

EXAMEN SEGUNDO TRIMESTRE

(DOSSIER - 1,5 pt) Encontrar cinco errores de normas de estilo en el fichero loto.cs, indicando número de línea, error encontrado y solución.

```

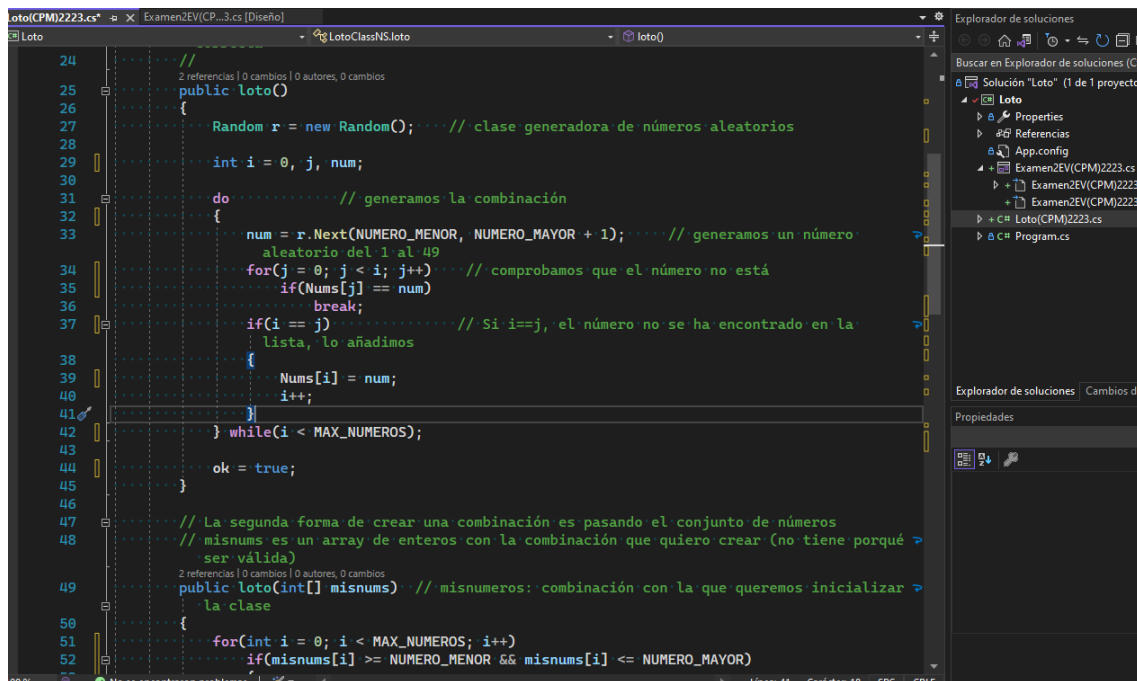
int i=0, j, num;

do ..... // generamos la combinación
{
    num = r.Next(NUMERO_MENOR, NUMERO_MAYOR + 1);
    aleatorio del 1 al 49
    for (j=0; j<i; j++) ..... // comprobamos que el nú
    if (Nums[j]==num)
        break;
    if (i==j) ..... // Si i==j, el número n
    lo añadimos
    {
        Nums[i]=num;
        i++;
    }
} while (i<MAX_NUMEROS);

ok=true;

```

Añadir espacios, con (control+k+d) o añadiendo espacios a mano



Cambiar misnums por misnumeros para que este claro que es

Líneas 49, 52, 56 y 59

```

23.cs* X Examen2EV(CP...3.cs [Diseño]
LotoClassNS.loto loto(int[] misnums)
...// misnums es un array de enteros con la combinación que quiero crear (no tiene porqu
...ser válida)
2 referencias | 0 cambios | 0 autores, 0 cambios
...public loto(int[] misnums) // misnumeros: combinación con la que queremos inicializa
...la clase
{
...for(int i = 0; i < MAX_NUMEROS; i++)
...if(misnums[i] >= NUMERO_MENOR && misnums[i] <= NUMERO_MAYOR)
...{
...int j;
...for(j = 0; j < i; j++)
...if(misnums[i] == Nums[j])
...break;
...if(i == j)
...Nums[i] = misnums[i]; // validamos la combinación
...else
...{
...ok = false;
...return;
...}
...}
}

