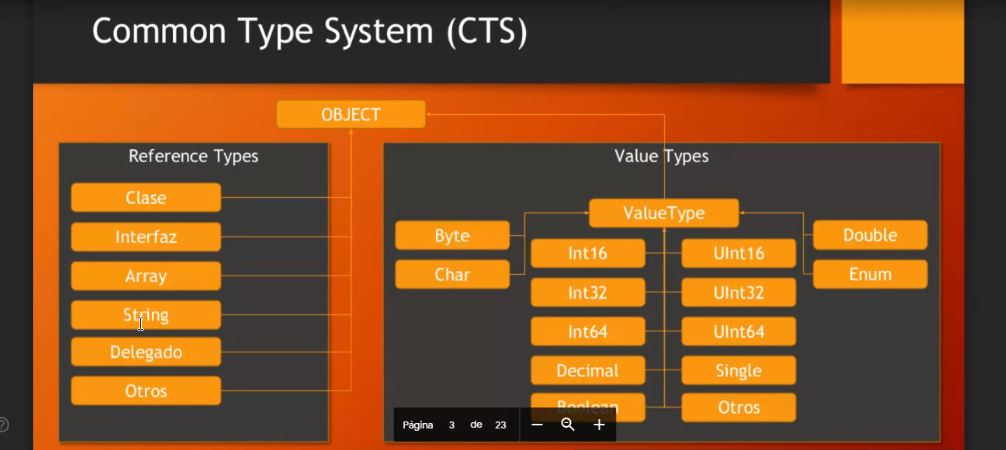
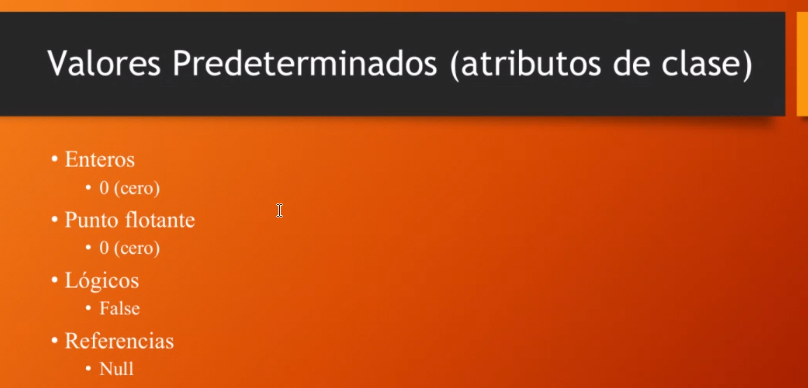
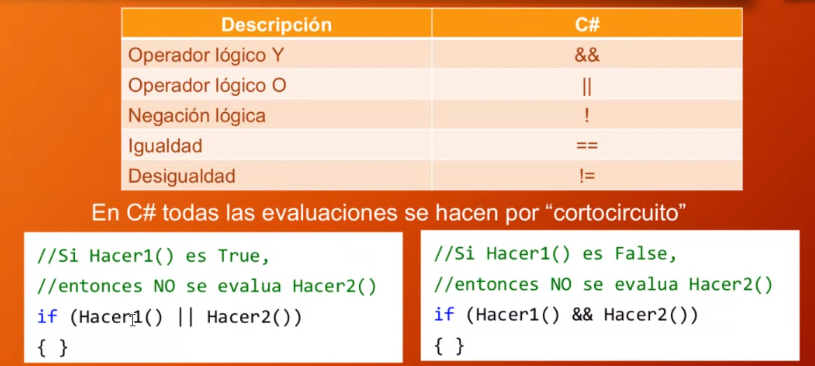
CTS (COMMON TYPE SYSTEM) – VALORES DE LA CLASE OBJETO, ya no se utiliza puntero como en C

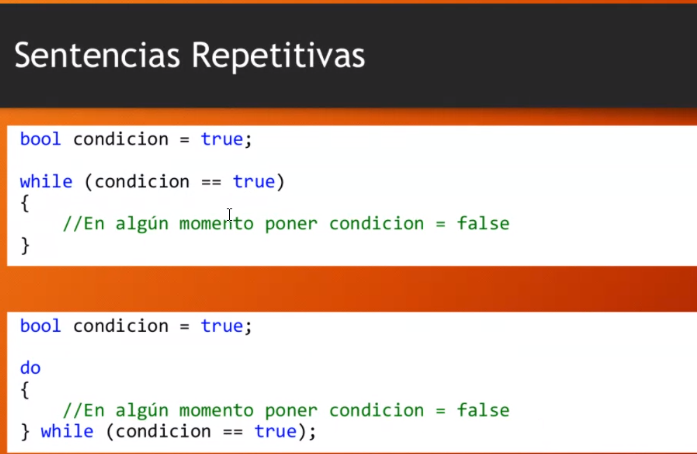


Valor: aloja en una ubicación de memoria esa variable declarada.

