PARCIAL 2 DE PROGRAMACIÓN 2

- Indique cuáles de las siguientes afirmaciones sobre interfaces son correctas:
  - **La clase que las implemente deberá darle una implementación a cada una de las operaciones especificadas por la interfaz.**
  - Los métodos se definirán con la palabra reservada abstract ya que no contienen código.
  - Al realizar la herencia de una clase abstracta que implementa una interfaz, podré implementar los métodos directamente en la clase que la hereda, sin deber implementarlos en la clase base.
  - **Todos los elementos que las componen son públicos.**
  - Contienen métodos, atributos y propiedades.
  - Ninguna respuesta.
  - **Son un contrato que establece una clase en el cual la misma asegura que implementará un conjunto de operaciones.**

- Al ejecutar un thread:
  - **Cuando finalice liberará el bloque de memoria que ocupaba.**
  - Proporcionaremos un medio apropiado para que los objetos puedan señalizar cambios de estado que pueden resultar útiles para los usuarios/clientes de ese objeto.
  - Finalizará si se culmina la ejecución del hilo principal.
  - Podremos ejecutar el método Abort más de una vez para el mismo método sin errores.
  - Ninguna respuesta.

- Teniendo en cuenta la siguiente línea de código, indicar las afirmaciones correctas:

```
System.IO.StreamWriter file = new System.IO.StreamWriter(archivo, true);
```

  - **Si declaro e instancio el objeto StreamWriter en una sentencia "using", no deberé preocuparme por cerrar el vínculo con el archivo.**
  - **Al finalizar, deberé cerrar el vínculo con el archivo para que no quede tomado.**
  - Si el archivo existe, lo sobrescribirá.
  - **Si el archivo existe, agregará información al mismo.**
  - Ninguna respuesta.
  - Me servirá para serializar en formato binario.

- Indique las afirmaciones correctas sobre tipos genéricos:
  - **Se pueden utilizar en clases, métodos, atributos y propiedades.**
  - Ninguna respuesta.
  - Sólo se puede tener un tipo genérico por clase.
  - Los métodos pueden recibir tipos genéricos, pero no retornarlos.
  - Las interfaces no pueden especificar tipos genéricos.
  - **Al generar la clase, reemplazaré por un comodín lo que podría ser un tipo de dato específico.**

- Al generar un método de extensión:
  - **Agrego funcionalidades a una clase preexistente.**
  - Anulo los métodos de la clase extendida.
  - Creo una nueva clase que hereda de una clase preexistente.
  - Hago referencia a la clase extendida mediante la palabra reservada base.
  - Ninguna respuesta.

- En el contexto de una competencia de programación sin IDEs, Victoria encontró 3 errores en el código que el resto de su equipo pasó por alto. Amelia, una de sus compañeras, señaló que cometieron una mala práctica. ¿Podrás encontrar los errores y corregirlos como hicieron Victoria y Amelia?

```
interface IExponer<S>
{
    S Datos { get; }
}

class ClaseBase<V>
{
    public V entero;
    public ClaseBase(long i)
    {
        this.entero = i;
    }
}

class ClaseEjecutor<int> : ClaseBase<T>, IExponer<T>
{
    public ClaseEjecutor(T e)
        : base(e)
    {
    }

    public T Datos
    {
        get
        {
            return base.entero;
        }
    }
}

static void Main(string[] args)
{
    ClaseEjecutor<string> ejecutor = new ClaseEjecutor("Hola");
    Console.WriteLine(ejecutor.Datos);
}
```

❖ Errores encontrados por Victoria:

- El constructor de ClaseBase debería recibir el tipo genérico V en vez de long.
- En la declaración de la clase genérica ClaseEjecutor se debería reemplazar <int> por <T>
- Cuando se instancia ClaseEjecutor en el método Main, se debe agregar <string> seguido del identificador de la clase.
  Quedaría:
  ClaseEjecutor<string> ejecutor = new ClaseEjecutor<string>("Hola");

❖ Errores encontrados por Amelia:

- Es una buena práctica que los atributos no tengan visibilidad pública. Por lo tanto, el atributo "entero" de ClaseBase debería ser private.