```

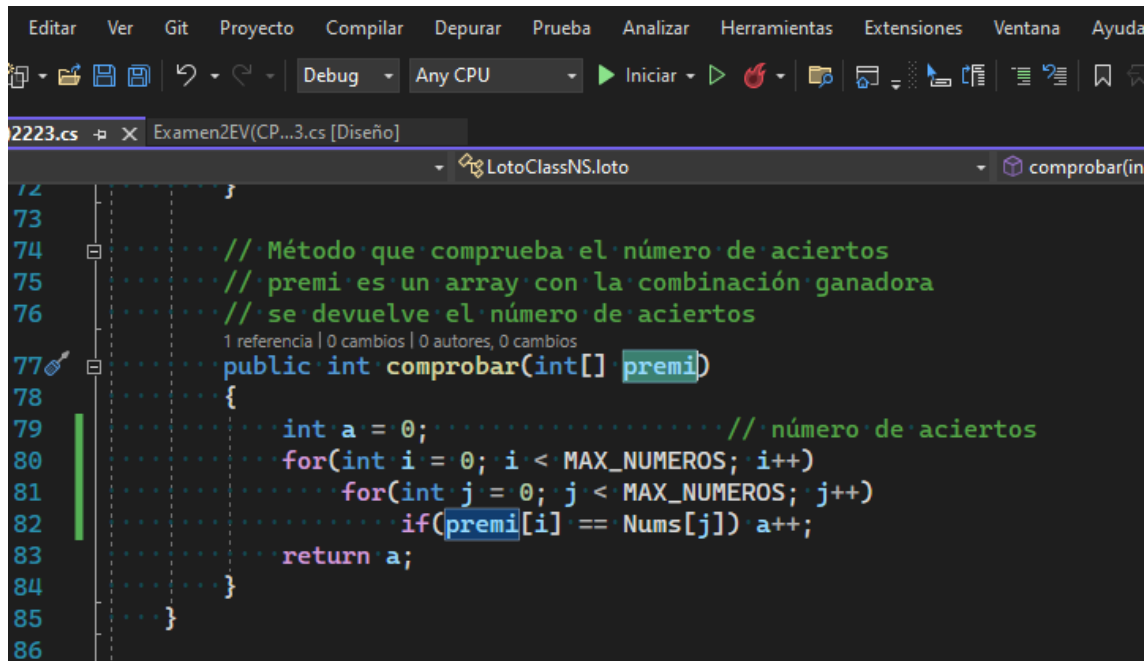
```

...// Constructor con parametro para crear la loteria
...//</summary>
...//<param name="misNumeros"></param>
3 referencias | Cristina Picazo, Hace menos de 5 minutos | 1 autor, 1 cambio
...public loto(int[] misNumeros) // misnumeros: combinación con la que queremos
...inicializar la clase
{
...for(int i = 0; i < MAX_NUMEROS; i++)
...if(misNumeros[i] >= NUMERO_MENOR && misNumeros[i] <= NUMERO_MAYOR)
...{
...int j;
...for(j = 0; j < i; j++)
...if(misNumeros[i] == ListaNumeros[j])
...break;
...if(i == j)
...ListaNumeros[i] = misNumeros[i]; // validamos la combinación
...else
...{
...esValida = false;
...return;
...}
...}
...else

```

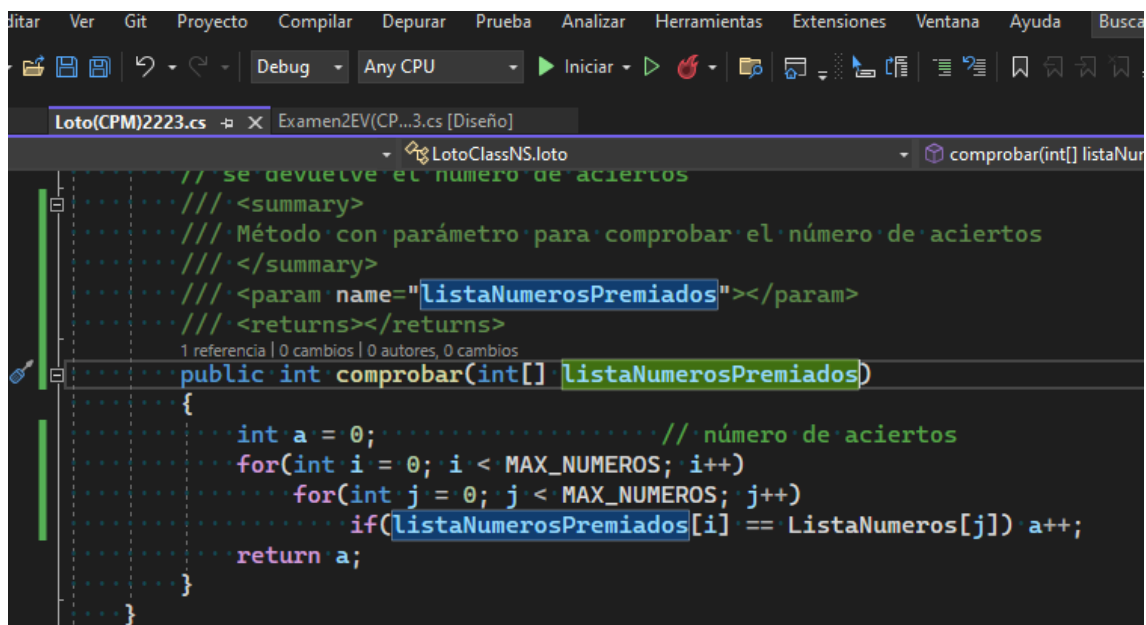
Cambiar premi por listaNumerosPremiados que es más descriptiva

