - IDE (Integrated Development Environment)
 - Editions (Tamaño, Funcionalidad, Costo)
 - Ultimate (\$)
 - Premium (\$)
 - Professional (\$, equipos pequeños)
 - Test Professional (\$)
 - Express (libre, Web, C#, VB, etc, no se puede utilizar extensiones)
- ➡ Community (libre, 2014, estudiantes, individuos, algunas restricciones)

Visual Studio Enterprise 2017

Visual Studio Professional 2017

Visual Studio Community 2017

Unidad I, Introducción Microsoft Visual Studio .NET

- Visual Studio Community (libre, 2014, estudiantes, individuos, algunas restricciones)
 - Web, Windows, Desktop y aplicaciones móviles
- Profesional con MSDN
 - Incluye todas las características para crear aplicaciones de todo tipo con todos los lenguajes de programación disponibles en la plataforma .NET (VB, C#, F#, TypeScript, C++)
- Enterprise con MSDN
 - Antes conocido como Ultimate, incluye las características centrales y herramientas avanzadas para crear aplicaciones y evitar “no se puede reproducir”.

Unidad I, Introducción Microsoft Visual Studio .NET

- Test Professional
 - Herramienta periférica desarrollada para “Testers”
- Visual Studio Core
 - Gratuita, Open Source, orientada para desarrollos Web sobre sistema operativos: Windows, Mac y Linux
 - <https://code.visualstudio.com/>
- Comparativo
 - <https://visualstudio.microsoft.com/vs/compare/>

Unidad I, Introducción Microsoft Visual Studio .NET

- Visual Studio Community (libre, 2014, estudiantes, individuos, algunas restricciones)
 - Web, Windows, Desktop y aplicaciones móviles
 - Estudiantes, equipos pequeños de desarrollo, contribuyentes al open source.
 - No incluye: SharePoint, Office, LightSwitch, y Cloud Business Apps.
 - <https://visualstudio.microsoft.com/es/vs/community/>

Unidad I, Introducción Microsoft Visual Studio .NET

- MSDN
 - Suscripción incluida con la versión Visual Studio Enterprise (\$150 USD) o Profesional (\$50 USD)
 - Permite acceso a TFS o Visual Studio online (VSO), así como también, acceso a: SharePoint, Exchange, Office, Dynamics, BizTalk, LightSwitch, y Cloud Business Apps.
 - <https://www.visualstudio.com/products/visual-studio-with-msdn-overview-vs>

Unidad I, Introducción Microsoft Visual Studio .NET

- TFS
 - Application Lifecycle Management (ALM) Environment
 - Gestionar y dar seguimiento a trabajo
 - Dos versiones:
 - On-premise-hosted TFS
 - Version online, llamada Visual Studio Online (VSO)
 - <https://www.visualstudio.com/en-us/products/what-is-visual-studio-online-vs.aspx>

Unidad I, Introducción Microsoft Visual Studio .NET

- Lenguajes de programación soportados en la plataforma .NET
 - C#, rapid application development,
 - Visual Basic .NET (VB.NET), productividad
 - C++, C background, Active Template Library (ATL), Microsoft Foundation Class (MFC) libraries, C Runtime (CRT) Library.
 - Visual F#, matemáticas, científico, ingeniería, y análisis (Machine Learning)
 - TypeScript, respuesta a JavaScript.

Unidad I, Introducción Microsoft Visual Studio .NET

Actividad:

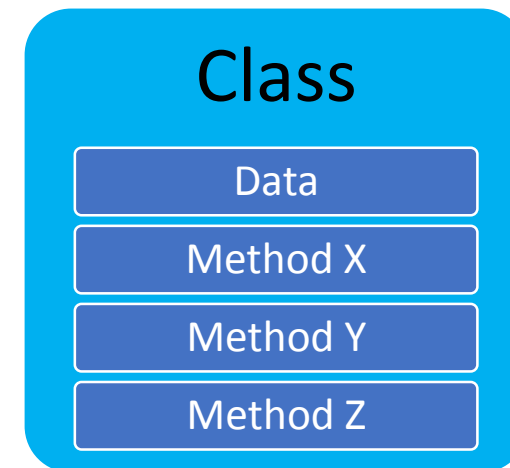
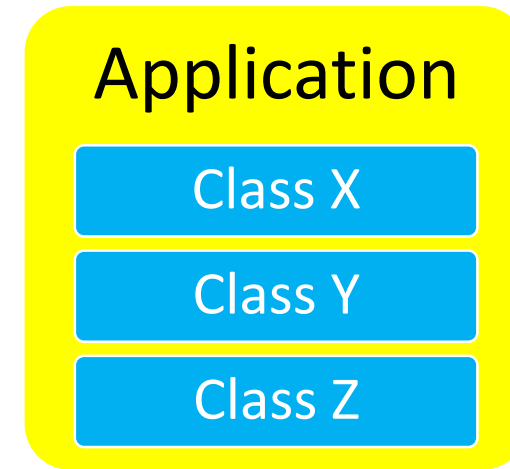
Instalación de Microsoft Visual Studio

Unidad I, Introducción Microsoft Visual Studio .NET

- CLR (Common Language Runtime)
 - Misma idea que la Java Virtual Machine (JVM)
 - Intermediate Language Code (IL Code)
 - CLR es una aplicación que traduce de IL código a código Nativo
 - Just In Time Compilation (JIT)

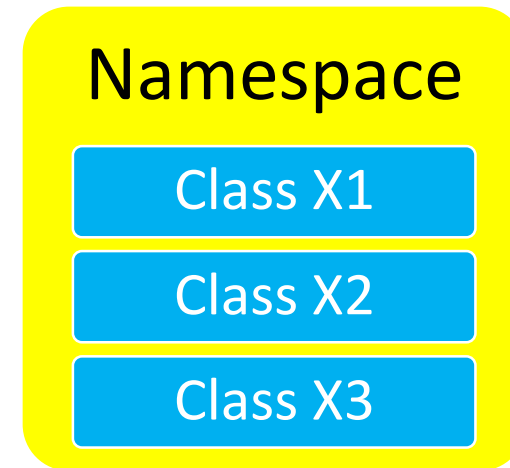
Unidad I, Introducción Microsoft Visual Studio .NET

- Arquitectura de las aplicaciones .NET
- Aplicaciones se forman a base de clases y su interacción mediante métodos



Unidad I, Introducción Microsoft Visual Studio .NET

- Namespaces
 - Colección de clases relacionadas
 - Para acceder a una clase de un namespace, se utiliza la siguiente nomenclatura:
 - `NamespaceName.ClassName.Method`
 - `using (Directive)`
 - Algunos Namespaces:
 - Databases
 - Graphics
 - Security



Unidad I, Introducción Microsoft Visual Studio .NET

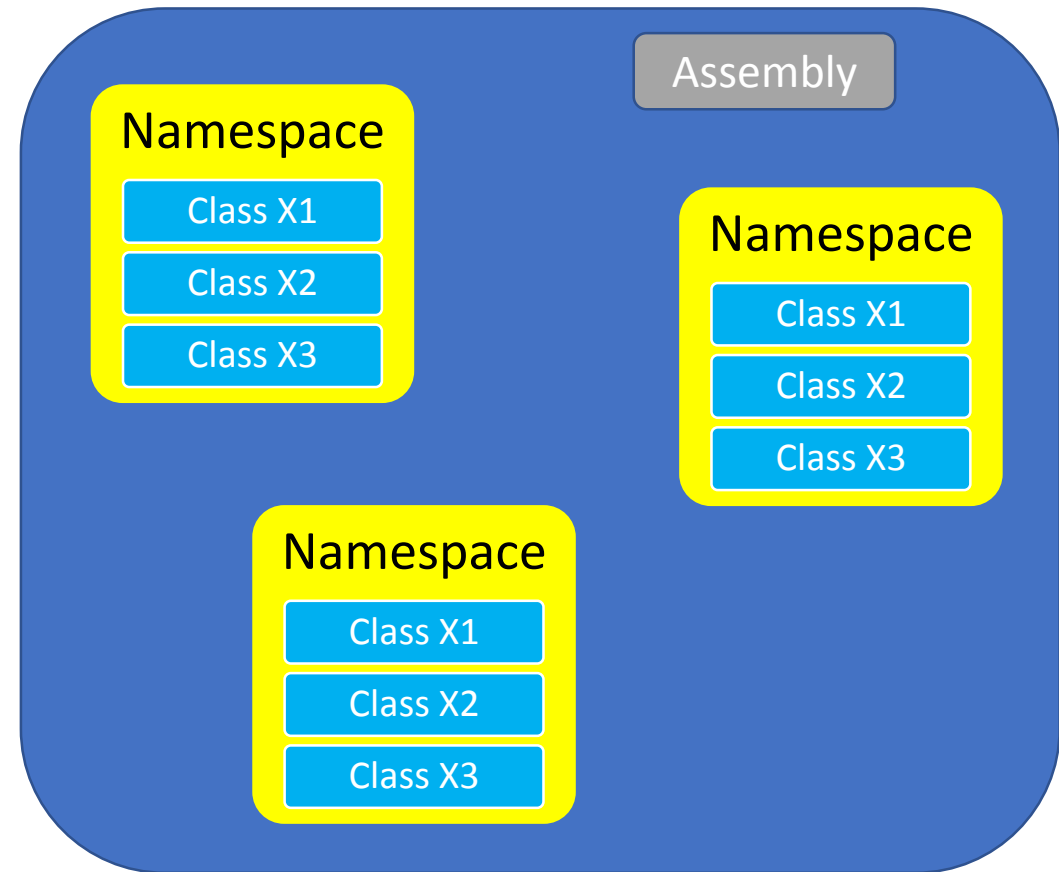
- El namespace global, es el namespace raíz (root)
- global::System se refiere al namespace de la plataforma .NET (System)

Ejemplo:

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

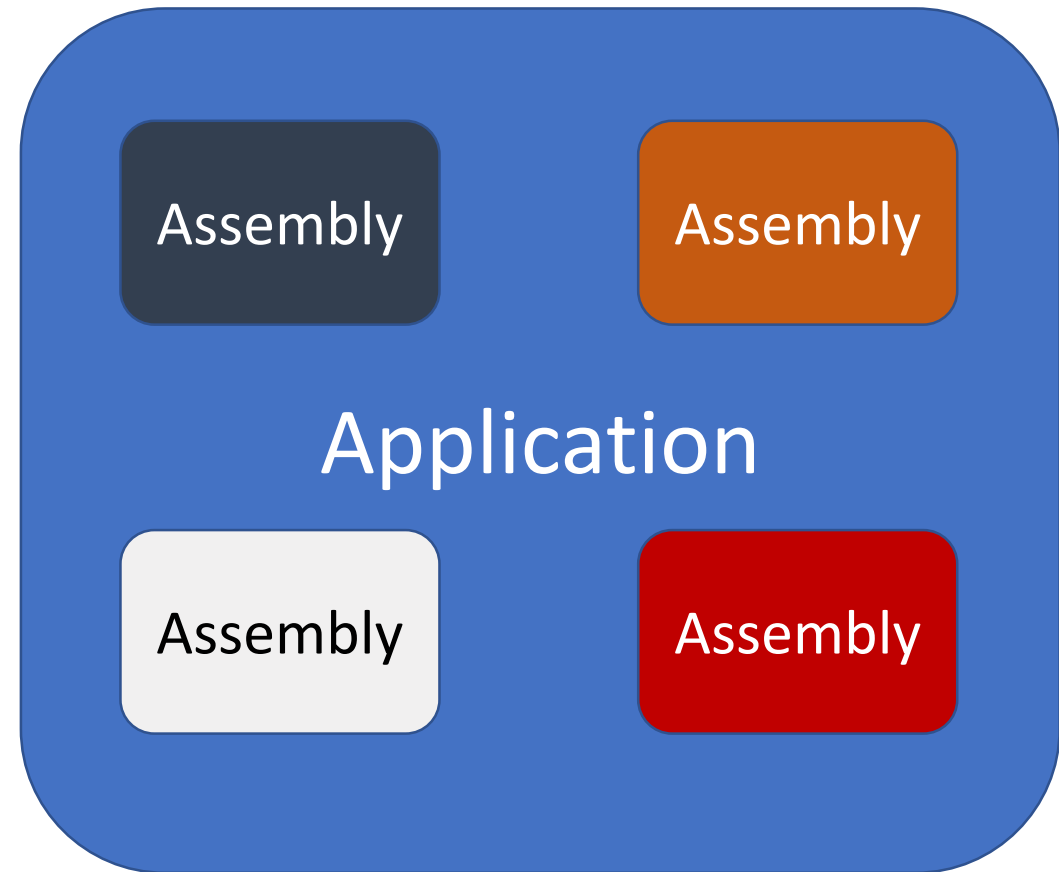
Unidad I, Introducción Microsoft Visual Studio .NET

