Comentarios

3 EN RAYA EN WINDOWSFORMS

Emiliano Montesdeoca del Puerto
1DAWB | CESAR MANRIQUE

Índice

Enunciado

Se pide:

Toman el código de uno de los programas que tienen hechos de programación. Por ejemplo Naves, Serpiente o juego del Tres en Raya de Windows Forms.
Le añaden a ese código los comentarios de tres barras a "///" a cada atributo, propiedad, método, constructor y clase, para que al ejecutar la utilidad de generación de documentación, genere el archivo XML correspondiente.

Añadir también comentarios estandar tipo C#, como son // o /* */ comentando el funcionamiento del programa.

A parte de esto crearemos un documento texto, en donde pondremos:

- Proyecto
- Fecha
- Autor
- Versión
- Clases que lo componen con sus métodos, atributos y propiedades. Sólo el esqueleto. Indicar los tipos de datos de estos atributos.
- Eventos que maneja el programa y a qué método llaman con sus parámetros.

El informe a realizare debe contener:

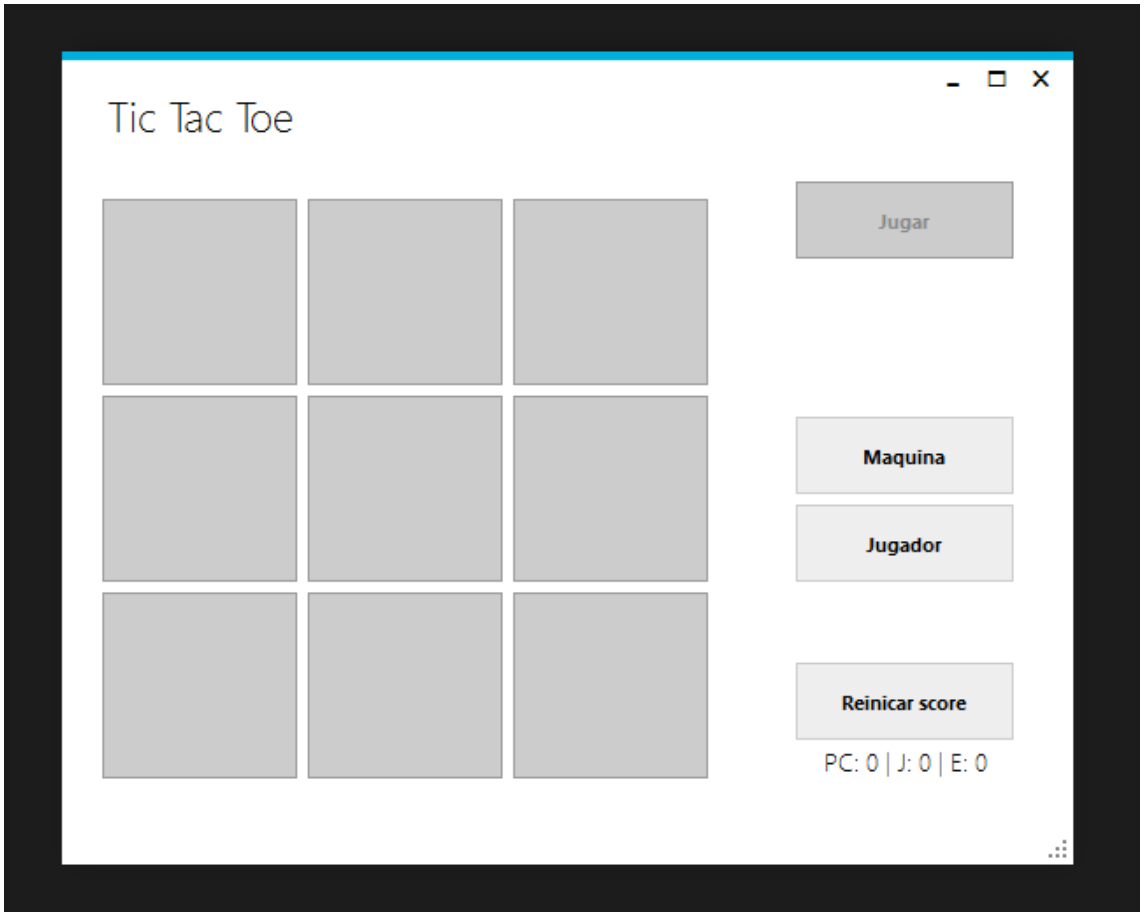
- Carátula inicial, índice, enunciado y objetivos de la práctica.
- Código original sin comentarios,
- Código con los comentarios,
- Archivo XML resultante,
- El archivo documento anteriormente descrito.
- La explicación del proceso de generación de documentación, de la documentación y cuál es el objetivo de la realización de la documentación de un programa.

Módulo de Entornos de Desarrollo 1º DAW.

Jorge Rivero, profesor del módulo.

Santa Cruz de Tenerife a 24 de Enero de 2017

Introduccion



El proyecto utilizado es el 3 en raya realizado como practica para la clase de programación, realizado en Enero por Emiliano Montesdeoca del Puerto.

Clases

Las clases que componen el programa son:

1. Juego

Inclue los siguientes atributos:

```
1. public static List<string> listaBtn
```

Incluye los siguientes métodos:

```
1. public Juego()
2. public static void CambioValorLista(int btn, string btnChar)
3. public static void CrearListaBotones()
4. public static void ReinicarValoresListaNull()
5. public static bool Comprobar()
6. public static int TurnoMaquina(int turno, bool Jugador)
7. public static int ComprobarSiExisteValorLista(int v)
```

2. Form1

Inclue los siguientes atributos:

```
1. List<Button> lButtons;
2. List<Button> lButtonsControles;
3. private int Turno;
4. private int scorePC;
5. private int scoreJ;
6. private int scoreTie;
7. private bool EmpezoJugador;
```

Incluye los siguientes métodos:

```
1. public Form1()
2. public void Form1_Load(object sender, EventArgs e)
3. public void StartJuego()
4. private string quienJuega()
5. private void CambioButton(int i)
6. private void ResetButtons()
7. private void ResetButtonsValue()
8. private void DisableButtons()
9. private void DisableSeleccionJugador()
10. private void EnableSeleccionJugador()
11. private void checkPartida()
12. private void JuegaMaquina()
```

Incluye los siguientes eventos

```
1.  private void upleft_Click(object sender, EventArgs e)
2.  private void button_2_Click(object sender, EventArgs e)
3.  private void button_3_Click(object sender, EventArgs e)
4.  private void button_4_Click(object sender, EventArgs e)
5.  private void button_5_Click(object sender, EventArgs e)
6.  private void metroButton4_Click(object sender, EventArgs e)
7.  private void button_7_Click(object sender, EventArgs e)
8.  private void button_8_Click(object sender, EventArgs e)
9.  private void button_9_Click(object sender, EventArgs e)
10. private void metroButton1_Click(object sender, EventArgs e)
11. private void metroButton2_Click(object sender, EventArgs e)
12. private void metroButton3_Click(object sender, EventArgs e)
13. private void metroButton4_Click_1(object sender, EventArgs e)
14. private void metroLabel1_Click(object sender, EventArgs e)
```

