

Al hablar de polimorfismo *

- ☐ Cuando el usuario puede elegir distintas opciones de funcionamiento del sistema.
- ☐ Cuando el programador elige distintas opciones de funcionamiento del sistema.
- ☐ Se refiere a la posibilidad de definir múltiples clases con funcionalidad diferente, pero con métodos o propiedades denominados de forma idéntica, que pueden utilizarse de manera intercambiable mediante código en tiempo de diseño
- ☒ Se refiere a la posibilidad de definir múltiples clases con funcionalidad diferente, pero con métodos o propiedades denominados de forma idéntica, que pueden utilizarse de manera intercambiable mediante código en tiempo de ejecución.
- ☐ Ninguna respuesta.

Indique el orden de los eventos del ciclo de vida de Windows Forms del primero en lanzarse hasta el último. *

- ☐ New, Paint, Load, Activated, FormClosing, FormClosed, Disposed
- ☒ New, Load, Paint, Activated, FormClosing, FormClosed, Disposed
- ☐ New, Load, Paint, Activated, FormClosing, Disposed, FormClosed
- ☐ Load, New, Paint, Activated, FormClosing, FormClosed, Disposed
- ☐ Ninguna respuesta
- ☐ Load, New, Paint, Activated, FormClosing, Disposed, FormClosed

Indique el tipo de clase al que corresponde la siguiente descripción: "No puede ser clase base, ni derivada, ni instanciarse" *

- ☐ Privada
- ☐ Abstracta
- ☐ Sellada
- ☒ Estática
- ☐ Ninguna respuesta.

Las clases selladas (sealed) *

- ☐ No pueden ser clase base.
- ☐ No pueden ser base ni derivada.
- ☐ No se pueden instanciar ni heredar.
- ☐ No pueden ser clase derivada.
- ☒ Ninguna respuesta.

¿Qué características son propias de los formularios? *

- ☒ Se implementan mediante clases parciales (Partial Class).
- ☒ Todos los formularios heredan de Form.
- ☒ Si lanzo un formulario con el método Show este impedirá que utilicemos el Form Principal hasta que sea cerrado.
- ☒ Todos los formularios heredan de Object.
- ☐ No se puede modificar su constructor.
- ☐ Ninguna respuesta.

Miembros virtuales *

- ☐ Su implementación se encuentra sólo en las clases derivadas.
- ☐ Se pueden sobrescribir desde cualquier clase.
- ☐ Las clases derivadas están obligadas a implementar los miembros virtuales de la clase base.
- ☐ Se puede aplicar en constructores
- ☒ Se puede aplicar en propiedades
- ☒ Se puede aplicar en metodos
- ☐ Ninguna respuesta.

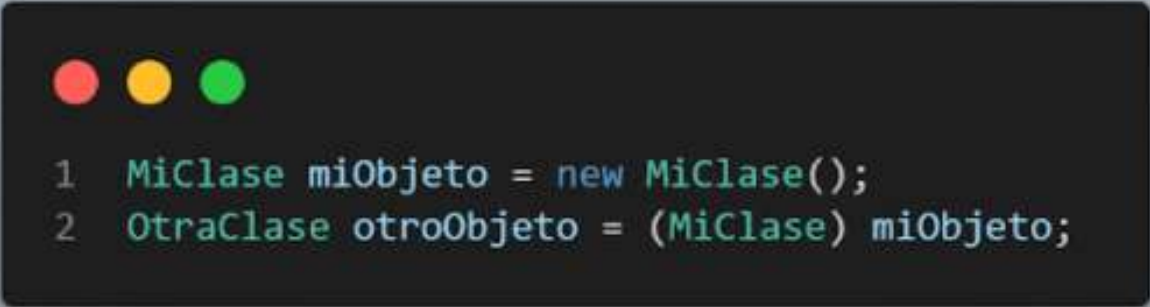
¿Cuál de las siguientes es una característica del Common Language Runtime de .NET? *

- ☐ Compila código fuente escrito en C#, F# o [VB.NET](#)
- ☐ Define los tipos de datos de .NET.
- ☐ Libera espacio de memoria ocupado por los tipos por valor (value types)
- ☒ Ninguna respuesta.

El lenguaje de programación C#: *

- ☐ Se compila a lenguaje intermedio en tiempo de ejecución y se compila a lenguaje máquina/nativo en tiempo de compilación
- ☐ Su ejecución es administrada por Visual Studio.
- ☐ Es un lenguaje interpretado y de tipado fuerte.
- ☒ Se compila a lenguaje intermedio en tiempo de compilación y se compila a lenguaje máquina/nativo en tiempo de ejecución.
- ☐ Ninguna respuesta.