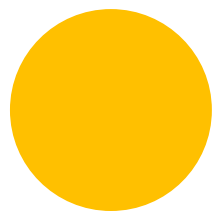
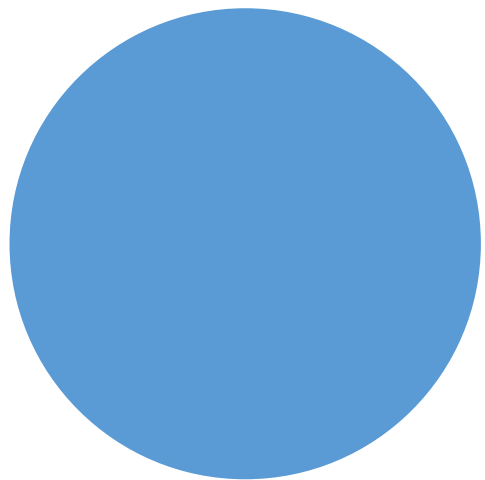
- Assembly (DLL o EXE)
 - Contenedor de namespaces relacionados
 - Puede ser en formato
 - DLL (Dynamic-link Library)
 - EXE (Ejecutable)




Unidad I, Introducción Microsoft Visual Studio .NET

- Application
 - Assembly
 - Namespace
 - Class
 - Methods
 - Assembly
 - Namespace
 - Class
 - Methods





Unidad II, Introducción a Microsoft Visual C#



Unidad II, Introducción a Microsoft Visual C#

- 2.1 Breve descripción del C#.
- 2.2 Tipos de datos básicos.
- 2.3 La biblioteca de clases.
- 2.4 Operadores.
- 2.5 Instrucciones de control de programa.
- 2.6 Clases, objetos y métodos en C#.
- 2.7 Arreglos y Matrices, Cadenas.
- 2.8 Sobrecarga de operadores.
- 2.9 Indizadores y propiedades.
- 2.10 Herencia en C#.
- 2.11 Interfaces, estructuras y enumeraciones.
- 2.12 Colecciones



Unidad II, Introducción a Microsoft Visual C#

- 2.1 Breve descripción del C#
- C# historia
 - C# se pronuncia “C-Sharp”
 - Lenguaje de programación orientado a objetos
 - Distribuido por Microsoft y se ejecuta sobre la plataforma .NET
 - Anders Hejlsberg es su fundador
 - Basado en C++ y Java
 - Se introdujo en 2002

Unidad II, Introducción a Microsoft Visual C#

2.1 Breve descripción del C#

- El porque estudiar C#
 - Combina las mejores características de otros lenguajes de programación ya existentes tales como: C++, Pascal, Java, Visual Basic
 - Sencillo
 - Lenguaje de programación Moderno
 - Orientado a objetos
 - Seguro
 - Tiene una basta biblioteca
 - Rápida ejecución

Unidad II, Introducción a Microsoft Visual C#

2.1 Breve descripción del C#

- C# se ha convertido en un standard, lo que permite que se puedan hacer otras implementaciones además de Microsoft C#.
 - <http://www.ecma-international.org>
- C# se puede utilizar además en plataformas no Windows a través de Mono Project y .NET core que son manejadas por la fundación .NET
 - <http://www.dotnetfoundation.org/>

Unidad II, Introducción a Microsoft Visual C#

- Algunas capacidades incluidas en su biblioteca:
 - Bases de datos
 - Debugging
 - Web Apps
 - Graphics
 - Input/Output
 - Networking
 - Permissions
- Continuación:
 - Mobile
 - Procesamiento de “Strings”
 - Multithreading
 - File Processing
 - Security
 - Graphical User Interface
 - Data Structures

Unidad II, Introducción a Microsoft Visual C#

| C# Version History | | |
|--------------------|-----------------------------------------------------------------------------------------------------|-----------------|
| Version | Features | Year of release |
| C# 1.0 | Basic Features | 2002 |
| C# 2.0 | Generics, Partial types, Anonymous methods, Nullable types, Static classes | 2005 |
| C# 3.0 | Var, LINQ, Lambda expression, Auto-implemented properties, Anonymous types, Extension methods | 2007 |
| C# 4.0 | Dynamic binding, Named and Optional arguments | 2010 |
| C# 5.0 | Asynchronous methods, Caller info attributes | 2012 |
| C# 6.0 | Auto-property initializers, Null-propagating operator, Exception filters, Using static members, ... | 2015 |

Unidad II, Introducción a Microsoft Visual C#

2.2 Tipos de datos básicos

- Primitive Types and Expressions
 - Variables y Constantes
 - Conversión entre tipos

Unidad II, Introducción a Microsoft Visual C#

| | C# Type | .NET Type | Bytes | Range |
|------------------|----------------|-----------|-------|-----------------------------------------------|
| Integral Numbers | byte | Byte | 1 | 0 to 255 |
| | short | Int16 | 2 | -32,768 to 32,767 |
| | int | Int32 | 4 | -2.1B to 2.1B |
| Real Numbers | long | Int64 | 8 | ... |
| | float | Single | 4 | -3.4×10^{38} to 3.4×10^{38} |
| | double | Double | 8 | ... |
| | decimal | Decimal | 16 | -7.9×10^{28} to 7.9×10^{28} |
| Character | char | Char | 2 | Unicode Characters |
| Boolean | bool | Boolean | 1 | True / False |

Unidad II, Introducción a Microsoft Visual C#

- Variables

- Variable, localidad de memoria usada para almacenar un valor que puede ser alterado en el tiempo.
- Tipos:
 - Decimal
 - Boolean
 - Integral: int, char, byte, short, long
 - Floating point: float, double
 - Nullable

- Ejemplo:

```
int number;  
int Number = 1;  
double d;  
float f;  
char ch;
```

Unidad II, Introducción a Microsoft Visual C#

- Constantes

- Valor constante asignado a un identificador.
- No cambia en el futuro.
- Declarado con la palabra reservada “const”.

- Ejemplo:

```
const float Pi = 3.14f;
```

Unidad II, Introducción a Microsoft Visual C#

- Consideraciones
 - En C#, no se hace una verificación de “overflowing”;
 - Al declarar un número flotante, se debe especificar que el formato del valor es flotante también, de lo contrario se trataría de usar “double”.

Unidad II, Introducción a Microsoft Visual C#

- Conversiones

- Conversión Implícita:

- `byte b = 1;`
 - `int i = b;`
 - `float f = i;`
 - `byte c = i;`

- Cont.

- Conversión Explícita (casting)

- `byte c = (byte) i;`
 - `int j = (int) f;`

Unidad II, Introducción a Microsoft Visual C#

- Conversiones

- Conversión entre no compatibles:
 - `string s = "1";`
 - `int i = (int) s; // won't compile`
- `int i = Convert.ToInt32(s);`
- `int j = int.Parse(s);`

- Cont.

- `Byte()`
- `ToInt16()`
- `ToInt32()`
- `ToInt64()`

UNIDAD I, introducción a la programación visual

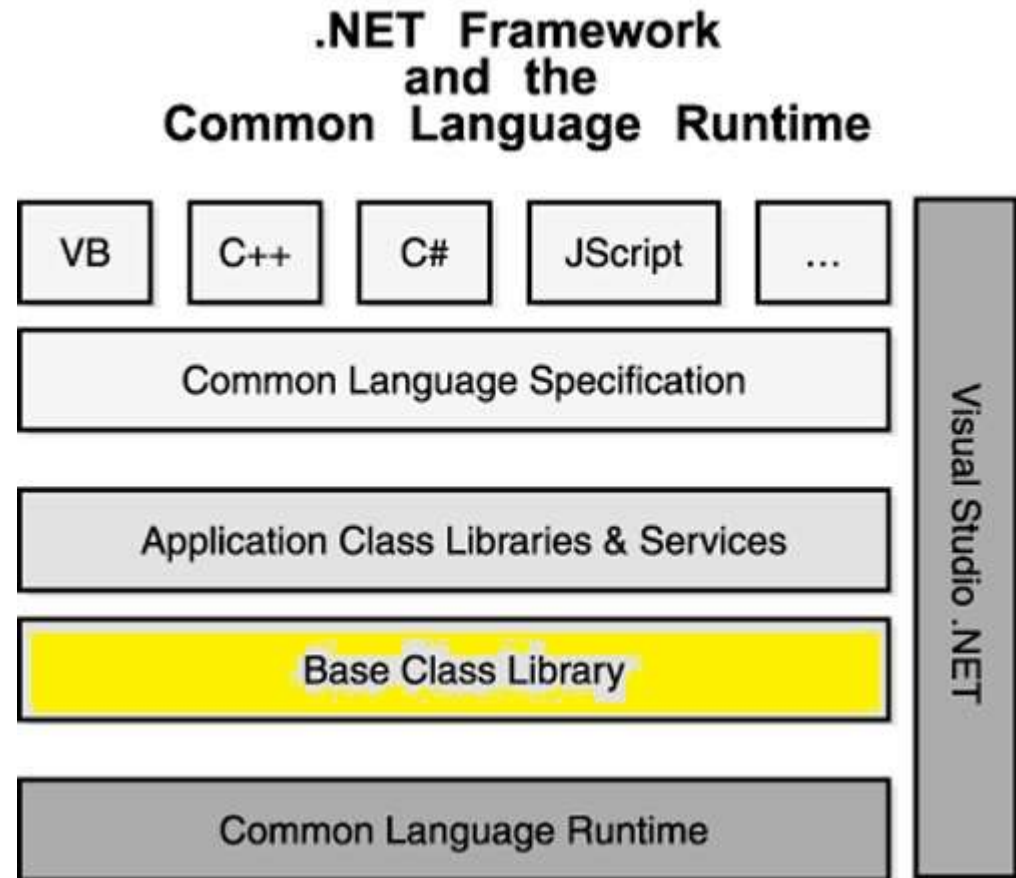
TAREA # 1 Tipos de datos utilizados en C#:

1. Realizar un programa que muestre los diferentes tipos de datos utilizados en C#, utilizar variables y constantes como parte del programa y realizar conversiones entre los mismos.
2. Subir el programa a github o bitbucket y enviar por email (daniel_nuno@hotmail.com) la liga al mismo.

Unidad II, Introducción a Microsoft Visual C#

2.3 La biblioteca de clases

Base Class Library (BCL)



Unidad II, Introducción a Microsoft Visual C#

2.3 La biblioteca de clases

La BCL está constituida por espacios de nombres (**namespaces**). Cada espacio de nombres contiene tipos que se pueden utilizar en el programa: clases, estructuras, enumeraciones, delegados e interfaces.

