Programación Orientada a Objetos: Clases

¿*Qué es la programación orientada a objetos*?   
Es un modelo de programación donde el diseño de software esta organizado en entorno a datos y objetos en lugar de funciones y lógica.

Un ejemplo es en vez de tener la lógica de guardar un nuevo cliente en base de datos directamente en el objeto codeBehind de la vista. Creamos una clase y una función dentro de esa clase para guardar los datos del cliente y solo hacemos referencia a la clase, eh invocamos la función dentro de la clase.

Al poner nuestro código en una clase nos permite reusar este código desde cualquier parte del proyecto y no está limitado a nuestro objeto codeBehind.

Los cuatro pilares de la programación orientada a objetos son:

* **Herencia:** las clases secundarias heredan datos y comportamientos de la clase principal
* **Encapsulación:** contener información en un objeto, exponiendo solo información seleccionada
* **Abstracción:** solo exponer métodos públicos de alto nivel para acceder a un objeto
* **Polimorfismo:** muchos métodos pueden hacer la misma tarea

En las próximas practicas estaremos aplicando los conceptos de Herencia, Encapsulación, Abstracción, Polimorfismo.

Nuestras Clases y Objetos tiene Access Modifiers(Modificadores de Acceso) dependiendo de cual acceso le demos a la clase determinara su disposición de uso.



Ejemplo: “public” so podrá utilizar desde cualquier parte del proyecto. “private” solo se podrá usar en la clase o objeto donde este función, clase, variable o objeto.

## 1.- Clases

Una clase es un contenedor de almacenamiento de campos y métodos definidos por el programador.  
Básicamente, una clase combina los campos y métodos (función miembro que define acciones) en una sola unidad. En C#, las clases admiten polimorfismo, herencia y también proporcionan el concepto de clases derivadas y clases base.  
  
*Creación de un proyecto de Clases en C#:* En explorador de soluciones Sobre el Proyecto principal le damos clic derecho y nos posicionamos en New Project(nuevo proyecto) y damos clic en Add(Agregar).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Seleccionamos Class Library (Librería de Clases) y nombramos el nuevo Proyecto.Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Aquí podemos ver que nos creo un Proyecto con una clase por default Class1

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

*Creación de Clases en C#:*

Nomenclatura de Clases: En los proyectos de Telepro dividimos la lógica en una clase y los objetos en una clase aparte.

* **Clase de logica “*bllEncrypt*”:** La palabra “bll” + (Entidad o descripción sobre qué es lo que hace la clase) = “bll” + Encrypt (la logica encripta).
* **Clase de Objeto “*clsDataEncrypt*”:** La palabra “cls” + (descripción sobre qué usa la clase) = “cls” + DataEncrypt (Se informa o obtiene los datos de encryptacion).

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Creamos una Clase de Lógica para guardar nuestro Cliente en BD.

Damos clic derecho sobre en nuevo Proyecto de clases y nos posicionamos en Add(Agregar) y seleccionamos Class(Clase).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Nombramos nuestra Clase “bll” + “Cliente” (Entidad o descripción)Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Nos crea una nueva clase con el nombre bllCliente

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Llamado de función sobre clases en codeBehind de vista.

1. Agregamos referencia a Proyecto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Damos check en el nuevo Proyecto y le damos en “OK”

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

1. Relacionamos el Proyecto de BD al Proyecto de Clases.  
   Interfaz de usuario gráfica, Texto, Aplicación

   Descripción generada automáticamente  
     
   En bllCliente le indicamos ala clase que estaremos utilizando el proyecto ProdeskNet.BD col la palabra “using” y instanciamos nuestra clase “clsManejaBD”  
     
   Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

   Descripción generada automáticamente
2. Le ponemos public a nuestra clase para poder utilizarla en nuestro código de logica de nuestra webform. Agregamos nuestras propiedades que esta viajando para guardar a BD. Creamos nuestra lógica.  
   Texto

   Descripción generada automáticamente
3. Solo invocamos nuestra función en el evento “click”

Texto

Descripción generada automáticamente

## Documentación

* Documentación de POO: [Programacion Orientada a Objetos](https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP)
* Documentación de OBJETOS-CLASES: [Objetos y Clases](https://www.geeksforgeeks.org/c-sharp-class-and-object/#:~:text=A%20class%20is%20a%20user,derived%20classes%20and%20base%20classes.)
* Documentación de MODIFICADORES DE ACCESO: [Modificadores de acceso](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers)