BOLETÍN DE EJERCICIOS TEMA 5

PROFESOR: MARTÍN GARCÍA FIGUEIRA
martin@ciclosmontecastelo.com

1. **Comprobando el código**

   ¿Qué sucedería si eliminamos la declaración local de la variable n en el siguiente código?

```
public partial class Form1 : Form
{
    private int n = 8;
    private void MiMétodo()
    {
        int n;
        n = 3;
    }
}
```

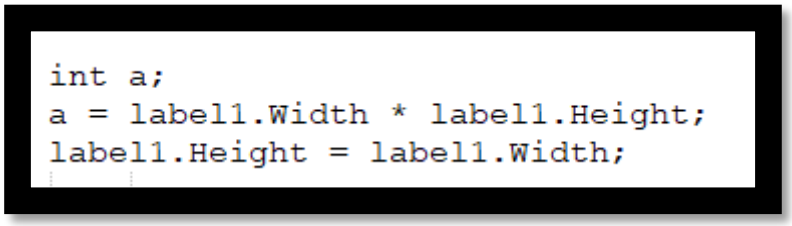
2. **Comprobando el código II**

   ¿Qué error encuentras en el siguiente código?

```
Public Form1()
{
    label1.Text = "38";
    InitializeComponent();
}
```

3. **Comprobando el código III**

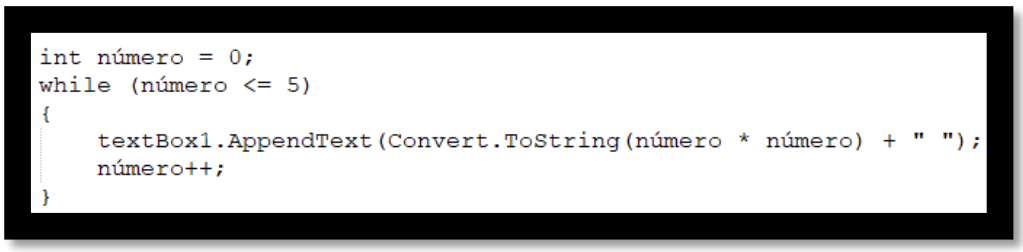
   ¿Qué funcionalidad tiene el siguiente código?



```
int a;
a = label1.Width * label1.Height;
label1.Height = label1.Width;
```

4. Comprobando el código IV

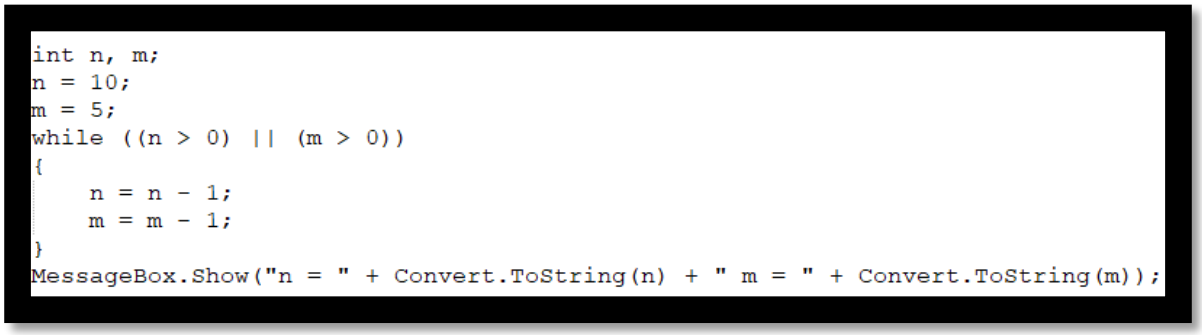
¿Qué función tiene el siguiente código?



```
int número = 0;
while (número <= 5)
{
    textBox1.AppendText(Convert.ToString(número * número) + " ");
    número++;
}
```

5. Comprobando el código V

¿Qué aparece en pantalla con el siguiente código?



```
int n, m;
n = 10;
m = 5;
while ((n > 0) || (m > 0))
{
    n = n - 1;
    m = m - 1;
}
MessageBox.Show("n = " + Convert.ToString(n) + " m = " + Convert.ToString(m));
```

6. Creando métodos y propiedades

Imagina que queremos hacer una clase de un reproductor MP3; ¿qué métodos y propiedades podría tener? ¿Cuáles de estos métodos y propiedades podrían ser miembros?