Cuando se crea un proyecto de Visual C# en Visual Studio, se sigue haciendo referencia a las DLL más comunes de la clase base, pero, si necesita usar un tipo incluido en una DLL a la que aún no se hace referencia, deberá agregar la referencia de esa DLL.

La plataforma .NET incluye una colección de clases bien organizada cuya parte independiente del sistema operativo ha sido propuesta para su estandarización.

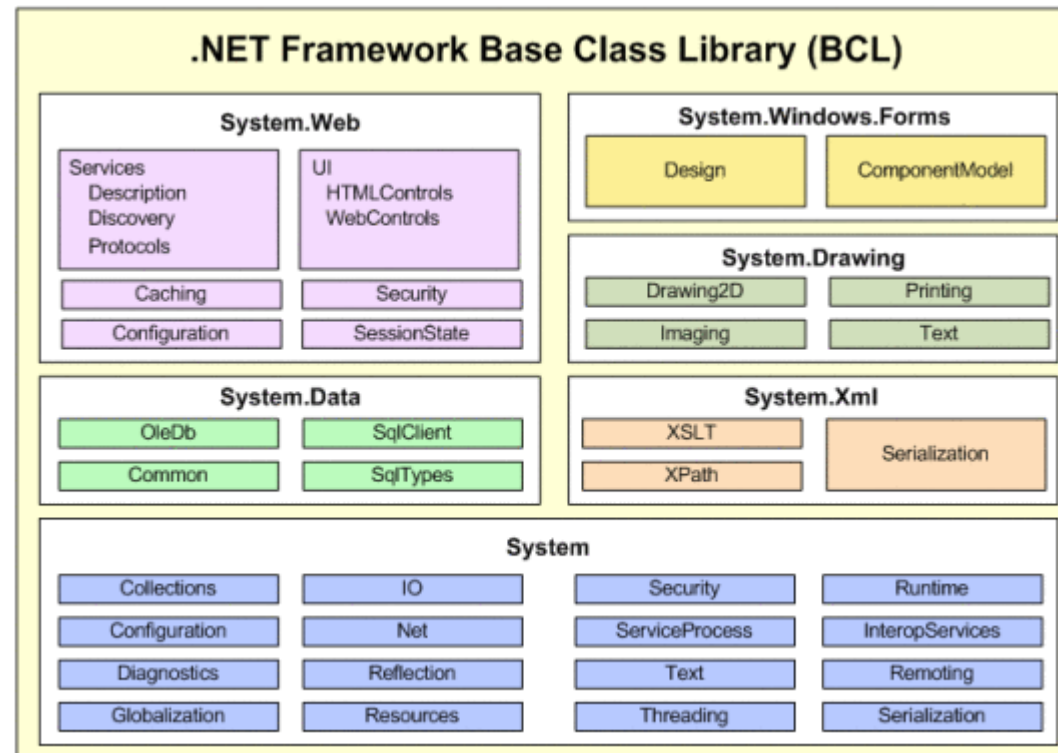
La BCL integra todas las tecnologías Windows en un marco único para todos los lenguajes de programación (Windows Forms, GDI+, Web Forms, Web Services, impresión, redes...).

La BCL proporciona un modelo orientado a objetos que sustituye a los componentes COM.

Unidad II, Introducción a Microsoft Visual C#

- System namespace
 - Namespace raíz para tipos de datos de .NET
 - Contiene clases que representan los tipos de datos base: Object, Byte, Char, Array, Int32, String
 - Contiene mas de 100 clases que van de un rango para manejo de excepciones y hasta ejecución.
 - <https://docs.microsoft.com/en-us/dotnet/api/system>
 - Además, contiene namespaces de segundo nivel.
 - <https://docs.microsoft.com/dotnet/api>

Unidad II, Introducción a Microsoft Visual C#



Unidad II, Introducción a Microsoft Visual C#

2.4 Operadores

- Operadores
 - Aritméticos
 - Comparación
 - Asignación
 - Lógicos
 - “Bitwise”
 - Misceláneos

Unidad II, Introducción a Microsoft Visual C#

- Operadores Aritméticos

| Significado | Operador | Ejemplo |
|-------------|----------|----------|
| Add | + | $a + b$ |
| Subtract | - | $a - b$ |
| Multiply | * | $a * b$ |
| Divide | / | a / b |
| Reminder | % | $a \% b$ |

Unidad II, Introducción a Microsoft Visual C#

- Operadores Aritméticos (unary)

| Significado | Operador | Ejemplo | Equivalencia |
|--------------------|----------|---------|---------------------------------------------|
| Incrementar (post) | ++ | a++ | int a = 1; int b = a++; a = 2, b = 1; |
| Decrementar (post) | -- | a-- | int a = 1; int b = a--; a = 0, b = 1; |
| Incrementar (pre) | ++ | ++a | |
| Decrementar (pre) | -- | --a | |

Unidad II, Introducción a Microsoft Visual C#

- Operadores de Comparación

| Comparación | Operador | Ejemplo |
|-----------------------|----------|---------|
| Equal | == | a==b |
| Not Equal | != | a != b |
| Greater Than | > | a > b |
| Greater or equal to | >= | a >= b |
| Less than | < | a < b |
| Less than or equal to | <= | a <= b |

Unidad II, Introducción a Microsoft Visual C#

- Operadores de Asignación

| Significado | Operador | Ejemplo | Same as |
|---------------------------|----------|---------|-----------|
| Assignment | = | a=1 | |
| Addition assignment | += | a+=3 | a=a + 3 |
| Subtraction assignment | -= | a-=3 | a=a - 3 |
| Multiplication assignment | *= | a *= 3 | a = a * 3 |
| Division assignment | /= | a /= 3 | a = a / 3 |

Unidad II, Introducción a Microsoft Visual C#

- Operadores Lógicos

| Significado | Operador | Ejemplo |
|-------------|----------|---------|
| And | && | a && b |
| Or | | a b |
| Not | ! | a != b |

Unidad II, Introducción a Microsoft Visual C#

- Operadores “Bitwise”

| Operador | Descripción | Ejemplo |
|----------|---------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| & | Binary AND Operator copies a bit to the result if it exists in both operands. | (A & B) = 12, which is 0000 1100 |
| | Binary OR Operator copies a bit if it exists in either operand. | (A B) = 61, which is 0011 1101 |
| ^ | Binary XOR Operator copies the bit if it is set in one operand but not both. | (A ^ B) = 49, which is 0011 0001 |
| ~ | Binary Ones Complement Operator is unary and has the effect of 'flipping' bits. | (~A) = 61, which is 1100 0011 in 2's complement due to a signed binary number. |
| << | Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand. | A << 2 = 240, which is 1111 0000 |
| >> | Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand. | A >> 2 = 15, which is 0000 1111 |

Unidad II, Introducción a Microsoft Visual C#

- Operadores Misceláneos

| Operador | Descripción | Ejemplo |
|----------|------------------------------------------------------|----------------------------------------------------------------------------------|
| sizeof() | Returns the size of a data type. | sizeof(int), returns 4. |
| typeof() | Returns the type of a class. | typeof(StreamReader); |
| & | Returns the address of an variable. | &a; returns actual address of the variable. |
| * | Pointer to a variable. | *a; creates pointer named 'a' to a variable. |
| ? : | Conditional Expression | If Condition is true ? Then value X : Otherwise value Y |
| is | Determines whether an object is of a certain type. | If(Ford is Car) // checks if Ford is an object of the Car class. |
| as | Cast without raising an exception if the cast fails. | Object obj = new StreamReader("Hello"); StreamReader r = obj as StreamReader; |

Unidad II, Introducción a Microsoft Visual C#

- Precedencia y asociación de operadores

| Category (By Precedence) | Operator(s) | Associativity |
|--------------------------|--------------------------------------|---------------|
| Unary | + - ! ~ ++ -- (type)* & sizeof | Right to Left |
| Additive | + - | Left to Right |
| Multiplicative | % / * | Left to Right |
| Relational | < > <= >= | Left to Right |
| Shift | << >> | Left to Right |
| Equality | == != | Right to Left |
| Logical AND | & | Left to Right |
| Logical OR | | Left to Right |
| Logical XOR | ^ | Left to Right |
| Conditional OR | | Left to Right |
| Conditional AND | && | Left to Right |
| Null Coalescing | ?? | Left to Right |
| Ternary | ?: | Right to Left |
| Assignment | = *= /= %= += -= <<= >>= &= ^= = => | Right to Left |

UNIDAD I, introducción a la programación visual

TAREA # 2 Operadores en C#:

1. Realizar un programa que muestre los diferentes tipos de operadores utilizados en C#.
2. Subir el programa a github o bitbucket y enviar por email (daniel_nuno@hotmail.com) la liga al mismo.

Unidad II, Introducción a Microsoft Visual C#

2.5 Instrucciones de control de programa

- Control de Flujo

- Condicionales

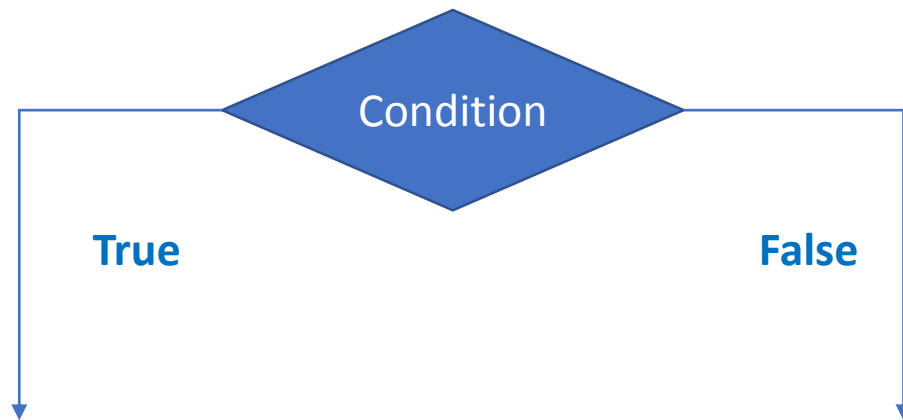
- If/else
 - switch/case
 - Operator a ? b : c

- Iteración

- Ciclo “for”
 - Ciclo “foreach”
 - Ciclo “while”
 - Ciclo “do-while”

Unidad II, Introducción a Microsoft Visual C#

- Control de Flujo
 - Condicionales
 - if/else



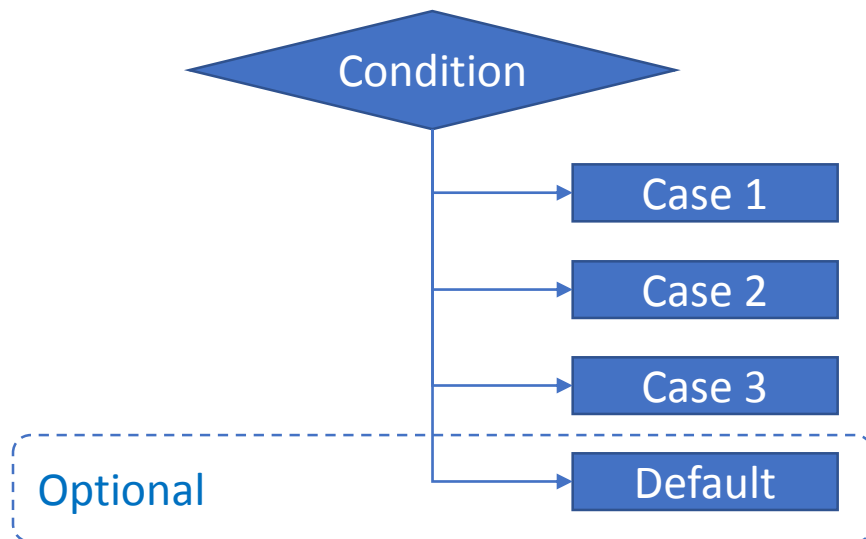
```
if(boolean_expression) {  
    /* statement(s) will execute if the boolean expression  
    is true */  
}
```

```
if(boolean_expression) {  
    /* statement(s) will execute if the boolean expression  
    is true */  
} else {  
    /* statement(s) will execute if the boolean expression  
    is false */  
}
```

```
if( boolean_expression 1) {  
    /* Executes when the boolean expression 1 is true */  
    if(boolean_expression 2) {  
        /* Executes when the boolean expression 2 is true */  
    }  
}
```