Referencia: aloja una dirección de memoria donde apunta al valor de la clase.

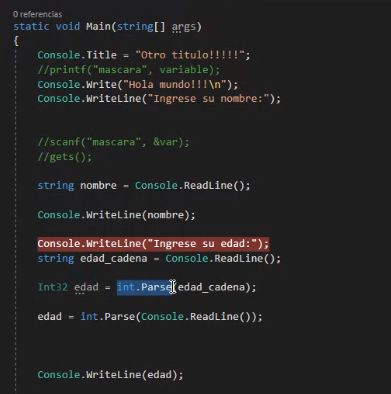
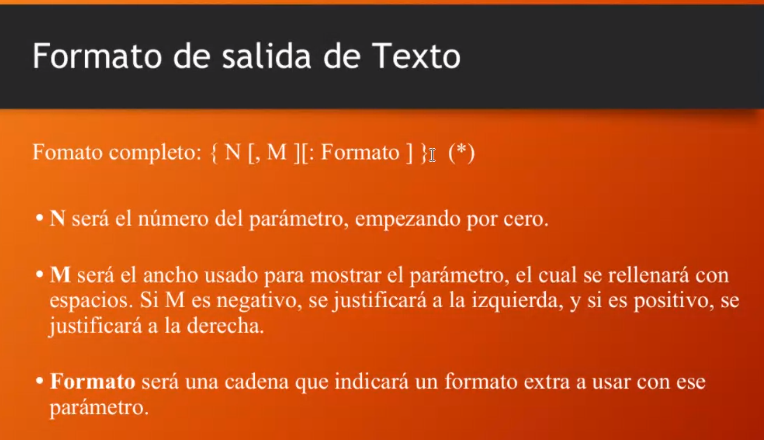




Foreach no permite utilizar índices.

Console.Write: permite escribir sobre la consola pero sin salto de línea, debo escribir \n.

Console.Writeline: permite escribir sobre la consola pero genera el enter, una línea nueva.



Los formatos varian solos dependiendo de la computadora de cada uno.

, para separar miles y . para separar caracteres

X = hexadecimal

E =

C = currency, formato moneda

D = decimal

G = fecha

N = decimales (N2 ,00)

O cambiar formato de fecha {0:dd/MM/yy},datetime.now;



Dd= valor numerico del dia (24)

Dddd=nombre del dia (Martes)

[\\de](file:///\\de) para mostrar literal de



Math.Pow = para elevar un numero (numerito,lo que elevo{2,3,etc})

Console.ReadKey(); para cuando finaliza el programa, para que lea

**POO: PROGRAMACION ORIENTADA A OBJETOS**

**(Clase seria el plano de una casa y un objeto es la casa completa)**

**Conceptos de poo**

-Abstraccion: nos permite diseñar nuestras distintas clases de la mejor forma posible. Se centra en las características fundamentales de su selección.

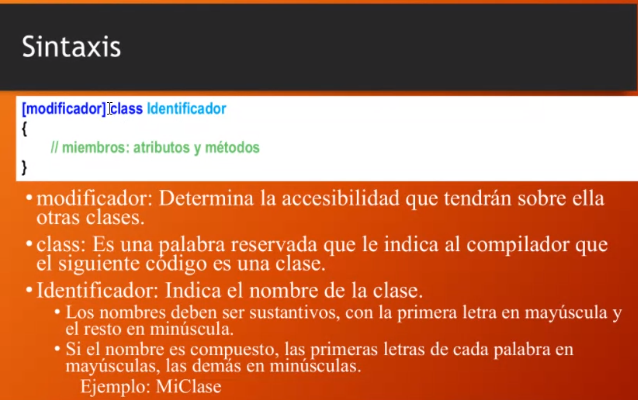
-Encapsulamiento: forma de ocultar funcionalidades al exterior de la clase

-Herencia: jerarquizar para reutilizar código

-Polimorfismo: manera de implementar un único método de distintas formas.

**Clase:** Estructura, esqueleto que van a tener mis objetos. A partir de la construcción de una clase, generamos un nuevo tipo de dato.