Codigo original

Clase Juego

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;

namespace _3enraya
{
    class Juego
    {
        public static List<string> listaBtn = new List<string>();
        public static void CrearListaBotones()
        {
            for (int i = 1; i < 10; i++)
            {
                listaBtn.Add("btn" + i);
                listaBtn[i - 1] = null;
            }

        }
        public Juego()
        {
            ReinicarValoresListaNull();
            CrearListaBotones();
        }
        public static void CambioValorLista(int btn, string btnChar)
        {
            listaBtn[btn] = btnChar;

        }
        public static void ReinicarValoresListaNull()
        {
            for (int i = 0; i < listaBtn.Count; i++)
            {
                listaBtn[i] = null;
            }
        }
        public static bool Comprobar()
        {
            bool res = false;
            ///Horizontales
            ///1+2+3
            if (listaBtn[1 - 1] == listaBtn[2 - 1] && listaBtn[2 - 1] ==
listaBtn[3 - 1] && listaBtn[2 - 1] != null)
            {
                return true;
            }
            ///4+5+6
            if (listaBtn[4 - 1] == listaBtn[5 - 1] && listaBtn[5 - 1] ==
listaBtn[6 - 1] && listaBtn[5 - 1] != null)
            {
```

```
                return true;
            }
            ///7+8+9
            if (listaBtn[7 - 1] == listaBtn[8 - 1] && listaBtn[8 - 1] ==
listaBtn[9 - 1] && listaBtn[8] != null)
            {
                return true;
            }
            ///Verticales
            ///1+4+7
            if (listaBtn[1 - 1] == listaBtn[4 - 1] && listaBtn[4 - 1] ==
listaBtn[7 - 1] && listaBtn[4 - 1] != null)
            {
                return true;
            }
            ///2+5+8
            if (listaBtn[2 - 1] == listaBtn[5 - 1] && listaBtn[5 - 1] ==
listaBtn[8 - 1] && listaBtn[5 - 1] != null)
            {
                return true;
            }
            ///3+6+9
            if (listaBtn[3 - 1] == listaBtn[6 - 1] && listaBtn[6 - 1] ==
listaBtn[9 - 1] && listaBtn[6 - 1] != null)
            {
                return true;
            }
            ///Centro hacia los diagonales
            ///7+5+3//1+5+9//
            if (listaBtn[7 - 1] == listaBtn[5 - 1] && listaBtn[5 - 1] ==
listaBtn[3 - 1] && listaBtn[5 - 1] != null)
            {
                return true;
            }
            if (listaBtn[1 - 1] == listaBtn[5 - 1] && listaBtn[5 - 1] ==
listaBtn[9 - 1] && listaBtn[5 - 1] != null)
            {
                return true;
            }
            return res;
        }
        public static int TurnoMaquina(int turno, bool Jugador)
        {
            int res = 0;

            if (turno == 0 && Jugador)
            {
                return -1;
            }
            if (turno == 1 && !Jugador)
            {
                listaBtn[5 - 1] = "O";
                return 5 - 1;
            }
            if (turno >= 3 || turno == 1 && Jugador)
            {
                ///1-5>9
```

```
                if (listaBtn[1 - 1] == listaBtn[5 - 1] && listaBtn[1 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[9 - 1] != "X" && listaBtn[9 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(9 - 1);
                }
                ///9-5>1
                if (listaBtn[9 - 1] == listaBtn[5 - 1] && listaBtn[9 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[1 - 1] != "X" && listaBtn[1 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(1 - 1);
                }
                ///3-5>7
                if (listaBtn[3 - 1] == listaBtn[5 - 1] && listaBtn[3 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[7 - 1] != "X" && listaBtn[7 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(7 - 1);
                }
                ///7-5>3
                if (listaBtn[7 - 1] == listaBtn[5 - 1] && listaBtn[7 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[3 - 1] != "X" && listaBtn[3 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(3 - 1);
                }
                ///1-2>3
                if (listaBtn[1 - 1] == listaBtn[2 - 1] && listaBtn[1 - 1] !=
null && listaBtn[2 - 1] != null && listaBtn[3 - 1] != "X" && listaBtn[3 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(3 - 1);
                }
                ///3-2>1
                if (listaBtn[3 - 1] == listaBtn[2 - 1] && listaBtn[3 - 1] !=
null && listaBtn[2 - 1] != null && listaBtn[1 - 1] != "X" && listaBtn[1 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(1 - 1);
                }
                ///7-8>9
                if (listaBtn[7 - 1] == listaBtn[8 - 1] && listaBtn[7 - 1] !=
null && listaBtn[8 - 1] != null && listaBtn[9 - 1] != "X" && listaBtn[9 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(9 - 1);
                }
                ///9-8>7
                if (listaBtn[9 - 1] == listaBtn[8 - 1] && listaBtn[9 - 1] !=
null && listaBtn[8 - 1] != null && listaBtn[7 - 1] != "X" && listaBtn[7 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(7 - 1);
                }
                ///4-5>6
                if (listaBtn[4 - 1] == listaBtn[5 - 1] && listaBtn[4 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[6 - 1] != "X" && listaBtn[6 - 1]
!= "O")
```

```
                {
                    return ComprobarSiExisteValorLista(6 - 1);
                }
                ///6-5>4
                if (listaBtn[6 - 1] == listaBtn[5 - 1] && listaBtn[6 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[4 - 1] != "X" && listaBtn[4 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(4 - 1);
                }
                ///1-4>7
                if (listaBtn[1 - 1] == listaBtn[4 - 1] && listaBtn[1 - 1] !=
null && listaBtn[4 - 1] != null && listaBtn[7 - 1] != "X" && listaBtn[7 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(7 - 1);
                }
                ///7-4>1
                if (listaBtn[7 - 1] == listaBtn[4 - 1] && listaBtn[7 - 1] !=
null && listaBtn[4 - 1] != null && listaBtn[1 - 1] != "X" && listaBtn[1 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(1 - 1);
                }
                ///3-6>9
                if (listaBtn[3 - 1] == listaBtn[6 - 1] && listaBtn[3 - 1] !=
null && listaBtn[6 - 1] != null && listaBtn[9 - 1] != "X" && listaBtn[9 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(9 - 1);
                }
                ///9-6>3
                if (listaBtn[9 - 1] == listaBtn[6 - 1] && listaBtn[9 - 1] !=
null && listaBtn[6 - 1] != null && listaBtn[3 - 1] != "X" && listaBtn[3 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(3 - 1);
                }
                ///2-5>8
                if (listaBtn[2 - 1] == listaBtn[5 - 1] && listaBtn[2 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[8 - 1] != "X" && listaBtn[8 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(8 - 1);
                }
                ///8-5>2
                if (listaBtn[8 - 1] == listaBtn[5 - 1] && listaBtn[8 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[2 - 1] != "X" && listaBtn[2 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(2 - 1);
                }
                /// 1-3>2
                if (listaBtn[1 - 1] == listaBtn[3 - 1] && listaBtn[1 - 1] !=
null && listaBtn[3 - 1] != null && listaBtn[2 - 1] != "X" && listaBtn[2 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(2 - 1);
                }
```

