

Programación II - REPASO Segundo Parcial

Puntos totales 10/10 ?

Este formulario es sólo un modelo de parcial a fin de repasar algunos de los contenidos.

Recibirás una copia de tus respuestas en tu dirección de correo electrónico.

Correo *

roabigail.diaz@gmail.com

PRIMERA PARTE

10 de 10 puntos

* Seleccione la opción correcta

* Cada pregunta tiene una sola respuesta correcta

✓ Excepciones: *

1/1

- El bloque finally sólo se ejecuta cuando se produce una excepción.
- El bloque catch sólo se ejecuta si se produjo una excepción dentro del bloque try ✓
- El bloque catch se ejecuta si se produjo una excepción sin importar si sucede dentro o fuera del bloque try

Comentarios

El bloque try contiene una expresión que puede generar la excepción. En caso de producirse la excepción, el runtime detiene la ejecución normal y empieza a buscar un bloque catch que pueda capturar la excepción pendiente (basándose en su tipo). Las instrucciones contenidas en el bloque finally siempre se ejecutarán sea cual sea el flujo de control.

✓ Según el patrón 'AAA' para la construcción de Unit Test. La sección 'Arrange': *

1/1

- Realiza la llamada al método a probar con los parámetros preparados para tal fin.
- Comprueba que el método ejecutado se comporta tal y como teníamos previsto que lo hiciera.
- Inicializa los objetos y establece los valores de los datos que vamos a utilizar en el Test. ✓

Comentarios

El patrón AAA sugiere dividir una prueba unitaria (un método de pruebas) en tres secciones.

Arrange: Inicializa los objetos y establece los valores de los datos que vamos a utilizar en el Test.
Act: Realiza la llamada al método a probar con los parámetros preparados para tal fin.
Assert: Comprueba que el método ejecutado se comporta tal y como teníamos previsto que lo hiciera.

✓ Si una clase implementa dos interfaces que coinciden en la firma de un método pero la implementación es diferente para cada interfaz: *

1/1

- Se implementan las dos interfaces de forma implícita
- Se implementa al menos una de las interfaces de forma explícita ✓
- Una clase no puede implementar más de una interfaz
- Una clase no puede implementar interfaces que tengan métodos con la misma firma

Comentarios

Si una clase implementa dos interfaces que contienen un miembro con la misma firma para llamar a una implementación diferente según la interfaz que esté en uso, puede implementar un miembro de la interfaz explícitamente. Una implementación de interfaz explícita es un miembro de clase que solo se llama a través de la interfaz especificada. Por ejemplo:

```
public class SampleClass : IControl, ISurface
{
    void IControl.Paint()
    {
        // Código para dibujar en la superficie
    }
}
```

```

        System.Console.WriteLine("IControl.Paint");
    }
    void ISurface.Paint()
    {
        System.Console.WriteLine("ISurface.Paint");
    }
}

```

La implementación explícita también se utiliza para resolver casos en los que dos interfaces declaran miembros diferentes con el mismo nombre, como una propiedad y un método. Para implementar ambas interfaces, una clase tiene que usar una implementación explícita para la propiedad P, el método P o ambos para evitar un error del compilador.

Por ejemplo:

```

interface ILeft
{
    int P { get; }
}
interface IRight
{
    int P();
}

class Middle : ILeft, IRight
{
    public int P() { return 0; }
    int ILeft.P { get { return 0; } }
}

```

- ✓ Cual de las siguientes declaraciones restringe el parámetro T a que sea 1/1 de tipo referencia? *

- class Mock<T> where T: class ✓
- class Mock<T> where T: object
- class Mock<T> where T = new()
- class Mock<T> where T: new()

- ✓ Marque la correcta sobre serialización en XML: * 1/1

- La clase a serializar debe tener un constructor sin parámetros ✓
- Se pueden serializar atributos públicos y privados.
- Se pueden serializar atributos y propiedades públicas y privadas.
- Todas son correctas

Comentarios

La serialización es el proceso de convertir las propiedades públicas y los campos de un objeto a un formato, en este caso XML, para su almacenamiento o transporte. La deserialización vuelve a crear el objeto en su estado original a partir de la salida XML. Una clase debe tener un constructor por defecto para que XmlSerializer pueda serializarla.

