

Que retorna el siguiente código al realizar la operación matemática. \*

```
O referencias
static void Main(string[] args)
{
    int numeroA = 25;
    int numeroB = 0;
    int numeroC = 5;
    int resultado;

    try
    {
        resultado = (numeroA / numeroB) * numeroC;
        Console.WriteLine(resultado);
    }
    catch (NullReferenceException ex)
    {
        Console.WriteLine(ex.Message);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    catch (DivideByZeroException ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

- ☐ 25
- ☐ Error en tiempo de Diseño. ✓
- ☐ Error en tiempo de Ejecución.
- ☐ 125
- ☐ Mensaje de NullReferenceException.
- ☐ Mensaje de DivideByZeroException.
- ☒ Mensaje de Exception. ✗
- ☐ Ninguna respuesta.

## Excepciones \*

- ☐ Algunas excepciones no tienen la propiedad InnerException.
- ☒ La propiedad InnerException nos permite almacenar una excepción dentro de otra. ✓
- ☐ La sentencia "throw;" dentro de un bloque catch, reiniciará el stack trace.
- ☒ La sentencia "throw ex;" dentro de un bloque catch, reiniciará el stack trace. ✓
- ☐ Si existen múltiples bloques catch que coincidan con el tipo de la excepción, se ejecutarán todos.
- ☐ Ninguna respuesta.

## Qué pasa si una excepción no es manejada \*

- ☐ Se abre un nuevo hilo de trabajo.
- ☐ El programa no compila.
- ☐ Sale un mensaje de error y continua el flujo.
- ☒ Se detiene el flujo y el programa deja de funcionar. ✓
- ☐ Ninguna respuesta.

## Test Unitarios \*

- ☐ Nuestra clase test heredará de UnitTest.
- ☐ A través de métodos estáticos de la clase Act podré verificar el resultado de cada Test.
- ☒ A través de métodos estáticos de la clase Assert podré verificar el resultado de cada Test. ✓
- ☒ Con ellos podré verificar cómo se comporta una porción del código interpretando valores de variables, excepciones lanzadas, etc. ✓
- ☐ Ninguna respuesta.

¿Qué son los test unitarios? \*

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

Como se Ejecutan las pruebas de un test unitario \*



- ☐ Corren automáticamente cuando se debuguea el proyecto.
- ☒ Desde el explorador de pruebas. ✓
- ☐ Desde la consola del proyecto.
- ☐ Desde la clase Test Unitarios.
- ☐ Ninguna respuesta.

Partiendo de la siguiente declaración ERRONEA!! de un método de extensión, seleccione las correcciones a aplicar para que dicha declaración funcione como un método de Extensión \*


```
O referencias
public class MiMetodoDeExtension
{
    O referencias
    public double ElevadoALa(int numero, int exponente)
    {
        return Math.Pow(numero, exponente);
    }
}
```

- ☐ Para poder hacer uso del método "ElevadoALa" deberé de instanciar un Objeto de tipo Int.
- ☐ La clase "MiMetodoDeExtension" debe ser de instancia.
- ☐ La clase "MiMetodoDeExtension" debe ser de instancia y heredar de extends.
- ☒ La clase "MiMetodoDeExtension" deberá cambiarse a estática. ✓
- ☐ La clase "MiMetodoDeExtension" deberá cambiarse a estática y heredar de extends.
- ☒ El método "ElevadoALa" deberá cambiarse a estático. ✓
- ☐ El método "ElevadoALa" debe ser de Instancia.
- ☒ El primer parámetro que recibe mi método debe anteponer la palabra reservada This. ✓
- ☐ Ninguna respuesta.



### Al generar un método de extensión \*

- ☐ Creo una nueva clase que hereda de una clase preexistente.
- ☐ Hago referencia a la clase extendida mediante la palabra reservada base.
- ☐ Anulo los métodos de la clase extendida.
- ☐ El método y la clase deben ser de instancia.
- ☒ Permiten extender clases estáticas. 
- ☐ Ninguna respuesta. 

### ¿Qué es un método de extensión? \*

- ☒ Son métodos definidos como estáticos pero que se los llama como si fuesen un método de instancia 
- ☐ Un método no sobrescribible.
- ☐ Un método estático creado con "extends".
- ☐ Un método parametrizado.
- ☐ Ninguna respuesta.

### Genéricos \*

- ☐ Sólo puedo declarar métodos genéricos dentro de clases que también sean genéricas
- ☐ Las interfaces no pueden especificar tipos genéricos.
- ☒ Se pueden utilizar en clases, métodos, atributos y propiedades. 
- ☒ Sólo se pueden aplicar restricciones o constraints a los tipos genéricos de una clase, no pudiéndose hacer en otros casos como interfaces o métodos.  Se puede aplicar a un metodo. Ejemplo: JsonSerializer
- ☐ Ninguna respuesta.

Que puedo ingresar en el parámetro "dato" de la siguiente declaración de clase: \*