**Sintaxis:**

****

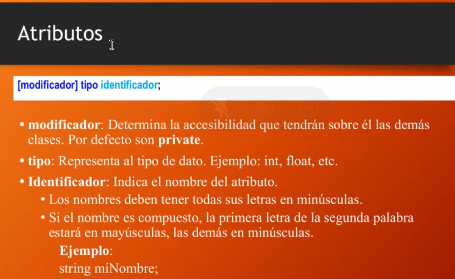
Modificadores de visibilidad: cuan accesible va a ser, si puedo verla en un ámbito o un ámbito mayor.

Modificadores de comportamientos: si va a cambiar la característica de la clase.

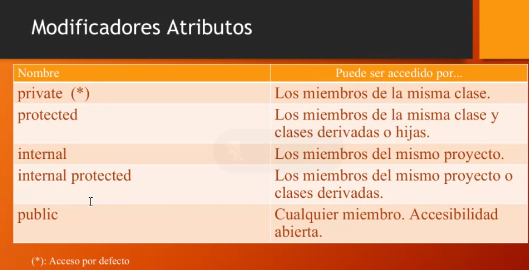


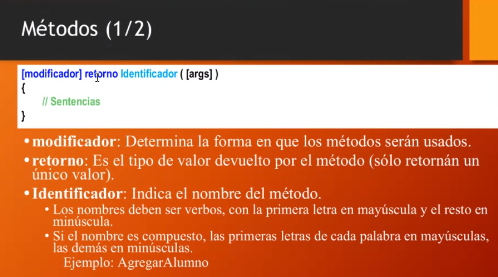
Los atributos son campos de variables como en C

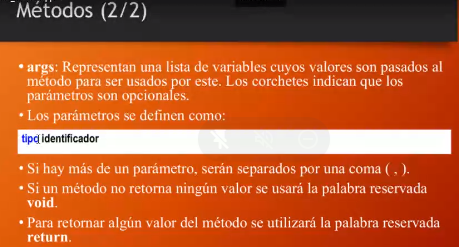
**PRIVATE ES EL POR DEFAULT, SE ASUME SI NO COLOCO NADA. Este implica que solo va a ser accedido dentro de los limites de la clase**



Modificadores de visibilidad para atributos:







Métodos no estáticos son llamados de instancia

Métodos estáticos son llamados de clase

Al llamar un método no estatico se usa la palabra reservada this.algun atributo

Para llamar un método no estatico en un lugar distinto que la clase, se requiere instanciar mi objeto con la palabra reservada new() para poder llamarlos. En cambio lo estáticos se los puede llamar con Miclase. Etc.

Lo estatico en todo el programa va a ser el mismo valor.

**WINDOWS FORMS**

**-necesitan tener al menos un constructor**

**-poseen objetos, métodos y atributos**

**- son objetos**

**-ciclo de vida de un form: Paint, new, activated, load, formClosing, otros**

**-partial en una clase nos permite definir parcialmente una clase, trabajar en la clase en diferentes archivos y separar la lógica de mi clase de los elementos que la conforman.**

**EXCEPCIONES: es un evento o acción que sucede en tiempo de ejecución.**

**Ocurren en tiempo de ejecución.**

**-Excepcion no controlada, finaliza la ejecución abruptamente.**

**Las excepciones son parte del objeto exception.**

**-Propiedad INNER EXCEPTION: almacena una excepción dentro de otra, asi la nueva exception puede contener info de la anterior. Para cargar la propiedad se debe pasar la instancia como argumento del constructor de la nueva excepción.**

**Si hay una sola excepción se devuelve null cuando no hubo otra, porque las excepciones son objetos y los objetos son tipos de referencia, por eso por defecto es null.**

**-Propiedad STACK TRACE: representa una pila de llamadas de forma LIFO. Lo bueno es que dice la clase, método y línea donde ocurrió. La primera de la lista es la última, ósea el origen del error.**

**-Bloque Try-Catch: try encapsula instrucciones que se podrían lanzar una excepción, catch maneja la excepción si ocurre.**

**Si en el bloque try no hay excepción, catch nunca se ejecuta y continua su flujo. Las excepciones son manejadas por el namespace System.Exception**

**-Bloque Finally, es la parte de código que debe ser ejecutada si hubo excepción o no. Es útil para evitar repetir instrucciones y para liberar los recursos luego de una excepción.**

**Problema grave de colecciones no genéricas: el recibir objetos provocaba un proceso de deboxing y unboxing y al querer recorrer un objeto para sacar un objeto es un proceso muy costoso. Por eso se recomienda usar colecciones en vez de usar un array.**

**INTERFACES:**

Es un medio común que nos permite agrupar distintas funcionalidades que podría implementar en una clase.