Contenido

[Introduccion a .net y C# 7](#_Toc114043200)

[¿Qué es .NET? ¿Cuáles son sus principales características? 7](#_Toc114043201)

[¿Cómo se categorizan las versiones de .NET? Explique su relación con las políticas de soporte y mantenimiento de dichas versiones. 7](#_Toc114043202)

[¿Qué componentes forman parte de una implementación de .NET? Describa brevemente cada uno. 7](#_Toc114043203)

[¿Qué es el Common Language Runtime? 8](#_Toc114043204)

[¿Qué es la Base Class Library? 8](#_Toc114043205)

[¿En qué se diferencia un marco de trabajo (framework) de una biblioteca (library)? 8](#_Toc114043206)

[Describa los estados y etapas del proceso de compilación de .NET. 8](#_Toc114043207)

[Explique y compare tiempo de compilación y tiempo de ejecución. 9](#_Toc114043208)

[C# 9](#_Toc114043209)

[Enumere y describa las características de C#. 9](#_Toc114043210)

[Explique la diferencia entre lenguajes de tipado estático y lenguaje de tipado dinámico. 9](#_Toc114043211)

[Defina sintaxis, semántica y vocabulario de un lenguaje de programación. 9](#_Toc114043212)

[Defina los siguientes términos: Sentencia (statement), variable y expresión (expression). ¿Cómo se relacionan? 10](#_Toc114043213)

[¿En qué se diferencian variables y constantes? 10](#_Toc114043214)

[¿Qué es el punto de entrada (entry point)? ¿Cuál es en los programas de C#? 10](#_Toc114043215)

[¿Qué es el Common Type System (CTS)? 10](#_Toc114043216)

[Explique las diferencias entre los tipos de referencia y los tipos de valor. 10](#_Toc114043217)

[¿Cuáles son las categorías de tipos de datos de .NET? Clasifique en tipos de valor y tipos de referencia. 11](#_Toc114043218)

[¿Qué es un alias? 11](#_Toc114043219)

[Ordene los siguientes tipos de datos del menos preciso al más preciso: decimal, float, double. 11](#_Toc114043220)

[¿Qué son los tipos object y dynamic? ¿En qué se diferencian? 11](#_Toc114043221)

[¿Para qué se usan los operadores nameof() y sizeof()? 12](#_Toc114043222)

[¿Cuál es el valor por defecto de los tipos de datos en .NET? 12](#_Toc114043223)

[¿En qué consiste el cortocircuito lógico de los operadores && y ||? 12](#_Toc114043224)

[¿Qué es el formato compuesto? ¿Cómo se aplica en el método WriteLine? Mencione cómo configurar el ancho de los campos, justificación a la izquierda o derecha, formatos estandar y formatos personalizados. 12](#_Toc114043225)

[Clases y métodos estáticos 13](#_Toc114043226)

[Métodos estáticos 13](#_Toc114043227)

[¿Qué propone el principio DRY? 13](#_Toc114043228)

[¿Qué modificadores podemos aplicar a un método y qué efecto tienen? 13](#_Toc114043229)

[Enumere y detalle las buenas prácticas a la hora de diseñar un método. 14](#_Toc114043230)

[¿Qué significa refactorizar? 14](#_Toc114043231)

[Namespaces 14](#_Toc114043232)

[¿Qué es un namespace y cuál es su función principal? 14](#_Toc114043233)

[¿Puedo tener distintos namespaces dentro de un mismo proyecto o ensamblado de .NET? 14](#_Toc114043234)

[¿Para qué se usa la directiva using? 14](#_Toc114043235)

[¿Para qué se usa la directiva alias? 15](#_Toc114043236)

[¿Puedo declarar dos clases independientes/distintas con el mismo identificador dentro del mismo namespace? ¿Y en namespaces distintos? 15](#_Toc114043237)

[Strings 16](#_Toc114043238)

[¿Es lo mismo declarar una variable como string (en minúscula) o como String (con la primera letra en mayúscula)? ¿Por qué? 16](#_Toc114043239)

[¿Se puede recorrer un string con un foreach? ¿Por qué? 16](#_Toc114043240)

[¿Qué significa que los strings son inmutables? ¿qué sucede en realidad cuando usamos métodos u operadores para modificar un string? 16](#_Toc114043241)

[Programación orientada a objetos 16](#_Toc114043242)

[¿Qué es un paradigma? ¿Qué propone el paradigma orientado a objetos? 16](#_Toc114043243)

[Nombre a los pilares del paradigma orientado a objetos. 16](#_Toc114043244)

[Describa el pilar de la abstracción y de un ejemplo que no se haya mencionado en los apuntes o en clase. 16](#_Toc114043245)

[¿Qué es una clase? 17](#_Toc114043246)

[¿Qué es un objeto? ¿En qué se relacionan con las clases? 17](#_Toc114043247)

[¿Qué significa instanciar un objeto? 17](#_Toc114043248)

[Explique qué es la abstracción en el contexto de programación orientada a objetos y cuál es su relación con las clases. 17](#_Toc114043249)

[¿Qué son los atributos o estado de un objeto? 17](#_Toc114043250)

[¿Qué son los métodos de un objeto? 17](#_Toc114043251)

[Windows Forms 18](#_Toc114043252)

[¿Los formularios son objetos? 18](#_Toc114043253)

[¿En qué parte del código se instancia el primer formulario? 18](#_Toc114043254)

[¿De qué clase heredan/derivan todos los formularios? 18](#_Toc114043255)

[¿Qué es una *partial class* o clase parcial? 18](#_Toc114043256)

[¿Puedo agregar parámetros de entrada al constructor de la clase del formulario? ¿Puedo sobrecargarlo? ¿Por qué? 18](#_Toc114043257)

[¿Se pueden declarar nuevos campos/atributos/propiedades dentro del formulario? ¿Por qué? 18](#_Toc114043258)

[¿Cuál es la diferencia entre los métodos Show y ShowDialog? 18](#_Toc114043259)

[¿Qué es un formulario MDI? ¿Y uno SDI? 18](#_Toc114043260)

[¿Con qué propiedad indico que un formulario es un contenedor MDI? 19](#_Toc114043261)

[¿Modificando qué propiedad del formulario hijo indico cuál es el formulario MDI padre? 19](#_Toc114043262)

[¿Qué es un *evento* en Windows Forms? Mencione algunos eventos de los controles o de los formularios y explique cómo trabaja con ellos. ¿Cuál es el *manejador del evento*? 19](#_Toc114043263)

[Explique el ciclo de vida de los formularios asociándolo a sus eventos. 20](#_Toc114043264)

[Colleciones 20](#_Toc114043265)

[Arrays 20](#_Toc114043266)

