7.C# Básico

1 El lenguaje

1.1Tipos de datos

Estos son los tipos de datos básicos de C#

| **Tipo** | **Tamaño** | **Rango** | **Significado** |
| --- | --- | --- | --- |
| byte | 8 bit | 0 a 255 | Entero sin signo |
| sbyte | 8 bit | -128 a 127 | Entero con signo |
| short | 16 bit | -32768 a 32767 | Entero corto con signo |
| ushort | 16 bit | 0 a 65535 | Entero corto sin signo |
| int | 32 bit | -2147483648 a 2147483647 | Entero medio con signo |
| uint | 32 bit | 0 a 4.294.967.295 | Entero medio sin signo |
| long | 64 bit | -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 | Entero largo con signo |
| ulong | 64 bit | 0 a 18.446.744.073.709.551.615 | Entero largo sin signo |
| float | 32 bit | ±1.401298E−45 a ±3.402823E+38 | Coma flotante corto |
| double | 64 bit | ±4.94065645841246E−324 a ±1.79769313486232E+308 | Coma flotante largo |
| decimal | 128 bit | −7.9228162514264337593543950335 a +7.9228162514264337593543950335 | Coma flotante monetario |
| char | 16 bit | ‘\u0000’ a ‘\uFFFF’ | Carácter unicode |
| bool | 8 bit | true o false | Verdadero o falso |

-Una variable tiene que empezar con una letra o guion

-No puede tener espacios

-No se pueden usar palabras reservadas para nombrarlas

-En C# son case sensitive

1.2 El lenguaje

Operadores:

Los más comunes son :

*- =* Asignación

*- +=* Incremento y asignación

*- -=* Decremento y asignación

*- ==* Igual

*- !=* Distinto

*- >* Mayor qué

*- >=* Mayor o igual que

*- <* Menor que

*- <=* Menor o igual que

*- &&* Y (And)

*- ||* O (Or)1.3

Estructuras condicionales:

En C#se utilizan los ifs else ifs o if else

Siempre devuelven un valor booleano

int number = 5*;*

if(number == 5) {

}

Estructuras de control (Bucles)

-For:Se ejecuta un número de veces.

-While: Se ejecuta hasta que la condición sea distinta del valor dado.

-For each: Sirve para recorrer colecciones.

1.4 Clases, Estructuras e interfaces

C# es un lenguaje orientado a objetos

-Clase

Para definir una clase dentro de C# se usa la palabra reservada class dentro podemos tener distintas propiedades, métodos.

Para definir una clase en C# se usa la palabra class, dentro podemos tener distintas propiedades , variables,métodos…

Tenemos varias palabras reservadas para definir una clase.

-abstract

No permite crear instancias de esa clase.

-partial

Puede estar formada por varios ficheros

-Interfaz

Nos permite crear un contrato que posteriormente implementarán nuestras clases, estos no van a tener contenido.

-Struct

Es un tipo de valor que normalmente se usa para encapsular pequeños grupos de variables relacionadas, creando un nuevo tipo de dato.

1.5 Herencia

Una de las características más importantes de la programación orientada a objetos es la de poder realizar herencia entre clases :

Al trabajar con herencia tenemos que tener constancia de una serie de palabras reservadas a la hora de nombrar las clases de nuestro programa:

public

Indica que la clase,propiedad o método es publico en todo nuestro proyecto

protected

Solo es visible para sí misma y sus hijos.

private

Solo se puede usar desde la propia clase

virtual

Marcar una propiedad o método con virtual, nos indicará que esa entidad de nuestra clase, estará preparada para la sobrecarga.

override

Cualquier método o propiedad marcado con Override significa que estará listo para la sobrecarga.

1.6 System.object

Es la clase base en C#, todos los objetos heredan de Object.

Object tiene 3 métodos que se pueden sobrecargar:

-ToString.Permite devolver un string que hace referencia al tipo de dato o clase que se esta utilizando.

-Equals.sirve para comparar entre objetos para saber si son iguales o no.

-GetHashCode. Sirve para obtener el identificador del objeto que se le pasa.

2 Particularidades C#

2.1.Genéricos: Uso básico

Public class PlayList<T>

{}

T-> Indica que se va a usar un elemento genérico,básicamente es como un comodín.

Engloba prácticamente cualquier tipo de dato.

Se puede añadir todos los genéricos que queramos desde 1 a n.

2.2.Colecciones:Lista,pila y cola

Lista:

-Es un listado que nos permite guardar los elementos que necesitemos, se pueden añadir objetos nuevos con el método Add y se pueden añadir conjuntos de objetos con AddRange.

-Se puede recorrer con un bucle for.

Pila:

Una pila es un LIFO ( last in first out) el último que entra es el primero que sale.

Para añadir objetos a una pila se usa Push.

