

Examen segunda evaluación

Adrián Ibáñez Cañizares

1. Encontrar cinco errores de normas de estilo en el fichero “lotoAICDAW2223.cs”, indicando el número de línea, error encontrado y solución.

- **Renombrado de variables:**

Tanto en el getter/setter de los números, como en la declaración de las variables(dentro y fuera de los getter/setter) tienen un nombre poco autodescriptivo, además es incorrecto iniciar el nombre con un “_”. Para corregirlo, seleccionamos la sección de código que se quiere renombrar y hacemos clic en “Editar> Refactorizar> Cambiar nombre”.



(Antes del cambio)

```
private int[] _nums = 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)

public int[] Nums {
    get => _nums;
    set => _nums = value;
}
```

(Después del cambio)

```
14 private int[] numeros = new int[MAX_NUMEROS]; // numeros de la combinación
15 public bool ok = false; // combinación válida (si es aleatoria, siempre es válida, si no, no tiene porqué)
16
17 public int[] Numeros {
18     get => numeros;
19     set => numeros = value;
20 }
```

Los cambios “numeros” se han realizado en la línea 14, 18 y 19.

Los cambios “Numeros” se han realizado en la línea 17, 34, 38, 54, 57, 79,

En la siguiente imagen se puede apreciar un error del mismo tipo, tanto el parámetro “premi” como el valor int “a” tienen nombres poco descriptivos, hacemos lo mismo que en el proceso anterior.

(Código antes del renombrado)

```
// Método que comprueba el número de aciertos
// premi es un array con la combinación ganadora
// se devuelve el número de aciertos
public int comprobar(int[] premi)
{
    int a=0; // número de aciertos
    for (int i=0; i<MAX_NUMEROS; i++)
        for (int j=0; j<MAX_NUMEROS; j++)
            if (premi[i]==Numeros[j]) a++;
    return a;
}
```

(Código tras el renombrado, hay más errores en estas imágenes, como los comentarios, pero eso se tratará más adelante)

```
74 public int comprobar(int[] premio)
75 {
76     int boleto=0; // número de aciertos
77     for (int i=0; i<MAX_NUMEROS; i++)
78         for (int j=0; j<MAX_NUMEROS; j++)
79             if (premio[i]==Numeros[j]) boleto++;
80     return boleto;
81 }
```

Los cambios “premio” se han realizado en la línea 74 y 79

Los cambios “boleto” se han realizado en la línea 76, 79 y 80.

“a” hacia referencia a un número, dentro de la función comprobar, por eso se renombra como boleto, para que sea más autodescriptivo.

Pasa lo mismo con “misnums”, lo renombramos como “misNumeros”.

(Código antes del renombrado)

```
public loto(int[] misnums) // misnumeros: combinación con la que queremos inicializar 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]==Numeros[j])
                    break;
            if (i==j)
                Numeros[i]=misnums[i]; // validamos la combinación
            else {
                ok=false;
                return;
            }
        }
        else
        {
            ok=false; // La combinación no es válida, terminamos
            return;
        }
    ok=true;
}
```

(Código tras el renombrado)

```
48 public loto(int[] misNumeros) // misNumeros: combinación con la que queremos inicializar la clase
49 {
50     for (int i=0; i<MAX_NUMEROS; i++)
51     if (misNumeros[i]>=NUMERO_MENOR && misNumeros[i]<=NUMERO_MAYOR) {
52         int j;
53         for (j=0; j<i; j++)
54             if (misNumeros[i]==Numeros[j])
55                 break;
56         if (i==j)
57             Numeros[i]=misNumeros[i]; // validamos la combinación
58         else {
59             ok=false;
60             return;
61         }
62     }
63     else
64     {
65         ok=false; // La combinación no es válida, terminamos
66         return;
67     }
68     ok=true;
69 }
```

Los cambios “misNumeros” se han realizado en la línea 48, 51(2), 54 y 57.

En la siguiente imagen se van a renombrar las siguientes variables: r, num, ya que tienen un nombre poco descriptivo.