```
                ///1-7>4
                if (listaBtn[1 - 1] == listaBtn[7 - 1] && listaBtn[1 - 1] !=
null && listaBtn[7 - 1] != null && listaBtn[4 - 1] != "X" && listaBtn[4 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(4 - 1);
                }
                ///7-9>8
              if (listaBtn[7 - 1] == listaBtn[9 - 1] && listaBtn[7 - 1] !=
null && listaBtn[9 - 1] != null && listaBtn[8 - 1] != "X" && listaBtn[8 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(8 - 1);
                }
                ///3-9>6
                if (listaBtn[3 - 1] == listaBtn[9 - 1] && listaBtn[3 - 1] !=
null && listaBtn[9 - 1] != null && listaBtn[6 - 1] != "X" && listaBtn[6 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(6 - 1);
                }
                ///4-6>5
                if (listaBtn[4 - 1] == listaBtn[6 - 1] && listaBtn[4 - 1] !=
null && listaBtn[6 - 1] != null && listaBtn[5 - 1] != "X" && listaBtn[5 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(5 - 1);
                }
                ///2-8>5
                if (listaBtn[2 - 1] == listaBtn[8 - 1] && listaBtn[2 - 1] !=
null && listaBtn[8 - 1] != null && listaBtn[5 - 1] != "X" && listaBtn[5 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(5 - 1);
                }
                else
                {
                    Random r = new Random();
                    res = r.Next(0, 9);
                    return ComprobarSiExisteValorLista(res);
                }

            }
            return res;
        }
        public static int ComprobarSiExisteValorLista(int v)
        {
            while (true)
            {
                if (listaBtn[v] == "O" || listaBtn[v] == "X")
                {
                    Random r = new Random();
                    v = r.Next(0, 9);
                }
                else
                {
                    listaBtn[v] = "O";
                    return v;
                }
```

```
                }
            }
        }

    }
```

Clase Form1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MetroFramework.Forms;
using System.Diagnostics;

namespace _3enraya
{


    public partial class Form1 : MetroForm
    {
        List<Button> lButtons;
        List<Button> lButtonsControles;

        private int Turno = 0;
        private int scorePC = 0;
        private int scoreJ = 0;
        private int scoreTie = 0;
        private bool EmpezoJugador = false;


        public Form1()
        {
            InitializeComponent();
        }

        public void Form1_Load(object sender, EventArgs e)
        {
            new Juego();
            StartJuego();
        }

        public void StartJuego()
        {

            lButtons = new List<Button> { button_1, button_2, button_3,
button_4, button_5, button_6, button_7, button_8, button_9 };
            ResetButtonsValue();
            lButtonsControles = new List<Button> { metroButton1,
metroButton2, metroButton3, metroButton4 };
            DisableButtons();
            lButtonsControles[0].Enabled = false;
            EnableSeleccionJugador();
        }

        #region BOTONES DE JUEGO

        private void upleft_Click(object sender, EventArgs e)
```

```
{
    CambioButton(0);
    Juego.CambioValorLista(0, lButtons[0].Text);
    //checkPartida();
    JuegaMaquina();
}

private void button_2_Click(object sender, EventArgs e)
{
    CambioButton(1);
    Juego.CambioValorLista(1, lButtons[1].Text);
    //checkPartida();
    JuegaMaquina();
}

private void button_3_Click(object sender, EventArgs e)
{
    CambioButton(2);
    Juego.CambioValorLista(2, lButtons[2].Text);
    //checkPartida();
    JuegaMaquina();
}

private void button_4_Click(object sender, EventArgs e)
{
    CambioButton(3);
    Juego.CambioValorLista(3, lButtons[3].Text);
    //checkPartida();
    JuegaMaquina();
}

private void button_5_Click(object sender, EventArgs e)
{
    CambioButton(4);
    Juego.CambioValorLista(4, lButtons[4].Text);
    //checkPartida();
    JuegaMaquina();
}
private void metroButton4_Click(object sender, EventArgs e)
{
    CambioButton(5);
    Juego.CambioValorLista(5, lButtons[5].Text);
    //checkPartida();
    JuegaMaquina();
}
private void button_7_Click(object sender, EventArgs e)
{
    CambioButton(6);
    Juego.CambioValorLista(6, lButtons[6].Text);
    //checkPartida();
    JuegaMaquina();
}

private void button_8_Click(object sender, EventArgs e)
{
    CambioButton(7);
    Juego.CambioValorLista(7, lButtons[7].Text);
    //checkPartida();
    JuegaMaquina();
```

```
}

private void button_9_Click(object sender, EventArgs e)
{
    CambioButton(8);
    Juego.CambioValorLista(8, lButtons[8].Text);
    //checkPartida();
    JuegaMaquina();
}
#endregion

#region BOTONES DE SELECCION

private void metroButton1_Click(object sender, EventArgs e)
{
    ResetButtons();
    Juego.ReinicarValoresListaNull();
    DisableSeleccionJugador();
    JuegaMaquina();

}
/// <summary>
/// Boton de seleccion de la X
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void metroButton2_Click(object sender, EventArgs e)
{
    Turno = 0;
    metroButton1.Enabled = true;
    DisableSeleccionJugador();
    EmpezoJugador = true;

}

private void metroButton3_Click(object sender, EventArgs e)
{

    scoreJ = 0; /
    scorePC = 0;
    scoreTie = 0;
    metroLabel1.Text = "PC: 0 | J: 0 | E: 0";
}
private void metroButton4_Click_1(object sender, EventArgs e)
{
    Turno = 1;
    metroButton1.Enabled = true;
    DisableSeleccionJugador();
    EmpezoJugador = false;
}
#endregion

#region METODOS DE CONTROL DE VISTA

private string quienJuega()
{
    int calculo = Turno % 2;
    /
```

```
    switch (calculo)
    {
        case 0:
            return "X";
        default:
            return "O";
    }
}

private void CambioButton(int i)
{
    lButtons[i].Text = quienJuega();
    lButtons[i].Enabled = false;
    Turno++;
}


private void ResetButtons()
{
    for (int i = 0; i < lButtons.Count; i++)
    {
        lButtons[i].Text = null;
        lButtons[i].Enabled = true;
    }
}
private void ResetButtonsValue()
{
    for (int i = 0; i < lButtons.Count; i++)
    {
        lButtons[i].Text = null;
    }
}
private void DisableButtons()
{
    for (int i = 0; i < lButtons.Count; i++)
    {
        lButtons[i].Enabled = false;
    }
}
private void DisableSeleccionJugador()
{
    for (int i = 1; i < lButtonsControles.Count; i++)
    {
        lButtonsControles[i].Enabled = false;
    }
}

private void EnableSeleccionJugador()
{
    for (int i = 1; i < lButtonsControles.Count; i++)
    {
        lButtonsControles[i].Enabled = true;
    }

}

private void checkPartida()
{
    if (Juego.Comprobar())
```

```
            {

                if (Turno % 2 == 0)
                {
                    MessageBox.Show("Gana la Maquina", "Fin de partida",
MessageBoxButtons.OK);
                    scorePC++;
                }
                else
                {
                    MessageBox.Show("Gana el Jugador", "Fin de partida",
MessageBoxButtons.OK);
                    scoreJ++;
                }
                StartJuego();
            }
            else if (Juego.Comprobar() == false && Turno == 10 || Turno == 9)

            {
                MessageBox.Show("Empate!", "Fin de partida",
MessageBoxButtons.OK);
                scoreTie++; .
                StartJuego();
            }
            metroLabel1.Text = "PC: " + scorePC + " | J: " + scoreJ + " | E:
" + scoreTie;
        }

        private void JuegaMaquina()
        {
            if (Juego.Comprobar() == false)
            {
                int a = 0;
                if (Turno % 2 != 0 && Turno != 9)
                {

                    a = Juego.TurnoMaquina(Turno, EmpezoJugador);
                    if (a == -1)
                    {

                    }
                    else
                    {
                        CambioButton(a);
                    }
                }

            }
            checkPartida();
        }

        #endregion

        #region LABELS
        private void metroLabel2_Click(object sender, EventArgs e)
        {

        }
```

```
        private void metroLabel1_Click(object sender, EventArgs e)
        {
            metroLabel1.Text = "PC: " + scorePC + " | J: " + scoreJ + " | E:
" + scoreTie;
        }

        #endregion
    }
```

