Introducción a C#

4.1.- *CONCEPTOS DE P.O.O.*

OBJETO : Bloque que contiene datos y 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#

→ Es posible establecer un acceso de sólo lectura y de sólo escritura sobre las propiedades, especificando o no los métodos get y set.

→ Es posible tener distinta accesibilidad en los métodos get y set que nos definen la propiedad.

→ Una práctica común es definir campos privados (sólo accesible desde dentro de la clase) y proporcionar acceso controlado fuera de la clase mediante propiedades

MÉTODOS

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

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

Introducción a Visual C#

En C# y en .NET Framework TODO es un 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#

La definición de la clase contiene como mínimo un constructor

→ **Constructor por defecto**

→ Método que tiene el mismo nombre que la clase y no contiene parámetros

→ No es necesario definirlo, viene **implicito**

→ Podemos suministrar tantos constructores distintos al de por defecto como queramos.

→ Métodos que tienen el mismo nombre que la clase, pero con diferentes parámetros.

En C# los constructores se llaman a través de la palabra clave **new,** momento en el que se crea un objeto

Fuera de la clase no se pueden utilizar constructores definidos como private.

Se puede obligar a utilizar un constructor distinto del de por defecto definiendo el constructor por defecto como private

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#

MIEMBROS ESTATICOS (static)

→ Miembros de una clase que se comparte entre todas las instancias de una misma clase

- No es necesario instanciar las clases para utilizar los miembros estáticos de la misma

- Para dar un valor a un campo estático se puede dar de forma directa o a través de un constructor estático.
    - Una clase sólo puede tener un único constructor estático y no puede tener ni parámetros ni modificadores de acceso. Nunca son accesibles desde fuera de la clase, se ejecutan cuando se crea una instancia de la clase que tiene algún miembro estático o previamente cuando se accede a un miembro estático

- Las clases que todos sus miembros son estáticos son clases estáticas y no se pueden instanciar, pero sí podrían tener el constructor estático, ningún otro.

Introducción a Visual C#

INTERFACES

- Un interfaz es la definición de un grupo de métodos y propiedades que proporcionan una determinada funcionalidad

- Una clase puede implementar un interfaz, la clase debe soportar todos los métodos y propiedades del mismo.

- Un interfaz no se puede instanciar y los métodos o propiedades no pueden contener código alguno, solo su definición

- Una clase puede implementar múltiples interfaces y múltiples clase pueden implementar el mismo interfaz

- Un interfaz puede extender (heredar) de otro interfaz para ampliar la funcionalidad

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)

- Los métodos de una clase pueden ser virtual (se pueden sobrescribir en la clase hija mediante la palabra override)
    - Un miembro virtual no puede ser private

Introducción a Visual C#

CLASES ABSTRACTAS

- Una clase abstracta es una clase que tenga algún miembro abstracto

- Una clase abstracta no se puede instanciar, pero sí puede servir como clase base para heredar

- Los miembros abstractos no contienen ninguna implementación, debiendo proporcionar en la clase hija

CLASES SEALED

- Una clase sealed es aquella que no se permite heredar de ella.

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
    - A diferencia de los Arrays 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

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

Introducción a Visual C#

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

EVENTOS

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

- 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