[¿Qué es un array "jagged"? ¿En qué valor se inicializan sus elementos? 20](#_Toc114043267)

[¿Los arrays son objetos? 21](#_Toc114043268)

[¿Qué significa que en C# los arrays son de "base-cero"? 21](#_Toc114043269)

[Los arrays implementan la interfaz IEnumerable, ¿qué es lo que esto les permite hacer? 21](#_Toc114043270)

[¿Cómo se declara e instancia un array multidimensional? 21](#_Toc114043271)

[¿El método Array.Resize cambia el tamaño de la instancia de array proveída o genera una nueva con distinto tamaño? ¿Por qué? 22](#_Toc114043272)

[Considerando que la propiedad Length devuelve la cantidad TOTAL de elementos de TODAS las dimensiones del array, ¿qué valor vamos a mostrar por consola en el siguiente código? 22](#_Toc114043273)

[COLECCIONES 22](#_Toc114043274)

[¿Cuál es la diferencia entre colecciones y matrices? 22](#_Toc114043275)

[¿Cuál es la diferencia entre las colecciones genéricas y las no-genéricas? 22](#_Toc114043276)

[¿Cuál es la diferencia entre una cola (Queue) y una pila (Stack). Asocie con los conceptos "FIFO" y "LIFO". 22](#_Toc114043277)

[Describa las siguientes colecciones: List, Dictionary, SortedList, Arraylist, Hashtable. Clasifíquelas en genéricas o no-genéricas. 23](#_Toc114043278)

[¿Por qué piensa que las colas y pilas no traen un método para ordenarlas? Piense en el uso de dichas colecciones. 23](#_Toc114043279)

[¿Cuál es la diferencia entre colas y pilas genéricas y no-genéricas? 23](#_Toc114043280)

[¿Cual es la salida del siguiente código? 23](#_Toc114043281)

[ENCAPSULAMIENTO 24](#_Toc114043282)

[ENCAPSULAMIENTO 24](#_Toc114043283)

[Defina encapsulamiento / encapsulación en el contexto del paradigma orientado a objetos. 24](#_Toc114043284)

[Defina cada nivel de ocultamiento / accesibilidad de la programación orientada a objetos. Incluya la función del modificador internal en C#. 24](#_Toc114043285)

[PROPIEDADES 24](#_Toc114043286)

[¿Qué es y para qué sirve una propiedad? 24](#_Toc114043287)

[¿Para qué sirve el descriptor de acceso get? 25](#_Toc114043288)

[¿Para qué sirve el descriptor de acceso set? ¿Cuál es el papel de la palabra clave value? 25](#_Toc114043289)

[¿Cómo declaro una propiedad de sólo lectura? 25](#_Toc114043290)

[¿Cómo declaro una propiedad de sólo escritura? 25](#_Toc114043291)

[ENUMERADOS 25](#_Toc114043292)

[¿Qué es un enumerado? ¿Cuál es su función? 25](#_Toc114043293)

[¿Un enumerado sólo puede estar anidado dentro de una clase? 25](#_Toc114043294)

[¿Los tipos enumerados son de valor o de referencia? 25](#_Toc114043295)

[Por defecto, ¿cuál es el primer valor numérico de un enumerado? ¿se pueden cambiar los valores por defecto? 25](#_Toc114043296)

[Indique los valores asociados a cada constante: 26](#_Toc114043297)

[Excepciones 26](#_Toc114043298)

[¿Qué es una excepción? 26](#_Toc114043299)

[¿Qué es el manejo de excepciones (exception handling)? (Desde lo conceptual / teórico) 26](#_Toc114043300)

[¿Las excepciones son objetos? 26](#_Toc114043301)

[¿Qué tienen en común todas las excepciones? 26](#_Toc114043302)

[¿Qué es el Stack Trace o pila de llamadas? ¿En qué orden se lee? 26](#_Toc114043303)

[¿Cuál es la función de cada bloque dentro de una estructura try-catch? 26](#_Toc114043304)

[En el caso de una misma estructura try-catch que tiene más de un tipo de bloque catch, ¿Se podría ejecutar más de un bloque catch? ¿Por qué? 27](#_Toc114043305)

[¿En qué parte del código continúa la ejecución del programa una vez manejada una excepción? 27](#_Toc114043306)

[¿Existe una forma de capturar cualquier excepción sin importar su tipo? ¿Qué habría que considerar en ese caso si se tiene más de un tipo de bloque catch en la misma estructura? 27](#_Toc114043307)

[¿Qué sucede cuando se lanza una excepción? ¿Qué sucede si no la manejo / controlo? 27](#_Toc114043308)

[¿Cómo lanzo una excepción? 27](#_Toc114043309)

[Dentro de un bloque catch, ¿cuál es la diferencia entre "throw;" y "throw ex;"? (ex es un identificador para la instancia de una excepción capturada) 27](#_Toc114043310)

[¿Se lanzan en tiempo de compilación o de ejecución? ¿Por qué? 27](#_Toc114043311)

[¿Cómo creo una excepción propia? 27](#_Toc114043312)

[¿Qué es la propiedad InnerException? Describa a qué clase pertenece, su contenido y qué tipo de dato almacena. 27](#_Toc114043313)

[¿Cómo se carga un valor en la propiedad InnerException? 27](#_Toc114043314)

[¿Qué sucede si no se le proporciona un valor a la propiedad InnerException? ¿Por qué? 28](#_Toc114043315)

[De existir un bloque finally. ¿Bajo qué condiciones se ejecutará el código que contiene? 28](#_Toc114043316)

[¿Para qué se utiliza el bloque finally? 28](#_Toc114043317)

[¿Dónde se ubica el bloque finally dentro de una estructura de manejo de excepciones? 28](#_Toc114043318)

# Introduccion a .net y C#

## ¿Qué es .NET? ¿Cuáles son sus principales características?

.NET es una plataforma gratuita y de código abierto que provee una serie de herramientas y programas para construir software de forma fácil. También nos provee el entorno necesario para ejecutarlo sobre distintos SO y tipos de arquitecturas.

Podemos nombrar como características que .NET es multiplataforma, open source y multilenguaje.

## ¿Cómo se categorizan las versiones de .NET? Explique su relación con las políticas de soporte y mantenimiento de dichas versiones.

Se categorizan en dos grandes grupos que determinan por cuantos años tendrán soporte.

* ***Long-Term Support (LTS):*** Version estable que se actualiza con poca frecuencia. Tiene soporte durante 3 años.
* ***Current:*** Tiene las últimas mejoras y se actualizan con frecuencia. Reciben actualización hasta 18 meses después de su reléase.