Líneas 77 y 82



This screenshot shows the original implementation of the `comprobar` method in the `LotoClassNS.loto` file. The method signature is `public int comprobar(int[] premi)`. The code includes XML documentation comments in Spanish: `/// Método que comprueba el número de aciertos`, `/// premi es un array con la combinación ganadora`, and `/// se devuelve el número de aciertos`. The logic involves a nested loop where the outer loop iterates over the `premi` array and the inner loop iterates over the `Nums` array, incrementing a counter `a` for each match. The method returns the final count `a`.

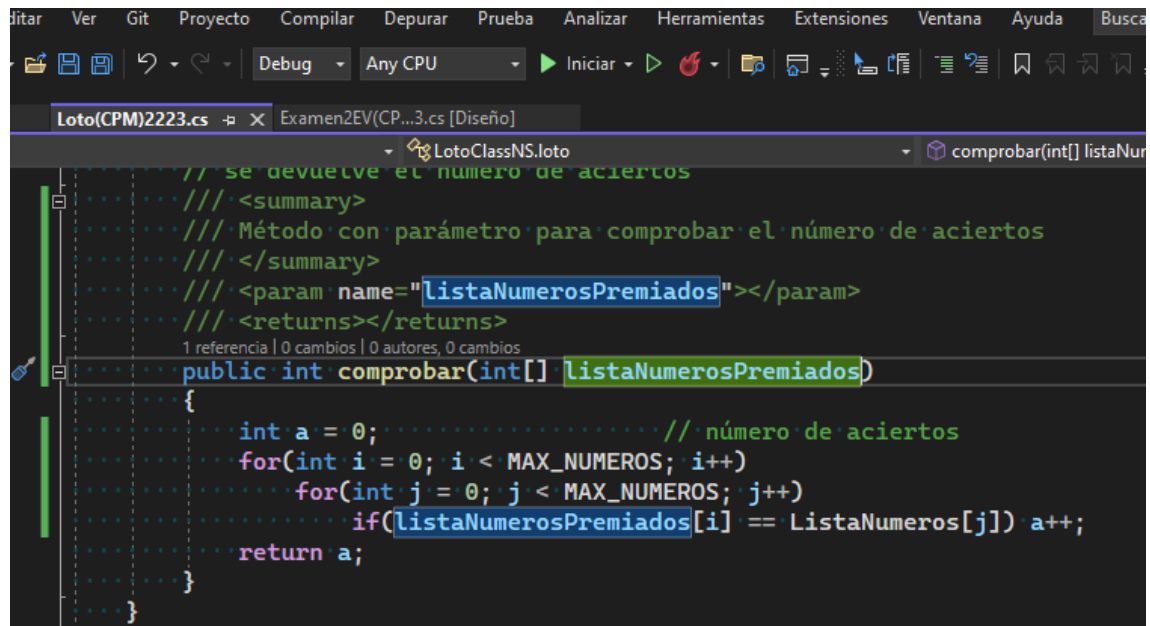
```
72 }
73
74 /// Método que comprueba el número de aciertos
75 /// premi es un array con la combinación ganadora
76 /// se devuelve el número de aciertos
77 public int comprobar(int[] premi)
78 {
79     int a = 0; // número de aciertos
80     for (int i = 0; i < MAX_NUMEROS; i++)
81     {
82         for (int j = 0; j < MAX_NUMEROS; j++)
83             if (premi[i] == Nums[j]) a++;
84     }
85     return a;
86 }
```



This screenshot shows the modified version of the `comprobar` method. The parameter `premi` has been replaced with `listaNumerosPremiados` throughout the code. The XML documentation comments have been updated to reflect this change, with `<param name="listaNumerosPremiados"></param>` and `if (listaNumerosPremiados[i] == ListaNumeros[j])`. The rest of the logic remains the same.

```
/// se devuelve el número de aciertos
/// <summary>
/// Método con parámetro para comprobar el número de aciertos
/// </summary>
/// <param name="listaNumerosPremiados"></param>
/// <returns></returns>
public int comprobar(int[] listaNumerosPremiados)
{
    int a = 0; // número de aciertos
    for (int i = 0; i < MAX_NUMEROS; i++)
    {
        for (int j = 0; j < MAX_NUMEROS; j++)
            if (listaNumerosPremiados[i] == ListaNumeros[j]) a++;
    }
    return a;
}
```