Codigo comentado

Clase Juego

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Diagnostics;

namespace _3enraya
{
    class Juego
    {
        public static List<string> listaBtn = new List<string>();
        /// <summary>
        /// Crea la lista de botones
        /// </summary>
        public static void CrearListaBotones()
        {
            for (int i = 1; i < 10; i++)
            {
                listaBtn.Add("btn" + i);
                listaBtn[i - 1] = null;
            }

        }
        /// <summary>
        /// El constructor asigna valores nulos a todos los valores de los
botones
        /// </summary>
        public Juego()
        {
            ReinicarValoresListaNull();
            CrearListaBotones();
        }
        /// <summary>
        /// Metodo que asigna *************
        /// </summary>
        /// <param name="btn"></param>
        /// <param name="btnChar"></param>
        public static void CambioValorLista(int btn, string btnChar)
        {
            listaBtn[btn] = btnChar;

        }
        /// <summary>
        /// Resetea los botones a nulo
        /// </summary>
        public static void ReinicarValoresListaNull()
        {
```

```
            for (int i = 0; i < listaBtn.Count; i++)
            {
                listaBtn[i] = null;
            }
        }
        /// <summary>
        /// Comprueba si hay 3 en raya
        /// </summary>
        /// <returns></returns>
        public static bool Comprobar()
        {
            bool res = false;
            ///Horizontales
            ///1+2+3
            if (listaBtn[1 - 1] == listaBtn[2 - 1] && listaBtn[2 - 1] ==
listaBtn[3 - 1] && listaBtn[2 - 1] != null)
            {
                return true;
            }
            ///4+5+6
            if (listaBtn[4 - 1] == listaBtn[5 - 1] && listaBtn[5 - 1] ==
listaBtn[6 - 1] && listaBtn[5 - 1] != null)
            {
                return true;
            }
            ///7+8+9
            if (listaBtn[7 - 1] == listaBtn[8 - 1] && listaBtn[8 - 1] ==
listaBtn[9 - 1] && listaBtn[8] != null)
            {
                return true;
            }
            ///Verticales
            ///1+4+7
            if (listaBtn[1 - 1] == listaBtn[4 - 1] && listaBtn[4 - 1] ==
listaBtn[7 - 1] && listaBtn[4 - 1] != null)
            {
                return true;
            }
            ///2+5+8
            if (listaBtn[2 - 1] == listaBtn[5 - 1] && listaBtn[5 - 1] ==
listaBtn[8 - 1] && listaBtn[5 - 1] != null)
            {
                return true;
            }
            ///3+6+9
            if (listaBtn[3 - 1] == listaBtn[6 - 1] && listaBtn[6 - 1] ==
listaBtn[9 - 1] && listaBtn[6 - 1] != null)
            {
                return true;
            }
            ///Centro hacia los diagonales
            ///7+5+3//1+5+9//
            if (listaBtn[7 - 1] == listaBtn[5 - 1] && listaBtn[5 - 1] ==
listaBtn[3 - 1] && listaBtn[5 - 1] != null)
            {
                return true;
            }
            if (listaBtn[1 - 1] == listaBtn[5 - 1] && listaBtn[5 - 1] ==
listaBtn[9 - 1] && listaBtn[5 - 1] != null)
```

```
            {
                return true;
            }
            return res;
        }
        /// <summary>
        /// Metodo encargado de realizar el turno de la maquina
        /// </summary>
        /// <param name="turno">Turno en el que esta la partida</param>
        /// <param name="JStarted">Booleano que significa si el Jugador
empieza</param>
        /// <returns></returns>
        public static int TurnoMaquina(int turno, bool Jugador)
        {
            int res = 0;

            if (turno == 0 && Jugador)
            {
                return -1;
            }
            if (turno == 1 && !Jugador) //Cuando la maquina es la primera y
jugador es falso pone el circulo en el medio
            {
                listaBtn[5 - 1] = "O";
                return 5 - 1;
            }
            if (turno >= 3 || turno == 1 && Jugador)
            // Cuando la maquina es la primera en tirar significa que juega
en los pares, y si no es primera signfica que juega en los impares
            // Asignamos el valor del bool Jugador para poder cambiar si
juega o no.
            {
                ///1-5>9
                if (listaBtn[1 - 1] == listaBtn[5 - 1] && listaBtn[1 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[9 - 1] != "X" && listaBtn[9 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(9 - 1);
                }
                ///9-5>1
                if (listaBtn[9 - 1] == listaBtn[5 - 1] && listaBtn[9 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[1 - 1] != "X" && listaBtn[1 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(1 - 1);
                }
                ///3-5>7
                if (listaBtn[3 - 1] == listaBtn[5 - 1] && listaBtn[3 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[7 - 1] != "X" && listaBtn[7 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(7 - 1);
                }
                ///7-5>3
                if (listaBtn[7 - 1] == listaBtn[5 - 1] && listaBtn[7 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[3 - 1] != "X" && listaBtn[3 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(3 - 1);
```

```
                }
                ///1-2>3
                if (listaBtn[1 - 1] == listaBtn[2 - 1] && listaBtn[1 - 1] !=
null && listaBtn[2 - 1] != null && listaBtn[3 - 1] != "X" && listaBtn[3 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(3 - 1);
                }
                ///3-2>1
                if (listaBtn[3 - 1] == listaBtn[2 - 1] && listaBtn[3 - 1] !=
null && listaBtn[2 - 1] != null && listaBtn[1 - 1] != "X" && listaBtn[1 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(1 - 1);
                }
                ///7-8>9
                if (listaBtn[7 - 1] == listaBtn[8 - 1] && listaBtn[7 - 1] !=
null && listaBtn[8 - 1] != null && listaBtn[9 - 1] != "X" && listaBtn[9 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(9 - 1);
                }
                ///9-8>7
                if (listaBtn[9 - 1] == listaBtn[8 - 1] && listaBtn[9 - 1] !=
null && listaBtn[8 - 1] != null && listaBtn[7 - 1] != "X" && listaBtn[7 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(7 - 1);
                }
                ///4-5>6
                if (listaBtn[4 - 1] == listaBtn[5 - 1] && listaBtn[4 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[6 - 1] != "X" && listaBtn[6 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(6 - 1);
                }
                ///6-5>4
                if (listaBtn[6 - 1] == listaBtn[5 - 1] && listaBtn[6 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[4 - 1] != "X" && listaBtn[4 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(4 - 1);
                }
                ///1-4>7
                if (listaBtn[1 - 1] == listaBtn[4 - 1] && listaBtn[1 - 1] !=
null && listaBtn[4 - 1] != null && listaBtn[7 - 1] != "X" && listaBtn[7 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(7 - 1);
                }
                ///7-4>1
                if (listaBtn[7 - 1] == listaBtn[4 - 1] && listaBtn[7 - 1] !=
null && listaBtn[4 - 1] != null && listaBtn[1 - 1] != "X" && listaBtn[1 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(1 - 1);
                }
                ///3-6>9
```