```
1 referencia
public class ClaseGenerica<T> where T : Persona, new ()
{
    private T dato;

    0 referencias
    public ClaseGenerica(T dato)
    {
        this.dato = dato;
    }
}
```

- ☐ Una instancia de Persona.
- ☐ Una instancia de Persona, si tiene constructor por defecto. ✓
- ☒ Una instancia de un hijo de la Clase Persona que tengan constructor por defecto. ✓
- ☐ Una instancia de un hijo de la Clase Persona.
- ☐ Ninguna respuesta.

Cual/es de las siguientes declaraciones de clases es correcta \*

- ☐ public class Dinosaurio<T,J> where T,J : new() {}
- ☐ public class Dinosaurio<T> where T is new() {}
- ☒ public class Dinosaurio<T, J> where T : new() where J : new() {} ✓
- ☒ public class Dinosaurio<T> where T: new() {} ✓
- ☐ public class Dinosaurio<T, J> where T : new() where J : T {} ✓
- ☐ Ninguna respuesta.





Responda en base al código de la Imagen. IConexiones es una Interfaz \*

```
3 referencias
public interface IConexiones
{
    3 referencias
    void Encender();
    2 referencias
    void Apagar();
}

0 referencias
public class Bluetooth : IConexiones
{
    1 referencia
    public void Apagar()
    {
        //implementacion
    }
    2 referencias
    public void Encender()
    {
        //implementacion
    }
}

0 referencias
public class Wifi : IConexiones
{
    1 referencia
    public void Apagar()
    {
        //implementacion
    }
    2 referencias
    public void Encender()
    {
        //implementacion
    }
}

0 referencias
public class TelefonoMovil
{
    0 referencias
    public void ActivarConexion(IConexiones conectar)
    {
        conectar.Encender();
    }
}
```

- ☐ El metodo "ActivarConexion" de "TelefonoMovil" podrá recibir objetos de Tipo Wifi como objetos de tipo Bluetooth 
- ☐ No es obligatorio que Bluetooth implemente todas las operaciones definidas en IConexiones
- ☐ No es obligatorio que Wifi implemente todas las operaciones definidas en IConexiones
- ☐ El código da error ya que un método no puede tener una interfaz como tipo de parámetro de entrada
- ☐ La implementacion de IConexiones en Wifi debe ser de forma explicita
- ☐ TelefonoMovil tambien debe implementar IConexiones
- ☒ Ninguna respuesta. 

## Interfaces \*

- ☐ Las interfaces no pueden especificar tipos genéricos.
- ☒ Todos los elementos que las componen son públicos. ✓
- ☒ La clase que las implemente deberá darle una implementación a cada una de las operaciones especificadas por la interfaz. ✓
- ☐ Contienen atributos, propiedades y métodos abstractos.
- ☐ Todos sus métodos deben ser virtuales.
- ☐ 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.
- ☐ Ninguna respuesta.

## Cuántas interfaces pueden implementarse en una clase \*

- ☐ Una
- ☐ Las interfaces no se implementan
- ☐ Dos
- ☐ Todas las que hereden del mismo padre.
- ☒ Todas las que se requieran. ✓
- ☐ Ninguna respuesta.



Teniendo en cuenta el siguiente método, indicar las afirmaciones correctas. \*

2 referencias