Unidad II, Introducción a Microsoft Visual C#

- Control de Flujo
 - Condicionales
 - switch/case



```
switch(expression) {  
    case constant-expression :  
        statement(s);  
        break; /* optional */  
    case constant-expression :  
        statement(s);  
        break; /* optional */
```

```
    /* you can have any number of case  
statements */  
    default : /* optional */  
        statement(s);  
}
```

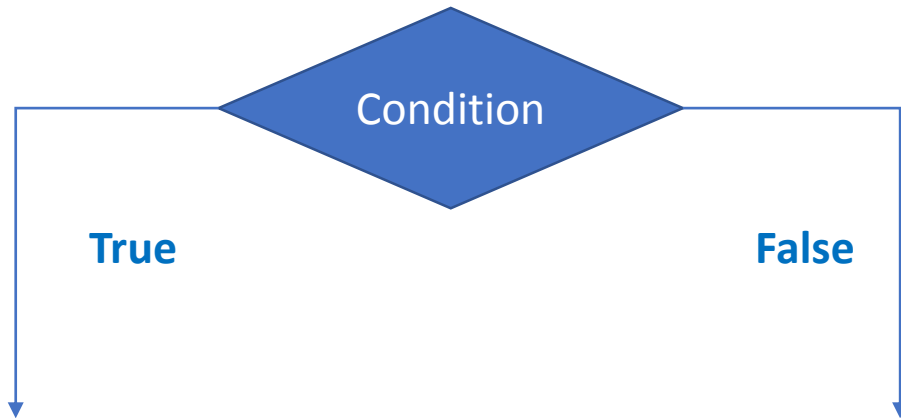
Unidad II, Introducción a Microsoft Visual C#

- Control de Flujo

- Condicionales

- `Exp1 ? Exp2 : Exp3;`

`Exp1 ? Exp2 : Exp3;`



Unidad II, Introducción a Microsoft Visual C#

- Control de Flujo

- Iteración

- Ciclo “for”
 - Ciclo “foreach”

```
for ( init; condition; increment ) {  
    statement(s);  
}
```

```
foreach (var number in numbers) {  
    statement(s);  
}
```

Unidad II, Introducción a Microsoft Visual C#

- Control de Flujo
 - Iteración
 - Ciclo “while”
 - Ciclo “do-while”
 - Se ejecuta al menos una vez

```
while(condition) {  
    statement(s);  
}
```

```
do {  
    statement(s);  
} while( condition );
```

Unidad II, Introducción a Microsoft Visual C#

2.6 Clases, objetos y métodos en C#.

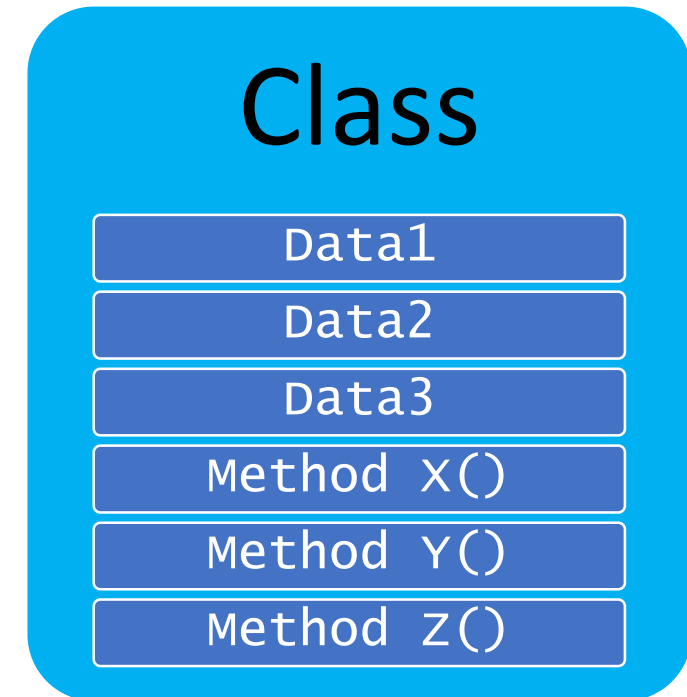
- Non-Primitive Types
 - Classes
 - Structs
 - Arrays
 - Strings
 - Enums

Unidad II, Introducción A Microsoft Visual C#

Non-Primitive Types

Clases

- Una clase es una **plantilla (molde)**, que define ***atributos*** (variables) y ***métodos***(funciones)
- La clase **define los atributos y métodos** comunes a los objetos de ese tipo, pero luego, cada objeto tendrá sus propios valores y compartirán las mismas funciones.
- ***Debemos crear una clase*** antes de poder crear objetos (***instancias***) de esa clase. Al crear un objeto de una clase, se dice que se crea una instancia de la clase o un objeto propiamente dicho.



Unidad II, Introducción A Microsoft Visual C#

Clases

- La sintaxis de una clase en C# es:

```
class[nombre de la clase] {  
  [atributos variables de la clase]  
  [métodos funciones de la clase]  
  [main]  
}
```

- Un **método** es un *conjunto de instrucciones* a las que se les asocia un nombre de modo que si se desea ejecutarlas basta referenciarlas a través de dicho nombre en vez de tener que escribirlas.
- Dentro de estas instrucciones es posible acceder con total libertad a la información almacenada en los campos pertenecientes a la clase dentro de la que el método se ha definido, *los métodos permiten manipular los datos almacenados en los objetos.*

Unidad II, Introducción A Microsoft Visual C#

Clases y constructores

- *Cada vez que se crea una clase se llama a su **constructor***
- Un constructor es un método que se utiliza para inicializar la clase.
- Una clase puede tener **varios constructores** que toman argumentos diferentes.
- Los **constructores** permiten al programador establecer valores predeterminados, limitar la creación de instancias y escribir código flexible y fácil de leer.

Unidad II, Introducción A Microsoft Visual C#

Características de los Constructores

- Tiene el mismo nombre de la clase.
- Es el primer método que se ejecuta.
- Se ejecuta en forma automática.
- No puede retornar datos.
- Se ejecuta una única vez.
- Un constructor tiene por objetivo inicializar atributos.

• SINTAXIS

```
Modificador NombredeLaClase(Parámetros)
{
    Instrucciones
}
```

Unidad II, Introducción A Microsoft Visual C#

Constructores

- Las clases pueden definir constructores que acepten parámetros.
- Las clases pueden definir varios constructores y no se requiere ninguno para definir un constructor predeterminado.
- Se debe llamar a constructores que toman parámetros a través de una instrucción “new”.
- Al igual que los métodos, los constructores también pueden ser sobrecargados.

```
public class Employee
{
    public int salary;

    public Employee(int annualSalary)
    {
        salary = annualSalary;
    }

    public Employee(int weeklySalary, int numberOfWeeks)
    {
        salary = weeklySalary * numberOfWeeks;
    }
}
```

```
Employee e1 = new Employee(30000);
Employee e2 = new Employee(500, 52);
```

Unidad II, Introducción A Microsoft Visual C#

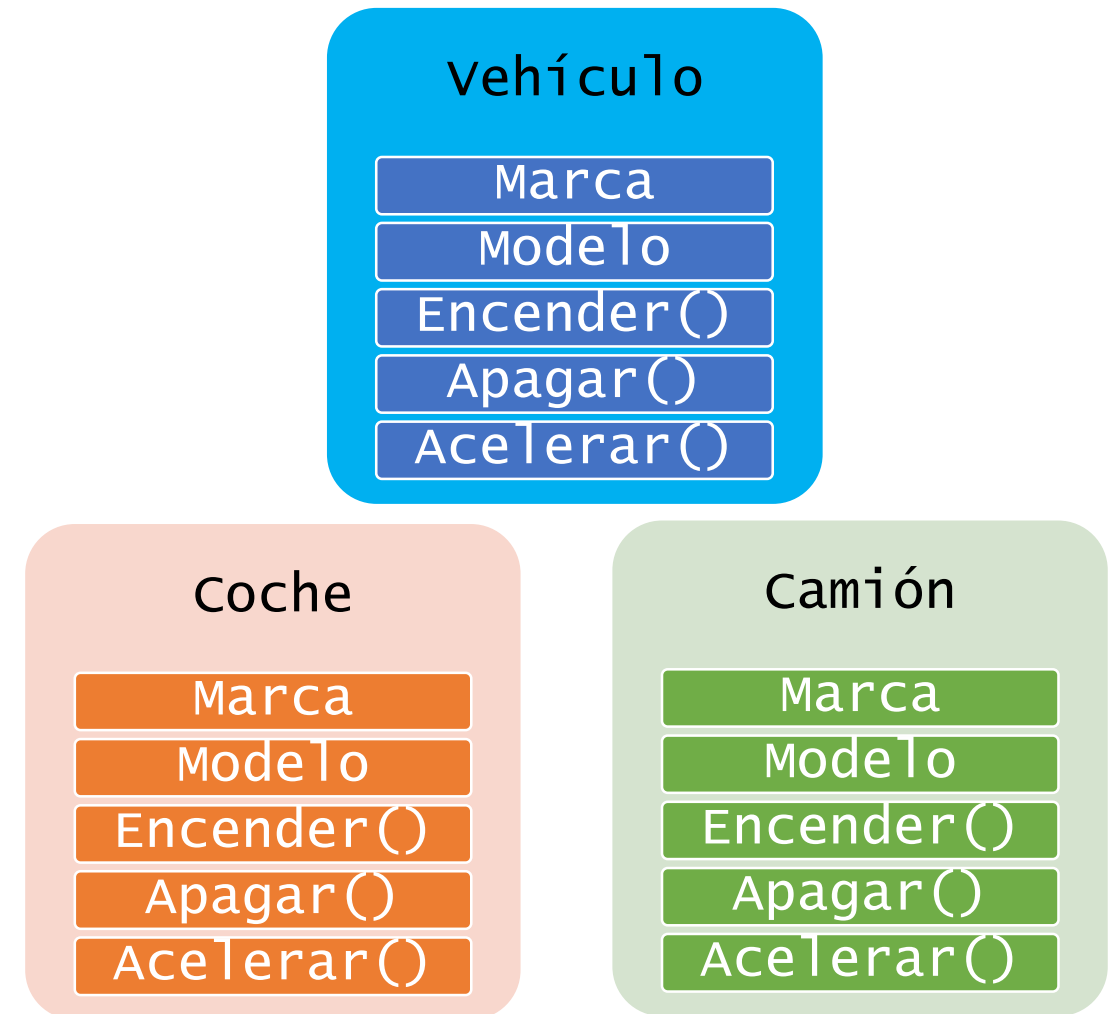
Destruyores

- Un destructor se declara como un constructor, aunque va precedido por un signo de tilde ~.
- Se emplea una desasignación de memoria de objetos no referenciados (recolección de basura), y cuando esto ocurre se ejecuta el destructor de dicha clase.
- El destructor de una clase no se llama cuando un objeto sale del ámbito.
- Todos los destructores se llamarán antes de que finalice un programa.
- La palabra clave “**this**” es un apuntador al mismo objeto en el cual se usa.
- C# permite la sobrecarga de operadores con la palabra clave **operator**

