**NOTAS DE DESARROLLO**

***Bootstrap:***

1. mx-auto: Funciona para centrar en el eje x(Hay que denominar altura y ancho del elemento para que funcione).

***CSS:***

1. vh: altura pantalla
2. vw: ancho pantalla

***VS:***

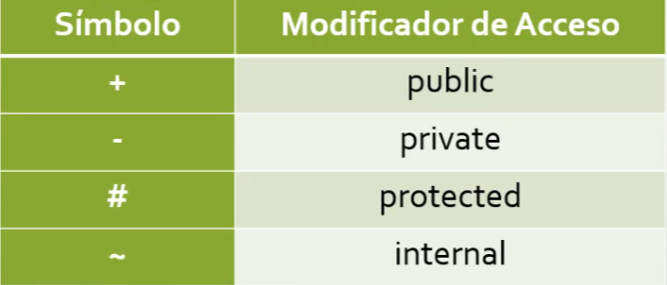
1. ViewData["Title"] = "Nombre";

***Flexbox:***

1. Inline-flex: El width se adapta al contenido que tiene dentro
2. Flex-wrap: wrap; Los elementos se adaptan al width
3. Order: orden;
4. Flex-grow: espacio entre columnas;
5. Flex-shrink: reducir tamaño caja;
6. Flex-basis: ancho base;
7. Justify-content: flex end,center,start, space-around(distribuye contenido con espaciado), space-evenly;
8. Align-items: center, flex-end,start; verticalmente(para que funcione se debe hacer el el div con # donde se define el display)
9. Align-self;

***C#:***

***Modificadores de Acceso:***

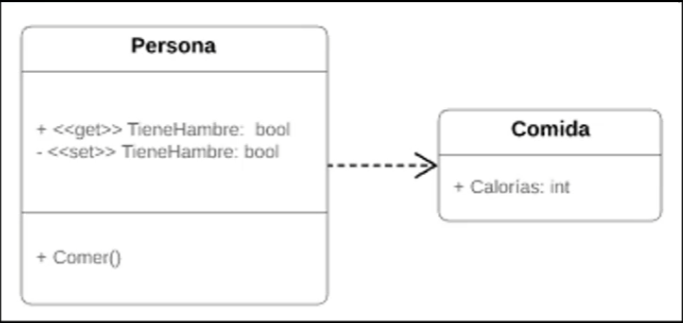
******

1. Public: puede ser utilizado desde donde sea
2. Private: puede ser accedido por código desde la misma clase
3. Internal : puede ser accedido por código del mismo proyecto
4. Protected: puede ser accedido por código de la misma clase o clase derivada
5. Protected Internal: puede ser accedido por código del mismo proyecto o clase derivada de otro proyecto
6. Private Protected : puede ser accedido por código del mismo proyecto desde la misma clase o clase derivada
7. Abstracción: Solo lo necesario en las clases
8. Encapsulamiento: Ocultar detalles de implementación de nuestras clases
9. Polimorfismo: programar contra abstracciones o generalizaciones y en tiempo de ejecución se invocarán los miembros de un tipo concreto adecuado

**Tipos de relaciones:**

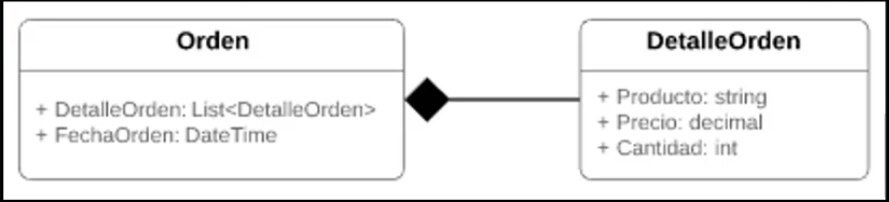
***Asociación:***

Una clase usa a la otra

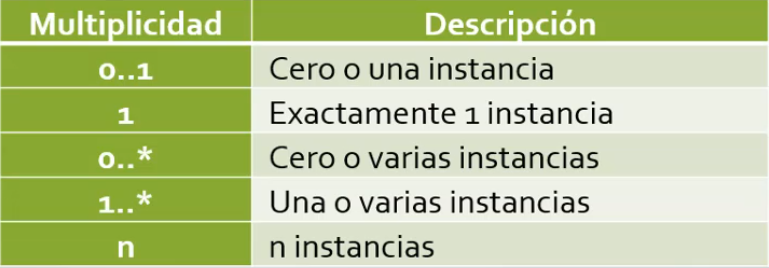


***Composición:***

Un objeto se compone del otro, donde la parte no tiene sentido sin la clase huésped

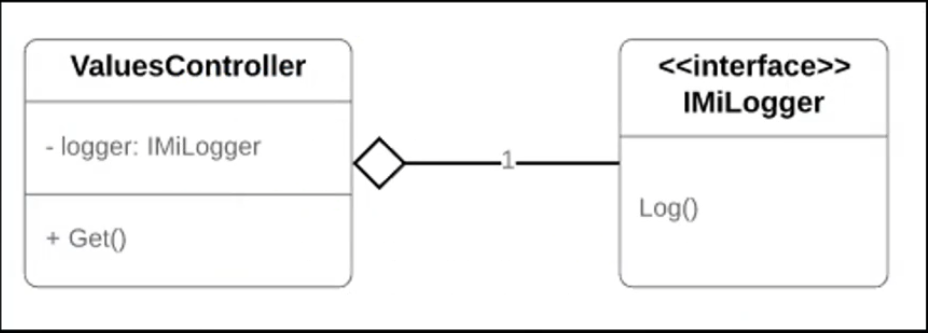


***Multiplicidad:***

******

***Agregado:***

Un objeto se compone del otro

******

**Tipos de Objetos:**

***Entidad:***

Representan conceptos muy relacionados con el problema en el que trabaja la aplicación

***Control:***

Coordinan las interacciones de varios objetos para realizar alguna tarea

**Frontera:**

Se encargan de interactuar con sistemas externos como bases de datos y Web Services

**Principios SOLID**

***Responsabilidad Única:***

Indica que los módulos deberían tener una sola razón para cambiar

***Abierto Cerrado:***

Indica que debemos implementar la variación protegida en nuestras clases, donde los detalles de implementación que pueden variar los colocamos en una clase aparte

***Luzhkov:***

Indica que si programamos contra una abstracción, cualquier tipo concreto adecuado debe de funcionar con nuestro código sin necesidad de alterarlo

***Segregación de interfaces:***

Indica que los clientes no deben ser forzados de depender de miembros los cuales no utilizan

***Inversión de dependencias:***

Indica que las clases deben depender de abstracciones

**Arquitectura N-Capas**

***Capas:***

Distintas partes de nuestra aplicación

***Dominio-Céntrica:***

Se les da mayor protagonismo a las entidades de negocio y a los objetos de control que coordinan los procesos. Los objetos de frontera que trabajan con sistemas externos tienden a ser detalles de implementación o infraestructura