- ✓ La Primary Key / Clave Primaria: * 1/1

- Es auto-incremental por defecto
- Puede ser de cualquier tipo que admita el motor de base de datos. ✓
- Puede ser null.
- Sólo puede ser de tipo int.

Comentarios

Primary Key es un campo o conjunto de campos que identifica a cada fila de una tabla. Puede ser de cualquier tipo de dato. No todas las primary key son de tipo Identity, es decir son auto-incrementales.

- ✓ Eventos: * 1/1

- Los eventos permiten que una clase u objeto notifique a otras clases u objetos cuando ocurre algo de interés
- Un evento puede tener varios suscriptores/manejadores. Un suscriptor/manejador puede manejar varios eventos de varios emisores.
- Los eventos que no tienen suscriptores/manejadores nunca se generan.
- Todas son correctas. ✓

Comentarios

Un evento es el modo que tiene una clase dada de proporcionar notificaciones a sus usuarios cuando ocurre algo en particular dentro del objeto. Cada evento tiene un emisor que produce el evento y un receptor que lo captura. Para asociar un evento a un manejador de eventos en tiempo de ejecución, hay que 'agregarlo' al evento del emisor usando `+=`. Un Evento puede tener múltiples manejadores y viceversa. Los eventos son un tipo especial de delegado que solo se puede invocar desde la clase que los declaró.

- ✓ ¿Cuál es la razón de utilizar consultas parametrizadas cuando trabajamos 1/1 con las clases SqlConnection y SqlCommand? *

- Evitar la inyección de sql ✓
 Generar código más limpio y ordenado
 Las consultas parametrizadas son la única forma que tenemos de realizar consultas SQL
 Todas son correctas

Comentarios

Una de las razones más importantes de usar las consultas parametrizadas es evitar la inyección de SQL. Una inyección SQL es una vulnerabilidad de seguridad que permite a un atacante interferir con las consultas que una aplicación realiza a una base de datos. Generalmente le permite al atacante acceder a datos que normalmente no debería ser capaz de acceder. Estos podrían ser información sobre usuarios o sobre la aplicación en sí. Las consultas parametrizadas nos permiten resolver esto ya que en vez de concatenar los valores que ingresa el usuario directamente en la consulta los vamos a reemplazar en tiempo de ejecución removiendo la posibilidad de código SQL que podría ser inseguro.

- ✓ ¿Cuál es la función de los delegados? *

1/1

- Nos permiten encapsular las referencias a métodos de una instancia dentro de un objeto de tipo delegado
 Nos permite encadenar varios métodos manejadores disparando un solo evento
 Nos permiten definir la firma que todos nuestros métodos manejadores van a tener dándonos una seguridad de tipos
 Todas son correctas ✓

Comentarios

Los delegados en .NET son los intermediarios entre nuestros eventos y manejadores, nos permiten almacenar y encapsular todas las referencias a los métodos manejadores de nuestros eventos. Además de que al tener seguridad de tipos nos aseguramos que todos los métodos manejadores asociados tengan la misma firma y como podemos asociar más de un método manejador a un evento estos mismos pueden ser encadenados cuando ocurre la invocación de dicho evento.

- ✓ Marque la respuesta correcta *

1/1

```
1 referencia
public class Pista
{
    List<Thread> hilos;
    bool sigueLaCarrera;

    0 referencias
    public Pista()
    {
        this.hilos = new List<Thread>();
        this.sigueLaCarrera = false;
    }

    0 referencias
    public void Correr(Deportista deportista)
    {
        if (this.sigueLaCarrera)
        {
            Thread nuevoHilo = new Thread(deportista.Moverse);
            this.hilos.Add(nuevoHilo);
            nuevoHilo.Start();
        }
    }
}
```

- Si ejecuto ".Abort()" una vez que terminó la ejecución del método Moverse, se lanzará una excepción. ✓
 Una vez finalizado el hilo podrá volver a iniciar ejecutando "nuevoHilo.Start()"
 Dará error en tiempo de compilación ya que el método Start de Thread no recibe argumentos.
 Todas son correctas

SEGUNDA PARTE

0 de 0 puntos

- * Describa y desarrolle los conceptos utilizando sus palabras.
- * De ser necesario, dar ejemplos que no se hayan dado en clase.