```
                if (listaBtn[3 - 1] == listaBtn[6 - 1] && listaBtn[3 - 1] !=
null && listaBtn[6 - 1] != null && listaBtn[9 - 1] != "X" && listaBtn[9 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(9 - 1);
                }
                ///9-6>3
                if (listaBtn[9 - 1] == listaBtn[6 - 1] && listaBtn[9 - 1] !=
null && listaBtn[6 - 1] != null && listaBtn[3 - 1] != "X" && listaBtn[3 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(3 - 1);
                }
                ///2-5>8
                if (listaBtn[2 - 1] == listaBtn[5 - 1] && listaBtn[2 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[8 - 1] != "X" && listaBtn[8 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(8 - 1);
                }
                ///8-5>2
                if (listaBtn[8 - 1] == listaBtn[5 - 1] && listaBtn[8 - 1] !=
null && listaBtn[5 - 1] != null && listaBtn[2 - 1] != "X" && listaBtn[2 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(2 - 1);
                }
                /// 1-3>2
                if (listaBtn[1 - 1] == listaBtn[3 - 1] && listaBtn[1 - 1] !=
null && listaBtn[3 - 1] != null && listaBtn[2 - 1] != "X" && listaBtn[2 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(2 - 1);
                }
                ///1-7>4
                if (listaBtn[1 - 1] == listaBtn[7 - 1] && listaBtn[1 - 1] !=
null && listaBtn[7 - 1] != null && listaBtn[4 - 1] != "X" && listaBtn[4 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(4 - 1);
                }
                ///7-9>8
                if (listaBtn[7 - 1] == listaBtn[9 - 1] && listaBtn[7 - 1] !=
null && listaBtn[9 - 1] != null && listaBtn[8 - 1] != "X" && listaBtn[8 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(8 - 1);
                }
                ///3-9>6
                if (listaBtn[3 - 1] == listaBtn[9 - 1] && listaBtn[3 - 1] !=
null && listaBtn[9 - 1] != null && listaBtn[6 - 1] != "X" && listaBtn[6 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(6 - 1);
                }
                ///4-6>5
                if (listaBtn[4 - 1] == listaBtn[6 - 1] && listaBtn[4 - 1] !=
null && listaBtn[6 - 1] != null && listaBtn[5 - 1] != "X" && listaBtn[5 - 1]
!= "O")
```

```
                {
                    return ComprobarSiExisteValorLista(5 - 1);
                }
                ///2-8>5
                if (listaBtn[2 - 1] == listaBtn[8 - 1] && listaBtn[2 - 1] !=
null && listaBtn[8 - 1] != null && listaBtn[5 - 1] != "X" && listaBtn[5 - 1]
!= "O")
                {
                    return ComprobarSiExisteValorLista(5 - 1);
                }
                else
                {
                    Random r = new Random();
                    res = r.Next(0, 9);
                    return ComprobarSiExisteValorLista(res);
                }

            }
            return res;
        }
        /// <summary>
        /// Si existe el valor en la lista, genera un nuevo valor hasta que
este libre.
        /// </summary>
        /// <param name="v">Valor a comprobar</param>
        /// <returns>Nuevo valor</returns>
        public static int ComprobarSiExisteValorLista(int v)
        {
            while (true)
            {
                if (listaBtn[v] == "O" || listaBtn[v] == "X")
                {
                    Random r = new Random();
                    v = r.Next(0, 9);
                }
                else
                {
                    listaBtn[v] = "O";
                    return v;
                }
            }
        }
    }