- ¿Cuáles de las siguientes estructuras son válidas?
  LOS COMENTARIOS TAMBIÉN DEBEN SER CORRECTOS

```
// OPCION A
try { ... }

// OPCION B
try { /* aquí se lanza la excepción */ }
catch (Exception e) { /* aquí se controla la excepción */ }

// OPCION C
try { /* aquí se controla la excepción */ }
catch (Exception e) { /* aquí se lanza la excepción */ }

// OPCION D
try { ... }
catch (Exception e) { ... }
finally { /* se ejecutará si no se lanzó ninguna excepción  */ }

// OPCION E
try { ... }
finally { ... }

// OPCION F
try { ... }
catch (Exception e) { ... }
finally { /* se ejecutará siempre  */ }
```

  - Opción A
  - **Opción B**
  - Opción C
  - Opción D
  - **Opción E**
  - **Opción F**
  - Ninguna respuesta.

- Indique cuáles de las siguientes afirmaciones sobre serialización son correctas:
  - Ninguna respuesta.
  - Podemos serializar en archivos de texto plano.
  - Sólo se pueden serializar en formato binario atributos y propiedades públicas.
  - **Los objetos que se deseen serializar en formato XML deben tener un constructor público sin parámetros.**
  - Los objetos que se deseen serializar en formato binario no necesitan ningún agregado o característica en particular.

- Al ejecutar las siguientes líneas de código dentro de una clase del tipo Form, sucederá que:

MiMetodo NO contiene un bucle infinito.

```
Thread t = new Thread(MiMetodo);
t.Start();
```

  - **Se lanzará un nuevo hilo, totalmente independiente del actual.**
  - Ninguna respuesta.
  - **Se podrá cancelar el hilo a través del método Abort.**
  - Se ejecutará el método MiMetodo hasta que se cancele el hilo por código.
  - **Si se quiere cancelar un thread inactivo, lanzará una excepción.**
  - **La propiedad IsAlive me retornará el estado de ejecución del hilo.**

- Cuando realice un test unitario:
  Utilizando Visual Studio Unit Testing Framework (MSTest).
  - Nuestra clase test heredará de UnitTest.
  - **Podré verificar cómo se comporta una porción del código interpretando valores de variables, excepciones lanzadas, etc.**
  - **Cada método llevará la etiqueta [TestMethod].**
  - Ninguna respuesta.
  - **A través de métodos estáticos de la clase Assert podré verificar el resultado de cada test.**

- Indique cuáles de las siguientes afirmaciones sobre eventos son correctas:
  - **Para asociar un manejador a un evento se usa el operador +=.**
  - **Su definición depende de un delegado.**
  - Es un tipo que representa referencias a métodos con una lista de parámetros determinada y un tipo de valor devuelto.
  - **Los eventos permiten a una clase u objeto enviar notificaciones a otras clases u objetos cuando sucede algo en particular.**
  - Ninguna respuesta.
  - Sólo puede haber un manejador o subscriptor por cada evento.
  - Cada manejador o subscriptor puede estar asociado a un sólo evento.

- Suponiendo que el siguiente código se encuentra dentro de un proyecto de Consola, teniendo las referencias a las librerías necesarias implementadas correctamente, contestar que hará el siguiente código según las opciones dadas:

```
class Prueba
{
    public Prueba(string mensaje)
    {
        Thread thread = new Thread(new ParameterizedThreadStart(Imprime));

        thread.Start(mensaje);
    }

    void Imprime(object o)
    {
        Console.WriteLine((string)o);
    }
}
```

  - **Imprime en un hilo secundario un mensaje por consola.**
  - Ninguna respuesta.
  - Error en tiempo de diseño.
  - Error en tiempo de ejecución.
  - Imprime en el hilo principal un mensaje por consola.

