Tema 7; .NET y Primeros programas con C#

.NET

-permite el desarrollo de aplicaciones software y librerías
-contiene el compilador y las herramientas necesarias para construir, depurar y ejecutar estas aplicaciones
-independiente del lenguaje
-permite combinar código escrito en diferentes lenguajes
-No está orientado a un Hardware/Sistema Operativo concreto

Visual Studio .NET

-Entorno de desarrollo para desarrollar aplicaciones que se ejecutan sobre el .NET Framework.
-Simplifica la creación e implantación de Aplicaciones y Servicios Web y Aplicaciones basadas en Windows

Common Language Specification (CLS)

-mínimos estándares que deben satisfacer los lenguajes y desarrolladores si desean que sus componentes y aplicaciones sean ampliamente utilizados por otros lenguajes compatibles con .NET.
-permite crear aplicaciones con la seguridad de que no habrá problemas con la integración de los diferentes lenguajes
-permite heredar de clases desarrolladas en lenguajes diferentes.

.NET Framework

-Es el motor de ejecución
-Proporciona un conjunto de servicios comunes para los proyectos generados en .net, con independencia del lenguaje.
-Extensible
-La jerarquía del .NET Framework no queda oculta al desarrollador
-Herencia y Herencia multilenguaje
-código está organizado en espacios de nombres jerárquicos y clases
-Sistema de tipos común,denominado sistema de tipos unificado, que utiliza cualquier lenguaje compatible con .NET. En el sistema todo es un objeto

Colecciones de datos

-Existen 3 tipos principales de colecciones: ICollection, interfaz IList e interfaz IDictionary
-las colecciones de tipo IList (y las directamente derivadas de ICollection) solo almacenan un valor, mientras que las colecciones de tipo IDictionary guardan un valor y una clave relacionada con dicho valor
-IList se utiliza para acceder a los elementos de un array mediante un índice numérico
-Existen 3 tipos de colecciones que implementan esta interfaz: sólo lectura (no se pueden modificar. Basadas en ReadOnlyCollectionBase), de tamaño fijo (no se pueden quitar ni añadir elementos, pero sí modificarlos) y las de tamaño variable (permiten cualquier acción. Son la mayoría)
-ArrayList, CollectionBase y StringCollection implementan IList
-ArrayList representa una lista de datos, puede modificar su tamaño dinámicamente. Empieza en 0 y se añaden elementos consecutivamente
-using System.Collections ArrayList arraylist = new ArrayList();
-El constructor de ArrayList también puede utilizar un parámetro entero indicando el tamaño del objeto
-Add añade el elemento en la última posición
-Insert lo inserta en una posición indicada
-remove elimina el objeto pasado como parámetro
-removeAt elimina el elemento en la posición especificada
-RemoveRange elimina un grupo de elementos
-Todos los objetos de ArrayList son tratados como objetos
-a diferencia de los array no todos los elementos deben ser del mismo tipo
-Count sirve para conocer la cantidad de elementos que contiene
-Capacity indica la capacidad máxima de la colección
-si Capacity devuelve menos que Count se producirá la excepción ArgumentOutOfRangeException
-Si es un escenario donde no conocemos el tamaño que tendrá la colección y si además será muy probable que el tamaño varíe, entonces será recomendable bajo todas las demás circunstancias usar un ArrayList
-para escenarios donde se conoce de antemano la cantidad total de elementos a almacenar y si todos son del mismo tipo, se debe usar un array normal

Common Language Runtime (CLR)