Describa la pirámide de pruebas de calidad. Alcance, tiempos, costos.

Existen 3 tipos de test de software en la pirámide. En la base nos encontramos con los test unitarios, son los que se encargan de probar el funcionamiento de cada método/función por separado, es decir independientemente del resto, son las pruebas más baratas y más rápidas de realizar.

En el medio nos encontramos con las pruebas integrales, son aquellas que se llevan a cabo luego de haber sido realizadas las unitarias, estas pruebas verifican que los métodos funcionen correctamente probandolos juntos en bloque.

Por último en la cima nos encontramos con las pruebas funcionales, son las más costosas y las que me llevan tiempo ya que son las encargadas de testear todas las funcionalidades del software

Comentarios

Las PRUEBAS FUNCIONALES también llamadas End-To-End, se encuentran en la cúspide de la pirámide porque son las más lentas y costosas de elaborar y mantener. Verifican el flujo completo de la aplicación, simulan un usuario real.

Se prueba el ciclo completo, desde la interfaz de usuario, la interacción con los datos, la conexión con servicios externos, etc.

Las PRUEBAS INTELIGENTES se encuentran en el centro de la pirámide, son menos costosas que las Funcionales. Verifican que las relaciones existentes entre las diferentes clases y servicios funcionan correctamente. Son capaces de probar la mayoría de la lógica sin pasar por la interfaz de usuario.

Las PRUEBAS UNITARIAS o UNIT TEST se ubican en la base de la pirámide porque son más baratas y rápidas de hacer, se encuentran en mayor cantidad que las Inteligentes y Funcionales. Comprueban la funcionalidad específica de una pequeña parte de una aplicación, como clases y métodos.

Desarrolle qué es y cuándo se produce una Excepción. Describa InnerException y StackTrace.

Una excepción es la indicación de un error que ocurre cuando se está ejecutando un programa. El stackTrace brinda información de la pila de llamadas durante la ejecución de un programa antes de que se produzca una excepción y una InnerException es una propiedad que almacena info sobre la excepción interna

Comentarios

Se producen por un error en el programa en tiempo de ejecución. Las excepciones detienen el flujo actual del programa, y si no son controladas el programa dejará de funcionar.

La propiedad InnerException obtiene la instancia de Exception que provocó la excepción actual. Devuelve el mismo valor que se pasó al constructor Exception (String, Exception), o null si el valor de la excepción interna no se proporcionó al constructor. Esta propiedad es de sólo lectura.

La propiedad StackTrace es una colección con orden de acceso LIFO que registra los métodos invocados previo al lanzamiento de una excepción.

Desarrollar el concepto de Generics, dar un ejemplo

Los tipos genericos son aquellos que son utilizados para declarar métodos y/o clases parametrizadas con seguridad de tipo, al momento de su declaración no es necesario especificar que de qué tipo van a hacer, ya que estos son especificados luego depende el uso que sea necesario.

A las clases parametrizadas se le pueden agregar restricciones para asegurarse que se usen los tipos de datos adecuados, estas restricciones se especifican utilizando la palabra reservada where y alguna de ellas son:

new() --> donde el argumento de tipo debe contener un constructor sin parámetros públicos
class --> el argumento debe ser tipo X referencia
<nombre clase> --> el argumento debe ser derivada de la clase base especificada o la propia clase base

Un ej de los genericos pueden ser las List<tipo_dato> ya que estas aceptan cualquier tipo de dato dependiendo el uso que se le quiera dar
O si tengo un mismo metodo que cumple la misma función pero para diferentes tipos , solo creo un metodo con tipo generico donde luego puedo usar el dato que desee

Aclaracion: se pueden declarar métodos genéricos en clases no genéricas

Comentarios

C# Nos permite definir clases, interfaces, atributos, métodos, propiedades usando el operador de tipo y sin especificar ningún tipo de dato específico. Un operador de tipo va a ser una plantilla que luego será reemplazado por tipo específico. Esto nos permite diseñar clases y métodos que difieren de una especificación por uno o más tipos hasta que la clase o método es declarada en el código. Generics también nos ofrece ciertas restricciones de tipos para achicar el espacio de datos que pueden ser reemplazados.