(Código antes del renombrado)

```
public loto()
{
    Random r = new Random(); // clase generadora de números aleatorios
    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 (Numeros[j]==num)
                break;
        if (i==j) // Si i==j, el número no se ha encontrado en la lista, lo añadimos
        {
            Numeros[i]=num;
            i++;
        }
    } while (i<MAX_NUMEROS);

    ok=true;
}
```

(Código tras el renombrado)

```
24 public loto()  
25 {  
26     Random aleatorio = new Random(); // clase generadora de números aleatorios  
27  
28     int i=0, j, numero;  
29  
30     // generamos la combinación  
31     do  
32     {  
33         numero = aleatorio.Next(NUMERO_MENOR, NUMERO_MAYOR + 1); // generamos un número aleatorio  
34         for (j = 0; j<i; j++) // comprobamos que el número no está  
35             if (Numeros[j]==numero)  
36                 break;  
37         if (i==j) // Si i==j, el número no se ha encontrado en la lista, lo añadimos  
38         {  
39             Numeros[i]=numero;  
40             i++;  
41         }  
42     } while (i<MAX_NUMEROS);  
43     ok=true;  
44 }
```

Los cambios “numero” se han realizado en la línea 28, 32, 34 y 38.

Los cambios “aleatorio” se han realizado en la línea 26 y 32

El mismo tipo de renombrado, pero en la clase form1

(Antes)

```
59 {  
60     int[] nums = new int[6];  
61     for (int i = 0; i < 6; i++)  
62         nums[i] = Convert.ToInt32(combinacion[i].Text);  
63     miLoto = new loto(nums);  
64     if (miLoto.ok)  
65     {  
66         nums = new int[6];  
67         for (int i = 0; i < 6; i++)  
68             nums[i] = Convert.ToInt32(combinacion[i].Text);  
69         int aciertos = miGanadora.comprobar(nums);  
70         if (aciertos < 3)  
71             MessageBox.Show("No ha resultado premiada");  
72         else  
73             MessageBox.Show(";Enhorabuena! Tiene una combinación con " + Convert.ToString(aciertos));  
74     }  
75     else  
76         MessageBox.Show("La combinación introducida no es válida");  
77 }
```

(Después)

```
59 {  
60     int[] numeros = new int[6];  
61     for (int i = 0; i < 6; i++)  
62         numeros[i] = Convert.ToInt32(combinacion[i].Text);  
63     miLoto = new loto(numeros);  
64     if (miLoto.ok)  
65     {  
66         numeros = new int[6];  
67         for (int i = 0; i < 6; i++)  
68             numeros[i] = Convert.ToInt32(combinacion[i].Text);  
69         int aciertos = miGanadora.comprobar(numeros);  
70         if (aciertos < 3)  
71             MessageBox.Show("No ha resultado premiada");  
72         else  
73             MessageBox.Show(";Enhorabuena! Tiene una combinación con " + Convert.ToString(aciertos) + " acie");  
74     }  
75     else  
76         MessageBox.Show("La combinación introducida no es válida");  
77 }
```

Se han realizado los cambios en las líneas : 60, 62, 63, 66, 68 y 69.

- **Eliminación/ cambios en los comentarios.**

En la línea 6 tenemos dos barras “//” pero nada comentado en ellas, ya que sobran vamos a eliminarlas y el comentario de la línea superior lo pasamos a la línea donde estaban las barras “//”.

(Antes)

```
5 // Clase que almacena una combinación de la lotería
6 //
7 public class Loto
```

(Después) Línea 5/6

```
6 // Clase que almacena una combinación de la lotería
7 public class Loto
```

Adjunto el mismo tipo de error en distintas líneas del código.

(Antes)

```
22 // En el caso de que el constructor sea vacío, se genera una combinación aleatoria correcta
23 //
24 public Loto()
```

(Después) Líneas 22/23

```
23 // En el caso de que el constructor sea vacío, se genera una combinación aleatoria correcta
24 public Loto()
```

En la siguiente imagen se puede apreciar la existencia de comentarios declarados de forma contigua a las líneas de código, puesto que esto es un error de estilo, reubicaremos los comentarios, cada comentario encima del código en cuestión.