## ¿Qué componentes forman parte de una implementación de .NET? Describa brevemente cada uno.

Todas las implementaciones de .NET tienen los siguientes componentes:

1. Uno o más entornos de ejecución o runtime
2. Una biblioteca de clase base
3. Infraestructura y componentes comunes
4. Uno a mas frameworks para desarrollo de aplicaciones
5. Herramientas de desarrollo adicionales.
6. Common language runtime (CLR) es el programa encargado de administrar recursos, abstraer a nuestras aplicaciones de la maquina sobre la que se ejecutan, compilar nuestro código, manejar errores, soporte multi-hilo y cuestiones de seguridad y rendimiento.
7. Base Class Library (BCL) es una serie de bibliotecas con funcionalidades de uso general. Funciona como framework de bajo nivel sobre el que están construidos los de alto nivel.
8. Integra herramientas como
   * Lenguajes y sus compiladores
   * .NET CLI: interfaz de línea de comandos
   * MSBuild: un motor para cargar y construir aplicaciones
   * NuGet: administrador de paquetes para incorporar librerías externas.
9. Frameworks como Windows forms, ASP.NET core y WPF
10. Acepta “plugins”

## ¿Qué es el Common Language Runtime?

Common language runtime (CLR) es el programa encargado de administrar recursos, abstraer a nuestras aplicaciones de la maquina sobre la que se ejecutan, compilar nuestro código, manejar errores, soporte multi-hilo y cuestiones de seguridad y rendimiento.

## ¿Qué es la Base Class Library?

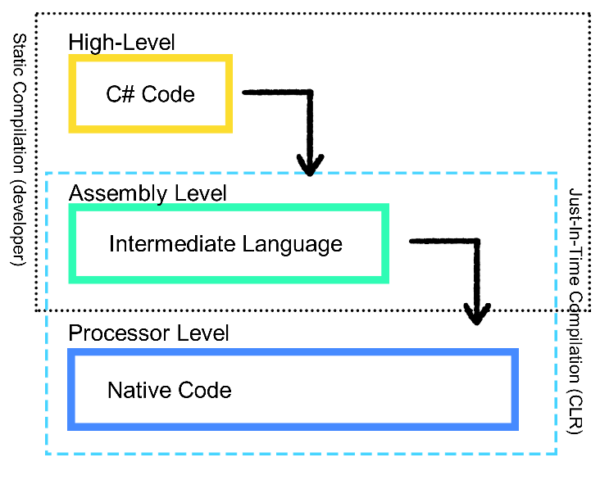
Base Class Library (BCL) es una serie de bibliotecas con funcionalidades de uso general. Funciona como framework de bajo nivel sobre el que están construidos los de alto nivel.

## ¿En qué se diferencia un marco de trabajo (framework) de una biblioteca (library)?

La librería contiene funcionalidades específicas y yo tengo la libertad de implementarla como me guste y a necesidad. Mientras que un framework tiene un conjunto de herramientas las cuales tengo que usarlas en el estándar definido de dicho framework

## Describa los estados y etapas del proceso de compilación de .NET.

El proceso de compilación pasa por tres estados y se divide en dos etapas.



Etapa 1 : Compilación estatica:

El código de alto nivel (C#) se compila en un lenguaje ensamblador intermedio. Esto se logra mediante el build y su resultado son archivos en este lenguaje intermedio que se distribuirá a usuarios finales. Los tan famosos .exe y dll para las bibliotecas.

Etapa 2: Just in time (JIT):

EL SO invoca al CLR, el cual compila el lenguaje intermedio a uno nativo por la maquina en un proceso llamada JIT. Esto abstrae al código fuente de la maquina y el SO sobre el que se terminará ejecutando el programa. Una desventaja es que es un poco mas lento ya que debe traducir ese lenguaje intermedio a uno que entienda la maquina.

## Explique y compare tiempo de compilación y tiempo de ejecución.

* ***Tiempo de compilación:*** Es cuando compilamos nuestro código a un ejecutable. Se realizan optimizaciones y chequeos de sintaxis.
* ***Tiempo de ejecución (runtime):*** Es el tiempo en el que un programa se ejecuta en el SO. Comienza cuando el programa se pone en memoria y termina cuando este finaliza, sea por que así se quiso o por algún error.

# C#

## Enumere y describa las características de C#.

* Orientado a objetos con soporte a herencia, polimorfismo y encapsulamiento.
* Orientado a componentes donde un componente encapsula un conjunto de datos o funciones relacionadas.
* Tipado estático/fuerte
* Garbage collector que permite la liberación de memoria no utilizadas en el heap.
* Sistema de tipo unificado donde todos heredan de System.Object
* Case sensitive donde se distinguen mayus de minus.

## Explique la diferencia entre lenguajes de tipado estático y lenguaje de tipado dinámico.

Tipado estático es cuando tenemos que definir el tipo de dato antes de compilar el programa, mientras que Tipado dinámico se definen durante la ejecución del programa.

## Defina sintaxis, semántica y vocabulario de un lenguaje de programación.

En un lenguaje de programación la sintaxis nos define las reglas en el que se escribe, combinaciones correctas y símbolos.

La semántica es el significado que surge de la combinación de estas reglas (sintaxis) y el vocabulario son las palabras reservadas y operadores que nos permite construir expresiones siguiendo una sintaxis valida.

## Defina los siguientes términos: Sentencia (statement), variable y expresión (expression). ¿Cómo se relacionan?

Una sentencia (statement) es el análogo a una oración en el español. Puede estar compuesta por variables (lugar reservado reservado en memoria para luego ser usado) y expresiones. La finalidad de esta está marcada con punto y coma.

## ¿En qué se diferencian variables y constantes?

Ambas nos sirven para almacenar datos en memoria pero la principal diferencia es que la variable como su nombre lo indica puede cambiar en el tiempo y una constante permanece inmutable, es decir que sus valores se definen en tiempo de compilación y no cambian durante el resto de la vida del programa.

## ¿Qué es el punto de entrada (entry point)? ¿Cuál es en los programas de C#?

El método Main es el entry point de todos los programas en C#.. Es el primer método en ejecutarse.

## ¿Qué es el Common Type System (CTS)?

Define el conjunto de tipo de datos en todos los lengaujes soportados de .NET

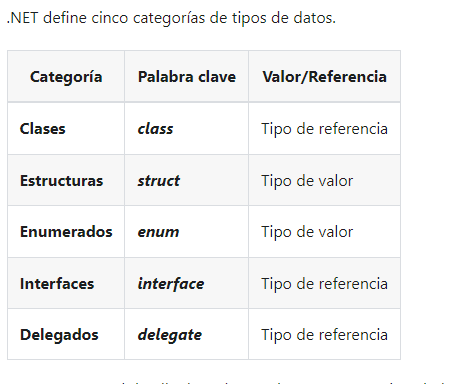
* Establece un marco de herramientas que habilita la ejecución de los distintos lenguajes de la platforma
* Provee modelo orientado a objetos
* Define conjunto de relgas que todos los lenguajes deben seguir en lo referido a tipos de datos
* Provee los tipos primitivos básicos.
* Define tipos de datos en dos categorías. Por valor y por referencia.