Generics combina la reusabilidad, seguridad de tipos y eficiencia que sus contrapartes no genéricas no pueden lograr. Generics es muy frecuentemente utilizado con colecciones y métodos que operan con ellas, las cuales nos permiten crear nuestras colecciones genéricas fuertemente tipadas que nos provee una eficiencia superior a sus contrapartes no genéricas.

Desarrollar que son los métodos de extensión, como se utilizan y dar un ejemplo

Los métodos de extensión como lo indica su nombre, son métodos que se crean para agregarle funcionalidad a un tipo ya existente, son métodos estáticos pero en su llamada se invocan como si fuera de instancia sobre el tipo extendido, por ejemplo puedo crear un método que no existe para agregarle funcionalidad a un tipo de dato string que cuente la cantidad de oración de un texto y lo invoco de la siguiente manera:

```
string texto = "ggsahgsah.gaggsfdg.gsgagag";  
int total = texto.ContarParrafos();
```

Comentarios

Los métodos de extensión son métodos estáticos definidos en clases estáticas que son llamados como si fueran métodos de instancia del tipo extendido y su primer parámetro va a ser el tipo que queremos extender antecedido por el modificador this. Los métodos de extensión nos permiten añadir métodos a tipos existentes sin la necesidad de modificar el tipo original. Si por ejemplo quisieramos extender un método del tipo int para hacer una validación la firma del método sería la siguiente: public static bool [EsMayorQue(this int entero, int numero)]. Los métodos de extensión pueden ser usados en cualquier parte de la aplicación incluyendo el namespace del método de extensión

Explicar La funcionalidad del siguiente código tomando como referencia un formulario que tiene dos controles, un botón y un textBox

```
private delegate void DelegadoSeguro(string texto);  
  
private void Button1_Click(object sender, EventArgs e)  
{  
    Thread thread2 = new Thread(new ThreadStart(SetearTexto));  
    thread2.Start();  
}  
  
private void EscribirTextoSeguro(string texto)  
{  
    if (this.textBox1.InvokeRequired)  
    {  
        DelegadoSeguro d = new DelegadoSeguro(EscribirTextoSeguro);  
  
        object[] args = new object[]  
        {  
            texto  
        };  
  
        this.textBox1.Invoke(d, args);  
    }  
    else  
    {  
        this.textBox1.Text = texto;  
    }  
}  
  
private void SetearTexto()  
{  
    this.EscribirTextoSeguro("Este texto se seteo de una forma segura");  
}
```

Comentarios

Dentro de un formulario, cada vez que disparemos el evento click del control Button1 se va a crear un hilo secundario en nuestro programa que va a tratar de escribir un mensaje en nuestro control de texto textBox1. Pero como no podemos acceder a los controles de windows form desde un subproceso distinto al cual ese control fue creado tenemos que realizar una validación, por medio de la propiedad InvokeRequired nosotros podemos preguntar si estamos tratando de acceder a nuestro control desde un subproceso distinto a donde ese control fue creado. Si estamos en un subproceso distinto la propiedad InvokeRequired nos va a devolver true, en ese caso vamos a realizar una instancia de nuestro delegado DelegadoSeguro pasándole como parámetro la referencia a el método donde estamos actualmente y luego armamos un array de object con los argumentos requeridos por nuestro método para así poder lanzar un evento en nuestro control por medio del invoke pasandole el delegado y los argumentos requeridos para el método EscribirTextoSeguro() por medio del delegado donde el control fue creado, como estamos en el hilo principal del control la propiedad InvokeRequired nos va a devolver false y el flujo de nuestro programa va a seguir, pudiendo escribir el mensaje en el control de tipo textBox

Este contenido no ha sido creado ni aprobado por Google. - [Términos del Servicio](#) - [Política de Privacidad](#)