Para recuperar los elementos se usa Pop.

Cola:

Una cola es un FIFO (First in first out) el primero que entra es el primero que sale.

Para añadir objetos a la cola se usa Enqueue y para quitarlos se usa Dequeue.

2.3 Colecciones:Diccionario

Dictionary es básicamente un Hashmap en C#, funciona añadiendo claves/valor.

Se recorre con un for each.

2.4 Métodos de extensión.

Básicamente sirven para añadir métodos que no tienen a los tipos existentes sin tener que recompilar o modificar el original.

Son clases especiales estáticas pero se les llama como métodos de instancia.

2.5 System.Linq

Linq es un conjunto de métodos que se aplican sobre la interfaz IEnumerable y que permiten realizar operaciones sobre las colecciones que implementan Ienumerable.

La manera más común de usar las consultas Linq es usando funciones lambda.

Using System.linq->añade muchos métodos de extensión.

Algunas de las funciones que se pueden realizar con Linq son:

Where:

Hace una búsqueda en una colección

var songs = mediaList.Where(x=>x.Category == category)

return songs.toList()

Select

Convierte un listado en otro

mediaList.Select(x=>x.Title)

SelectMany

Convierte un conjunto de listados en uno de otro tipo

var namesOfArtists = mediaList.SelectMany(x => x.artists).Select(x => x.Name)

return namesOfArtists.ToList();

// Esto me recuerda a las consultas de Javi

Any

Any sirve para devolver cualquier valor encontrado

return mediaList.Any(x => x.Title == title);

First/FirstOrDefault

Es mejor usar FirstOrDefault ya que First si no encuentra el valor devuelve una excepción

mediaList.firstOrDefault( x=> x.Seconds <60);

Last/LastOrDefault

mediaList.lastOrDefault(x= x.Seconds <60);

haría lo mismo que el firstordefault pero devolvería la ultima que dure menos de 60 segundos

Skip

return mediaList.skip(10).take(10).ToList

en este caso el Skip sirve para saltarse 10 elementos y el take para coger los 10 siguientes

Take

return mediaList.take(10).ToList

El take recogerá los 10 primeros valores de la lista.

Se suelen utilizar en paginados.

Orderby/OrderByDescending

return mediaList.OrderBy(x => x.Visits).Take(10).ToList().

Ordena por el número de visitas si es un número, si fuese un string ordinaria alfabéticamente

OrderBy es para las primeras , OrderbyDescending para las últimas.

GroupBy

mediaList.GroupBy(X => x.Category).ToDictionary(x=>x.Key)

Groupby ordena por una categoría y el ToDictionary lo convierte a un Diccionario con la key pasada.

ToList y ToArray hacen lo mismo solo que uno a listas y el otro a arrays

// Si, son las consultas de Javi pero aplicadas a C# en consola

2.6 Genéricos:Funcionalidades

Los Constraints son restricciones a los tipos de datos que se pueden introducir lo que nos permite limitar los tipos de datos

public class PlayList<T> where T: Media

En este caso estaríamos diciendo que el genérico T es de tipo Media.

2.7 Async and await

IAsyncResult, Thread y BackgroundWorker estan por asi decirlo deprecated, para procesar la concurrencia en C# se usa el async/await.

Para programar un Async hay que usar la clase Task

public Task<byte[]> DownloadAsync(string songName){

Thread.Sleep(1000);

return Task.FromResult(new byte[]{});

}

private async static void DownloadSongAsync(){

DownloadService downloadServe = new DownloadService();

byte[] futureSong = await downloadServe.DoanloadAsync()

}

Básicamente haríamos una petición de espera mientras se descarga la canción y podríamos seguir codificando sin problemas y el programa funcionará sin congelarse ni pararse y que el usuario no se enfade.

2.8 System.Reflection

Activator:

Permite crear instancias del tipo T, esto nos permite crear instancias de distintos objetos sin tener ni idea del mismo.

static void doDomethingWithMyClass<T>() where T: new(){

Activator.CreateInstance<T>();

}

Atributos

[My(Count = 10)]

public class MyAttribute : Attribute

{

public string Name{}

public int Number

}

Básicamente sirve para crear nuestros propios atributos y habría que recogerlo usando por ejemplo un GetFirstOrDefault,

El uso del mismo es inimaginable debido a la variedad de opciones que tiene.

Se puede acceder a propiedades,métodos y campos usando los métodos GetProperties,GetMethods y GetFields independientemente de su nivel de acceso.

Assembly:

Permite conocer el ensamblado al que pertenece, su nombre y sus diferentes capacidades.

Performance

La reflexión es muy potente para realizar diferentes tareas, pero tiene un problema, su costra en tiempo de ejecución es mayor que acceder directamente, asi que tampoco conviene abusar de él.