}
```

Clase Form1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MetroFramework.Forms;
using System.Diagnostics;

namespace _3enraya
{


    public partial class Form1 : MetroForm
    {
        List<Button> lButtons; // Lista de botones 1-9
        List<Button> lButtonsControles; // Lista de botones de
control(Juegar-Maquina-Jugador-Restear)

        private int Turno = 0; // Variable de Turno.
        private int scorePC = 0; // Variable de puntuaciones de la maquina.
        private int scoreJ = 0; // Variable de puntuaciones del jugador.
        private int scoreTie = 0; // Variable de puntuaciones de empates.
        //private bool quienJuegaBol = false; // Variable para saber que
boton de seleccion de jugador ha pulsado, PC = false, Jug = true
        //private bool JStarted = false; //Variable para saber si ha empezado
el Jugador, este bool es para controlar si en los Turnos en
        ////los que no puede atacar ni defender, realize un movimiento
aleatorio.
        private bool EmpezoJugador = false;


        public Form1()
        {
            InitializeComponent();
        }

        public void Form1_Load(object sender, EventArgs e)
        {
            new Juego();
            StartJuego();
        }

        public void StartJuego()
        {

            ///Lista de botones del juego, del 1 al 9
            lButtons = new List<Button> { button_1, button_2, button_3,
button_4, button_5, button_6, button_7, button_8, button_9 };
            ResetButtonsValue();
            ///Lista de botones de control: Jugar - maquina y resetear la
puntuacion
            lButtonsControles = new List<Button> { metroButton1,
metroButton2, metroButton3, metroButton4 };
```

```
            ///Desactiva los botones de juego
            DisableButtons(); //Desactiva el boton de jugar hasta que se
selecciona un juegador o letra
            lButtonsControles[0].Enabled = false; //Metodo que deshabilita el
boton de Jugar hasta que se selecciona un jugador
            EnableSeleccionJugador(); //Metodo que habilita los botones de
seleccion de jugador
        }

        #region BOTONES DE JUEGO

        private void upleft_Click(object sender, EventArgs e)
        {
            CambioButton(0);
            Juego.CambioValorLista(0, lButtons[0].Text);
            //checkPartida();
            JuegaMaquina();
        }

        private void button_2_Click(object sender, EventArgs e)
        {
            CambioButton(1);
            Juego.CambioValorLista(1, lButtons[1].Text);
            //checkPartida();
            JuegaMaquina();
        }

        private void button_3_Click(object sender, EventArgs e)
        {
            CambioButton(2);
            Juego.CambioValorLista(2, lButtons[2].Text);
            //checkPartida();
            JuegaMaquina();
        }

        private void button_4_Click(object sender, EventArgs e)
        {
            CambioButton(3);
            Juego.CambioValorLista(3, lButtons[3].Text);
            //checkPartida();
            JuegaMaquina();
        }

        private void button_5_Click(object sender, EventArgs e)
        {
            CambioButton(4);
            Juego.CambioValorLista(4, lButtons[4].Text);
            //checkPartida();
            JuegaMaquina();
        }
        private void metroButton4_Click(object sender, EventArgs e)
        {
            CambioButton(5);
            Juego.CambioValorLista(5, lButtons[5].Text);
            //checkPartida();
            JuegaMaquina();
        }
        private void button_7_Click(object sender, EventArgs e)
        {
```

```
            CambioButton(6);
            Juego.CambioValorLista(6, lButtons[6].Text);
            //checkPartida();
            JuegaMaquina();
        }

        private void button_8_Click(object sender, EventArgs e)
        {
            CambioButton(7);
            Juego.CambioValorLista(7, lButtons[7].Text);
            //checkPartida();
            JuegaMaquina();

        }

        private void button_9_Click(object sender, EventArgs e)
        {
            CambioButton(8);
            Juego.CambioValorLista(8, lButtons[8].Text);
            //checkPartida();
            JuegaMaquina();
        }
        #endregion

        #region BOTONES DE SELECCION
        /// <summary>
        /// Boton de JUGAR
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void metroButton1_Click(object sender, EventArgs e)
        {
            ResetButtons(); //Resetea los botones, les da valores nulos.
            Juego.ReinicarValoresListaNull(); // Cambia los valores de la
estructura en la clase Juego.
            DisableSeleccionJugador(); // Deshabilita los botones de jugar.
            JuegaMaquina(); // Metodo para que juege la maquina.

        }
        /// <summary>
        /// Boton de seleccion de la X
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void metroButton2_Click(object sender, EventArgs e)
        {
            Turno = 0; // Empieza en el Turno 0, ignorando el primer Turno de
la maquina y asi el tercero es aleatorio.
            metroButton1.Enabled = true; //Una vez clicado este boton, activa
el de Jugar.
            DisableSeleccionJugador(); // Deshabilita los botones de
seleccion de jugador.
            EmpezoJugador = true;

        }
        /// <summary>
        /// Boton de reinicio de puntos
        /// </summary>
        /// <param name="sender"></param>
```

```
        /// <param name="e"></param>
        private void metroButton3_Click(object sender, EventArgs e)
        {

            scoreJ = 0; //Asigna valor 0 a la puntuacion del Jugador.
            scorePC = 0; //Asigna valor 0 a la puntuacion de la Maquina.
            scoreTie = 0; //Asigna valor 0 a la puntuacion de empate.
            metroLabel1.Text = "PC: 0 | J: 0 | E: 0";
        }
        /// <summary>
        /// Boton de seleccion de la maquina
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void metroButton4_Click_1(object sender, EventArgs e)
        {
            Turno = 1; // Empieza en el Turno 1, empezando asi una O en el
medio, ya que empieza la Maquina.
            metroButton1.Enabled = true; // Una vez clicado este boton,
activa el de Jugar.
            DisableSeleccionJugador(); // Deshabilita los botones de
seleccion de jugador.
            EmpezoJugador = false;
        }
        #endregion

        #region METODOS DE CONTROL DE VISTA

        /// <summary>
        /// Metodo que devuelve a los botones la letra que ponen dependiendo
del Turno.
        /// </summary>
        /// <returns>Retorna X u O</returns>
        private string quienJuega()
        {
            int calculo = Turno % 2;
            // Este calculo determina la letra que se va a escribir
dependiendo del Turno en el que estamos
            // en el juego, si empieza la Maquina el Turno empieza en 1 por
lo que 1%2!=0 escribe O, y se va sumando la variable
            // Turno al final del turno, asi que en el siguiente Turno sera
una X ya que 2%2=0.
            // La forma de controlar esto es cambiando el valor de
quienJuegaInt
            switch (calculo)
            {
                case 0:
                    return "X";
                default:
                    return "O";
            }
        }

        /// <summary>
        /// Metodo encargado de darle X u O y deshabilita el boton
        /// </summary>
        /// <param name="i"></param>
        private void CambioButton(int i)
        {
```

```
            lButtons[i].Text = quienJuega(); // Asigna el texto llamando al
metodo quienJuega que lo devuelve dependiendo del Turno.
            lButtons[i].Enabled = false; // Desactiva el boton
            Turno++;
        }

        /// <summary>
        /// Resetea los botones, les asigna - y los habilita
        /// </summary>
        private void ResetButtons()
        {
            for (int i = 0; i < lButtons.Count; i++) // Bucle que resetea los
botones.
            {
                lButtons[i].Text = null; // Le asigna valor nulo.
                lButtons[i].Enabled = true; // Activa los botones.
            }
        }
        /// <summary>
        /// Resetea los botones
        /// </summary>
        private void ResetButtonsValue()
        {
            for (int i = 0; i < lButtons.Count; i++) // Bucle que resetea los
botones.
            {
                lButtons[i].Text = null; // Le asigna valor nulo.
            }
        }
        /// <summary>
        /// Deshabilitar los botones
        /// </summary>
        private void DisableButtons()
        {
            for (int i = 0; i < lButtons.Count; i++)
            {
                lButtons[i].Enabled = false; // Desactiva los botones de
jugar(del tablero).
            }
        }
        /// <summary>
        /// Deshabilitar los botones de seleccion de jugador
        /// </summary>
        private void DisableSeleccionJugador()
        {
            for (int i = 1; i < lButtonsControles.Count; i++)
            {
                lButtonsControles[i].Enabled = false; // Desactiva los
botones de seleccion de jugador.
            }
        }
        /// <summary>
        /// Habilita los botones de seleccion de letra
        /// </summary>
        private void EnableSeleccionJugador()
        {
            for (int i = 1; i < lButtonsControles.Count; i++)
            {
```

```
                lButtonsControles[i].Enabled = true; // Activa los botones de
seleccion de jugador
            }

        }
        /// <summary>
        /// Metodo que mira si alguien ha ganado
        /// </summary>
        private void checkPartida()
        {
            if (Juego.Comprobar()) // Si el comprobar de la clase juego
devuelve verdadero siginifica que hay una combinacion ganadora
                                   // por lo que mirando el turno podemos
deducir quien es el ganador, si el turno que acaba es par significa que el
ganador
                                   // es la maquina, si es impar ha ganado el
Jugador. Esto depende de quien empieza la partida ya que el valor de Turno
                                   // depende del boton que selecciona a la
hora de jugar.
            {
                // Si el turno finalizado es par ha ganado la maquina.
                if (Turno % 2 == 0)
                {
                    MessageBox.Show("Gana la Maquina", "Fin de partida",
MessageBoxButtons.OK);
                    scorePC++; // Suma un punto a la puntuacion de la
maquina.
                }
                else
                {
                    MessageBox.Show("Gana el Jugador", "Fin de partida",
MessageBoxButtons.OK);
                    scoreJ++; // Suma un punto a la puntuacion del jugador.
                }
                StartJuego(); // Llamada al metodo de reincio de juego.
            }
            else if (Juego.Comprobar() == false && Turno == 10 || Turno == 9)
            // Este caso es para cuando es empate, si el Comprobar es falso y
el turno es 10, por lo que todos
            // los botones estan ocupados. La segunda parte del OR es para
cuando el jugador empieza a jugar.
            // Cuando se empata habiendo empezado el jugador el ultimo turno
no se llega a ser 10 por lo que hay
            // que acabar el juego en el turno 9 y utilizando la variable
JStarted que asignamos a verdadero cuando le damos
            // al boton de seleccionar que el Jugador empieza la partida.
            {
                MessageBox.Show("Empate!", "Fin de partida",
MessageBoxButtons.OK);
                scoreTie++; // Suma un punto a la puntuacion de empate.
                StartJuego(); // Llamada al metodo de reincio de juego.
            }
            metroLabel1.Text = "PC: " + scorePC + " | J: " + scoreJ + " | E:
" + scoreTie; // Actualizacion de label de puntuaciones.
        }

        private void JuegaMaquina()
        {
```

```
            if (Juego.Comprobar() == false) // Si la comprobacion de una
combinacion ganadora es falsa entonces jugara la maquina.
            {
                int a = 0;
                if (Turno % 2 != 0 && Turno != 9) // Si empezo el jugador y
el turno es 0 significa que no tiene que hacer nada
                {
                    // Variable entera para traer el boton que hay que pulsar
y actualizar
                    a = Juego.TurnoMaquina(Turno, EmpezoJugador); // Le
pasamos el turno para que el metodo decida que hacer.
                    if (a == -1)
                    {

                    }
                    else
                    {
                        CambioButton(a);
                    }
                }
                //if (Turno % 2 == 0 && EmpezoJugador )
                //{
                //    a = Juego.TurnoMaquina(Turno, EmpezoJugador) - 1;


                //}

            }
            checkPartida(); // Comprueba que el movimiento que ha hecho la
maquina es ganador
        }

        #endregion

        #region LABELS
        private void metroLabel2_Click(object sender, EventArgs e)
        {

        }


        private void metroLabel1_Click(object sender, EventArgs e)
        {
            metroLabel1.Text = "PC: " + scorePC + " | J: " + scoreJ + " | E:
" + scoreTie;
        }

        #endregion
    }
}
```