(Antes)

```
13
14 private int[] numeros = new int[MAX_NUMEROS]; // numeros de la combinación
15 public bool ok = false; // combinación válida (si es aleatoria, siempre es válida, si no, no tiene porqué)
16
```

(Después) Líneas 14/17

```
13
14 // numeros de la combinación
15 private int[] numeros = new int[MAX_NUMEROS];
16 // combinación válida (si es aleatoria, siempre es válida, si no, no tiene porqué)
17 public bool ok = false;
```

A continuación se muestra el mismo error en distintas partes del código.

(Antes)

```

26 public loto()
27 {
28     Random aleatorio = new Random(); // clase generadora de números aleatorios
29
30     int i=0, j, numero;
31
32     do // generamos la combinación
33     {
34         numero = aleatorio.Next(NUMERO_MENOR, NUMERO_MAYOR + 1); // generamos un número aleatorio del 1 al 49
35         for (j = 0; j<i; j++) // comprobamos que el número no está
36             if (Numeros[j]==numero)
37                 break;
38         if (i==j) // Si i==j, el número no se ha encontrado en la lista, lo añadimos
39         {
40             Numeros[i]=numero;
41             i++;
42         }
43     } while (i<MAX_NUMEROS);
44
45     ok=true;
46 }
47

```

(Después)

```

28 // clase generadora de números aleatorios
29 Random aleatorio = new Random();
30
31 int i=0, j, numero;
32
33 // generamos la combinación
34 do
35 {
36     // generamos un número aleatorio del 1 al 49
37     numero = aleatorio.Next(NUMERO_MENOR, NUMERO_MAYOR + 1);
38     // comprobamos que el número no está
39     for (j = 0; j<i; j++)
40         if (Numeros[j]==numero)
41             break;
42     // Si i==j, el número no se ha encontrado en la lista, lo añadimos
43     if (i==j)
44     {
45         Numeros[i]=numero;
46         i++;
47     }
48 } while (i<MAX_NUMEROS);
49
50 ok=true;
51

```

Cambios en las líneas 28, 33, 38 y 42.

(Antes)

```

54 // misNumeros es un array de enteros con la combinación que queremos crear (no tiene porque ser válida)
55 public loto(int[] misNumeros) // misNumeros: combinación con la que queremos inicializar la clase
56 {
57     for (int i=0; i<MAX_NUMEROS; i++)
58     {
59         if (misNumeros[i]>=NUMERO_MENOR && misNumeros[i]<=NUMERO_MAYOR) {
60             int j;
61             for (j=0; j<i; j++)
62                 if (misNumeros[i]==Numeros[j])
63                     break;
64             if (i==j)
65                 Numeros[i]=misNumeros[i]; // validamos la combinación
66             else {
67                 ok=false;
68                 return;
69             }
70         }
71         else
72         {
73             ok=false; // La combinación no es válida, terminamos
74             return;
75         }
76     }
77

```

(Después)

```
55 // misnumeros: combinación con la que queremos inicializar la clase
56 public loto(int[] misNumeros)
57 {
58     for (int i=0; i<MAX_NUMEROS; i++)
59         if (misNumeros[i]>=NUMERO_MENOR && misNumeros[i]<=NUMERO_MAYOR) {
60             int j;
61             for (j=0; j<i; j++)
62                 if (misNumeros[i]==Numeros[j])
63                     break;
64             // validamos la combinación
65             if (i==j)
66                 Numeros[i]=misNumeros[i];
67             else {
68                 ok=false;
69                 return;
70             }
71         }
72     else
73     {
74         // La combinación no es válida, terminamos
75         ok = false;
76         return;
77     }
78     ok=true;
79 }
```

Se han realizado cambios en la línea 55, 63 y 73.

(Antes)

```
85
86 int boleto=0; // número de aciertos
87 for (int i=0; i<MAX_NUMEROS; i++)
88     for (int i=0; i<MAX_NUMEROS; i++)
```

(Después)

```
86 // número de aciertos
87 int boleto =0;
88 for (int i=0; i<MAX_NUMEROS; i++)
89     for (int j=0; j<MAX_NUMEROS; j++)
```