## Explique las diferencias entre los tipos de referencia y los tipos de valor.

Por valor: Representan su valor real. Si se asignan a una variable esta obtiene una nueva copia del valor. Se almacenan en la pila.

Por referencia: representados por una referencia a un sector de la memoria en donde se aloja el valor real. Se almacenan en el heap.

## ¿Cuáles son las categorías de tipos de datos de .NET? Clasifique en tipos de valor y tipos de referencia.



## ¿Qué es un alias?

Son palabras clave que representan tipos proveidos por la plataforma .NET. Por ejemplo el tipo de dato que conocemos como INT en realidad es un alias de System.Int32.

## Ordene los siguientes tipos de datos del menos preciso al más preciso: decimal, float, double.

Float < doublé < decimal

## ¿Qué son los tipos object y dynamic? ¿En qué se diferencian?

Son tipos “comodin” que permiten inferir el tipo de dato sin importar cual sea. Su diferencia radica en que object necesita un casteo explicito para poder usar algún método del tipo inferido mientras que dynamic no.

Ejemplo: Object palabra = sads  
para usar el método .length necesito castear palabr a string .

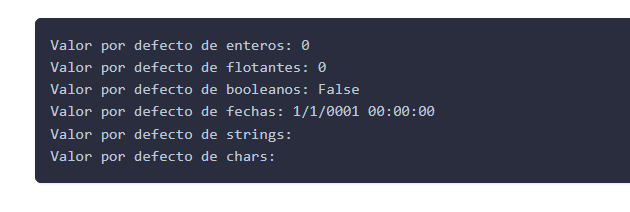
((string ) palabra). length

## ¿Para qué se usan los operadores nameof() y sizeof()?

Nameof() me permite saber como se llama esa variable pasada por argumento.

Sizeof() permite saber la cantidad de bytes en memoria que ocula el tipo de dato pasado por parámetro.

## ¿Cuál es el valor por defecto de los tipos de datos en .NET?

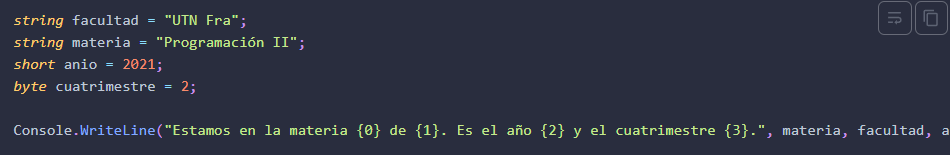


## ¿En qué consiste el cortocircuito lógico de los operadores && y ||?

Con ser al menos uno true o false ya no verifica la otra condición

## ¿Qué es el formato compuesto? ¿Cómo se aplica en el método WriteLine? Mencione cómo configurar el ancho de los campos, justificación a la izquierda o derecha, formatos estandar y formatos personalizados.

Es una combinación de texto y marcadores de posición los cuales corresponden a cada variable pasada para que se escriban.

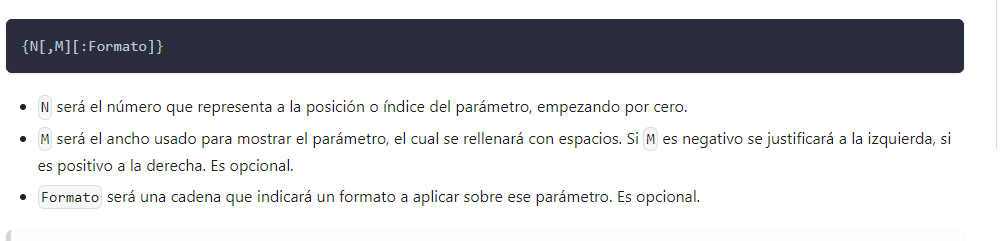


Primero se debe escribir entre llaves la posición o índice de la variable que queremos imprimir. {1} indica que allí se deberá insertar la representación en texto de la variable que está en la segunda posición.

Se puede definir el ancho del campo agregando una coma y un valor. Si el valor es positivo la cadena se alinea a la derecha, si es negativo se alinea a la izquierda. {1,10} aplica al valor de la variable en la segunda posición un ancho de columna de 10 caracteres y alineación a la derecha.

También se puede aplicar un formato específico agregando al marcador el símbolo de dos puntos : seguido de la cadena de formato. {1,10:N2} aplica al valor del marcador el formato N2 que representa un número con separadores de dígitos y hasta 2 decimales.





# Clases y métodos estáticos

## Métodos estáticos

## ¿Qué propone el principio DRY?

Significa “no te repitas”. Toda funcionalidad debe ser única y no debe de tener más de una interpretación. Debe ser entendida sin posibilidad de duda o equivocación. Tambien significa que se puede confiar. Es decir que si se hace un cambio, no se necesitaría actualizar multiples cosas en paralelo.

## ¿Qué modificadores podemos aplicar a un método y qué efecto tienen?

Existen modificadores de acceso los cuales limitan desde donde se puede interactuar con un métodos u otros tipos de miembros.

* ***Public:*** Accesible desde cualquier parte.
* ***Private:*** Solo accesible desde dentro del mismo tipo
* ***Internal:*** Solo accesible desde dentro del mismo ensamblado/proyecto.
* ***Protected:*** solo accesible desde dentro del mismo tipo o tipos derivados(hijos)

## Enumere y detalle las buenas prácticas a la hora de diseñar un método.

El nombre del método debe ir en pascal case (NombreMetodo) y los parámetros en camelCase (paramUno)

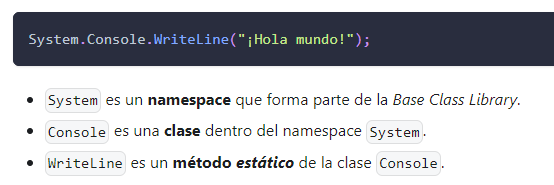
## ¿Qué significa refactorizar?

Es una técnica para reestructurar el código alterando su estructura interna sin cambiar comportamiento externo. No arregla errores ni añade funcionalidad, solo mejora la comprensión, estructura y diseño del código.

## Namespaces

## ¿Qué es un namespace y cuál es su función principal?

Es una agrupación lógica de clases y otros elementos del código fuente. Su función es la organización de código, permitiendo la reducción de coalision de nombres y la posibilidad de trabajar en un mismo programa con componentes de distinta procedencia.