```
public void Escribir(string dato, string path)
{
    try
    {
        using (StreamWriter streamWriter = new StreamWriter(path, true))
        {
            streamWriter.WriteLine(dato);
        }
    }
    catch (Exception)
    {
        throw;
    }
}
```

- ☒ Si declaro e instancio el objeto StreamWriter en una sentencia "using", no deberé preocuparme por cerrar el vínculo con el archivo. ✓
- ☐ Si el archivo existe lo sobrescribirá.
- ☒ Si el archivo existe agregará información al mismo. ✓
- ☐ Me servirá para serializar en formato XML.
- ☒ Me servirá para serializar en formato JSON. ✗
- ☐ Ninguna respuesta.

En que namespace se encuentra StreamReader \*

- ☒ System.IO ✓
- ☐ System.Data
- ☐ System.Collections.Generic
- ☐ System.Collections.Generic
- ☐ Microsoft.Data.SqlClient
- ☐ Ninguna respuesta.

Que código es correcto para verificar la existencia de un directorio \*

- ☐ Directory D = new Directory(); D.Exists(path);
- ☐ File.Exists(path);
- ☐ Directory.Exists(path); ✓
- ☒ Path.Exists(path); ✗
- ☐ Path P = new Path(); P.Exists(path);
- ☐ Ninguna respuesta




Como podemos acceder a las carpetas del sistema sin conocer su ruta completa \*

- ☐ A través del metodo GetFolderPath de la clase File.
- ☐ A través del metodo GetFolderPath de la clase Directory.
- ☒ A través del metodo GetFolderPath de la clase Environment. ✓
- ☐ Ninguna respuesta.



Serialización Binaria \*

- ☐ Una de las ventajas de la serialización binaria es que es más segura que la serialización XML.
- ☐ Cuando serializamos a binario, los atributos que se deseen serializar deberán ser públicos (o publicados a través de propiedades de lectura-escritura)
- ☒ La clase "BinaryFormatter" sera la encargada de serializar objetos en formato binario. ✓
- ☒ Uno de los requisitos es que la clase posea la etiqueta [Serializable]. ✓
- ☒ Exporta todos los métodos y propiedades ✗
- ☐ Exporta todos los métodos y propiedades públicos
- ☐ Ninguna respuesta.

### Serialización XML \*

- ☒ Los objetos que se deseen serializar en formato XML deben tener un constructor público sin parametros 
- ☐ Solo se podrán serializar los atributos y propiedades privadas.
- ☒ Solo se podrán serializar los atributos y propiedades públicas. 
- ☒ Una de las ventajas de la serialización binaria es que es más segura que la serialización JSON. 
- ☐ Podemos serializar en archivos de texto plano.
- ☐ Exporta todos los métodos y propiedades públicos
- ☐ Ninguna respuesta.

### Serialización JSON \*

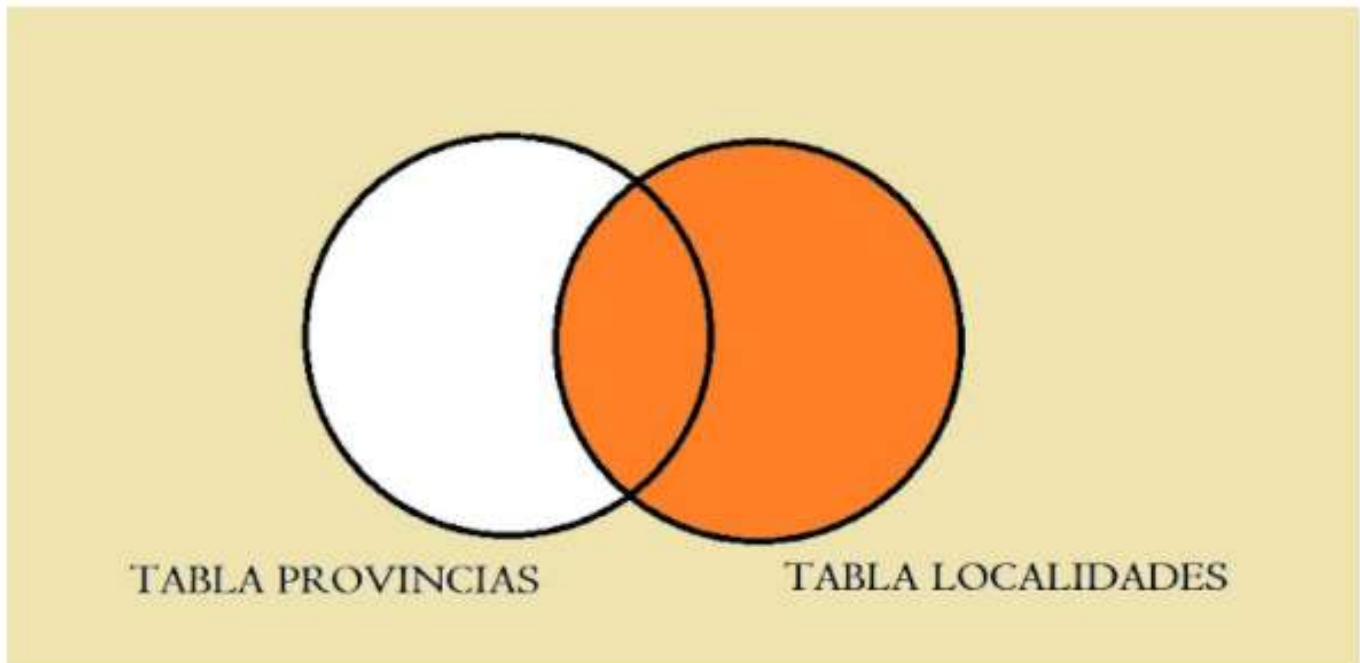
- ☒ Se deberá adicionar el espacio de nombres System.Text.Json 
- ☐ Se deberá adicionar el espacio de nombres System.Text
- ☐ Los objetos que se deseen serializar en formato JSON deben tener un constructor público sin parametros
- ☐ Una de las ventajas de la serialización JSON es que es más segura que la serialización XML.
- ☐ Deberé de generar una instancia de la clase "JsonSerializer", para poder serializar y deserializar.
- ☒ La deserialización la podre realizar a través del método de clase "Deserialize", indicándole el tipo de dato a deserializar. 
- ☐ Ninguna respuesta.

Que retorna el siguiente método \*