Cambiar a por numeroAciertos



```

// Se devuelve el número de aciertos
///<summary>
/// Método con parámetro para comprobar el número de aciertos
///</summary>
///<param name="listaNumerosPremiados"></param>
///<returns></returns>
1 referencia | 0 cambios | 0 autores, 0 cambios
public int comprobar(int[] listaNumerosPremiados)
{
    int a = 0; // número de aciertos
    for(int i = 0; i < MAX_NUMEROS; i++)
        for(int j = 0; j < MAX_NUMEROS; j++)
            if(listaNumerosPremiados[i] == ListaNumeros[j]) a++;
    return a;
}

```

Cambiar `_nums` por `_listaNumeros` y lo mismo con su propiedad `Nums` por `ListaNumeros` para que se entienda que hace

```

private int[] _listaNumeros = new int[MAX_NUMEROS]; // numeros de la combinación
public bool ok = false; // combinación válida (si es aleatoria, siempre es
    válida, si no, no tiene porqué)

7 referencias | 0 cambios | 0 autores, 0 cambios
public int[] ListaNumeros
{
    get => _listaNumeros;
    set => _listaNumeros = value;
}

```

Cambiar `num` por `numeroAleatorio`

Líneas 29,33 y 35

```

int i = 0, j, num;

do // generamos la combinación
{
    num = r.Next(NUMERO_MENOR, NUMERO_MAYOR + 1); // generamos un número
        aleatorio del 1 al 49
    for(j = 0; j < i; j++) // comprobamos que el número no está
        if(ListaNumeros[j] == num)
            break;
    if(i == j) // Si i==j, el número no se ha encontrado en la
        lista, lo añadimos
    {
        ListaNumeros[i] = num;
        i++;
    }
} while(i < MAX_NUMEROS);

ok = true;

```

```

Random r = new Random(); // clase generadora de números aleatorios

int i = 0, j, numeroAleatorio;

do // generamos la combinación
{
    numeroAleatorio = r.Next(NUMERO_MENOR, NUMERO_MAYOR + 1); // generamos un
        número aleatorio del 1 al 49
    for(j = 0; j < i; j++) // comprobamos que el número no está
        if(ListaNumeros[j] == numeroAleatorio)
            break;
    if(i == j) // Si i==j, el número no se ha encontrado en la
        lista, lo añadimos
    {
        ListaNumeros[i] = numeroAleatorio;
        i++;
    }
} while(i < MAX_NUMEROS);

ok = true;

```

Cambiar ok por esValida ya que debe es más explicativa así

```

... private int[] _listaNumeros = new int[MAX_NUMEROS]; ... // numeros de la combinación
... public bool ok = false; ... // combinación válida (si es aleatoria, siempre es
...     válida, si no, no tiene porqué)

7 referencias | 0 cambios | 0 autores, 0 cambios
... public int[] ListaNumeros
... {
...     get => _listaNumeros;
...     set => _listaNumeros = value;

```

```

public class Loto
{
    ... // definición de constantes
    ... public const int MAX_NUMEROS = 6;
    ... public const int NUMERO_MENOR = 1;
    ... public const int NUMERO_MAYOR = 49;

    ... private int[] _listaNumeros = new int[MAX_NUMEROS]; ... // numeros de la combinación
    ... public bool esValida = false; ... // combinación válida (si es aleatoria, siempre es
    ...     válida, si no, no tiene porqué)

```

Ver Git Proyecto Compilar Depurar Prueba Analizar Herramientas Extensiones Ventana Ayuda Buscar (Ctrl)

Debug Any CPU Iniciar

Examen2EV(CP...3.cs [Diseño]

LotoClassNS.loto esValida

```

...     esValida = true;
... }

... // La segunda forma de crear una combinación es pasando el conjunto de n
... // misnums es un array de enteros con la combinación que quiero crear (n
... // ser válida)
2 referencias | 0 cambios | 0 autores, 0 cambios
... public Loto(int[] misnumeros) // misnumeros: combinación con la que que
...     inicializar la clase
... {
...     for(int i = 0; i < MAX_NUMEROS; i++)
...     {
...         if(misnumeros[i] >= NUMERO_MENOR && misnumeros[i] <= NUMERO_MAYOR
...         {
...             int j;
...             for(j = 0; j < i; j++)
...             {
...                 if(misnumeros[i] == ListaNumeros[j])
...                     break;
...                 if(i == j)
...                     ListaNumeros[i] = misnumeros[i]; // validamos la combina
...             }
...             else
...             {
...                 esValida = false;
...                 return;
...             }
...         }
...     }
...     else
...     {
...         esValida = false; ... // La combinación no es válida, termin
...         return;
...     }
...     esValida = true;

