PROGRAMACION II MARKETING

LECTURAS SEXTA SESION

https://www.ecured.cu/Clase_(Programaci%C3%B3n)

https://www.youtube.com/watch?v=oMWrJwMPd6k

https://www.youtube.com/watch?v=GZybFJJrWNc

Clase (Programación)

-

Clase. Es una construcción que permite crear tipos personalizados propios mediante la agrupación de variables de otros tipos, métodos y eventos. Una clase es como un plano. Define los datos y el comportamiento de un tipo. Si la clase no se declara como estática, el código de cliente puede utilizarla mediante la creación de objetos o instancias que se asignan a una variable. La variable permanece en memoria hasta todas las referencias a ella están fuera del ámbito. Si la clase se declara como estática, solo existe una copia en memoria y el código de cliente solo puede tener acceso a ella a través de la propia clase y no de una variable de instancia.

Clases

Concepto:	Una clase es una construcción que permite crear tipos personalizados propios mediante la agrupación de variables de otros tipos, métodos y eventos.

Tipos de clases

Una parte muy importante de la programación orientada a objetos son las clases, si no fuera por ellas ni siquiera habría objetos. Cada una tiene sus propias características y ventajas. Un programador que conoce estas características sabe cuando debe usar una y no otra, de manera que su desarrolla un buen software. Los tipos de clases son:

- **Class. Public**: Son muy comunes, accesibles desde cualquier otra clase en la misma librería (de otro modo hay que importarlas).
- **Class. Abstract**: Aquellas que tienen por lo menos un método abstracto. No implementan sus métodos, sino que dan las bases para que sean implementados en la herencia.
- **Class. Final**: Son las que terminan la cadena de herencia. Útiles por motivos de seguridad y eficiencia de un programa, ya que no permiten crear más sub-divisiones por debajo de esta clase.
- **Class. Synchronizable**: Especifica que sus métodos son sincronizados, evitando problemas con los thread (hilo de ejecución), de forma que estos no pueden empezar a correr un método si no ha acabado el otro.

Características fundamentales

- **Nombre de la clase**. Sirve para identificar a todos los objetos que tengan unas determinadas características.
- **Conjunto de atributos**. Datos miembros. El valor de los atributos representan el estado de cada objeto.
- **Conjunto de métodos**. Funciones miembro. Permite que los objetos cambien de estado, dependiendo del estado anterior que tuviera el objeto.
- **Niveles de acceso** para proteger ciertos miembros de la clase. Normalmente, se definirán como ocultos (privados) los atributos y visibles (públicos) los métodos.

Declaración

Las clases se declaran mediante la palabra clave class. El nivel de acceso precede a la palabra clave class. Como, en este caso, se utiliza public, cualquiera puede crear objetos a partir de esta clase. El nombre de la clase sigue a la palabra clave class. El resto de la definición es el cuerpo de clase, donde se definen el comportamiento y los datos. Los campos, propiedades, métodos y eventos de una clase se conocen colectivamente como miembros de clase. Aunque se utilizan a veces de forma intercambiable, una clase y un objeto son cosas diferentes. Una clase define un tipo de objeto, pero no es propiamente un objeto. Un objeto es una entidad concreta basada en una clase y, a veces, se denomina instancia de una clase.

Herencia

La herencia se realiza a través de una derivación, lo que significa que una clase se declara utilizando una clase base de la cual hereda los datos y el comportamiento. Una clase base se especifica anexando dos puntos y el nombre de la clase base. Cuando una clase declara una clase base, hereda todos los miembros de la clase base excepto los constructores. A diferencia de C++, una clase en C# solo puede heredar directamente de una clase base. Sin embargo, dado que una clase base puede heredar de otra clase, una clase puede heredar indirectamente de varias clases base. Además, una clase puede implementar directamente más de una interfaz. Una clase se puede declarar como abstracta. Una clase abstracta contiene métodos abstractos que incluyen una definición de firma pero ninguna

implementación. No se pueden crear instancias de las clases abstractas. Solo se pueden utilizar a través de clases derivadas que implementan los métodos abstractos. Por el contrario, una clase sellada no permite que otras clases deriven de ella.

Ejemplo

```
public class Person
{
    // Field
    public string name;

    // Constructor
    public Person()
    {
        name = "unknown";
    }

    // Method
    public void SetName(string newName)
    {
        name = newName;
    }
}
class TestPerson
{
    static void Main()
    {
        Person person = new Person();
        Console.WriteLine(person.name);

        person.SetName("John Smith");
        Console.WriteLine(person.name);

        // Keep the console window open in debug mode.
        Console.WriteLine("Press any key to exit.");
        Console.ReadKey();
    }
}
```

Ejemplo de una clase en C#