```
1 referencia
public static string GetProvincia(int id)
{
    string returnAux = string.Empty;
    string cadenaConexion = "Server=.;Database=Provincias;Trusted_Connection=True;";
    SqlConnection conexion = new SqlConnection(cadenaConexion);
    SqlCommand sqlCommand = new SqlCommand();
    try
    {
        conexion.Open();
        SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();
        while (sqlDataReader.Read())
        {
            returnAux = sqlDataReader.GetString(1);
        }
        return returnAux;
    }
    catch (Exception)
    {
        return "Error en la sentencia SQL";
    }
    finally
    {
        if (conexion != null && conexion.State == System.Data.ConnectionState.Open)
        {
            conexion.Close();
        }
    }
}
```

- ☐ El nombre de la provincia respecto del ID enviado por parámetro
- ☐ La ultima provincia de la tabla Provincias
- ☒ Error en la sentencia SQL ✓
- ☐ Error en tiempo de ejecución
- ☐ Ninguna respuesta.

Indique a qué consulta corresponde el siguiente diagrama de Venn \*



- ☒ `SELECT * FROM PROVINCIAS P RIGHT JOIN LOCALIDADES L ON P.Provinciald = L.Provinciald;` ✓
- ☐ `SELECT * FROM PROVINCIAS P LEFT JOIN LOCALIDADES L ON P.Provinciald = L.Provinciald;`
- ☐ `SELECT * FROM PROVINCIAS P INNER JOIN LOCALIDADES L ON P.Provinciald = L.Provinciald;`
- ☐ `SELECT * FROM PROVINCIAS;`
- ☐ `DELETE FROM PROVINCIAS WHERE Pronvinciald = LocalidadesId`
- ☐ `DELETE FROM PROVINCIAS *`
- ☐ Ninguna Respuesta

### Que realiza el connectionString \*

- ☐ Realiza la conexión a la base
- ☒ Es un string que contiene la ruta y los datos de logeo a la base ✓
- ☐ Crea un enlace con la base de datos
- ☐ Realizar una consulta en la BD
- ☐ Ninguna respuesta.

### Delegados \*

- ☐ Son métodos ejecutables.
- ☐ Son colecciones de métodos.
- ☒ Es un tipo que encapsula referencias a métodos. ✓
- ☐ Solo se podrán asignar a este, métodos que reciban los mismos tipos de parámetros sin importar la cantidad, ni el tipo de retorno, respecto de la declaración
- ☐ Solo se podrán asignar a este, métodos que reciban los mismos tipos y cantidad de parámetros pero sin importar el tipo de retorno, respecto de la declaración.
- ☒ Solo se podrán asignar a este, métodos que reciban los mismos tipos, cantidad de parámetros y retornen el mismo tipo, respecto de la declaración. ✓
- ☐ Ninguna respuesta



Indique el resultado de la siguiente operación: \*

```
public delegate void DelegadoCalcular(int numero, ref int resultado);
```

0 referencias

```
static void Main(string[] args)
```

```
{  
    int numero = 2;  
    int resultado=0;  
    DelegadoCalcular realizarCalculo = SumarNumero;  
    realizarCalculo += MultiplicarNumero;  
  
    realizarCalculo(numero, ref resultado);
```

```
    Console.WriteLine(resultado);  
    Console.ReadKey();  
}
```

1 referencia