```

(COMMIT - 1,5 pt) Documentar el fichero loto.cs. Sólo se debe documentar los constructores y los métodos públicos.

```

... //La segunda forma de crear una combinación es pasando el conjunto de números
... //misnums es un array de enteros con la combinación que quiero crear (no tiene por
... //ser válida)
... ///<summary>
... ///Constructor con parametro para crear la lotería
... ///</summary>
... ///<param name="misnumeros"></param>
... 2 referencias | 0 cambios | 0 autores, 0 cambios
... public loto(int[] misnumeros) // misnumeros: combinación con la que queremos
...     inicializar la clase
... {
...     for(int i = 0; i < MAX_NUMEROS; i++)
...         if(misnumeros[i] >= NUMERO_MENOR && misnumeros[i] <= NUMERO_MAYOR)
...         {
...             int j;
...             for(i = 0; i < i; i++)

```

```

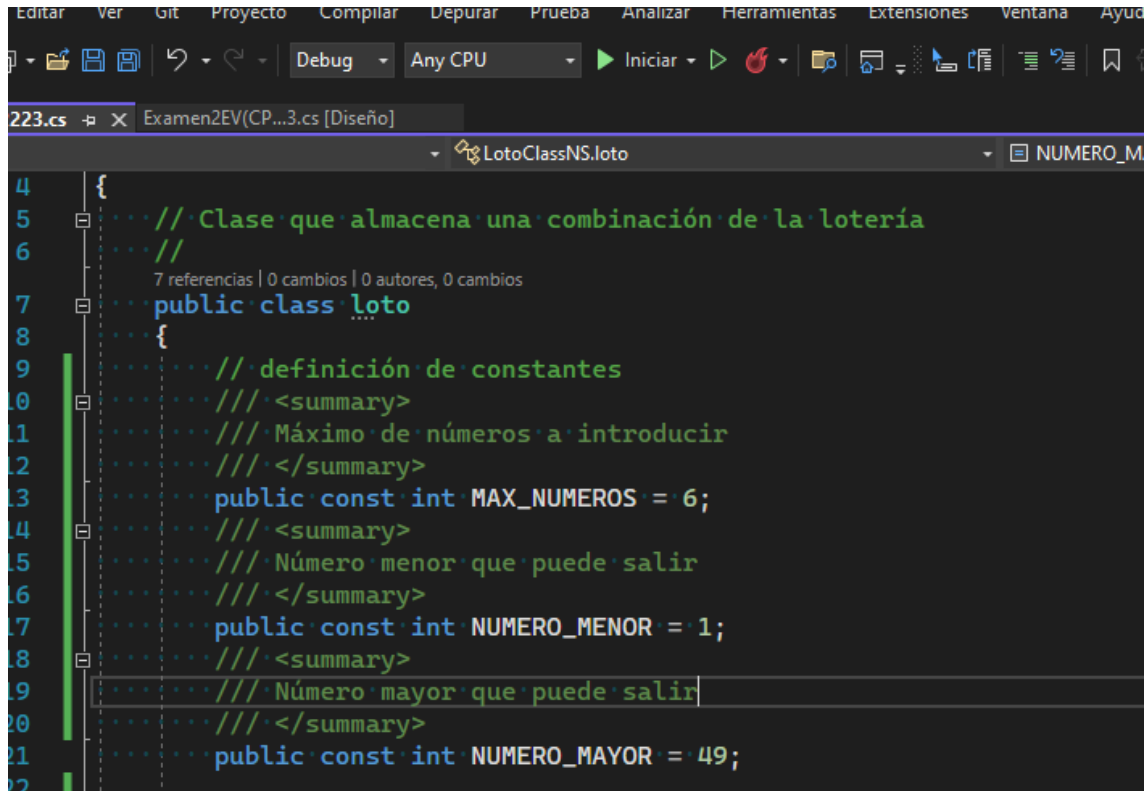
... //En el caso de que el constructor sea vacío, se genera una combinación aleatoria
... //correcta
... //
... ///<summary>
... ///Constructor vacío para crear la lotería
... ///</summary>
... 2 referencias | 0 cambios | 0 autores, 0 cambios
... public loto()
... {
...     Random r = new Random(); // clase generadora de números aleatorios
...
...     int i = 0, j, numeroAleatorio;
...
...     do // generamos la combinación
...     {