## ¿Puedo tener distintos namespaces dentro de un mismo proyecto o ensamblado de .NET?

Si, puede haber mas de un namespace por proyecto asi como también podría llamarse diferente al nombre del proyecto.

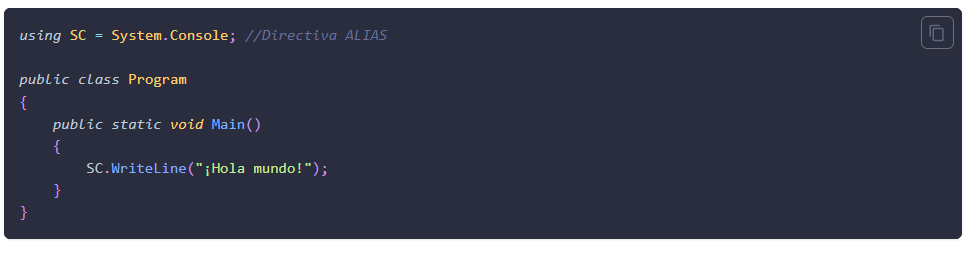
## ¿Para qué se usa la directiva using?

Permite la llamada a un método sin el uso obligatorio de su nombre completo.



## ¿Para qué se usa la directiva alias?

Permite utilizar un nombre distinto para un espacio de nombres. Se suele utilizar para abreviar nombres largos.



## ¿Puedo declarar dos clases independientes/distintas con el mismo identificador dentro del mismo namespace? ¿Y en namespaces distintos?

Dentro del mismo namespace no pero si están en diferentes si.

## Strings

## ¿Es lo mismo declarar una variable como string (en minúscula) o como String (con la primera letra en mayúscula)? ¿Por qué?

No, el lenguaje es case sensitive, por lo tanto se tomará como si fueran diferentes.

## ¿Se puede recorrer un string con un foreach? ¿Por qué?

Se puede recorrer con un foreach ya que un string es un array de caracteres.

## ¿Qué significa que los strings son inmutables? ¿qué sucede en realidad cuando usamos métodos u operadores para modificar un string?

Significa que no se puede cambiar su valor. Para “hacerlo” el método que lo hace copia su valor, destruye el anterior y retorna el modificado en base al anterior.

# Programación orientada a objetos

## ¿Qué es un paradigma? ¿Qué propone el paradigma orientado a objetos?

Un paradigma es una teoría o conjunto de ellas que se acepta sin cuestionar que se utiliza para resolver problemas y avanzar en el conocimiento.

En el caso de la programación define la forma o metodologías con el que se resolverán los problemas utilizando un lenguaje de programación.

La POO propone resolver problemas través de identificar objetos de la vida real dándole atributos y comportamientos.

## Nombre a los pilares del paradigma orientado a objetos.

La POO se sostiene sobre 4 pilares:

* ***Abstracción:*** capturar la idea principal de un objeto dentro de un contexto e ignorando detalles que no sean necesarios.
* ***Encapsulamiento:*** Se trata de agrupar datos y acciones relacionados a un objeto. También se define desde donde y como puede ser accedido y/o modificado.
* ***Herencia:*** Reutilización de código por parte de una clase más general que comparte estructura y funcionalidad a una clase más específica. Se pregunta “es un”.
* ***Polimorfismo:*** Una clase hija redefine una acción heredad en base a un comportamiento.

## Describa el pilar de la abstracción y de un ejemplo que no se haya mencionado en los apuntes o en clase.

Abstraccion significa capturaar la idea principal de un objeto dentro de un contexto e ignorando los detalles que no son necesarios.

Por ejemplo cuando hablo de alguna “persona”. Yo digo su nombre y algún dato que necesite en el contexto dado. No voy a describir detalles innecesarios como cuanto mide o que signo zodiacal tiene.

## ¿Qué es una clase?

Una clase es un molde, una plantilla la cual es utilizada para saber que atributos y de qué forma se va a comportar un objeto.

## ¿Qué es un objeto? ¿En qué se relacionan con las clases?

Un objeto es una instancia de la clase, una entidad concreta que basada en una clase. Un bloque de memoria que se ha asignado y configurado de acuerdo al molde.

## ¿Qué significa instanciar un objeto?

Crear un objeto basada en la clase y reservar/referenciar su espacio en memoria mediante la palabra new.

## Explique qué es la abstracción en el contexto de programación orientada a objetos y cuál es su relación con las clases.

Las clases las voy a crear basándome en un modelo de negocio y prescindiendo de ciertos detalles que no voy a necesitar en el contexto actual.

## ¿Qué son los atributos o estado de un objeto?

Los atributos del objeto son los datos que pueden o no variar durante su ciclo de vida. Estos me describen como es el objeto y se definen como sustantivos.

## ¿Qué son los métodos de un objeto?

Un método es una función dentro de una clase/objeto. Este me indica cómo se va a comportar y se definen como verbos en PascalCase. Ejemplo HacerCalculo()

# Windows Forms

## ¿Los formularios son objetos?

Tanto los formularios como todos los elementos que se le pueden agregar son objetos.

## ¿En qué parte del código se instancia el primer formulario?

Dentro del program, el main

## ¿De qué clase heredan/derivan todos los formularios?

Los formularios heredan de la clase Form

## ¿Qué es una *partial class* o clase parcial?

Una clase parcial es una clase subdividida en dos partes. La parte del código y el manejo del formulario y una .designer.cs que lo que contiene son todos los controles y diseños del formulario.

## ¿Puedo agregar parámetros de entrada al constructor de la clase del formulario? ¿Puedo sobrecargarlo? ¿Por qué?

Si se puede agregar parámetros de entrada o sobrecargar el constructor. Esto me permite poder pasar información de un formulario a otro.

## ¿Se pueden declarar nuevos campos/atributos/propiedades dentro del formulario? ¿Por qué?

Si se puede. Esto permite personalizar mas al formulario para lograr mi cometido.

## ¿Cuál es la diferencia entre los métodos Show y ShowDialog?

Show: Me muestra al formulario. Si yo tengo 3 forms con show, se me van a mostrar los 3 y puedo cambiar su foco entre ellos.

ShowDialog: Me muestra un formulario pero este no pierde el foco hasta que sea terminado.

## ¿Qué es un formulario MDI? ¿Y uno SDI?

Un formulario MDI (multiple document interface) puede mostrar muchos ventanas/forms hijas dentro de el.

Un formulario SDI (single document interface) puede manipular solamente un documento a la vez.