Archivo XML

```
<?xml version="1.0"?>

-<doc>


-<assembly>

<name>3enraya</name>

</assembly>


-<members>


-<member
name="M:_3enraya.Form1.metroButton1_Click(System.Object,System.EventArgs)">

<summary> Boton de JUGAR </summary>

<param name="sender"/>

<param name="e"/>

</member>


-<member
name="M:_3enraya.Form1.metroButton2_Click(System.Object,System.EventArgs)">

<summary> Boton de seleccion de la X </summary>

<param name="sender"/>

<param name="e"/>

</member>


-<member
name="M:_3enraya.Form1.metroButton3_Click(System.Object,System.EventArgs)">

<summary> Boton de reinicio de puntos </summary>

<param name="sender"/>

<param name="e"/>

</member>


-<member
name="M:_3enraya.Form1.metroButton4_Click_1(System.Object,System.EventArgs)">
```

```
<summary> Boton de seleccion de la maquina </summary>

<param name="sender"/>

<param name="e"/>

</member>


-<member name="M:_3enraya.Form1.quienJuega">

<summary> Metodo que devuelve a los botones la letra que ponen dependiendo
del Turno. </summary>

<returns>Retorna X u O</returns>

</member>


-<member name="M:_3enraya.Form1.CambioButton(System.Int32)">

<summary> Metodo encargado de darle X u O y deshabilita el boton </summary>

<param name="i"/>

</member>


-<member name="M:_3enraya.Form1.ResetButtons">

<summary> Resetea los botones, les asigna - y los habilita </summary>

</member>


-<member name="M:_3enraya.Form1.ResetButtonsValue">

<summary> Resetea los botones </summary>

</member>


-<member name="M:_3enraya.Form1.DisableButtons">

<summary> Deshabilitar los botones </summary>

</member>


-<member name="M:_3enraya.Form1.DisableSeleccionJugador">

<summary> Deshabilitar los botones de seleccion de jugador </summary>

</member>


-<member name="M:_3enraya.Form1.EnableSeleccionJugador">

<summary> Habilita los botones de seleccion de letra </summary>
```

```
</member>

-<member name="M:_3enraya.Form1.checkPartida">

<summary> Metodo que mira si alguien ha ganado </summary>

</member>

-<member name="F:_3enraya.Form1.components">

<summary> Variable del diseñador necesaria. </summary>

</member>

-<member name="M:_3enraya.Form1.Dispose(System.Boolean)">

<summary> Limpiar los recursos que se estén usando. </summary>

<param name="disposing">true si los recursos administrados se deben desechar;
false en caso contrario.</param>

</member>

-<member name="M:_3enraya.Form1.InitializeComponent">

<summary> Método necesario para admitir el Diseñador. No se puede modificar
el contenido de este método con el editor de código. </summary>

</member>

-<member name="M:_3enraya.Juego.CrearListaBotones">

<summary> Crea la lista de botones </summary>

</member>

-<member name="M:_3enraya.Juego.#ctor">

<summary> El constructor asigna valores nulos a todos los valores de los
botones </summary>

</member>

-<member
name="M:_3enraya.Juego.CambioValorLista(System.Int32,System.String)">

<summary> Metodo que asigna ************* </summary>

<param name="btn"/>

<param name="btnChar"/>
```

```
</member>

-<member name="M:_3enraya.Juego.ReinicarValoresListaNull">

<summary> Resetea los botones a nulo </summary>

</member>

-<member name="M:_3enraya.Juego.Comprobar">

<summary> Comprueba si hay 3 en raya </summary>

<returns/>

</member>

-<member name="M:_3enraya.Juego.TurnoMaquina(System.Int32,System.Boolean)">

<summary> Metodo encargado de realizar el turno de la maquina </summary>

<param name="turno">Turno en el que esta la partida</param>

<param name="JStarted">Booleano que significa si el Jugador empieza</param>

<returns/>

</member>

-<member name="M:_3enraya.Juego.ComprobarSiExisteValorLista(System.Int32)">

<summary> Si existe el valor en la lista, genera un nuevo valor hasta que
este libre. </summary>

<param name="v">Valor a comprobar</param>

<returns>Nuevo valor</returns>

</member>

-<member name="M:_3enraya.Program.Main">

<summary> Punto de entrada principal para la aplicación. </summary>

</member>

-<member name="T:_3enraya.Properties.Resources">

<summary> Clase de recurso fuertemente tipado para buscar cadenas traducidas,
etc. </summary>

</member>
```

```
-<member name="P:_3enraya.Properties.Resources.ResourceManager">

<summary> Devuelve la instancia ResourceManager almacenada en caché utilizada
por esta clase. </summary>

</member>


-<member name="P:_3enraya.Properties.Resources.Culture">

<summary> Invalida la propiedad CurrentUICulture del subproceso actual para
todas las búsquedas de recursos usando esta clase de recursos fuertemente
tipados. </summary>

</member>

</members>

</doc>
```