```

```

... public bool esvalida = false; // combinación válida (si es ale
...     válida, si no, no tiene por qué)
... ///<summary>
... ///Método público de la lista de números | privada
... ///</summary>
... 7 referencias | 0 cambios | 0 autores, 0 cambios
... public int[] ListaNumeros
... {
...     get => _listaNumeros;
...     set => _listaNumeros = value;
... }
...
... //En el caso de que el constructor sea vacío, se genera una combin
... //correcta
... //
... ///<summary>

```



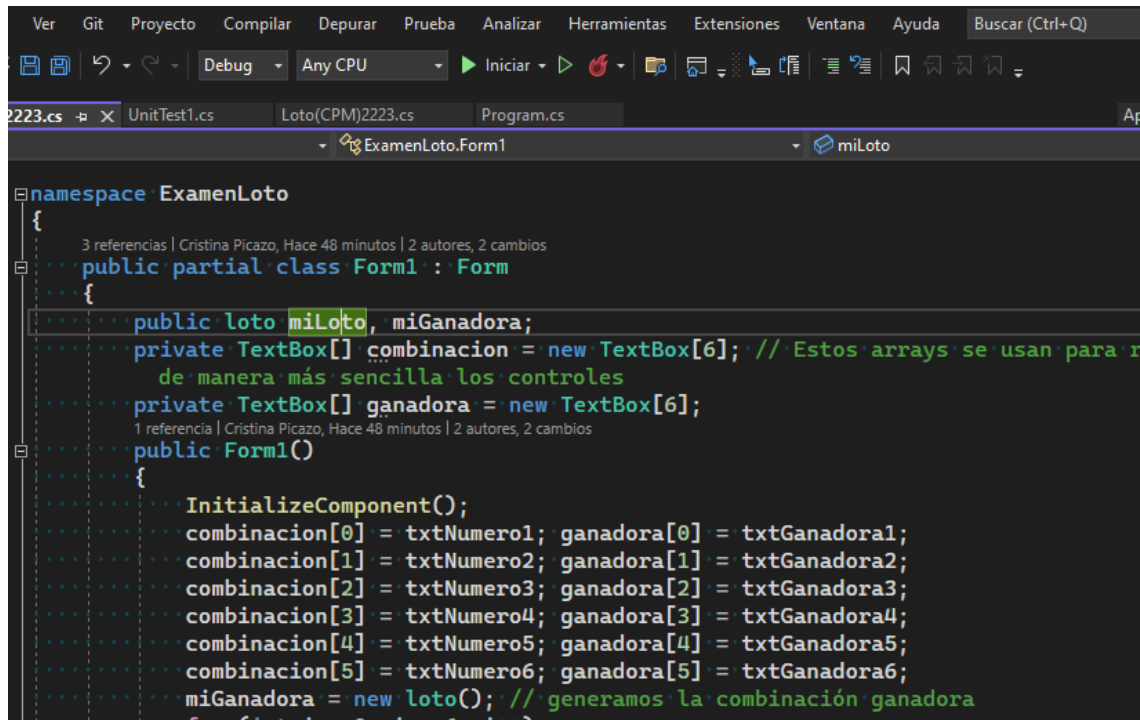
The image shows a screenshot of the Visual Studio Code editor. The top menu bar includes 'Editar', 'Ver', 'Git', 'Proyecto', 'Compilar', 'Depurar', 'Prueba', 'Analizar', 'Herramientas', 'Extensiones', 'Ventana', and 'Ayuda'. Below the menu bar is a toolbar with icons for file operations, debugging, and running. The active file is 'Examen2EV(CP...3.cs [Diseño]'. The editor shows a C# class named 'loto' with the following code:

```
4 {  
5     /// <summary>  
6     /// Clase que almacena una combinación de la lotería  
7     /// </summary>  
8     public class loto  
9     {  
10        /// <summary>  
11        /// definición de constantes  
12        /// </summary>  
13        public const int MAX_NUMEROS = 6;  
14        /// <summary>  
15        /// Número menor que puede salir  
16        /// </summary>  
17        public const int NUMERO_MENOR = 1;  
18        /// <summary>  
19        /// Número mayor que puede salir  
20        /// </summary>  
21        public const int NUMERO_MAYOR = 49;  
22    }
```


(DOSSIER+COMMIT - 1,5 pt) Si existen, detectar y aplicar al menos tres patrones de refactorización (tanto en el fichero Loto.cs como en el fichero Form1.cs), indicando el patrón que se aplica y, si es posible aplicarlo con Visual Studio, la opción que se usa.