Unidad II, Introducción A Microsoft Visual C#

Herencia

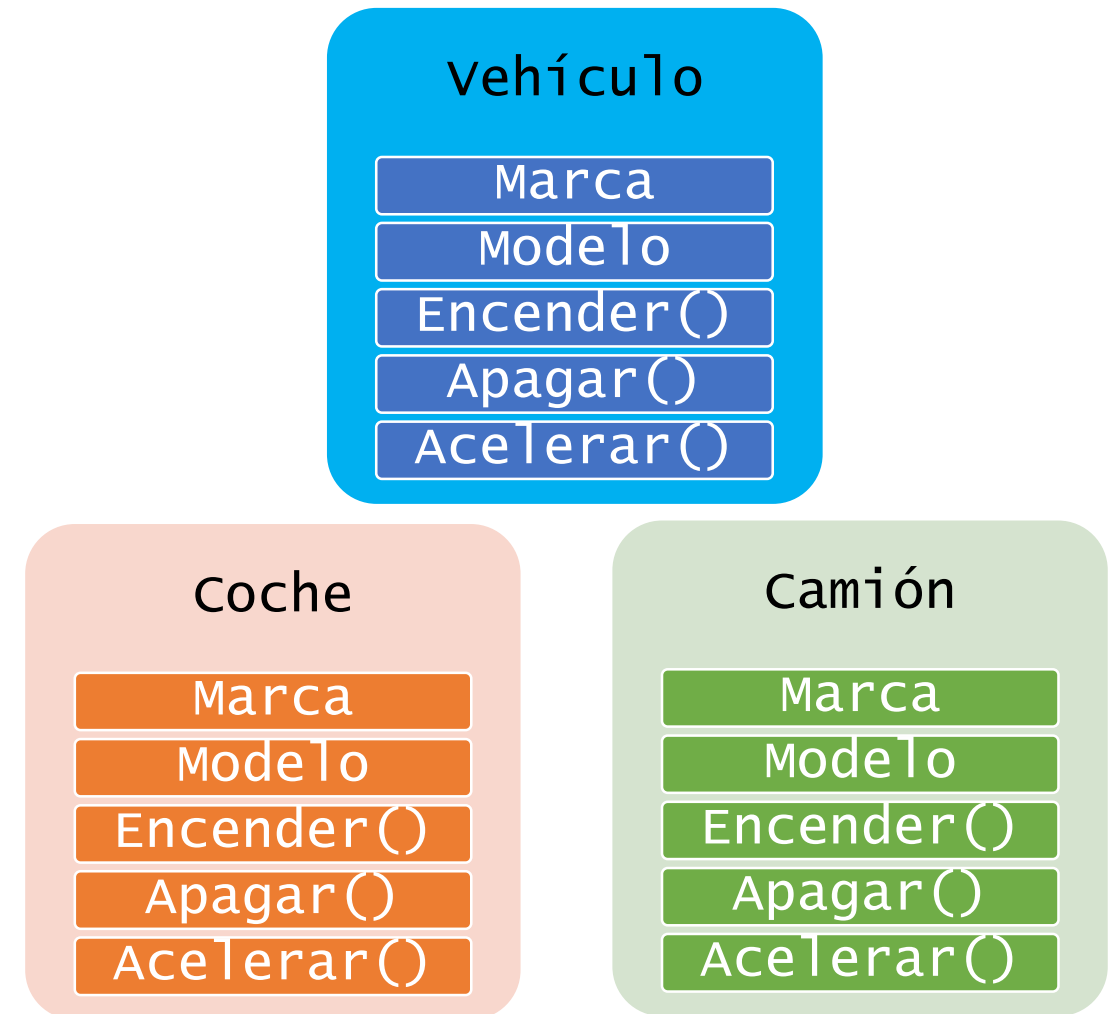
- **La herencia** nos permite derivar una nueva clase a partir de una existente.
- La clase existente es conocida como clase madre, clase padre, o superclase, o clase base.
- La clase derivada también es conocida como clase hija, o subclase.
- Una clase puede ser derivada de más de una clase o interface, lo que quiere decir que puede heredar datos y métodos de múltiples clases base.



Unidad II, Introducción A Microsoft Visual C#

Herencia

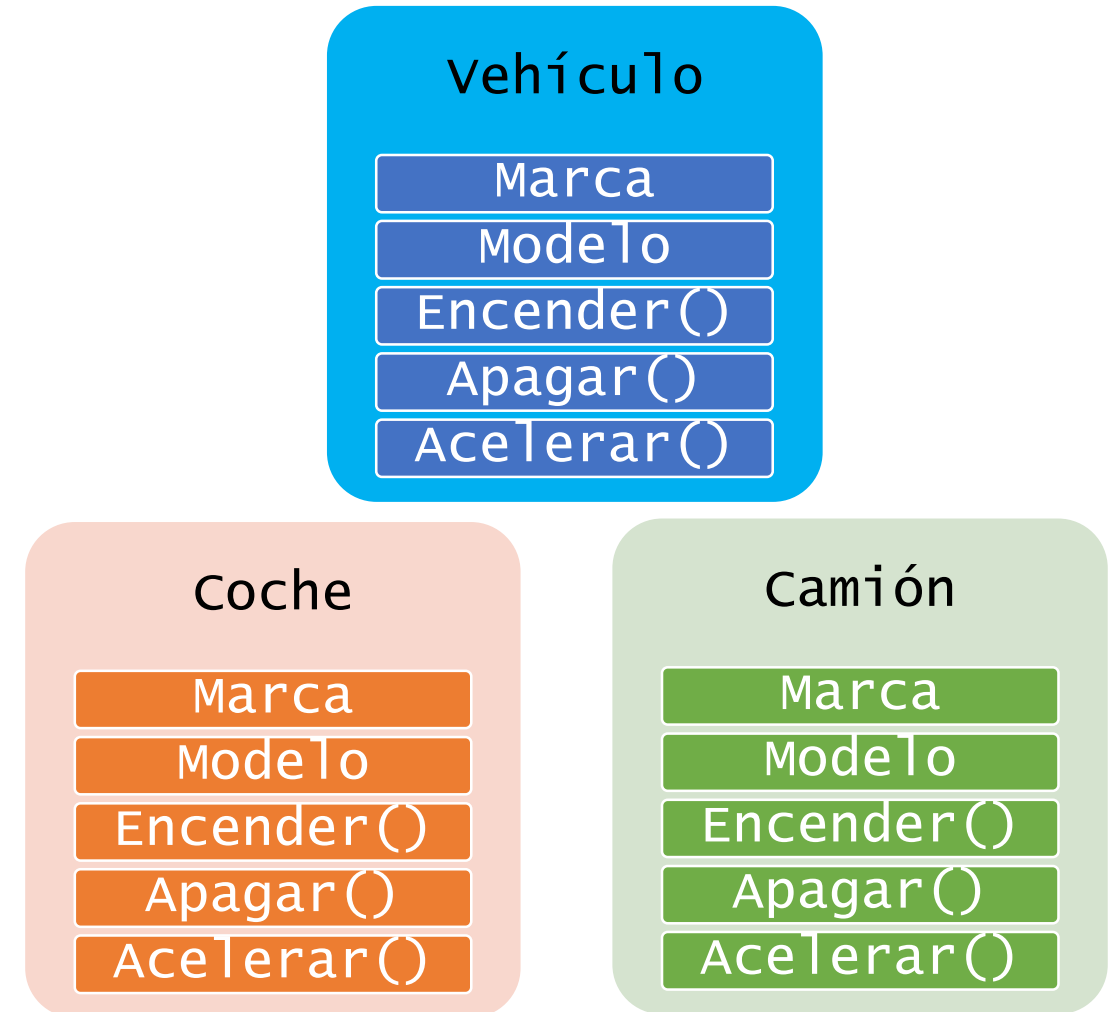
- **La herencia es la columna vertebral de la POO.** Permite a los programadores crear una jerarquía entre un grupo de clases que tienen características similares.
- La herencia **es una forma de reutilización de código.**



Unidad II, Introducción A Microsoft Visual C#

Encapsulamiento

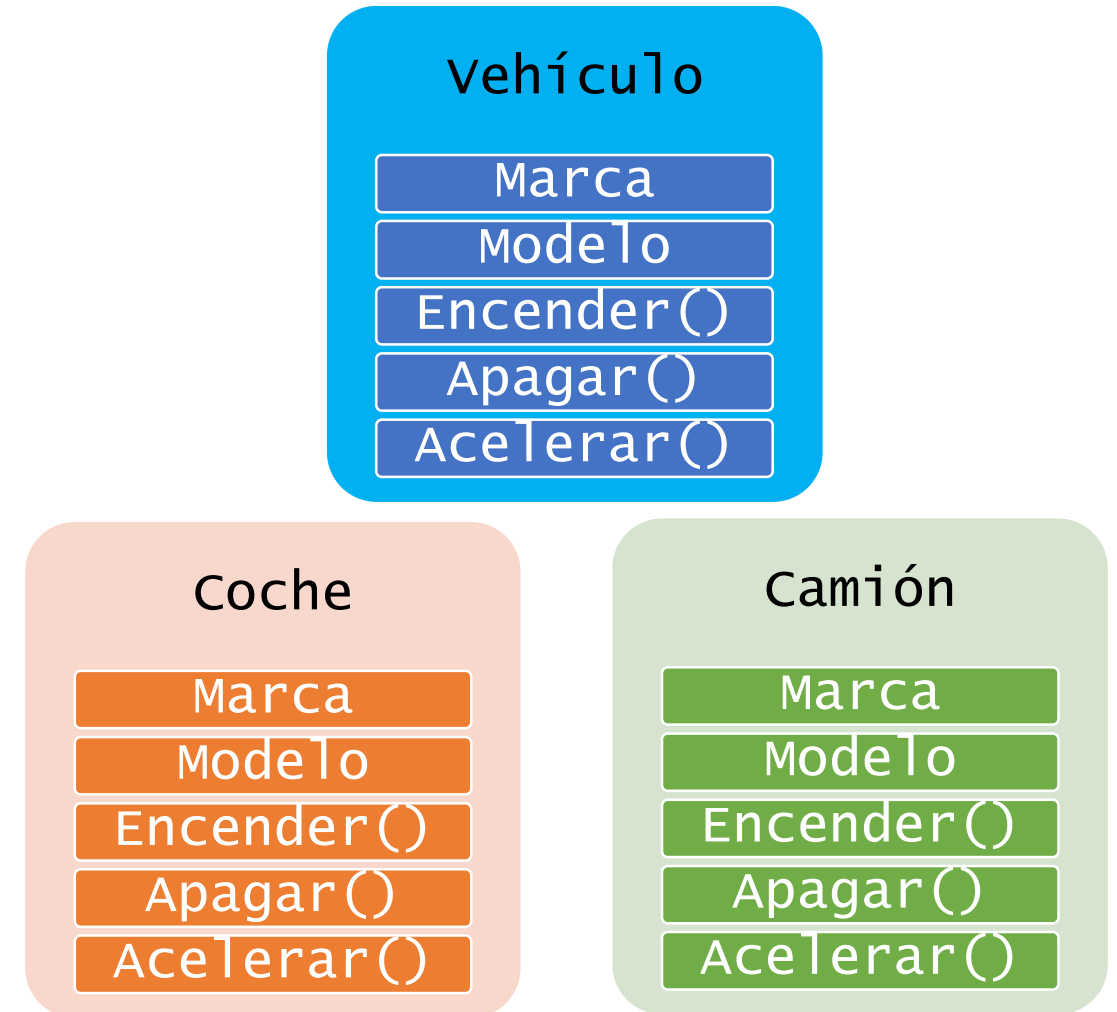
- Es la propiedad que tienen los objetos, de contener tanto datos como métodos, los cuales pueden manipular o cambiar estos datos.
- **Consiste en separar los aspectos externos de un objeto**(que pueden ser accedidos desde otros objetos) **de los detalles internos** de implementación del mismo.