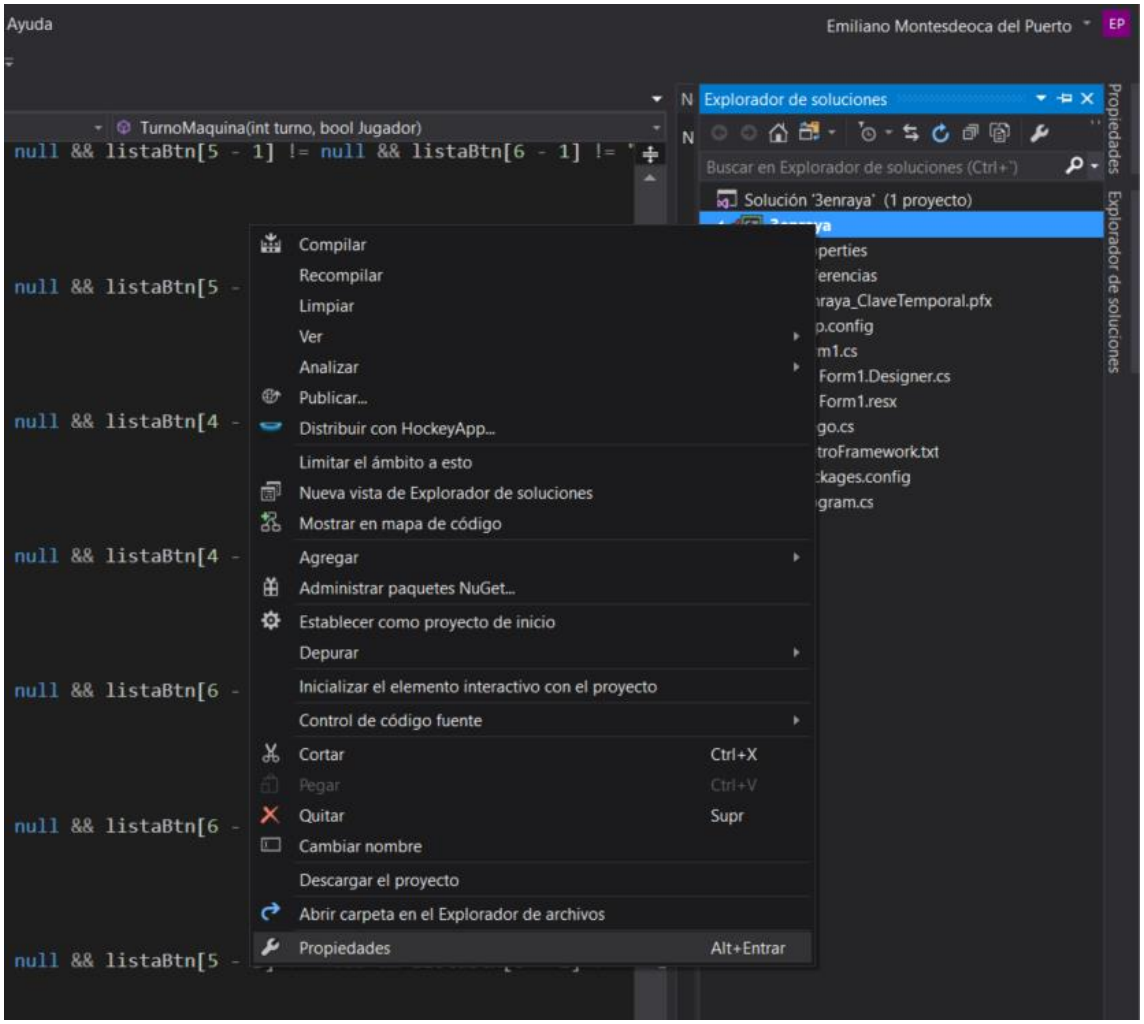
Generar archivo XML

La propiedad Generar archivo de documentación XML determina si un archivo XML se generará durante la compilación. Esta propiedad se establece en la página Compilación (para Visual Basic) o Generación (para Visual C#) del Diseñador de proyectos.

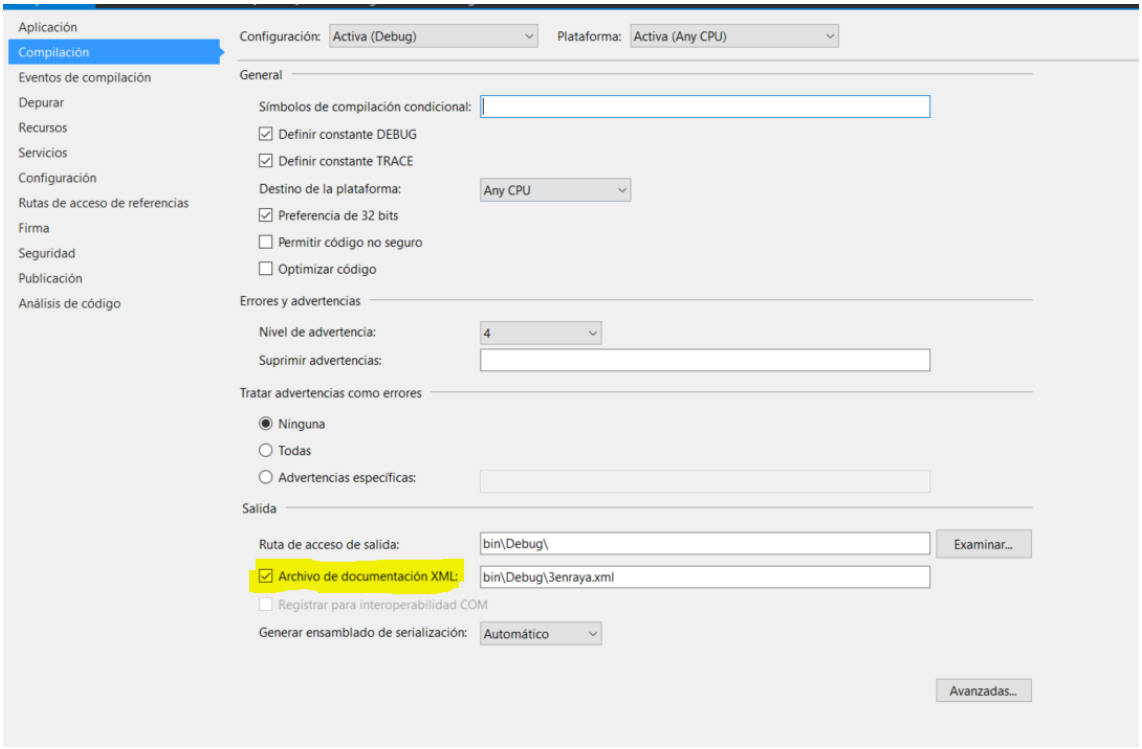
Cuando esta opción está seleccionada, la documentación XML se emite automáticamente en un archivo XML que tendrá el mismo nombre que el proyecto y la extensión .xml.

Para generar un archivo de documentación XML para un proyecto de Visual C#

1. Con un proyecto seleccionado en el **Explorador de soluciones**, en el menú **Proyecto** haga clic en **Propiedades**.



2. Haga clic en la ficha **Generar**.



3. En la página **Generación**, seleccione **Archivo de documentación XML**. De forma predeterminada, el archivo se crea bajo la ruta de acceso de resultados especificada, por ejemplo, "bin\Debug\Projectname.XML".