Cambios en la línea 86

(Antes) En la clase Form1

```
15 {
16     public loto miLoto, miGanadora;
17     private TextBox[] combinacion = new TextBox[6]; // Estos arrays se usan para recorrer de manera más sencilla lo
18     private TextBox[] ganadora = new TextBox[6];
19     public Form1()
20     {
21         InitializeComponent();
22         combinacion[0] = txtNumero1; ganadora[0] = txtGanadora1;
23         combinacion[1] = txtNumero2; ganadora[1] = txtGanadora2;
24         combinacion[2] = txtNumero3; ganadora[2] = txtGanadora3;
25         combinacion[3] = txtNumero4; ganadora[3] = txtGanadora4;
26         combinacion[4] = txtNumero5; ganadora[4] = txtGanadora5;
27         combinacion[5] = txtNumero6; ganadora[5] = txtGanadora6;
28         miGanadora = new loto(); // generamos la combinación ganadora
29         for (int i = 0; i < 6; i++)
30             ganadora[i].Text = Convert.ToString(miGanadora.Numeros[i]);
31     }
32 }
33
34 private void btGenerar_Click(object sender, EventArgs e)
35 {
36     miLoto = new loto(); // usamos constructor vacío, se genera combinación aleatoria
37     for (int i=0; i<6; i++)
38         combinacion[i].Text = Convert.ToString(miLoto.Numeros[i]);
```

a
a

(Después)

```
17 // Estos arrays se usan para recorrer de manera más sencilla los controles
18 private TextBox[] combinacion = new TextBox[6];
19 private TextBox[] ganadora = new TextBox[6];
20 public Form1()
21 {
22     InitializeComponent();
23     combinacion[0] = txtNumero1; ganadora[0] = txtGanadora1;
24     combinacion[1] = txtNumero2; ganadora[1] = txtGanadora2;
25     combinacion[2] = txtNumero3; ganadora[2] = txtGanadora3;
26     combinacion[3] = txtNumero4; ganadora[3] = txtGanadora4;
27     combinacion[4] = txtNumero5; ganadora[4] = txtGanadora5;
28     combinacion[5] = txtNumero6; ganadora[5] = txtGanadora6;
29     // generamos la combinación ganadora
30     miGanadora = new Loto();
31     for (int i = 0; i < 6; i++)
32         ganadora[i].Text = Convert.ToString(miGanadora.Numeros[i]);
33 }
34
35
36 private void btGenerar_Click(object sender, EventArgs e)
37 {
38     // usamos constructor vacío, se genera combinación aleatoria
39     miLoto = new Loto();
40     for (int i=0; i<6; i++)
41         combinacion[i].Text = Convert.ToString(miLoto.Numeros[i]);
42 }
43
```

2. Documentar el fichero Loto.cs. Sólo se debe documentar los constructores y los métodos públicos.

Documentación del primer constructor, este constructor funciona sin parámetros.

Los comentarios marcados en rojo los eliminamos, ya que no son necesarios tras la documentación del código.

```
24
25 // En el caso de que el constructor sea vacío, se genera una combinación aleatoria correcta
26 /// <summary>
27 /// Constructor vacío, que genera una combinación aleatoria correcta.
28 /// </summary>
29 public Loto()
30 {
31     // clase generadora de números aleatorios
32     Random aleatorio = new Random();
33
34     int i=0, j, numero;
35
36     // generamos la combinación
37     do
38     {
39         // generamos un número aleatorio del 1 al 49
40         numero = aleatorio.Next(NUMERO_MENOR, NUMERO_MAYOR + 1);
41         // comprobamos que el número no está
42         for (j = 0; j<i; j++)
43             if (Numeros[j]==numero)
```


Así se ve el segundo constructor tras documentarlo, los comentarios marcados en rojo los eliminamos, ya que tras la documentación no son necesarios.

```
52
53 // La segunda forma de crear una combinación es pasando el conjunto de números
54 // misnums es un array de enteros con la combinación que quiero crear (no tiene porqué ser válida)
55 // misnumeros: combinación con la que queremos inicializar la clase
56 /// <summary>
57 /// Constructor de un objeto loto con parámetros, este constructor recibe un vector con seis números.
58 /// Dicho vector puede ser o no en un inicio, pero se comprueba si es válido dentro del constructor.
59 /// </summary>
60 /// <param name="misNumeros"> Es el vector en cuestión con seis números.</param>
61 public loto(int[] misNumeros)
62 {
63     for (int i=0; i<MAX_NUMEROS; i++)
64     if (misNumeros[i]>=NUMERO_MENOR && misNumeros[i]<=NUMERO_MAYOR) {
65         int j;
66         for (j=0; j<i; j++)
67             if (misNumeros[i]==Numeros[j])
68                 break;
69         // validamos la combinación
70         if (i==j)
71             Numeros[i]=misNumeros[i];
72         else {
73             ok=false;
74             return;
75         }
76     }
77     else
78     {
79         // La combinación no es válida, terminamos
80         ok = false;
```

