

Cuestionario Teórico

Recomendación sobre cómo usar el cuestionario:

- ✓ Repase la clase en cuestión.
- ✓ Realice la ejercitación práctica.
- ✓ Estudie como si fuera a dar un parcial sobre ese tema específico.
- ✓ Responda el cuestionario tratando de no ayudarse con ninguna otra fuente que sus propios conocimientos.
- ✓ Si lo intentó y aun así no puede responder la pregunta, puede ayudarse del material teórico o investigar.
- ✓ Verifique sus respuestas contrastando con los apuntes y el material teórico.
- ✓ Refuerce sus conocimientos y respuestas con la puesta en común en clase, de ser necesario consulte sus dudas con el profesor.
- ✓ Repase lo aprendido antes del parcial.

Tabla de contenido

Módulo 24 – Excepciones	2
Módulo 25 – Pruebas de calidad	2
Módulo 26 – Tipos genéricos	2
Módulo 27 – Interfaces	3
Módulo 28 – Archivos y serialización	4
Módulo 29 – Bases de datos.....	4
Módulo 30 – Delegados	5
Módulo 31 – Hilos	5
Módulo 32 – Eventos.....	6
Módulo 33 – Métodos de extensión.....	6

Módulo 24 – Excepciones

- (001) ¿Qué es una excepción? ¿Qué es el manejo de excepciones (exception handling)?
- (002) ¿Qué son las excepciones en C#? ¿Qué tienen en común todas las excepciones?
- (003) ¿Qué es el Stack Trace o pila de llamadas? ¿En qué orden se lee?
- (004) ¿Cómo capturo una excepción? ¿Cuál es la función de cada bloque?
- (005) En el caso de una estructura try-catch se tiene más de un tipo de bloque catch, ¿se podría ejecutar más de un bloque catch que forme parte de la misma estructura?
- (006) ¿En qué parte del código continúa la ejecución del programa una vez manejada una excepción?
- (007) ¿Existe una forma de capturar cualquier excepción sin importar su tipo? ¿Qué habría que considerar si se tiene más de un tipo de bloque catch?
- (008) ¿Qué sucede cuando se lanza una excepción? ¿Qué sucede si no la manejo/controlo?
- (009) ¿Cómo lanzo una excepción?
- (010) Dentro de un bloque catch, ¿cuál es la diferencia entre “throw;” y “throw ex;” (ex es un identificador para una excepción capturada)?
- (011) ¿Se lanzan en tiempo de compilación o de ejecución? ¿Por qué?
- (012) ¿Cómo creo una excepción propia?
- (013) ¿Qué es la propiedad InnerException? Describa a qué clase pertenece, su contenido y cómo se carga. ¿Qué sucede si no se le proporciona un valor?
- (014) Bloque finally: ¿En qué condiciones se ejecutará el código que contiene? ¿Cómo se ubica dentro de una estructura de manejo de excepciones? ¿Para qué es útil?

Módulo 25 – Pruebas de calidad

- (015) Indique el orden y describa brevemente qué sucede en cada etapa del ciclo de vida general de los sistemas. (mantenimiento – diseño – desarrollo/construcción – implementación – análisis – pruebas)
- (016) ¿Qué es una prueba unitaria?
- (017) ¿Qué es una prueba integral?
- (018) ¿Qué es una prueba funcional?
- (019) ¿Qué es el patrón AAA? Describa cada una de sus etapas.
- (020) ¿Qué es la clase Assert? ¿Para qué sirve?

Módulo 26 – Tipos genéricos

- (021) ¿Qué es una clase genérica? ¿Qué permite?
- (022) ¿Se puede tener más de un parámetro genérico en una clase o un método?
- (023) ¿Qué es una restricción o constraint?

(024) ¿Qué sucede si no hay restricciones?

(025) ¿Qué sucede si intentamos instanciar una clase genérica pasando como argumento un tipo que no cumple con las restricciones?

(026) ¿Puedo declarar métodos genéricos en clases no-genéricas?

(027) ¿Sólo se puede aplicar una sola restricción por parámetro?

(028) ¿Qué nombre pueden tener los comodines o parámetros genéricos?

(029) Complete la siguiente tabla:

Restricción	Descripción
where T : struct	
where T : class	
where T : notnull	
where T : new()	
where T : <clase base >	
where T : <interface >	
where T : U	

Módulo 27 – Interfaces

(030) ¿Qué es una interfaz?

(031) ¿Qué puede especificar una interfaz (atributos, métodos, propiedades, etc)?

(032) ¿Qué nivel de visibilidad/acceso pueden tener los miembros especificados?

(033) ¿Las clases sólo pueden implementar una interfaz?

(034) ¿Se puede elegir cuáles de las operaciones definidas en la interfaz implementar en una clase?

(035) ¿Una clase puede heredar de otra y al mismo tiempo implementar una o más interfaces? Describa la sintaxis para lograr esto.

(036) Si implementé una interfaz en la clase base, ¿debo implementarla también en las derivadas?

(037) ¿Qué significa implementar una interfaz de forma explícita? ¿Qué utilidad tiene? ¿Qué consecuencias o efectos tiene?


- (038) ¿Una interfaz puede implementar otra interfaz? ¿Qué sucede con las operaciones especificadas (se suman o se anulan)?
- (039) ¿Se pueden tener interfaces genéricas? ¿Se puede restringir sus parámetros de tipo? ¿Puede implementarlas cualquier clase o sólo clases que también sean genéricas?
- (040) ¿En qué se diferencian una interfaz y una clase abstracta?
- (041) Si un método tiene un parámetro del tipo de una interfaz:
- a) ¿Qué argumentos podré pasarle?
 - b) Si no casteo el argumento a otro tipo, ¿podré acceder a todos los métodos y propiedades del objeto?
 - c) Considerando que una interfaz no tiene implementación, ¿si llamo a uno de sus métodos qué implementación se ejecutará? Asocie con polimorfismo.