Unidad II, Introducción A Microsoft Visual C#

Encapsulamiento

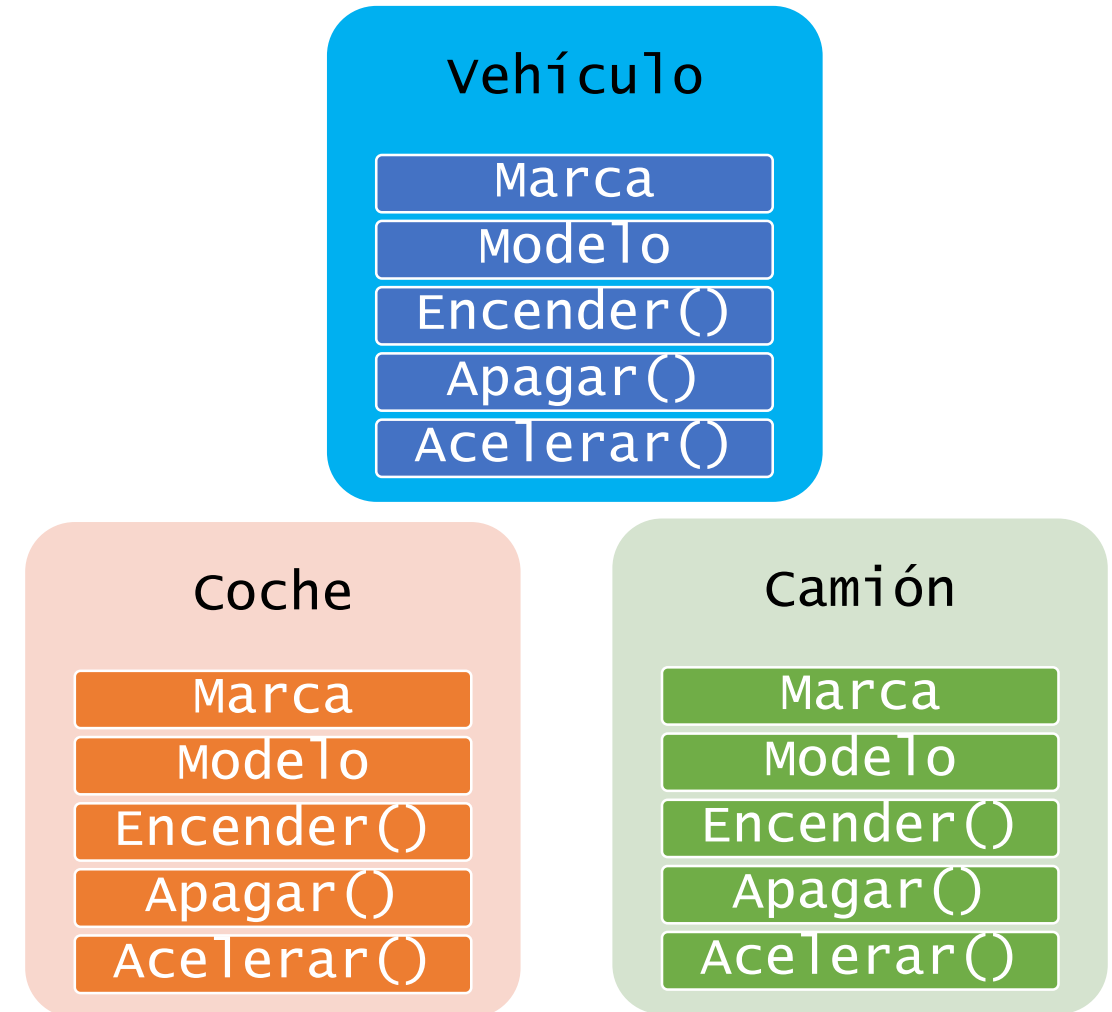
- Mediante esta propiedad, los objetos, tienen el control necesario, de la integridad de los datos contenidos en estos.
- Los clientes de una clase sólo conocen la interfaz de la misma, es decir, conocen los prototipos de las operaciones pero no cómo están implementadas



Unidad II, Introducción A Microsoft Visual C#

Polimorfismo

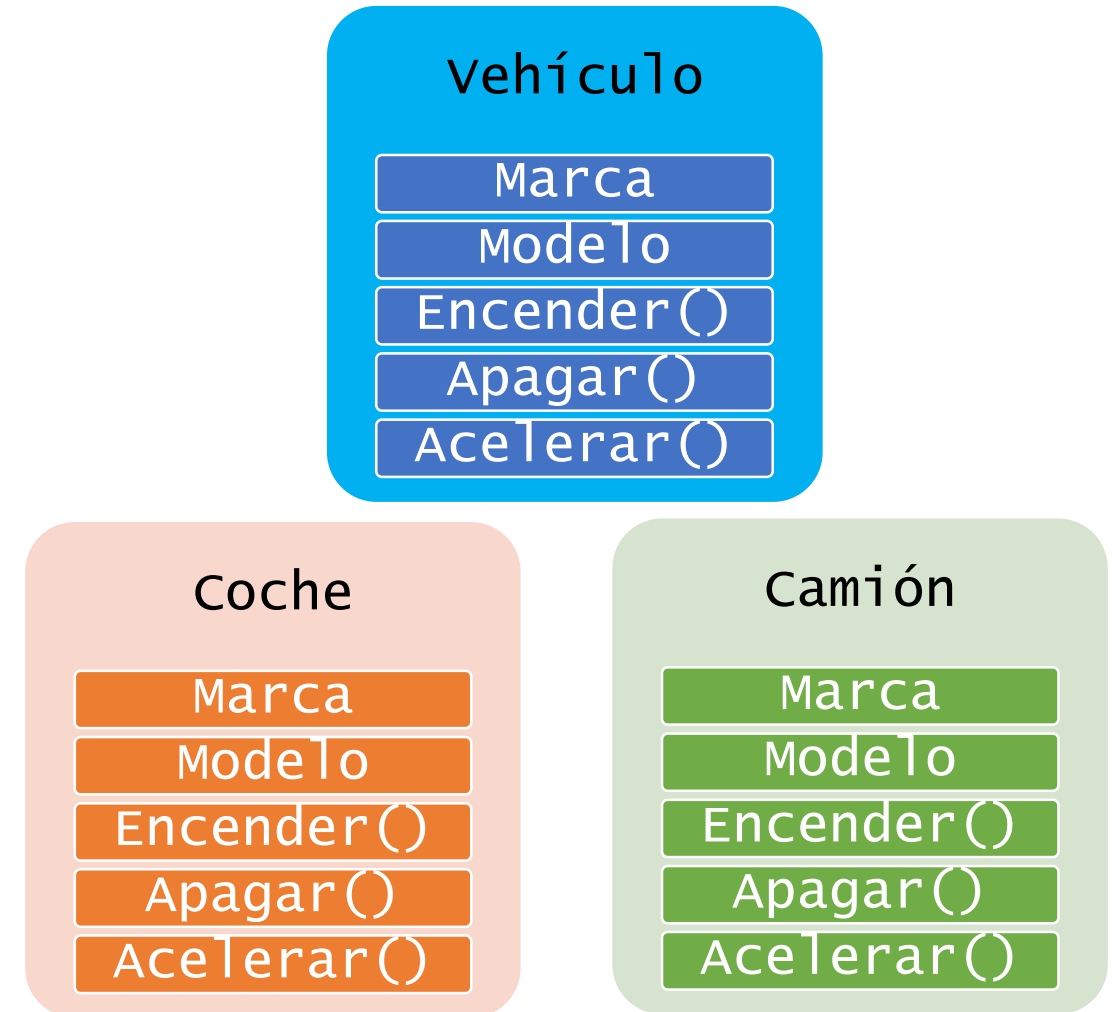
- Significa que la misma operación puede comportarse diferente en clases distintas.
- El polimorfismo está muy ligado a la herencia.
- Distintas instancias del mismo tipo interpretan el mismo mensaje en diferentes formas.



Unidad II, Introducción A Microsoft Visual C#

Polimorfismo

- El polimorfismo requiere enlace dinámico
- Enlace dinámico: la llamada se resuelve en tiempo de ejecución.
- Enlace estático: la llamada se resuelve en tiempo de compilación.



UNIDAD I, introducción a la programación visual

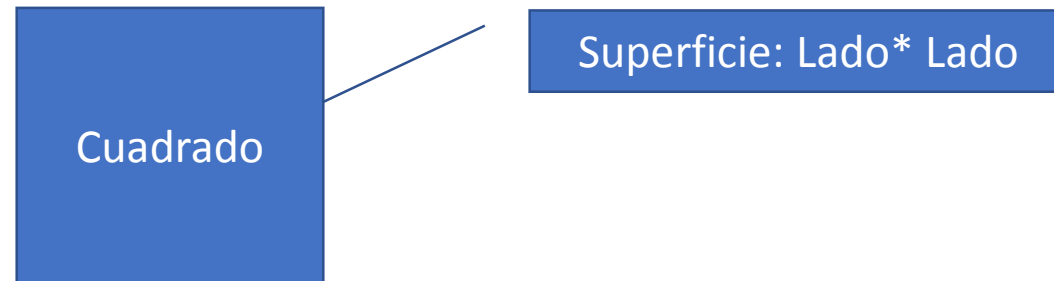
TAREA # 3 Clases en C#:

1. Realizar un programa que muestre el uso de Clases en C#, se puede reusar el código ya hecho en la tarea 1 y 2.
2. Subir el programa a github o bitbucket y enviar por email (daniel_nuno@hotmail.com) la liga al mismo.

UNIDAD I, introducción a la programación visual

Ejercicio # 3 Clases en C#:

1. Desarrollar un programa que tenga una clase que represente un Cuadrado y tenga los siguientes métodos:
2. ingresar valor a su lado,
3. imprimir su perímetro y su superficie.
4. Subir el programa a github o bitbucket y enviar por email (daniel_nuno@hotmail.com) la liga al mismo.



Unidad II, Introducción a Microsoft Visual C#

Non-Primitive Types

Structs

- Sirven para que una variable, pueda tener información relacionada.
- Se utilizan para representar un registro.
- No pueden recibir herencia de otras estructuras.

```
public struct RGBColor
{
    public int Red;
    public int Green;
    public int Blue;
}
```

Unidad II, Introducción a Microsoft Visual C#

Structs

- No pueden ser usadas como base de otras estructuras
- Las estructuras pueden tener un constructor pero no un destructor
- El constructor es definido automáticamente y no se puede modificar
- Ideales para definir un objeto pequeño, del que se crearían cientos

```
public struct RGBColor
{
    public int Red;
    public int Green;
    public int Blue;
}
```


Unidad II, Introducción a Microsoft Visual C#

Clases vs Structs

- Las clases son tipos de referencia, las estructuras son tipos de valor.
- Las estructuras no soportan herencia.
- Las estructuras no pueden tener un constructor default.
- El constructor en las estructuras es automáticamente definido y no se puede cambiar.