7. Creando clases I

Entré en un concesionario y después de ver varios coches compré un Ford de 2010, con capacidad para 5 personas, de 100CV y de color azul:

- ¿Cómo sería el constructor Coche?
- ¿El constructor cuenta con parámetros?
- ¿Cómo instanciarías la clase Coche para representar el ejemplo?

8. Creando clases II

Amplía el objeto Globo de manera que tenga una variable que describa el color del globo.

```
public class Globo
{
    private int x = 50;
    private int y = 50;
    private int diámetro = 20;
}
```

9. Creando clases III

- Escribe un método que mueva el globo hacia arriba por una cantidad que se proporcionará como parámetro. Usa el nombre MoverArriba.
- Escribe un método que permita cambiar el color del globo.
- Escribe el método Mostrar, de manera que despliegue un globo de color.
- Escribe una propiedad para permitir que un usuario sólo tenga acceso get a la coordenada “y” de un globo.
- Escriba un método constructor para crear un nuevo globo, especificando únicamente su diámetro.
- Escribe una lista de operaciones que puedan realizarse con un objeto de la clase Globo, y da ejemplos de cómo usarlas.

10. Creando clases IV

- Crea una clase 'Cuadrado' con propiedades 'Altura' y 'Ancho' y un método 'Area()'. Controla que no permita usar valores negativos.
- Crea un objeto de la clase Cuadrado y modifica sus propiedades. Visualiza el área del mismo.
- Sustituye el método por defecto 'ToString()', para que muestre información válida del objeto.

11. ¿Método o propiedad?

¿Cuáles de los siguientes elementos deberían ser métodos y cuáles propiedades al diseñar una clase llamada Cuenta para representar una cuenta bancaria?

- AbonarEnCuenta
- RetirarDeCuenta
- SaldoActual
- CalcularInterés
- Nombre

12. Creando un objeto II

Queremos crear un objeto que:

- Tenga propiedades 'Tipo' (Bus, Coche, Tren, etc), 'Fabricante', 'N_Ruedas', 'Pasajeros', 'consumo', etc.
- Tenga una variable pública y estática 'PrecioGasolina'
- Tenga un método estático PrecioPorKm()'.

Crea la clase que nos permita instanciarla para crear ese objeto.

13. Problema I: Pantalla de amplificador

Algunos amplificadores estereofónicos cuentan con un dispositivo visual que muestra el volumen de salida. Ese dispositivo aumenta y disminuye su nivel de acuerdo con el volumen en cualquier momento dado. De igual manera, ciertas pantallas tienen indicadores que muestran los valores máximo y mínimo alcanzados desde que el amplificador se encendió.

Escribe un programa que muestre en cuadros de texto los valores máximo y mínimo a los que se haya establecido una barra de seguimiento.

Escribe el código que tenga los valores y los compare en una clase. Esta clase debe tener un método llamado NuevoValor junto con las propiedades MenorValor y MayorValor.

14. Problema II: Cuenta bancaria

Escribe un programa que simule una cuenta bancaria. Un cuadro de texto debe permitirnos realizar depósitos (un número positivo) en la cuenta y hacer retiros (un número negativo) de la misma. El estado de la cuenta debe mostrarse de manera continua, y si entra en números rojos (saldo negativo) desplegar un mensaje apropiado.

Crea una clase llamada Cuenta para representar cuentas bancarias. Debe tener los métodos Depositar y Retirar, junto con una propiedad llamada SaldoActual.