Por último, documentamos los métodos de esta clase, que en este caso tenemos el método “comprobar”.

```
86
87 // Método que comprueba el número de aciertos
88 // premi es un array con la combinación ganadora
89 // se devuelve el número de aciertos
90 /// <summary>
91 /// Método que comprueba el número de aciertos del vector pasado por parámetro
92 /// </summary>
93 /// <param name="premio"> Vector que recibe con parámetro, el cual contiene seis números</param>
94 /// <returns> Devuelve un valor numérico entero en función de los aciertos</returns>
95 public int comprobar(int[] premio)
96 {
97     // número de aciertos
98     int boleto =0;
99     for (int i=0; i<MAX_NUMEROS; i++)
100         for (int j=0; j<MAX_NUMEROS; j++)
101             if (premio[i]==Numeros[j]) boleto++;
102     return boleto;
103 }
104 }
```

3. 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.

El primer error que vemos, en la clase loto, es la declaración de variables de forma pública, esto es un error debido a que al acceder directamente a los datos puede dar lugar a error.

Para solucionarlo, seleccionamos el código a cambiar, le damos a “Editar> Refactorizar> Encapsular campo”

(Código antes del cambio)

```
8
9      // definición de constantes
10     public const int MAX_NUMEROS = 6;
11     public const int NUMERO_MENOR = 1;
12     public const int NUMERO_MAYOR = 49;
13
14     // numeros de la combinación
15     private int[] numeros = new int[MAX_NUMEROS];
16     // combinación válida (si es aleatoria, siempre
17     public bool ok = false;
18
```

(Código tras el cambio)

```
9      // definición de constantes
10     private const int mAX_NUMEROS = 6;
11     private const int nUMERO_MENOR = 1;
12     private const int nUMERO_MAYOR = 49;
13
14     // numeros de la combinación
15     private int[] numeros = new int[MAX_NUMEROS];
16     // combinación válida (si es aleatoria, siempre es válida, si no, no tiene porqué)
17     private bool ok = false;
18
19     public int[] Numeros {
20         get => numeros;
21         set => numeros = value;
22     }
23
24     public static int MAX_NUMEROS => mAX_NUMEROS;
25
26     public static int NUMERO_MENOR => nUMERO_MENOR;
27
28     public static int NUMERO_MAYOR => nUMERO_MAYOR;
29
30     public bool Ok { get => ok; set => ok = value; }
31
32
33
```

Uno de los errores encontrados es, en la línea 63, se crea un nuevo vector llamado números, pero antes de usarse, se crea un vector con el mismo nombre sin usarse el anterior, además se crea otro vector en el botón “btValidar”, en la línea 46.

Para evitar la duplicidad de código, declaramos el vector “int[] numeros= new int [6]” como variable privada al inicio y eliminamos estas.

(Antes del cambio)

```
44 private void btValidar_Click(object sender, EventArgs e)
45 {
46     int[] numeros = new int[6];
47     for (int i = 0; i < 6; i++)
48         numeros[i] = Convert.ToInt32(combinacion[i].Text);
49     miLoto = new Loto(numeros);
50     if (miLoto.Ok)
51         MessageBox.Show("Combinación válida");
52     else
53         MessageBox.Show("Combinación no válida");
54 }
55
56 private void Form1_Load(object sender, EventArgs e)
57 {
58 }
59
60
61 private void btComprobar_Click(object sender, EventArgs e)
62 {
63     int[] numeros = new int[6];
64     for (int i = 0; i < 6; i++)
65         numeros[i] = Convert.ToInt32(combinacion[i].Text);
66     miLoto = new Loto(numeros);
67     if (miLoto.Ok)
68     {
69         numeros = new int[6];
70         for (int i = 0; i < 6; i++)
71             numeros[i] = Convert.ToInt32(combinacion[i].Text);
72         int aciertos = miGanadora.comprobar(numeros);
```