Las capturas están arriba, perdón

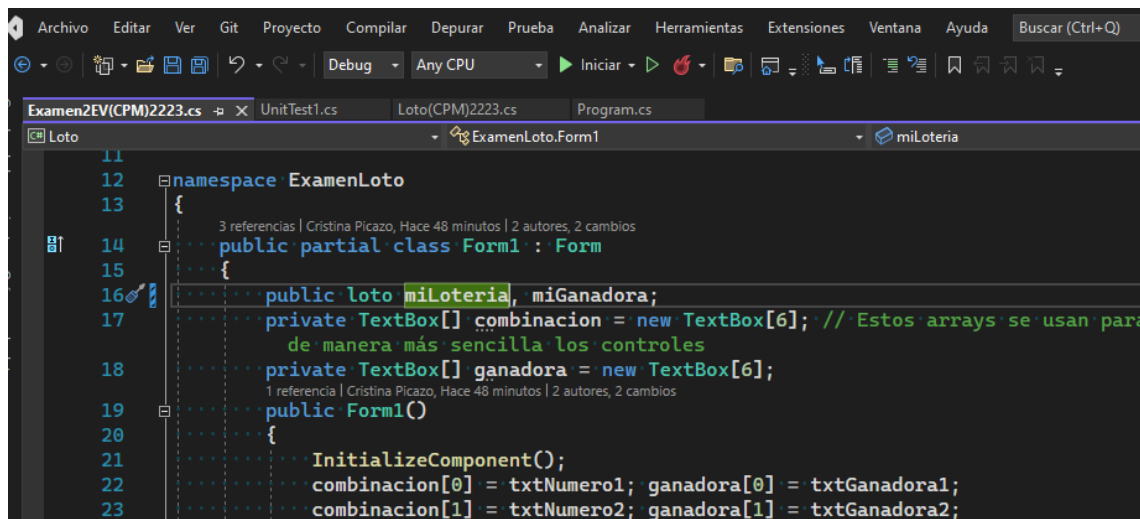
Cambiar miLoto por miLoteria



```

namespace ExamenLoto
{
    3 referencias | Cristina Picazo, Hace 48 minutos | 2 autores, 2 cambios
    public partial class Form1 : Form
    {
        public Loto miLoto, miGanadora;
        private TextBox[] combinacion = new TextBox[6]; // Estos arrays se usan para
        de manera más sencilla los controles
        private TextBox[] ganadora = new TextBox[6];
        1 referencia | Cristina Picazo, Hace 48 minutos | 2 autores, 2 cambios
        public Form1()
        {
            InitializeComponent();
            combinacion[0] = txtNumero1; ganadora[0] = txtGanadora1;
            combinacion[1] = txtNumero2; ganadora[1] = txtGanadora2;
            combinacion[2] = txtNumero3; ganadora[2] = txtGanadora3;
            combinacion[3] = txtNumero4; ganadora[3] = txtGanadora4;
            combinacion[4] = txtNumero5; ganadora[4] = txtGanadora5;
            combinacion[5] = txtNumero6; ganadora[5] = txtGanadora6;
            miGanadora = new Loto(); // generamos la combinación ganadora
        }
    }
}