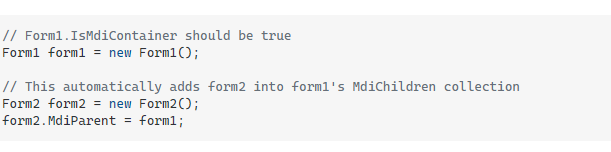
Un ejemplo de esto podría ser el ide de visual studio (MDI) y el notepad (SDI).

## ¿Con qué propiedad indico que un formulario es un contenedor MDI?

Al componente padre se le debe setear la propiedad ISMdiContainer = true;

## ¿Modificando qué propiedad del formulario hijo indico cuál es el formulario MDI padre?

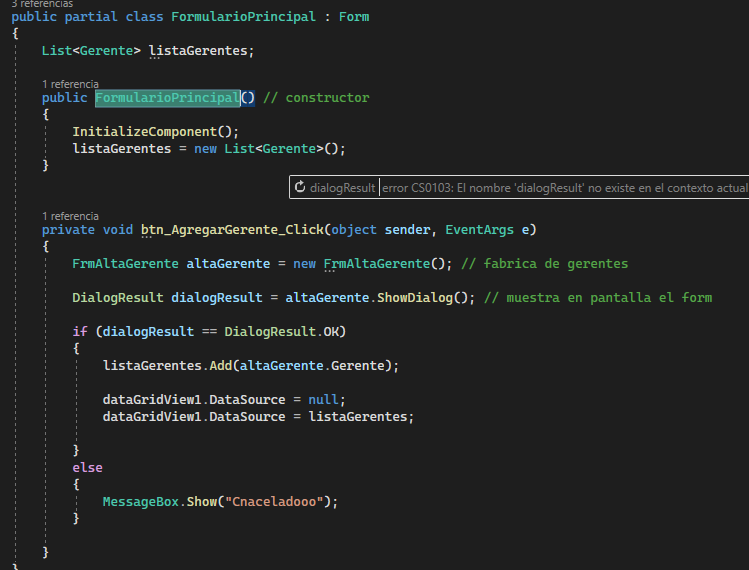
Una vez instanciado el formulario hijo, debo setear ese formulario con la propiedad MdiParent = formuPadre



## ¿Qué es un *evento* en Windows Forms? Mencione algunos eventos de los controles o de los formularios y explique cómo trabaja con ellos. ¿Cuál es el *manejador del evento*?

Un evento es una acción a la que yo podría responder o manejar (handle) mediante código. Es decir, un evento es la form que tiene una clase de notificar que algo ha ocurrido en el objeto. Los eventos pueden ser generados por un usuario al hacer click o presionando una tecla.

El manejador no es mas que un método que se ejecuta al dispararse el evento. Se crea una función dentro de la clase con dicho evento



## Explique el ciclo de vida de los formularios asociándolo a sus eventos.

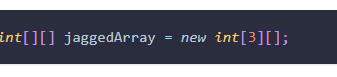
# Colleciones

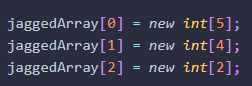
## Arrays

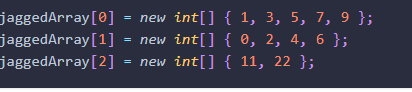
## ¿Qué es un array "jagged"? ¿En qué valor se inicializan sus elementos?

Significa “escalonada” y es una matriz cuyos elementos podrían llegar a ser de diferente tamaño. Tambien se denomina matriz de matrices.

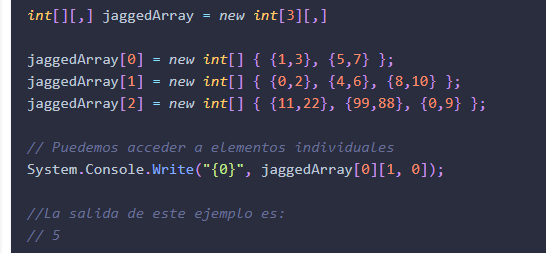
Primero se deben inicializar para saber cuantos elementos puede alojar en sus dimensiones.





En caso de querer poner datos explicitos  


En caso de querer hcer una matriz de matriz en donde la anidada tenga 2 posiciones



## ¿Los arrays son objetos?

Si, debido a ello tenemos que instanciarlos con new [tipo dato] [posiciones]

## ¿Qué significa que en C# los arrays son de "base-cero"?

Sus elementos pueden ser accedidos a través de un índice numérico que comienza con el numero 0

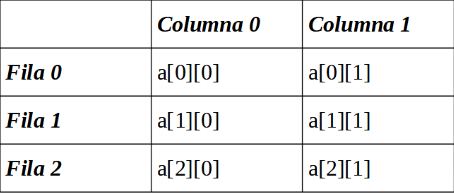
## Los arrays implementan la interfaz IEnumerable, ¿qué es lo que esto les permite hacer?

Permite recorrerlos con un foreach

## ¿Cómo se declara e instancia un array multidimensional?

Se declaran poniéndole comas (, ) por cada dimensión dentro del corchetes

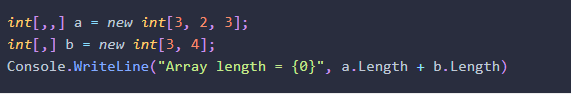




## ¿El método Array.Resize cambia el tamaño de la instancia de array proveída o genera una nueva con distinto tamaño? ¿Por qué?

Se genera una nueva instancia con los datos anteriores. Esto es asi porque un array tiene una longitud fija y para poder “agrandarlo” debo copiar los datos y crear una nueva instancia con la nueva longitud para pegar los datos de la instancia anterior.

## Considerando que la propiedad Length devuelve la cantidad TOTAL de elementos de TODAS las dimensiones del array, ¿qué valor vamos a mostrar por consola en el siguiente código?



## COLECCIONES

## ¿Cuál es la diferencia entre colecciones y matrices?

Las matrices tienen una longitud fija mientras que las colecciones su longitud es diinamica.

## ¿Cuál es la diferencia entre las colecciones genéricas y las no-genéricas?

Colecciones genéricas: Permiten declarar colecciones con un tipo determinado a elegir.

Colecciones no genéricas: Permite declarar colecciones de solo tipo Object

## ¿Cuál es la diferencia entre una cola (Queue) y una pila (Stack). Asocie con los conceptos "FIFO" y "LIFO".

La queue o cola me permite agregar elementos donde el primero en entrar será el primero en salir. FIFO first in first out

La stack o pila me permite agregar elementos donde el ultimo en entrar es el primero en salir. Last in first out.

## Describa las siguientes colecciones: List, Dictionary, SortedList, Arraylist, Hashtable. Clasifíquelas en genéricas o no-genéricas.

GENERICAS

List: Listta de elementos que se pueden acceder por un índice o mediante foreach. Tiene métodos para buscar, agregar, eliminar,etc.

