**[Módulo - 18] Indexadores:**

**(101) -** ¿Qué significa **indexar**?

Asignar un valor a algo para formar su indice

**(102) -** ¿Qué permite un **indexador** (función)?

Los indexadores permiten a la instancia de una clase ser indexada tal cómo un array.

**(103) -** ¿Cuál es la diferencia a la hora de declarar un indexador y una propiedad?

La declaración de un indexador luce cómo una propiedad, sólo que:

•Este recibe parámetros.

•La palabra clave this se utiliza para su definición

**(104) -** ¿Los indexadores solo se pueden indexar por valores numéricos?

No

**(105) -** ¿Un indexador puede recibir más de un parámetro (ser multidimensionales)?

Si

**(106) -** ¿Los indexadores pueden ser sobrecargados?

Si, para ordenar de diferentes manera , cambiando roden tipo igual q los metoods

**(107) -** ¿Cuál es el papel de la palabra clave “this” en un indexador?

El this es usado para definir el indexador.Hace referencia a la misma instancia de la clase

**(108) -** ¿Cuál es el papel de la palabra clave “value” en un indexador?

El “Value” es usado para definir el valor que se le esta asignando desde el Set

**(109) -** ¿Se pueden declarar indexadores estáticos? ¿Por qué?

No , porque tenes que usar el this.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:::::\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**[Módulo - 19] Encapsulamiento:**

**(110) -** Defina **encapsulamiento / encapsulación** en el contexto del paradigma orientado a objetos.

Es el acto de encerrar uno o más ítems en un contenedor.

En Programación Orientada a Objetos significa agrupar los datos del objeto junto con los métodos que operan sobre esos datos.

**(111) -** Defina cada **nivel de ocultamiento / accesibilidad** de la programación orientada a objetos. Incluya la función del modificador “internal” en C#.

Público: todos pueden acceder a los datos o métodos de una clase que se definen con este nivel, este es el nivel más bajo, esto es lo que tu quieres que la parte externa vea.

•Protegido: podemos decir que estas no son de acceso público, solamente son accesibles dentro de su clase y por subclases.

•Privado: en este nivel se puede declarar miembros accesibles sólo para la propia clase.

•internal es para el alcance del ensamblado (es decir, solo accesible desde el código en el mismo .exe o .dll). Los miembros internos solo son accesibles dentro del ensamblado por herencia (su tipo derivado) o por instancia de clase.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:::::\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**[Módulo - 20] Herencia:**

**(112) -** ¿Qué es la **herencia** en el contexto de la programación orientada a objetos? ¿Cuál es su propósito?

Es una relación entre clases en la cual una clase comparte la estructura y comportamiento definido en otra clase.

•Cada clase que hereda de otra posee:

•Los atributos de la clase base además de los propios.

•Soporta todos o algunos de los métodos de la clase base.

El propósito principal de la herencia es el de organizar mejor las clases que componen una determinada realidad, y poder agruparlas en función de atributos y comportamientos comunes. A la vez que cada una se especializa según sus particularidades.

•La herencia permite crear nuevas clases a partir de otras ya existentes (en lugar de crearlas partiendo de cero).

**(113) -** ¿Qué nombre recibe la clase que hereda y qué nombre recibe la clase que es heredada?

•La clase en la que está basada la nueva clase se la conoce como clase base o padre, mientras que la clase hija se conoce como clase derivada.

**(114) -** ¿Qué significa que la herencia es **transitiva**?

La herencia es transitiva: Si C hereda de B, y B hereda de A, entonces C también hereda de A.

**(115) -** ¿Se heredan los constructores?

Se hereda TODO menos los constructores y finalizadores. Si queres usar el constructor de la clase padre usas Base:

**(116) -** ¿Se heredan los miembros private de la clase base?

Los miembros private no son visibles en las clases derivadas (PERO SÍ SE HEREDAN).

**(117) -** ¿Qué es **herencia múltiple**? ¿Es posible en C#? ¿En qué se diferencia de la **herencia simple**?

Herencia Simple:

•Una clase derivada puede heredar sólo de una clase base (los lenguajes .NET soportan este tipo de herencia)

•Herencia Múltiple:

•Una clase derivada puede heredar de una o más clases base (En C# no se puede).

**(118) -** ¿Una clase pública puede heredar de una clase privada?

Una clase derivada no puede ser más accesible que su clase base. Por ejemplo, no es posible derivar una clase pública de una clase privada.

**(119) -** ¿Qué es una **clase sellada (sealed)**?

Es una clase que no puede ser heredada

**(120) -** ¿Una clase sellada puede heredar de otras clases? (Ser clase derivada)

Si

**(121) -** ¿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”.

El significado del modificador de acceso protected depende de la relación entre la clase que tiene el modificador y la clase que intenta acceder a los miembros que usan el modificador.

•Para una clase derivada, la palabra reservada protected es equivalente a la palabra public.

•Entre dos clases que no tengan una relación base-derivada, por el contrario, los miembros protegidos de una clase se comportan como miembros privados para la otra clase.

**(122) -** ¿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?

Si la clase derivada no hace una llamada explícita a un constructor de la clase base, el compilador de C# usará implícitamente un constructor de la forma :base(). , PERO:

El compilador generará un mensaje de error si el constructor indicado no coincide con ningún constructor de la clase base.