(Después del cambio)

```
16 public Loto miLoto, miGanadora;
17 // Estos arrays se usan para recorrer de manera más sencilla los controles
18 private TextBox[] combinacion = new TextBox[6];
19 private TextBox[] ganadora = new TextBox[6];
20 int[] numeros = new int[6];
```

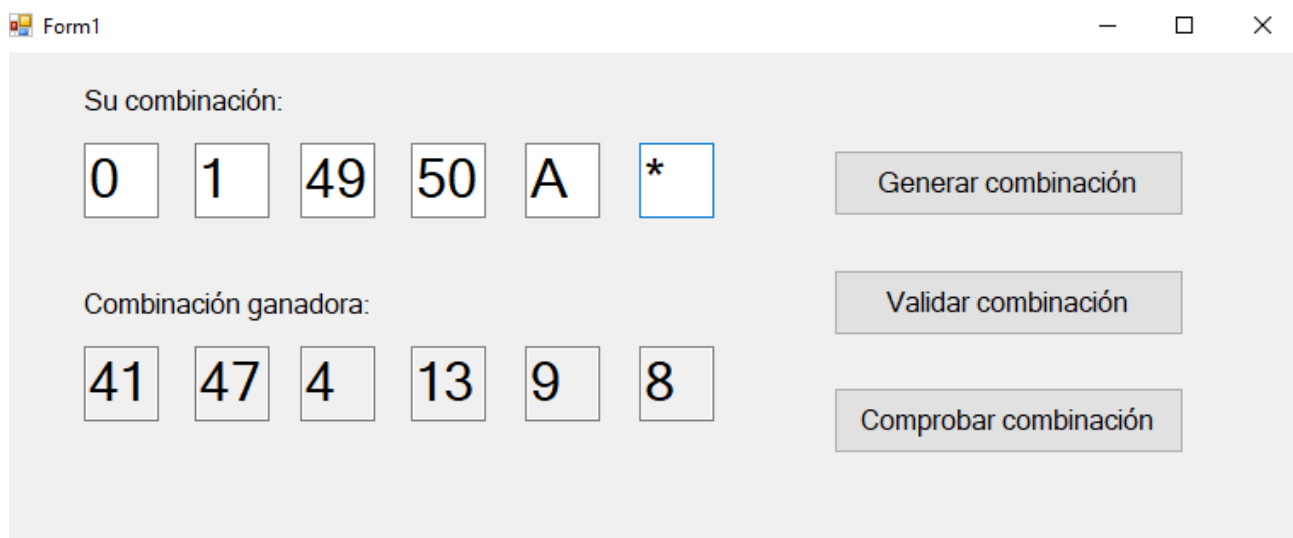
4. Realizar el diseño de pruebas (caja negra) para el constructor con parámetro de la clase `loto`.

```
// definición de constantes
private const int MAX_NUMEROS = 6;
private const int NUMERO_MENOR = 1;
private const int NUMERO_MAYOR = 49;
```

Como podemos ver en la imagen, los valores válidos son los valores comprendidos entre 1 y 49.

Para hacer pruebas de caja negra usaremos las siguientes clases de equivalencia:

- Valores válidos, valores entre 1 y 49
- Valores cercanos al límite, tanto válidos como no: 0, 1, 2, 48, 49, 50
- Valores alfabéticos y valores especiales: A-Z, a-z, ?, *,
- Valores nulos



Al hacer la comprobación, vemos que para el programa, esto es debido a la falta de implementación de excepciones.

```
for (int i = 0; i < 6; i++)
    numeros[i] = Convert.ToInt32(combinacion[i].Text);
miLoto = new Loto(numeros);
if (miLoto.Ok)
    MessageBox.Show("Combinación válida");
```

El procedimiento para hacer estas pruebas no es como aparece en la imagen, se deben hacer distintas pruebas, con las 6 casillas con valores válidos, otra prueba con las seis casillas con valores límite, otra con las seis casillas con valores alfabéticos y otra con valores nulos.

El apartado de las excepciones se aplicaría en el siguiente punto.