Dictionary: Colección de pares de clave / valor. Se accede mediante la calve y cuenta con diferentes métodos de inserción, eliminación, etc.

SortedList: Colección de pares clave / valor que se ordenan por clave

NO GENERICAS

ArrayList: Matriz de objetos cuyo tamaño aumenenta dinámicamente según sea necesario

Hashtable: Colección de pares clave/valor donde se organiza en función del código de hash de su clave

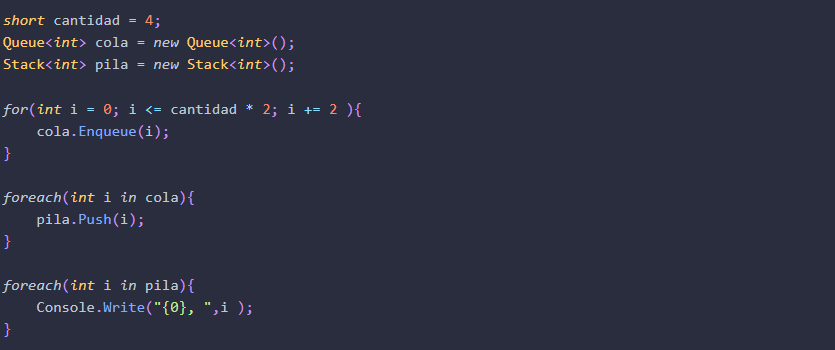
## ¿Por qué piensa que las colas y pilas no traen un método para ordenarlas? Piense en el uso de dichas colecciones.

Porque perderían la funcionalidad de pila o cola. Es decir que si la ordeno ya no seria FIFO o LIFO

## ¿Cuál es la diferencia entre colas y pilas genéricas y no-genéricas?

Las genéricas hay que definirle un tipo de daato, podría ser cualquiera mientras que las no genéricas solo acepta Object

## ¿Cual es la salida del siguiente código?



# ENCAPSULAMIENTO

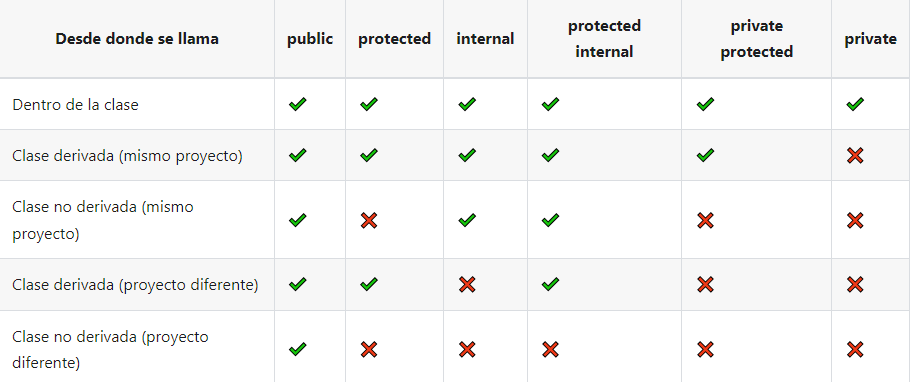
## ENCAPSULAMIENTO

## Defina encapsulamiento / encapsulación en el contexto del paradigma orientado a objetos.

Es agrupar los datos/atributos de la clase con los métodos que operan sobre estos. Es decir, poder decidir desde donde podré usar el atributo mediante un método. (accesibilidad)

## Defina cada nivel de ocultamiento / accesibilidad de la programación orientada a objetos. Incluya la función del modificador internal en C#.

* **Public:** Se puede acceder desde cualquier parte del código. Desde otra clase, la misma u otro proyecto.
* **Private:** El acceso está limitado solamente a su contenedor. Es decir que se puede acceder solamente desde su clase.
* **Protected:** Se puede acceder desde la misma clase o una clase hija (herencia).
* **Internal:** Se puede acceder dentro del proyecto actual. Miembros private se podrán acceder desde cualquier código siempre dentro del mismo proyecto.
* **Protected internal:** Se puede acceder dentro de la clase y una clase hija como no hija dentro del mismo proyecxto. Tambien se puede acceder clase hija de difertne proyecto.
* **Private protected:** Se puede acceder en la misma clase o hija dentro del mismo proyecto.



## PROPIEDADES

## ¿Qué es y para qué sirve una propiedad?

La propiedad es un miembro que proporciona una forma flexible de leer, escribir o calcular el valor de un campo. Son el análogo de los getters y setters y se utilizan para dar visibilidad a atributos privados.

## ¿Para qué sirve el descriptor de acceso get?

Analogo a getter y es un método de acceso a un atributo que podría estar privado. Devuelve un tipo de dato igual al atributo.

## ¿Para qué sirve el descriptor de acceso set? ¿Cuál es el papel de la palabra clave value?

Analogo a setter y es un meotdo de acceso de un atributo que podría estar privado. Se utiliza para asignar un valor a dicho atributo mediante el parámetro implícito llamado “value”

## ¿Cómo declaro una propiedad de sólo lectura?

Se declara mediante una propiedad de get   
[accesibilidad] [tipo de daato] [nombre de propiedad en pascal case]  
{  
 get { return nombre pripiedad }

}

## ¿Cómo declaro una propiedad de sólo escritura?

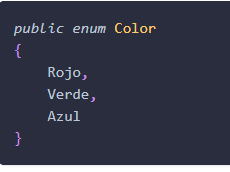
Se declara mediante una propiedad de set   
[accesibilidad] [tipo de daato] [nombre de propiedad en pascal case]  
{  
 set { nombre de propiedad = value }

}

## ENUMERADOS

## ¿Qué es un enumerado? ¿Cuál es su función?

Un enumerado es un tipo de valor de que define un conjunto de constantes con nombre



## ¿Un enumerado sólo puede estar anidado dentro de una clase?

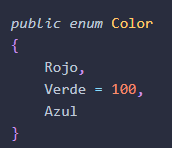
No, y se recomienda tener un archivo con solo el enumerado

## ¿Los tipos enumerados son de valor o de referencia?

Es de tipo valor

## Por defecto, ¿cuál es el primer valor numérico de un enumerado? ¿se pueden cambiar los valores por defecto?

Por defecto los valores están asociados a un tipo int de índice 0. Es decir que como el array comienza en 0. Esto podría cambiarse de forma explicita.



## Indique los valores asociados a cada constante:



Sat : 0  
sun : 1  
mon : 15  
tue: 16  
wed: 17  
thu : 2  
fri: 3

# Excepciones

## ¿Qué es una excepción?

Es la manera de controlar la gestión de errores.