-es el núcleo de la plataforma .NET
-gestiona la ejecución de las aplicaciones desarrolladas
-ofrece servicios que simplifican el desarrollo la fiabilidad y seguridad de las aplicaciones.
-Ejecución multiplataforma, actúa como una máquina virtual, cualquier plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación .NET.
-Existe para todas las versiones de Windows
-para sistemas Unix existe Mono
-la arquitectura del CLR es abierta.
-La integración de lenguajes provee que es posible escribir una clase en C# que herede de otra escrita en Visual Basic.NET que, a su vez, herede de otra escrita en C++ con extensiones gestionadas.
-El CLR incluye un recolector de basura que evita errores de programación muy comunes (Intentos de borrado o acceso de objetos ya borrados o agotamiento de memoria
-comprueba que toda conversión de tipos que se realice durante la ejecución de una aplicación .NET se haga de modo que los tipos origen y destino sean compatibles
-los errores se propagan mediante excepciones

Biblioteca de clases

-expone características del entorno de ejecución y proporciona en una jerarquía de objetos otros servicios de alto nivel que todo programador necesita
-Se denomina espacio de nombres
-el espacio de nombres Collections que añade numerosas posibilidades nuevas, como clasificación, colas, pilas y matrices de tamaño automático
-La clase de sistema Threading también ofrece nuevas posibilidades para crear verdaderas aplicaciones multi-hilo
-El espacio de nombres System contiene clases fundamentales y clases base que definen tipos de datos valor y referencia comúnmente utilizados, eventos y descriptores de eventos, interfaces, atributos y procesamiento de excepciones. Es el espacio de nombres más utilizado

System	Tipos muy frecuentemente usados, como los los tipos básicos, tablas, excepciones, fechas, números aleatorios, recolector de basura, entrada/salida en consola, etc.
System.Collections	Colecciones de datos de uso común como pilas, colas, listas, diccionarios, etc.
System.Data	Manipulación de bases de datos. Forman la denominada arquitectura ADO.NET
System.IO	Manipulación de ficheros y otros flujos de datos.
System.Net	Realización de comunicaciones en red.
System.Reflection	Acceso a los metadatos que acompañan a los módulos de código.
System.Runtime.Remoting	Acceso a objetos remotos.
System.Security	Acceso a la política de seguridad en que se basa el CLR.
System.Threading	Manipulación de hilos.
System.Web.UI.WebControls	Creación de interfaces de usuario basadas en ventanas para aplicaciones Web.
System.Winforms	Creación de interfaces de usuario basadas en ventanas para aplicaciones estándar.
System.XML	Acceso a datos en formato XML.

-Object es un tipo del que derivan el resto de tipos
-String facilita la gestión de cadenas. Permite asignaciones directas y concatenaciones
-La clase Console solo se debe utilizar para aplicaciones en línea de comandos. Para aplicaciones Windows se debe utilizar el espacio de nombre System.Windows.Forms

Console	Entrada/Salida de la consola
Console.Read()	Lee un flujo de entrada y lo devuelve como un int
Console.ReadLine()	Lee una cadena de texto entera
Console.Write()	Permite escribir en pantalla
Console.WriteLine("La suma de {0} y {1} es {2} ",a,b, a+b) o ("La suma de " + a + " y " + b + " es " + (a+b)) {2,8:F2}	Escribe en la pantalla añadiendo el carácter retorno de carro o fin de línea. La salida se puede formatear utilizando el formato {n.m} donde n es el índice del parámetro y w el ancho de salida. Se puede indicar los decimales utilizando F y un número

-Dos clases importantes de excepciones

System.SystemException	Tienen naturaleza muy general y pueden ser lanzadas por cualquier aplicación
System.ApplicationException	Clase base de cualquier clase de excepción por terceros

Microsoft Intermediate Lenguage (MSIL)

-Los compiladores de .NET generan código escrito en el lenguaje intermedio conocido como Microsoft Intermediate Lenguage (MSIL)
-Incluye instrucciones que permiten trabajar directamente con objetos, tablas y excepciones

Sistema de Tipos Comunes (CTS)