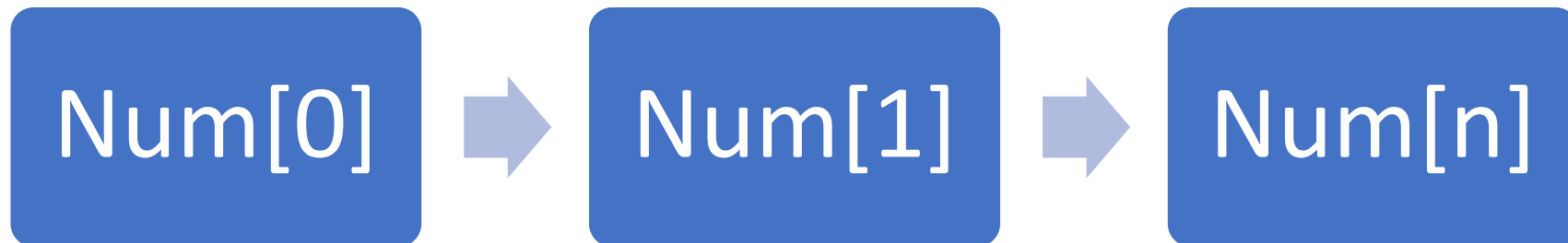
Unidad II, Introducción a Microsoft Visual C#

2.7 Arreglos y Matrices

Non-Primitive Types

Arrays,

- Un arreglo almacena una colección secuencial de elementos de un tamaño fijo y de un mismo tipo.
- Todos los arreglos consisten en localidades de memoria contigua.



Unidad II, Introducción a Microsoft Visual C#

Non-Primitive Types

Arrays, Matrices

- Los índices empiezan en cero.
- Cuando se declara un arreglo, los corchetes [] deben ir después del tipo, no después del identificador.
- Los arreglos no se inicializan automáticamente.

- Declaración válida:

```
datatype[] arrayName;
```

Ejemplo:

```
int[] valores;
```

- Declaración inválida:

```
int valores[];
```

Unidad II, Introducción a Microsoft Visual C#

Non-Primitive Types

Arrays, Matrices

- Los arreglos es una clase de referencia **System.Array**, se requiere utilizar la palabra “new” para instanciarlos.
- Se utiliza el índice para acceder a los valores almacenados.
- [Propiedades y métodos de System.Array](#)

- Declaración válida:

```
datatype[] arrayName;
```

Ejemplo:

```
int[] valores;
```

- Declaración inválida:

```
int valores[];
```

Unidad II, Introducción a Microsoft Visual C#

Non-Primitive Types

Arreglos Multidimensionales,

- Los arreglos multidimensionales son realmente objetos **System.Array** que es el tipo base abstracto de todos los tipos de arreglos.
- Las propiedades y otros miembros de la clase **System.Array** se pueden utilizar cuando sea necesario.
- Ejem:
 - La propiedad Length para obtener la longitud de una matriz.

Unidad II, Introducción a Microsoft Visual C#

Non-Primitive Types

Arreglos Multidimensionales,

- Para declarar un arreglo multidimensional
- Dos Dimensiones: `datatype[,] arrayName;`
- Tres Dimensiones: `datatype[, ,] arrayName;`

• Jagged Arrays

- Arreglos de Arreglos
- `datatype[][] arrayName;`

Unidad II, Introducción a Microsoft Visual C#

Non-Primitive Types

Arreglos como parámetros

- Los arreglos pueden ser pasados como parámetros entre funciones.
- Facilitan el pasar argumentos cuando se desconoce la cantidad de los mismos.

```
class ParamArray {  
    public int AddElements(params int[] arr) {  
        // Logic  
    }  
  
int suma = AddElements(512, 720, 250, 567, 889);
```

Unidad II, Introducción a Microsoft Visual C#

- Non-Primitive Types
 - ~~Classes~~
 - ~~Structs~~
 - ~~Arrays~~
 - Strings
 - Enums

Unidad II, Introducción a Microsoft Visual C#

- Non-Primitive Types

- Strings

- Los strings se pueden utilizar como un arreglo de caracteres
 - Se puede utilizar también, la palabra reservada “string” para declarar una variable
 - La palabra reservada “string” es un alias de la clase **System.String**
 - Los strings son inmutables, es decir, que no se pueden cambiar, existen funciones que pueden manipularlos, pero el string resultante, será un nuevo string.

- Propiedades y métodos de System.String

Unidad II, Introducción a Microsoft Visual C#

- Non-Primitive Types
 - Strings
- **Se puede crear un “string” de la siguientes maneras:**
 - Al asignar un literal de cadena a una variable de cadena
 - Mediante el uso de un constructor de clase String
 - Mediante el uso del operador de concatenación de cadenas (+)
 - Recuperando una propiedad o llamando a un método que devuelve una cadena
 - Llamando a un método de formateo para convertir un valor o un objeto a su representación de cadena

Unidad II, Introducción a Microsoft Visual C#

- Non-Primitive Types
 - Strings
 - Caracteres de escape

| Escape Character | Description |
|------------------|-----------------------|
| \n | New line |
| \t | Tab |
| \\ | Backslash |
| \' | Single quotation mark |
| \" | Double quotation mark |

Unidad II, Introducción a Microsoft Visual C#

- Non-Primitive Types

- Strings

- Verbatim strings

- `string path = "c:\\proj\\proj1\\folder1";`

- `string path = @"c:\proj\proj1\folder1";`

Unidad II, Introducción a Microsoft Visual C#

- Non-Primitive Types

- Enums

- Un grupo de pares que consisten en un nombre y un valor.
 - Son constantes, no pueden ser cambiadas.
 - No pueden ser heredados
 - No pueden heredar

```
enum <enum_name> {  
    enumeration list  
};
```

Ejemplo:

```
enum Days { Sun, Mon, tue,  
Wed, thu, Fri, Sat };
```

Unidad II, Introducción a Microsoft Visual C#

```
const int RegularMail = 1;  
const int RegisterMail = 2;  
const int ExpressMail = 3;
```

```
public enum MailType : byte  
{  
    RegularMail = 1,  
    RegisterMail = 2,  
    ExpressMail = 3  
};
```

```
var method =  
MailType.ExpressMail;
```

Unidad II, Introducción a Microsoft Visual C#

- **Value Types**

- Structures

 - Primitive Types

 - int
 - char
 - float
 - bool

 - Custom Structures

- Allocated on stack
- Memory allocation automatically
- Immediately removed when out of scope

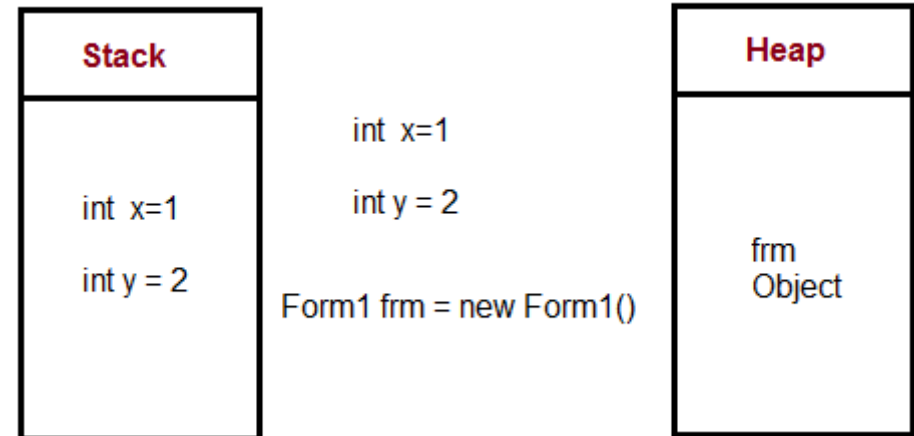
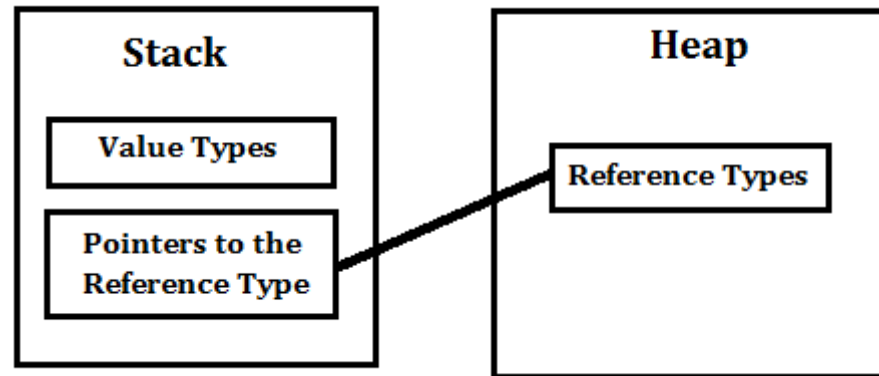
- **Reference Types**

- Classes

 - Array (System.Array)
 - Strings (System.String)
 - Custom classes

- You need to allocate memory
- Memory allocated on the heap
- Garbage collector

Unidad II, Introducción a Microsoft Visual C#



UNIDAD I, introducción a la programación visual

Ejercicio # 4 en clase:

- Realizar un programa que haga un recorrido en una matriz dada e imprima el contenido:

| | | | | |
|---|---|----|----|-----|
| 1 | 9 | 23 | 34 | 102 |
| 2 | 8 | 67 | 56 | 23 |
| 3 | 7 | 84 | 78 | 43 |
| 4 | 5 | 90 | 98 | 34 |

Resultado: 1,2,3,4,5,7,8,9,23,67,84,90,98,78,56,34,102,23,43,34

- Subir el programa a github o bitbucket y enviar por email (daniel_nuno@hotmail.com) la liga al mismo.

UNIDAD I, introducción a la programación visual

Ejercicio # 5 en clase:

1) Entrar a la página de Microsoft DeveloperNetwork MSDN:

<https://msdn.microsoft.com/es-mx>>Documentación > API y referencia >

2) En la página ubicar la sección de: Herramientas y Lenguajes de Desarrollo
>Seleccionar: Visual Studio 2015

3) Con apoyo de la herramienta de búsqueda ubicar las siguientes páginas de contenido:

Tutorial: [Crear una aplicación sencilla con Visual C# o Visual Basic](#)


4) Seguir las instrucciones para crear la aplicación

5) Subir el programa a github o bitbucket y enviar por email (daniel_nuno@hotmail.com) la liga al mismo.

UNIDAD I, introducción a la programación visual

TAREA # 4 palíndromo en C#:

1. Realizar un programa que haga diga si una cadena es un “palíndromo”.
 1. ¿Acaso hubo búhos acá?
 2. A la catalana banal, atácala.
 3. A mamá, Roma le aviva el amor a papá y a papá, Roma le aviva el amor a Mamá.
 4. A ti no, bonita.
 5. Amo la pacífica paloma.
 6. Amor a Roma.
 7. Ana lava lana.
2. Subir el programa a github o bitbucket y enviar por email (daniel_nuno@hotmail.com) la liga al mismo.



Unidad II, Introducción a Microsoft Visual C#

~~2.1 Breve descripción del C#.~~

~~2.2 Tipos de datos básicos.~~

~~2.3 La biblioteca de clases.~~

~~2.4 Operadores.~~

~~2.5 Instrucciones de control de programa.~~

~~2.6 Clases, objetos y métodos en C#.~~

~~2.7 Arreglos y Matrices, Cadenas.~~

2.8 Sobrecarga de operadores.

2.9 Indizadores y propiedades.

2.10 Herencia en C#.

2.11 Interfaces, ~~estructuras y~~
~~enumeraciones.~~

2.12 Colecciones (listas, hash, stack, queue)

Unidad II, Introducción a Microsoft Visual C#

2.8 Sobrecarga de operadores

- Se puede redefinir o sobrecargar la mayoría de los operadores.
- Los operadores sobrecargados son funciones con un nombre especial que incluye la palabra “operator” seguida del operador que esta siendo definido.
- Un operador sobrecargado regresa un tipo de dato y acepta parámetros.