- Si tengo el siguiente código. ¿Qué afirmaciones PODRÍAN ser correctas?

```
public class MiClase<T, S> : A<T>, B<S>, C<T, S>
    where T : MiTipo {}
```

  - **A es una clase, B y C interfaces.**
  - **T deberá ser de tipo "MiTipo" o de algún tipo que herede o implemente este tipo de dato.**
  - T y S deberán ser de tipo "MiTipo".
  - **A, B y C son interfaces.**
  - Ninguna respuesta.
  - A y B son interfaces, C es una clase.
  - A y B son clases.

- Indique cuáles de las siguientes afirmaciones sobre hilos son correctas:
  - **Si deseáramos detener un hilo en ejecución, utilizaríamos el método Abort.**
  - **Mediante estos, una tarea que puede ser ejecutada al mismo tiempo que otra tarea.**
  - Ninguna respuesta.
  - **En el momento en el que todos los hilos de ejecución finalizan, el proceso no existe más y los recursos son liberados.**
  - **Si quisiéramos lanzar un hilo parametrizado, deberíamos utilizar el delegado ParameterizedThreadStart.**

- Cuando accedemos a una base de datos utilizando la API ADO.NET:
  - **Se pueden ejecutar sentencias al menos de dos formas: ExecuteNonQuery y ExecuteReader.**
  - Ninguna respuesta.
  - **SqlConnection administrará la conexión con un servidor.**
  - **El objeto SqlCommand se conectará a la base de datos por medio de otro objeto.**
  - **Podremos utilizar el mismo SqlCommand para ejecutar consultas en distintos servidores, sólo cambiando la conexión.**

- La serialización binaria me servirá para:
  - Ninguna respuesta.
  - Serializar clases.
  - Generar archivos sólo con la extensión .bin
  - **Serializar objetos.**
  - **Guardar en formato de bytes distintos tipos de datos.**

- Indique cuáles de las siguientes afirmaciones sobre métodos de extensión son correctas:
  - Se utilizarán mediante el identificador de la clase extendida.
  - **Permiten extender clases selladas.**
  - Ninguna respuesta.
  - **Su utilización será mediante una instancia de la clase extendida.**
  - **Permiten adicionar métodos a tipos existentes sin crear un nuevo tipo derivado, recompilar o modificar de otra manera el tipo original.**
  - **El método y la clase donde es declarado deberán ser estáticos.**

- En el siguiente código hay 2 errores, encontrarlos y corregirlos.

```
interface IPrueba<Tu>
{
    private void MetodoInterfaz();
    void MetodoLanzaException(Tu aux);
}
public class MiClase : IPrueba<string>
{
    public void MetodoInterfaz()
    {
        string rta = this.MetodoLanzaException("Error!");
    }

    public void MetodoLanzaException(string aux)
    {
        throw new Exception(aux);
    }
}
```

  - No se pueden especificar métodos privados en una interfaz.

  - MetodoLanzaExcepcion no retorna un string ni nada, no se puede asignar a una variable.

- ¿Qué son los test unitarios?
  - **Son pruebas para cada función no trivial o método en el módulo, de forma que cada caso sea independiente del resto.**
  - Son una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software.
  - Son aquellos que prueban que todos los elementos unitarios que componen el software, funcionan juntos correctamente probándolos en grupo.
  - Ninguna respuesta.

- ¿Cómo hago una consulta en ANSI SQL a una tabla provincia filtrando por id igual a 2?
  - SELECT ALL FROM provincia WHERE id = 2;
  - Provincia.getAll(2);
  - GET ALL FROM provincia PROVINCIA WHERE id = 2;
  - **SELECT * FROM provincia WHERE id = 2;**
  - Ninguna respuesta.
  - SELECT 2 FROM provincia;
  - SELECT * provincia WHERE id = 2;