Introducción a C#

4.1.- CONCEPTOS DE P.O.O.

C# → Lenguaje Orientado a Objetos

OBJETO : Bloque que contiene tanto datos como métodos
→ Los objetos se crean de tipos definidos llamados **clases**
→ Objeto es una **instancia de una clase** que actúa de plantilla

PROPIEDADES Y CAMPOS dan acceso a los datos del objeto.

Diferentes objetos instanciados de la misma clase pueden tener distintos valores de sus datos → **estado de un objeto**

→ Tanto las propiedades como los campos tienen un tipo.

→ Las propiedades se diferencian de los campos que éstas se acceden a través de métodos set y get y no directamente como los campos. (las vemos luego más en detalle)

→ En principio es mejor utilizar propiedades a campos, ya que proporcionan mayor control y desde el punto de vista del código el acceso es idéntico

Introducción a Visual C#

MÉTODOS

- Es el nombre que reciben las funciones que proporciona un objeto.

- Los métodos se utilizan para proporcionar acceso a la funcionalidad del objeto

- En carácter punto (.) separa el nombre del objeto de los métodos, propiedades o campos que se quiere acceder.

CICLO DE VIDA DE UN OBJETO : Construcción , Ejecución y Destrucción.

CONSTRUCCIÓN

→ Ocurre cuando es inicializado y se lleva a cabo por una función que se denomina CONSTRUCTOR

→La inicialización básica se lleva a cabo de forma automática (posicionar el objeto en memoria)

→ El constructor se encarga de inicializar el estado del objeto.

Introducción a Visual C#

DESTRUCCIÓN

→ Ocurre cuando un objeto es eliminado (cuando acaba el ámbito de ese objeto y ya no es accesible por el código)
Proceso : Garbage Collector del .NET runtime

→La inicialización básica se lleva a cabo de forma automática (quitar el objeto de memoria)

→ El destructor no es necesario especificarlo
De especificarlo se ejecutaría justo antes de eliminar el objeto.

→ método con el mismo nombre de la clase precedido por el carácter tilde (~)

Introducción a Visual C#

HERENCIA

- Cualquier clase puede heredar de otra.

- Contendrá todos los miembros de la clase de la que hereda

- Hablamos de clase padre y clase hija

- En C# una clase sólo puede heredar de una única clase padre

- Cuando una clase hereda de otra hay que tener en cuenta los distintos modificadores de acceso de los que disponemos : private (privado a la propia clase), public (publico para acceder desde la clase hija como de forma externa), protected (solo accesible desde la clase hija, pero no desde fuera)

- Una clase puede heredar (se dice implentar) de uno o varios intefaces

- Un interfaz puede heredar de uno o varios interfaces

    Más adelante veremos qué son los interfaces y para qué se utilizan

Introducción a Visual C#

POLIMORFISMO

- El polimorfismo consiste en poder asignar una variable de la clase hija a un variable de la clase padre

- En principio, únicamente podrán ser llamados los miembros que estén contenidos en la clase padre

- Polimorfismo a nivel de Interfaz :

- Aunque no se puede instanciar un interfaz, sí podemos crear variables de un interfaz y asignarles cualquier objeto que implemente este interfaz, pudiendo acceder a los métodos definidos en el interfaz

- Las clases hijas heredan también los interfaces que implementa la clase padre

Introducción a Visual C#

CONTENEDORES Y COLECCIONES

- Una clase puede contener objetos de otra clase

- Si es privado, no se tendrá acceso de forma directa a ningún miembro de la clase contenida, aunque sean públicos

- Colecciones son clases que pueden almacenar muchos objetos de otra clase (ArrayList, Diccionarios)

- A diferencia de los Arrays tu tamaño no es invariable y nos proporcionan funcionalidad adicional para su gestión

SOBRECARGA DE OPERADORES

- Para poder utilizar los operadores en objetos de nuestra clase, tenemos que sobrecargarlo dentro de la clase

- También se pueden sobrecargar operadores para que trabajen con dos clases distintas

- Sólo se pueden sobrecargar los operadores existentes, no se pueden crear nuevos operadores.

Introducción a Visual C#

EVENTOS

- Los objetos en C# pueden lanzar eventos, de forma similar a como funcionan las excepciones

Unity tiene su propio sistema de eventos

- De este modo se pueden crear aplicaciones basadas en eventos, que es un paradigma de programación distinto

TIPOS POR REFERENCIA vs TIPOS POR VALOR

- Los objetos se almacenan siempre por referencia, si no está inicializado contiene **null**

- Los tipos por referencia lo que se almacena es la dirección de memoria donde está el contenido, mientras que los tipos por valor almacenan el dato en sí mismo