15. Problema III: Guardar puntuación

Diseña y escribe una clase que sirva para guardar la puntuación para un juego de ordenador.

Debe mantener un solo entero: la puntuación.

Además, es necesario que proporcione métodos para inicializar la puntuación en cero, para incrementar la puntuación, para reducir la puntuación, y para devolver la puntuación.

Escribe instrucciones para crear y utilizar un solo objeto.

16. Problema IV: Dados

Diseña y escribe una clase que actúe como un dado que se pueda lanzar para obtener un valor del 1 al 6.

Primero escribe la clase de manera que siempre obtenga el valor 6.

Crea un programa para crear y utilizar un objeto dado. La pantalla debe mostrar un botón que al ser oprimido "lance" el dado y despliegue su valor.

Después modifica la clase dado, de manera que proporcione un valor un punto mayor al que tenía la última vez que se lanzó; por ejemplo, 4 si la última ocasión cayó en 3.

Luego transforma una vez más la clase, de manera que utilice el generador de números aleatorios de la biblioteca.

Algunos juegos, como el backgammon y el monopolio, requieren dos dados. Escribe instrucciones de C# para crear dos instancias del objeto dado, lanzarlos y mostrar los resultados.

17. Problema V: Generador de números aleatorios

Escribe tu propio generador de números aleatorios, creando

para ello una clase que utilice una fórmula para obtener el siguiente número pseudoaleatorio a partir del anterior.

Los programas de números aleatorios funcionan empezando con cierto valor de "semilla". A partir de ese momento, el número aleatorio actual se utiliza como base para obtener el siguiente. Esto se lleva a cabo realizando ciertos cálculos con el número actual para convertirlo en alguno otro (aparentemente aleatorio). Una fórmula que podemos usar para los enteros es:

siguienteA = ((anteriorA * 25173) + 13849) % 65536;

Esta fórmula produce números en el rango de 0 a 65,535. Los números específicos derivados de la misma han demostrado su capacidad para producir buenos resultados de tipo aleatorio.

18. Problema VI: Números complejos

Escribe una clase llamada Complejo para representar números complejos (junto con sus operaciones).

Los números complejos consisten de dos partes: una parte real (un double) y una imaginaria (un double).

El método constructor debe crear un nuevo número complejo mediante el uso de los valores double que se proporcionen como parámetros, de la siguiente forma:

Complejo c = new Complejo(1.0, 2.0);

Escribe las propiedades Real e Imaginaria para obtener las partes respectivas de un número complejo, mismas que deben utilizarse de la siguiente manera:

double x = c.Real;

Crea un método para sumar dos números complejos y devolver el resultado. La parte real es la suma de las dos partes reales; la parte imaginaria es la suma de las dos partes imaginarias.

Una invocación al método tendría esta forma:

Complex c = c1.Sum(c2);

Escribe un método para calcular el producto de dos números complejos.

Si un número tiene los componentes x1 e y1, y el segundo tiene los componentes x2 e y2:

- Parte real del producto = $x_1 \times x_2 + y_1 \times y_2$
- Parte imaginaria del producto = $x_1 \times y_2 + x_2 \times y_1$

19. Problema VII: Persona

Crea una clase llamada Persona que siga las siguientes condiciones:

- Sus atributos son: nombre, edad, DNI, sexo (H hombre, M mujer), peso y altura. No queremos que se accedan directamente a ellos. Piensa que

modificador de acceso es el más adecuado, también su tipo. Si quieres añadir algún atributo puedes hacerlo.

- Por defecto, todos los atributos menos el DNI serán valores por defecto según su tipo (0 números, cadena vacía para String, etc.). Sexo sera hombre por defecto, usa una constante para ello.
- Se implantarán varios constructores:
    - Un constructor por defecto.
    - Un constructor con el nombre, edad y sexo, el resto por defecto.
    - Un constructor con todos los atributos como parámetro.