Módulo 28 – Archivos y serialización

- (042) ¿Qué es serializar?
- (043) ¿Para qué sirve serializar?
- (044) ¿En qué formatos se puede serializar (vistos en clase)? Indique qué miembros de la clase se incluyen en cada formato.
- (045) Indique qué características debe tener la clase para ser serializable en cada formato.

Módulo 29 – Bases de datos

- (046) ¿Qué es una base de datos?
- (047) ¿Qué es una tabla?
- (048) ¿Qué es una primary key o clave primaria?
- (049) ¿Para qué sirve la clase SqlConnection?
- (050) ¿Para qué sirve la clase SqlCommand?
- (051) ¿Qué es una “connection string” o cadena de conexión? ¿Qué clase de ADO.NET la utiliza?
- (052) ¿Qué es la “Inyección SQL”? ¿Cómo se puede evitar con ADO.NET (tecnología utilizada en clase para conectar a una base de datos)?
- (053) Explique las 2 formas vistas en clase de liberar los recursos de conexión incluso cuando ocurra una excepción. Escriba un ejemplo de cada una.
- (054) En base a la siguiente tabla:

	Column Name	Data Type	Allow Nulls
	id	int	<input type="checkbox"/>
	nombre_provincia	nvarchar(50)	<input type="checkbox"/>
	cantidad_habitantes	int	<input checked="" type="checkbox"/>

- a) Escriba las sentencias SQL para insertar las siguientes provincias teniendo en cuenta que “id” es clave primaria y NO es autoincremental:
- ✓ { ‘Formosa’, 573.823 }
 - ✓ { ‘Neuquen’ } (Sin cantidad de habitantes)
 - ✓ { ‘Entre Ríos’, 1.308.000 }
 - ✓ { ‘Tierra del Fuego’, 152.317 }
- b) Teniendo en cuenta las inserciones del punto anterior, modifique el registro de la Provincia ‘Neuquén’ con una nueva cantidad de habitantes de 619.745.
- c) Escriba la sentencia para consultar a todas las provincias con todas sus columnas.
- d) Escriba la sentencia para consultar sólo el nombre de todas las provincias.
- e) Escriba la sentencia para consultar todos los datos de todas las provincias que tengan más de 600mil habitantes. Dibuje una tabla con el resultado de la consulta (teniendo en cuenta que se ejecutaron las sentencias de los puntos anteriores).
- f) Escriba la sentencia para eliminar el registro con id igual a 3.
- g) Escriba la sentencia para eliminar todos los registros que no tengan datos en la columna “cantidad_habitantes”.
- h) ¿Puedo insertar un registro que no contenga datos para la columna “nombre_provincia”? ¿Por qué?

Módulo 30 – Delegados

(055) ¿Qué es un delegado? ¿Para qué sirve?

(056) ¿Qué papel juega el tipo de retorno y los parámetros en la declaración del delegado?

(057) ¿En qué se diferencian los delegados de los punteros a función vistos en el lenguaje C?

Módulo 31 – Hilos

(058) ¿Qué es un hilo o subproceso?

(059) ¿Qué es un proceso?

(060) ¿Cuáles son los posibles parámetros de entrada para el constructor de la clase Thread? ¿Qué son? ¿Para qué sirven? ¿En qué se diferencian?

(061) ¿Cuándo termina la vida de un hilo? Explique las distintas posibilidades.

(062) Explique el siguiente código línea a línea e indique para qué se utiliza:

```
private void WriteTextSafe(string text)
{
    if (textBox1.InvokeRequired)
    {
        var d = new SafeCallDelegate(WriteTextSafe);
        textBox1.Invoke(d, new object[] { text });
    }
    else
    {
        textBox1.Text = text;
    }
}
```

Módulo 32 – Eventos

- (063) ¿Qué es un evento?
- (064) ¿Qué es un manejador de un evento (event handler)?
- (065) ¿Por qué los eventos son de un tipo delegado? ¿Cómo impacta esto a los posibles manejadores de ese evento?
- (066) ¿Qué sintaxis se utiliza para subscribirse a un evento? ¿Y para desuscribirse?
- (067) ¿Un mismo manejador puede estar subscripto a distintos eventos simultáneamente?
- (068) ¿Un mismo evento puede tener más de un manejador distinto?
- (069) ¿Qué papel juega la clase emisora y cuál la clase receptora? Asocie con los conceptos de declarar el evento, lanzar el evento y subscribirse al evento.
- (070) ¿Puedo declarar eventos estáticos?
- (071) ¿Puedo definir un evento en una interfaz?
- (072) Si tengo un evento de instancia y 5 instancias de esa clase. ¿Cuántas veces voy a tener que asociar el evento al manejador en la clase receptora?

Módulo 33 – Métodos de extensión

- (073) ¿Qué es un método de extensión? ¿Para qué sirve?
- (074) ¿Qué características debe tener la clase que lo contiene?
- (075) ¿Qué características debe tener el método para ser considerado un método de extensión?
- (076) ¿Cómo se usa un método de extensión? Indique diferencias entre su declaración y su uso.
- (077) ¿Puedo usar un método de extensión sin referenciar previamente al namespace donde está declarado (con un “using”, por ejemplo)?