## ¿Qué es el manejo de excepciones (exception handling)? (Desde lo conceptual / teórico)

Es la forma para contrtolar cuando sucede un error sin que el flujo de datos se pare. Es decir, que cuando sucede un error, el programa no se va a detener. Sino que va a ser manejado por excepciones.

## ¿Las excepciones son objetos?

Si, las excepciones son objetos.

## ¿Qué tienen en común todas las excepciones?

Todas las excepciones heredan de la clase Exception.

## ¿Qué es el Stack Trace o pila de llamadas? ¿En qué orden se lee?

Es un reporte que provee información sobre las diferentes subrutinas del programa. Se puede usaar para debugear en donde van apareciendo en un orden de PILA los diferentes erroes. Se lee desde abajo hacia arriba.

## ¿Cuál es la función de cada bloque dentro de una estructura try-catch?

Dentro del TRY va las partes de código que podrían lanzar excepciones, como podría ser una conexión a base de datos o una apertura de archivos.

Dentro del CATCH se va a hacer el tratamiento de las excepciones y finalmente el bloque FINALLY es para ejecutar un conjunto de instrucciones sea cual sea el flujo de control.

## En el caso de una misma estructura try-catch que tiene más de un tipo de bloque catch, ¿Se podría ejecutar más de un bloque catch? ¿Por qué?

Se pueden encadenar catch pero no se puede ejecutar mas de uno a la vez debido a que la excepción se devuelve , se la trata e inmediatamente se sale de los bloque try/catch o sewguiria por el finally.

## ¿En qué parte del código continúa la ejecución del programa una vez manejada una excepción?

Podria continuar por el finally o en el flujo justo después del catch manejado.

## ¿Existe una forma de capturar cualquier excepción sin importar su tipo? ¿Qué habría que considerar en ese caso si se tiene más de un tipo de bloque catch en la misma estructura?

Se puede crear una excepción genérica usando la class Exception. Para poder usarlar con varias opciones, se debería de poner en ultimo orden.

## ¿Qué sucede cuando se lanza una excepción? ¿Qué sucede si no la manejo / controlo?

Cuando se lanza una excepción esta burbujeará intentando encontrar un catch que la maneje. De no ser asi el flujo de datos se cortará y el programa finalizará de forma prematura o se cerrará.

## ¿Cómo lanzo una excepción?

Para lanzar una excepción es necesario utilizar la sentencia Throw

## Dentro de un bloque catch, ¿cuál es la diferencia entre "throw;" y "throw ex;"? (ex es un identificador para la instancia de una excepción capturada)

Lanzandola solo con Throw no tendré acceso a diferentes propiedades o métodos de la excepción.

## ¿Se lanzan en tiempo de compilación o de ejecución? ¿Por qué?

Las excepciones se lanzan en tiempo de ejecución porque es el error es producido en ese instante.

## ¿Cómo creo una excepción propia?

Se puede hacer una custom exceptioncreando una clase y que esta herede de Exception.

## ¿Qué es la propiedad InnerException? Describa a qué clase pertenece, su contenido y qué tipo de dato almacena.

Innerexception es un propiedad la excepción que es la que retorna el string que pasamos en su constructor. Se encarga de devolver información acerca de ese error.

## ¿Cómo se carga un valor en la propiedad InnerException?

Pasandola por el constructor de la excepcion

## ¿Qué sucede si no se le proporciona un valor a la propiedad InnerException? ¿Por qué?

Quedaria como null.

## De existir un bloque finally. ¿Bajo qué condiciones se ejecutará el código que contiene?

Se ejecutará en cualquier caso

## ¿Para qué se utiliza el bloque finally?

Para evitar la repetición de instrucciones. Tambien libera recursos tras el lanzamiento de una excepcion

## ¿Dónde se ubica el bloque finally dentro de una estructura de manejo de excepciones?

Se ubica debajo del catch

# Herencia

## ¿Qué es la herencia en el contexto de la programación orientada a objetos? ¿cuál es su propósito?

Es la relación entre una o mas clases en la que comparten atributos y métodos. Se utiliza para reutilizar código y mantener una semántica correcta de tipo “es un”.

## ¿Qué nombre recibe la clase que hereda y qué nombre recibe la clase que es heredada?

A la clase que hereda se la llama clase base o padre y a la heredada derivada o hija.

## Explique el principio de sustitución de Liskov.

Una clase hija podría ir en lugar de la clase padre y su funcionalidad debería de ser correcta. Podria tner diferente implementación pero ambas dos podrían funcionar de la misma forma.

## ¿Qué significa que la herencia es transitiva?

Significa que si B hereda de A y C hereda de B , C hereda de A. Es decir que si un “abuelo” va a heredar a su hijo y este a su hijo, el nieto tiene lo mismo que el abuelo.

## ¿Se heredan los constructores?

No, no se heredan.

## ¿Se heredan los miembros private de la clase base?

Se heredan pero no se pueden utilizar por ser privados.

## ¿Qué es herencia múltiple? ¿es posible en C#? ¿en qué se diferencia de la herencia simple?

Herencia multiple se le llama cuando una clase puede heredar de dos padres. No es posible en C#.

## ¿Una clase pública puede heredar de una clase privada?

No, la accesibilidad de la hija no puede ser mayor que la de su padre.

## ¿Qué es una clase sellada (sealed)?

Significa que esa clase no hereda mas. Se termina la gestión de clases hijas (vasectomia)

## ¿Una clase sellada puede heredar de otras clases (ser clase derivada)?

Si puede heredar de otras clases pero esta no puede dar herencia.

## ¿Cómo actúa el modificador “protected” en los miembros de la clase base para una clase derivada y cómo para una clase no-derivada? Relacionar la respuesta con los modificadores “public” y “private”.

Para una clase padre el accesor protected me va a dejar acceder a esos campos desde su clase hija mientras que para una no derivada no me será posible.

## ¿Qué pasa si la clase derivada no hace una llamada explícita a un constructor de la clase base? En esta situación, ¿qué pasa si la clase base declaró explícitamente un constructor con parámetros de entrada?

## La clase Alumno hereda de Persona. ¿Una instancia de Alumno es también de tipo Persona? Justifique.

Si porque primero es tipo Persona por la herencia y luego es un alumno. Generalmente para que esto sea semánticamente correcto se debe hacer la pregunta “es un “.

## La clase Alumno hereda de Persona. ¿Se puede hacer Persona persona = new Alumno()? ¿por qué?

Si se puede porque un Alumno es primero una Persona.

## La clase Alumno hereda de Persona. ¿Se puede hacer Alumno alumno = new Persona()? ¿por qué?

No, no se puede porque un Alumno es una persona pero una Persona no necesariamente es un ALumno. La semántica falla y el orden de herencia va del mas genérico al mas especifico.