```
public static void SumarNumero(int numero, ref int resultado)
```

```
{  
    resultado = numero + numero;  
}
```

1 referencia

```
public static void MultiplicarNumero(int numero, ref int resultado)
```

```
{  
    resultado *= numero;  
}
```

☐ 2

☐ 0

☐ 16

☐ 4

☒ 8



☐ Error en tiempo de ejecucion

☐ 32

☐ Ninguna respuesta



Partiendo de la siguiente declaración. Seleccione la correcta declaración de delegado que permita encapsular el método asignado a "delegadoFunc": \*



```
Func <int,int,bool> delegadoFunc = (numA, numB) => numA >= numB;
```

- ☒ public delegate bool MiDelegado(int numA, int numB)
- ☐ public delegate int MiDelegado(bool numA, bool numB);
- ☐ public delegate bool MiDelegado(int numA, bool numB);
- ☐ public delegate bool MiDelegado(int numA);
- ☐ public delegate int MiDelegado(int bool);
- ☐ public delegate int MiDelegado(int numA, int numB)
- ☐ Ninguna respuesta.

## Expresiones Lambda \*

- ☒ Pueden ser convertidas a un tipo delegado. El tipo delegado deberá coincidir con los parámetros y el tipo de retorno de la expresión.
- ☐ Pueden ser convertidas a un tipo delegado. El tipo delegado NO siempre deberá coincidir con los parámetros y el tipo de retorno de la expresión.
- ☐ Cuando el cuerpo del método se compone de una sola sentencia, no lleva llaves y esta posee un return implícito.
- ☐ Cuando el cuerpo del método no se compone de una sola sentencia, no lleva llaves y esta posee un return implícito.
- ☒ Cuando el cuerpo del método no se compone de una sola sentencia, lleva llaves y debo realizar un return de forma explícita..
- ☐ Ninguna respuesta.

Quando creamos un hilo secundario \*

- ☒ Debemos esperar que termine para que no se aborte al cerrar el hilo principal. 
- ☐ Siempre debe tener un evento para que ambos hilos se comuniquen.
- ☐ Debemos abrirlo en un bloque try para que no afecte al hilo principal.
- ☐ Debemos garantizar que el método termine en algún momento para que no quede corriendo al cerrar el hilo principal. 
- ☐ Ninguna respuesta.

Teniendo en cuenta el siguiente código, seleccione la/las respuestas correctas. \*

```
0 referencias
static void Main(string[] args)
{
    CancellationSource cancelationTokenSource= new CancellationSource();
    CancellationToken cancellation = cancelationTokenSource.Token;
    int num = 0;
    Task.Run(() =>
    {
        do
        {
            num++;
            Console.Write($"{num}-");
            Thread.Sleep(1000);
        } while (!cancellation.IsCancellationRequested);
    },cancellation);




    do
    {
        } while (Console.ReadKey().Key != ConsoleKey.Escape);

    cancelationTokenSource.Cancel();


    Console.ReadKey();
}
```

- ☐ Error en tiempo de ejecución.
- ☒ En un hilo secundario se imprimirá por pantalla distintos números, seguidos de un guion. ✓
- ☒ Mientras que no presione la tecla "Escape", el hilo secundario continuará activo. ✓
- ☐ Error en tiempo de diseño, no puedo enviarle "cancellation" al metodo Run
- ☐ El hilo secundario nunca inicia ya que tengo que generar una instancia de Task e invocar al metodo Start.
- ☐ Ninguna respuesta.

## Eventos \*

- ☒ La asociación entre un manejador y un evento se hace a través de un tipo delegado que debe tener la misma firma que el manejador. 
- ☐ Cada manejador o subscriber puede estar asociado a un sólo evento.
- ☐ Es un tipo que representa referencias a métodos con una lista de parámetros determinada y un tipo de valor devuelto.
- ☐ Son la forma que tienen las clases para comunicarse  Hay otras formas, me dio el punto el profesor.
- ☐ Sólo se pueden utilizar en conjunto con hilos
- ☐ La asociación entre un manejador y un evento se hace a través de un tipo delegado que puede tener distinta firma que el manejador 
- ☐ Son métodos que se ejecutan asincrónicamente.
- ☐ Ninguna respuesta

## Como se lanza un Evento \*

- ☐ Con el método Event.
- ☐ Con el +=
- ☐ Llamando al metodo por defecto de la instancia.
- ☒ Con el método Invoke.  También puedo llamarlo de la siguiente manera: evento
- ☐ Ninguna respuesta.