Responda en base al código en la imagen, partiendo del supuesto de que no hay ningún error en el mismo: *



```
1  MiClase miObjeto = new MiClase();  
2  OtraClase otroObjeto = (MiClase) miObjeto;
```

- ☐ Es una conversión implícita, las cuales por definición pueden implicar pérdida de información.
- ☐ Es una conversión explícita, las cuales por definición pueden implicar pérdida de información.
- ☐ Todas las respuestas.
- ☐ Ninguna respuesta.
- ☒ Es una conversión explícita, las cuales por definición NO deberían implicar pérdida de información.
- ☐ Es una conversión implícita, las cuales por definición NO deberían implicar pérdida de información.

Indique cuáles de las siguientes afirmaciones sobre los pilares de la programación orientada a objetos son correctas. *

- ☒ La abstracción consiste en aislar un elemento de su contexto o del resto de los elementos que lo acompañan, seleccionando la información importante y descartando los detalles insignificantes.
- ☐ La abstracción es un concepto que propone acceder y/o modificar el estado de un objeto sólo a través de métodos y propiedades públicas.
- ☐ La abstracción es la habilidad de un objeto para cambiar su comportamiento en tiempo de ejecución
- ☐ El polimorfismo es la habilidad de crear una clase nueva a partir de una ya existente con características y/o comportamiento en común.
- ☒ El polimorfismo es la habilidad de un objeto para cambiar su comportamiento en tiempo de ejecución.
- ☐ Ninguna respuesta.

Los indexadores *

- ☒ Permiten a la instancia de una clase ser indexada tal cómo un array.
- ☐ Se utiliza la palabra reserva indexer en su definición.
- ☐ Es obligatorio indexarlos con un valor del tipo int.
- ☐ Es obligatorio que sean unidimensionales.
- ☐ Ninguna respuesta.

Una colección del tipo Stack *

- ☐ Mantiene un orden FIFO (primero en entrar, primero en salir).
- ☒ Mantiene un orden LIFO (último en entrar, primero en salir).
- ☐ Posee el método Sort para poder ordenar la colección según el criterio deseado.
- ☐ Tiene un par Clave valor asociados
- ☐ Ninguna respuesta.

Se tiene una Clase A que hereda de una Clase B (ClaseA : ClaseB). Indique cuáles de las siguientes afirmaciones son correctas. *

- ☒ La Clase derivada podrá ser instanciada mediante el constructor de la clase base.
- ☐ En C# una clase derivada podrá heredar directamente de dos clases base al mismo tiempo (Herencia múltiple).
- ☒ Si la Clase B hereda de una Clase C, entonces la Clase A también heredará indirectamente de la Clase C. Clase C -> Clase B -> Clase A (Transitividad).
- ☒ La Clase derivada heredará los miembros privados de la clase base.
- ☐ El objetivo de la herencia es repetir código, permitiendo definir una clase base a partir de una clase derivada.
- ☐ La Clase derivada puede ser más accesible que la clase base
- ☐ Ninguna respuesta.

Si tengo una clase B que hereda de una clase abstracta A, cada una de estas con un constructor estático, y 3 constructores en total con las visibilidades público, privado y protegido *

- ☒ El primer constructor que se llamará será el estático.
- ☐ El constructor privado de B no podrá llamar a ningún constructor de A.
- ☐ Al instanciar un objeto del tipo A no podré llamar a los constructores de B.
- ☒ El constructor privado de B sólo podrá ser llamado desde el constructor público de B.
- ☐ Al instanciar un segundo elemento del tipo B, el primer constructor que se llamará será el público.
- ☐ Ninguna respuesta.

El ciclo de vida de un objeto: *

- ☒ Comienza al asignar memoria mediante la palabra reservada new.
- ☐ Comienza al asignar memoria mediante la llamada al constructor.
- ☐ Su existencia finaliza al eliminar una o todas las referencias que tenemos a él.
- ☒ No desaparece de memoria hasta que el Garbage Collector decide liberarla.
- ☐ Ninguna respuesta.

Cuáles de estas definiciones sobre conversiones implícitas son ciertas: *

- ☐ Se utilizan cuando puede haber pérdida de información.
- ☐ Se invocan mediante la operación de casting.
- ☒ `float f = 10;` es una conversión implícita.
- ☒ Si deseo generar mi propia conversión, la firma será: `public static implicit operator(int a)`
- ☐ Ninguna respuesta.

Un método puede sobrecargarse *

- ☐ Solo en clases heredadas si esta declarado como virtual
- ☐ Solo en clases heredadas si esta declarado como override
- ☐ Modificando el orden de los nombre de los parametros
- ☒ Modificando el numero de parametros o el tipo
- ☐ Ninguna respuesta.

Que métodos puedo sobrescribir *

- ☐ Los declarados con el modificador virtual en una clase sellada, al heredar dicha clase
- ☐ Los declarados con el modificador abstract o virtual en una clase no abstracta, al heredar dicha clase
- ☒ Los declarados con el modificador virtual en una clase sin modificadores, al heredar dicha clase
- ☒ Los declarados con el modificador new una vez heredados.
- ☐ Ninguna respuesta.