```

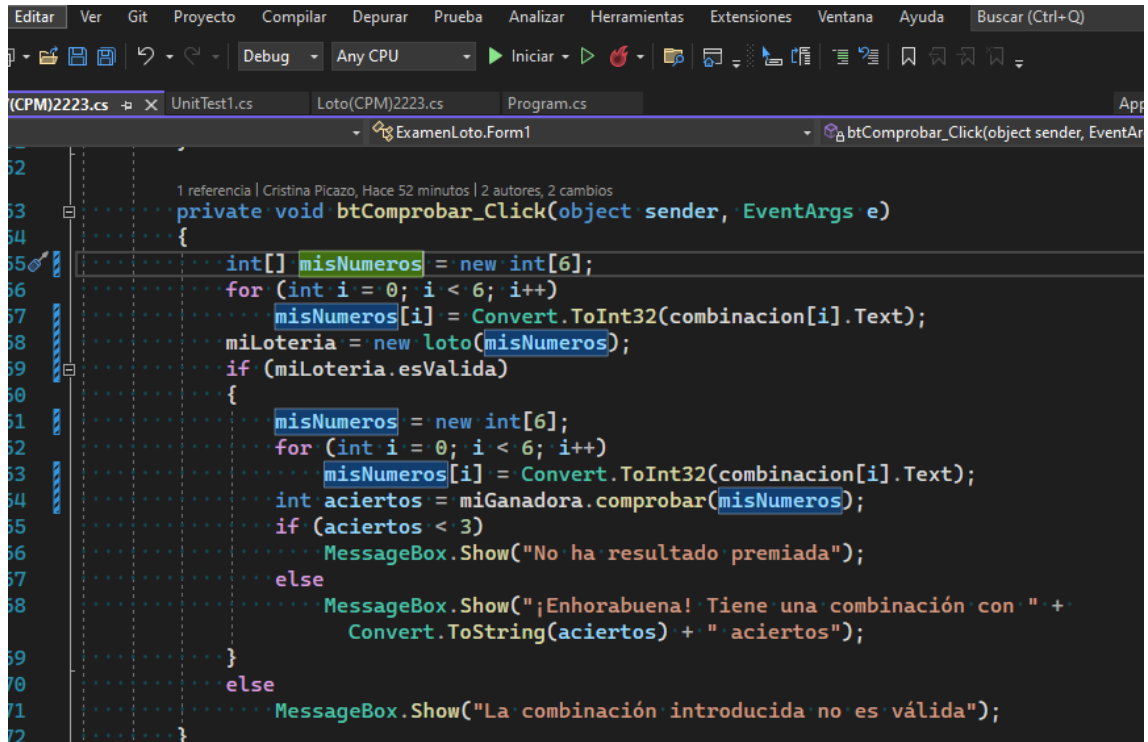


```

11
12 namespace ExamenLoto
13 {
    3 referencias | Cristina Picazo, Hace 48 minutos | 2 autores, 2 cambios
    14 public partial class Form1 : Form
    15 {
    16     public Loto miLoteria, miGanadora;
    17     private TextBox[] combinacion = new TextBox[6]; // Estos arrays se usan para
    de manera más sencilla los controles
    18     private TextBox[] ganadora = new TextBox[6];
    19     1 referencia | Cristina Picazo, Hace 48 minutos | 2 autores, 2 cambios
    20     public Form1()
    21     {
    22         InitializeComponent();
    23         combinacion[0] = txtNumero1; ganadora[0] = txtGanadora1;
        combinacion[1] = txtNumero2; ganadora[1] = txtGanadora2;
    }
}

```

Cambiar por misNumeros



The image shows a screenshot of the Visual Studio code editor. The top menu bar includes 'Editar', 'Ver', 'Git', 'Proyecto', 'Compilar', 'Depurar', 'Prueba', 'Analizar', 'Herramientas', 'Extensiones', 'Ventana', 'Ayuda', and 'Buscar (Ctrl+Q)'. The toolbar below the menu bar contains various icons for file operations, debugging, and development. The file explorer on the left shows a project structure with files like 'UnitTest1.cs', 'Loto(CPM)2223.cs', and 'Program.cs'. The active file is 'ExamenLoto.Form1'. The code editor displays a C# method named 'btComprobar_Click' which takes 'object sender' and 'EventArgs e' as parameters. The code logic involves creating an array 'misNumeros' of size 6, populating it with values from 'combinacion[i].Text' using 'Convert.ToInt32', creating a 'miLoteria' object, and then checking if it is valid. If valid, it checks the number of correct numbers ('aciertos') against a winning threshold (3). Depending on the result, it shows a message box indicating either a win or an invalid combination. The code is color-coded with syntax highlighting.

```
1 referencia | Cristina Picazo, Hace 52 minutos | 2 autores, 2 cambios
63 .....private void btComprobar_Click(object sender, EventArgs e)
64 .....{
65 .....    int[] misNumeros = new int[6];
66 .....    for (int i = 0; i < 6; i++)
67 .....        misNumeros[i] = Convert.ToInt32(combinacion[i].Text);
68 .....    miLoteria = new Loto(misNumeros);
69 .....    if (miLoteria.esValida)
70 .....    {
71 .....        misNumeros = new int[6];
72 .....        for (int i = 0; i < 6; i++)
73 .....            misNumeros[i] = Convert.ToInt32(combinacion[i].Text);
74 .....        int aciertos = miGanadora.comprobar(misNumeros);
75 .....        if (aciertos < 3)
76 .....            MessageBox.Show("No ha resultado premiada");
77 .....        else
78 .....            MessageBox.Show("¡Enhorabuena! Tiene una combinación con " +
79 .....                Convert.ToString(aciertos) + " aciertos");
80 .....    }
81 .....    else
82 .....        MessageBox.Show("La combinación introducida no es válida");
83 .....}
```

(DOSSIER - 1,5 pt) Realizar el diseño de pruebas (caja negra) para el constructor con parámetro de la clase loto.

- Comprobar que la lista de misnumeros no es > 6
- Número en la lista >= 1
- Número en la lista <= 49

misnumeros	6	true
misnumeros	7	false
lista	0	false
lista	1	true
lista	49	true
lista	50	false

(COMMIT - 1,5 pt) Por último, crear los métodos de prueba para aquellos casos en que se generan errores en el punto anterior.