-define un conjunto de tipos de datos predefinidos que se pueden utilizar para definir variables
-es una parte integral del runtime de lenguaje común y es compartido por los compiladores, las herramientas y el propio runtime
-define las reglas que sigue el runtime a la hora de declarar, usar y gestionar tipos
-establece un marco que permite la integración entre lenguajes, la seguridad de tipos y la ejecución de código con altas prestaciones
-admite tanto tipos de valor (datos directamente, copias en cada variable, operaciones sobre la variable) como de referencia (referencian a objetos, 2 variables pueden referenciar el mismo objeto, las operaciones pueden afectar a varias variables)
-Con signo: sbyte<short<int<long
-Sin signo: byte<ushort<uint<ulong
-float(prec. simple)<double(prec. doble)<decimal(prec. alta)
-un float debe inicializarse con una F al final de la parte decimal (numero = 28.67F)
-por defecto un número no entero es double
-los tipos bool no pueden ser entero o viceversa

C#

-diseñado para .NET
-Diseñado desde cero sin ningún condicionamiento

-primer lenguaje moderno orientado a componentes de la familia de C y C++ (y Java)
-Fácil de integrar con Visual Basic
-alto rendimiento
-permite el acceso a memoria de bajo nivel de C++
-Puede incrustarse en páginas ASP.NET
-Incluye clases, interfaces, delegados,espacios de nombres, propiedades, indexadores,eventos, sobrecarga de operadores, versionado,atributos, código inseguro
-documentación en formato XML ....
-Una aplicación C# es una colección de clases, estructuras y tipos
-Los comentarios son importantes. De una sola línea
-C# es un lenguaje de especificaciones seguras (type--- safe), lo que significa que el compilador de C# garantiza que los valores almacenados en variables son siempre del tipo adecuado
-El compilador C# exige que cualquier variable esté inicializada
-C# es mucho más seguro que C++
-VB inicializa a 0 por defecto las variables
-El valor de la constante se calcula en tiempo de compilación con variables del tipo readonly

Declaración de variables

-Usa letras, el signo de subrayado y dígitos
-Recomendaciones: Evita poner todas las letras en mayúsculas, Evita empezar con un signo de subrayado, Evita el uso de abreviaturas, Use PascalCasing (EstaEsUnaVariable) para nombres con varias palabras
-CamelCasing es lo mismo que PascalCasing, pero con la 1a letra en minúscula (estaEsUnaVariable)
-utilizar PascalCasing variables públicas y camelCasing para privadas
-El ámbito de una variable coincide con su clase contenedora
-Las constantes con la palabra const y se deben inicializar obligatoriamente en la declaración

Conversión de tipos

-Es implícita. Automática, cuando no hay posible pérdida de información
-Explicita. Se debe indicar que se desea realizar una conversión en la que puede haber pérdida de información. Se utiliza casting

Main

-debe ir con ‘M’
-es el punto de entrada al programa
-se declara como static void Main
-puede pertenecer a múltiples clases
-La aplicación termina cuando Main acaba o ejecuta un return

La Clase

-es un conjunto de datos y métodos
-Una aplicación C# puede incluir muchas clases
-Se hace referencia a clases por su espacio de nombres mediante la sentencia using

Estructuras de control

-condicionales simples: if/else
-condicionales múltiples: switch/case
-repetición: for/foreach, while/do
-cambio de secuencia: return, break, continue
-foreach permite recorrer todos los elementos de un contenedor con una variable del tipo de los elementos del contenedor
-Return devuelve el control a la rutina llamante actual
-Break sale de la actual estructura de bucle
-Continue obliga a ejecutar la siguiente iteración del bucle

Excepciones

-Es un objeto que se crea cuando se produce una situación de error específica
-el objeto contiene información que permite resolver el problema

-se utiliza try/catch. Try contiene el código normal del programa, Catch contiene el el código de los errores
-finally tiene el código que libera los recursos. Es opcional
-funcionamiento: ejecuta el código de try, si hay error va al catch, si no sigue el programa o salta al bloque finally si es que lo hay