Ver ejemplo:

https://www.tutorialspoint.com/csharp/csharp_operator_overloading.htm

Unidad II, Introducción a Microsoft Visual C#

2.8 Sobrecarga de operadores

- C# permite que los tipos definidos por el usuario sobrecarguen operadores al definir funciones miembro estáticas mediante la palabra clave [operator](#).
- No todos los operadores se pueden sobrecargar y algunos presentan restricciones.
- <https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/statements-expressions-operators/overloadable-operators>

Unidad II, Introducción a Microsoft Visual C#

2.9 Índices y propiedades.

Un índice permite que un objeto sea indexado como un arreglo.

Cuando se define un índice para una clase, esta clase se comporta similar a un arreglo virtual.

Entonces se puede acceder a la instancia de esta clase usando el operador “[]”.

Unidad II, Introducción a Microsoft Visual C#

2.9 Índices y propiedades.

```
element-type this[int index] {  
  
    // The get accessor.  
    get {  
        // return the value specified by index  
    }  
  
    // The set accessor.  
    set {  
        // set the value specified by index  
    }  
}
```


Unidad II, Introducción a Microsoft Visual C#

2.9 Índices y propiedades.

Uso de los índices.

Se pueden utilizar como a las propiedades de una clase, con un “get” y un “set”.

Sin embargo, las propiedades regresan o definen un miembro específico, y los índices regresan o definen un valor particular para la instancia del objeto.

Ejemplo:

https://www.tutorialspoint.com/csharp/csharp_indexers.htm

Unidad II, Introducción a Microsoft Visual C#

2.10 Herencia en C#

Unidad II, Introducción a Microsoft Visual C#

2.11 Interfaces, estructuras y enumeraciones.

Unidad II, Introducción a Microsoft Visual C#

2.12 Colecciones (list, hash, stack, queue)

Las colecciones son clases especializadas que se utilizan para manejar varios valores u objetos de diferentes series.

Existen dos tipos de colecciones:

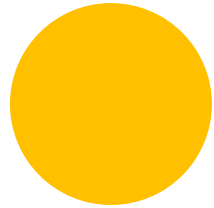
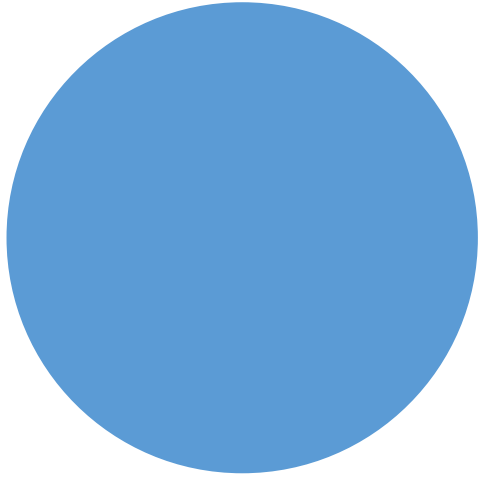
- non-generic collections
- generic collections

Cada clase de colecciones, implementa la interface "[IEnumerable](#)" por lo que los valores se pueden acceder mediante un "foreach"

Unidad II, Introducción a Microsoft Visual C#

2.12 Colecciones (list, hash, stack, queue)

| Non-generic collection | Uso |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| ArrayList | Almacena objetos de cualquier tipo a manera de arreglo. No se necesita especificar el tamaño del arreglo, ya que se incrementa automáticamente. |
| SortedList | Almacena pares donde uno es la llave y el otro es el valor. Automáticamente acomoda los elementos en modo ascendente utilizando la llave. |
| Stack | Almacena en forma de pila (LIFO). Provee métodos para almacenar, recuperar elementos. |
| Queue | Almacena en forma de cola (FIFO). Mantiene el orden en el que los elementos fueron agregados. Provee métodos para almacenar y recuperar. |
| Hashtable | Almacena pares donde uno es la llave y el otro es el valor. Recupera los valores utilizando la llave dada. |
| BitArray | Arreglo compacto de valores binarios, donde “true” indica un bit en “1”. |



Unidad III, Microsoft Visual C# Avanzado



Unidad III, Microsoft Visual C# Avanzado

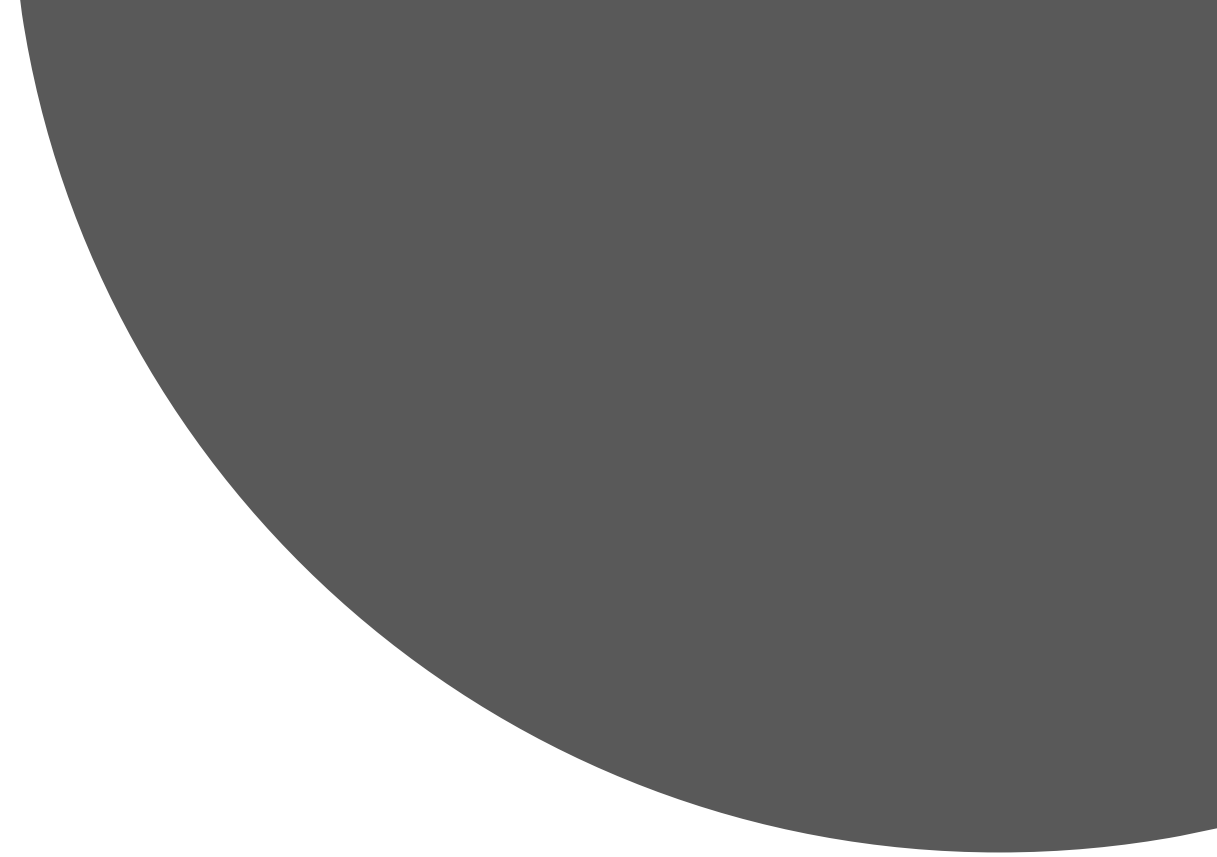
3.1 Control de excepciones.

3.2 Entrada y salida.

3.3 Delegados y eventos.

3.4 Espacios de nombres, preprocesador y ensamblados.

3.5 Código no seguro y punteros en C#.



Unidad IV, Exploración de la biblioteca C#



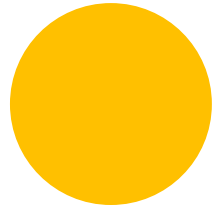
Unidad IV, Exploración de la biblioteca C#

4.1 Exploración del espacio de nombres System.

4.2 Uso de clases principales predefinidas.

4.3 Administración de memoria.

4.4 El recolector de basura de C#.



Unidad V, Programación Multiproceso



Unidad V, Programación Multiproceso

5.1 Fundamentos del multiproceso.

5.2 La clase Thread.

5.3 Propiedades de los procesos.


5.4 Comunicación entre procesos.

5.5 Suspensión, reanudación y detención de subprocesos.

5.6 Tareas independientes.



Unidad VI, Creación de aplicaciones Windows



Unidad VI, Creación de aplicaciones Windows

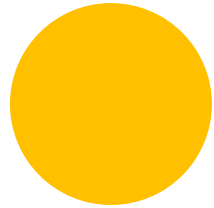
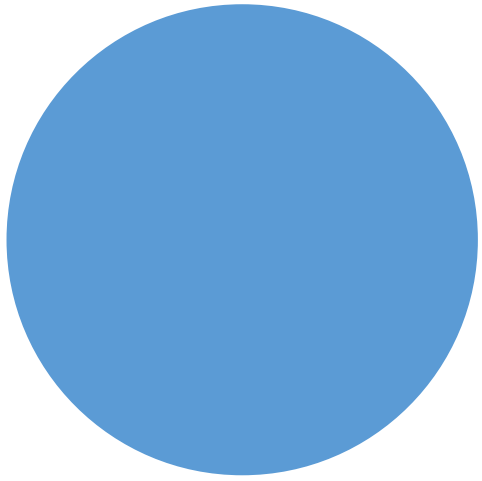
6.1 Formularios Windows.

6.2 Esqueletos de aplicaciones basadas en formularios.


6.3 La paleta de componentes.

6.4 Control de mensajes.

6.5 Principales componentes de una aplicación Windows.



Unidad VII, El modelo de acceso a datos ADO.NET



Unidad VII, El modelo de acceso a datos ADO.NET

7.1 Características de ADO.NET.

7.2 Modo desconectado.

7.3 Modo conectado.

7.4 Controles data y controles enlazados
con Windows forms.

7.5 Elementos XML.



Anexos

Bibliografía

- Visual C# How to Program
 - by Harvey Deitel, Paul Deitel
 - Publisher: Pearson
 - Release Date: August 2016
 - ISBN: 9780134628820
- Microsoft Visual Studio 2015 Unleashed, Third Edition
 - by Mike Snell, Lars Powers
 - Publisher: Sams
 - Release Date: August 2015
 - ISBN: 9780134133164

Bibliografía

- C# for Java Developers
 - by Adam Freeman, Allen Jones
 - Publisher: Microsoft Press
 - Release Date: August 2002
 - ISBN: 9780735617797
- .NET Common Language Runtime Unleashed
 - by Kevin Burton
 - Publisher: Sams
 - Release Date: April 2002
 - ISBN: 0672321246

Referencias

- <https://docs.microsoft.com/en-us/dotnet/index>
- <https://visualstudio.microsoft.com/>
- <https://www.tutorialspoint.com/csharp/>

Java versus C#

Miscellaneous Language Differences

- The following list presents some simple language differences, just in case you are looking for a key feature:
- Java does not have foreach or goto.
- Java has no pointers.
- C# can drop down to unsafe mode.
- With C#, all types are derived from System.Object.
- That means that even primitive types have ToString, GetHashCode, GetType, and Equals methods available to them.
- In addition, collections can be made including primitive types and other types.
- Java does not have that option.

Java versus C#

Miscellaneous Language Differences

- Java has no operator overloading.
- Java has no preprocessor directives.
- Although the potential exists to have cross-platform IL with the CLR, it does not exist now, and Java has a clear advantage on this issue.
- Java does not have a struct or enum.
- C# adds a decimal type that Java does not have.
- C# has verbatim strings where escape characters are not interpreted.
- C# has in, out, and ref keywords to explicitly control how arguments are passed