Los indexadores sólo se pueden indexar por *

- ☒ Valores numericos
- ☒ Objetos
- ☒ Valores string
- ☐ Ninguna respuesta.

El método ShowDialog() *

- ☒ Muestra un formulario de forma modal, permitiendo interactuar con otras ventanas.
- ☐ Muestra un formulario de forma modal, NO permitiendo interactuar con otras ventanas.
- ☐ Muestra una ventana emergente con un mensaje y botones para contestar "aceptar" "cancelar"
- ☐ Ninguna respuesta.

¿Cuál es la salida del programa de la imagen? *

0 referencias

```
static void Main(string[] args)
{
    int max = 5;
    Queue<double> data = new Queue<double>();
    int i;

    for (i = max; i > 0 ; i--)
    {
        data.Enqueue(i);
    }

    for (i = 0; i < max; i++)
    {
        Console.Write($"{data.Dequeue()}-");
    }

    Console.ReadKey();
}
```

- ☐ Error en tiempo de compilación
- ☐ 1-2-3-4-5-
- ☒ 5-4-3-2-1-
- ☐ Error en tiempo de ejecución
- ☐ Todas las respuestas

Responda en base al código en la imagen, partiendo del supuesto de que no hay ningún error en el mismo: *

```
public abstract class Club
{
    protected double precioBase;
    0referencias
    public Club()
    {
        this.precioBase = 900.50F;
    }
    2referencias
    public abstract double PrecioSubscripcion { get; }
}
0referencias
public class Gimnasio : Club
{
    1referencia
    public override double PrecioSubscripcion
    {
        get
        {
            double precioBruto = base.precioBase + base.precioBase * 0.5F;
            double precioNeto = precioBruto + precioBruto * 0.21F;
            return precioNeto;
        }
    }
}
0referencias
public class Natacion : Club
{
    1referencia
    public override double PrecioSubscripcion
    {
        get
        {
            double precioBruto = base.precioBase + base.precioBase * 0.1F;
            double precioNeto = precioBruto + precioBruto * 0.21F;
            return precioNeto;
        }
    }
}
```

- ☒ Se debe llamar al constructor de la clase base de forma explícita (:base), de otra forma precioBase valdrá 0.
- ☒ Sólo la clase "Gimnasio" está obligada a sobrescribir la propiedad "PrecioSubscripcion"
- ☐ Tanto la clase "Natacion" como la clase "Gimnasio" están obligadas a sobrescribir la propiedad "PrecioSubscripcion"
- ☐ Sólo la clase "Natacion" está obligada a sobrescribir la propiedad "PrecioSubscripcion"
- ☐ Ninguna respuesta.

¿Cuál es el resultado (numérico) del código en la imagen? NO tiene errores. *

```
namespace Clase_04
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            SuperCalculator superCalculator = new SuperCalculator();
            superCalculator += 3;
            superCalculator += 3;
            superCalculator += 1;
            Console.WriteLine($"Resultado {superCalculator.Resultado}");
            Console.ReadKey();
        }
    }
}

5 referencias
public class SuperCalculator
{
    private Stack<int> numeros;

    1 referencia
    public SuperCalculator()
    {
        this.numeros = new Stack<int>();
    }

    1 referencia
    public double Resultado
    {
        get
        {
            int resultado = 0;
            foreach (int numero in this.numeros)
            {
                resultado += numero;
            }
            return resultado;
        }
    }

    3 referencias
    public static SuperCalculator operator + (SuperCalculator superCalculator, int numero)
    {
        foreach (int item in superCalculator.numeros)
        {
            numero *= item;
        }
        superCalculator.numeros.Push(numero);
        return superCalculator;
    }
}
```

6+3+1 = 10.

Los Constructores: *

- ☒ Si no se declaró de forma explícita un constructor, existirá un constructor por defecto sin parámetros
- ☒ Los constructores de instancia, inicializan los atributos del objeto
- ☐ Estáticos no pueden ser invocados
- ☒ Estáticos son de la clase, no de una instancia específica
- ☐ Ninguna respuesta.

Indique los valores de para "OpcionTres" y "OpcionNueve" para la siguiente declaracion de Enumerado *

```
namespace Entidades
{
    0 referencias
    public enum Enumerado {OpcionUno = 1, OpcionDos=3, OpcionTres , OpcionDiez=11, OpcionQuince=22, OpcionNueve};
}
```

- ☒ OpcionTres = 4 y OpcionNueve = 23
- ☐ OpcionTres = 3 y OpcionNueve = 9
- ☐ OpcionTres = 2 y OpcionNueve = 6
- ☐ OpcionTres = null y OpcionNueve = Undefined
- ☐ Ninguna respuesta.

Al recorrer una coleccion Dictionary<int, string> con un foreach, cada item obtenido sera del tipo: *

- ☒ KeyValuePair<int,string>
- ☐ Dictionary<int, string>
- ☐ String
- ☐ Int
- ☐ No se puede recorrer con un foreach una colección de tipo Dictionary