- Los métodos que se implementaran son:
    - calcularIMC(): calculara si la persona está en su peso ideal (peso en kg/(altura^2 en m)), si esta fórmula devuelve un valor menor que 20, la función devuelve un -1, si devuelve un número entre 20 y 25 (incluidos), significa que está por debajo de su peso ideal la función devuelve un 0 y si devuelve un valor mayor que 25 significa que tiene sobrepeso, la función devuelve un 1. Te recomiendo que uses constantes para devolver estos valores.
    - esMayorDeEdad(): indica si es mayor de edad, devuelve un booleano.
    - comprobarSexo(char sexo): comprueba que el sexo introducido es correcto. Si no es correcto, sera H. No sera visible al exterior.
    - toString(): devuelve toda la información del objeto.
    - generaDNI(): genera un número aleatorio de 8 cifras, genera a partir de este su número su letra correspondiente. Este método será invocado cuando se construya el objeto. Puedes dividir el método para que te sea más fácil. No será visible al exterior.
    - Métodos set de cada parámetro, excepto de DNI.

Ahora, crea una clase ejecutable que haga lo siguiente:

- Pide por teclado el nombre, la edad, sexo, peso y altura.
- Crea 3 objetos de la clase anterior, el primer objeto obtendrá las anteriores variables pedidas por teclado, el segundo objeto obtendrá todos los anteriores menos el peso y la altura y el último por defecto, para este último utiliza los métodos set para darle a los atributos un valor.
- Para cada objeto, deberá comprobar si esta en su peso ideal, tiene sobrepeso o por debajo de su peso ideal con un mensaje.
- Indicar para cada objeto si es mayor de edad.
- Por último, mostrar la información de cada objeto.

20. Problema VIII: Password

Crea una clase llamada Password que siga las siguientes condiciones:

- Que tenga los atributos longitud y contraseña . Por defecto, la longitud será de 8.
- Los constructores serán los siguiente:
  - Un constructor por defecto.
  - Un constructor con la longitud que nosotros le pasemos. Generara una contraseña aleatoria con esa longitud.
- Los métodos que implementa serán:
  - esFuerte(): devuelve un booleano si es fuerte o no, para que sea fuerte debe tener mas de 2 mayúsculas, mas de 1 minúscula y mas de 5 números.
  - generarPassword(): genera la contraseña del objeto con la longitud que tenga.
  - Método get para contraseña y longitud.
  - Método set para longitud.

Ahora, crea una clase clase ejecutable:

- Crea un array de Passwords con el tamaño que tú le indiques por teclado.
- Crea un bucle que cree un objeto para cada posición del array.
- Indica también por teclado la longitud de los Passwords (antes de bucle).
- Crea otro array de booleanos donde se almacene si el password del array de Password es o no fuerte (usa el bucle anterior).
- Al final, muestra la contraseña y si es o no fuerte (usa el bucle anterior). Usa este simple formato:
    - contraseña1 valor_booleano1
    - contraseña2 valor_bololeano2
    - …

21. Visibilidad

Coloca dos botones sobre un formulario y cambia el código en el button_Click como se indica a continuación. Observa las regiones de código donde existen las variables tipo string.

```
public class Form1: Form
{
    string strClassString = "Clase String";
    private void button1_Click(object sender, EventArg e)
    {
        string strMetodoString = "Método String";
        MessageBox.Show(strClassString); // OK
        MessageBox.Show(strMetodoString); // OK
    }
    private void button2_Click(object sender, EventArg e)
    {
        MessageBox.Show(strClassString); // OK
        MessageBox.Show(strMetodoString); // Error
    }
}
```

Localiza las regiones de código donde puedes declarar string como public o private.

22. Declara una segunda ‘strClassString’ en la primera línea de ‘button2_Click’, y observa qué ocurre:

```
string strClassString = "String Nuevo";
```