

## Reporte Lab7

Bryan Villegas Alvarado – B78383

### **Archivo: BosqueEncantado.cs**

Método: BosqueEncantado

# Línea(s): 24

Tipo de error y descripción: Use consistent capitalization

Hacer un uso de mayúsculas consistente.

Método: BosqueEncantado

# Línea(s): 25

Tipo de error y descripción: Error de sintaxis.

int\_desicion está incorrecto.

Método: BosqueEncantado

# Línea(s): 12

Tipo de error y descripción: Avoid using bad names

Usar nombres significativos y legibles (pfilas no es pronunciable, etc)

Método: BosqueEncantado

# Línea(s): 17

Tipo de error y descripción: Avoid Hungarian notation

Evitar repetir el tipo en el nombre de la variable doubleAleatorio

Método: BosqueEncantado

# Línea(s): 23

Tipo de error y descripción: Falta exportar librería.

**Archivo: Hangman.cs**

Método: PlayHangman

# Línea(s): 15

Tipo de error y descripción: Avoid using bad names

Usar nombres significativos y fáciles de entender.

Método: PlayHangman

# Línea(s): 16

Tipo de error y descripción: Don't leave commented out code in your codebase

Línea de código antiguo comentado

Método: PlayHangman

# Línea(s): 11

Tipo de error y descripción: Avoid magic string

Se puede definir el mensaje de bienvenida de forma global o más general.

Método: PlayHangman

# Línea(s): 37-40

Tipo de error y descripción: Don't ignore caught errors

Se atrapa el error, pero no se hace nada con el mismo.

Método: PlayHangman

# Línea(s): 47-51

Tipo de error y descripción: "else" innecesario, se hace más complejo de leer el código sin necesidad.

**Archivo: Letters.cs**

Método: Count

# Línea(s): ---

Tipo de error y descripción: Functions should do one thing

Count hace todo y no está modularizado.

Método: Count

# Línea(s): 9

Tipo de error y descripción: Avoid Hungarian notation

Evitar repetir el tipo en el nombre de la variable strCaptured

Método: Count

# Línea(s): 13

Tipo de error y descripción: Se define "int i" fuera del "for" sin necesidad.

Método: Count

# Línea(s): 37

Tipo de error y descripción: Código innecesario que puede causar errores.

Se lee una línea de código sin darle ningún uso o motivo.

Método: Count

# Línea(s): 17-20, 28

Tipo de error y descripción: Avoid magic string

Todas las letras y condiciones se pueden definir.

**Archivo: NumberGuessing.cs**

Método: Play

# Línea(s): 11

Tipo de error y descripción: Avoid using bad names

Randno no es nombre significativo.

Método: Newnum

# Línea(s): 47

Tipo de error y descripción: No tiene comentario significativo o que agregue valor.

Método: Newnum

# Línea(s): 47

Tipo de error y descripción: Nombre del método no representa que hace en realidad.

Método: Play

# Línea(s): ---

Tipo de error y descripción: No se hace manejo de errores.

Método:

# Línea(s):

Tipo de error y descripción:

**Archivo: Pantalla.cs**

Método: Pantalla

# Línea(s): 3

Tipo de error y descripción: Sintaxis, el tipo de matriz no está definido.

Método: crear

# Línea(s): 6

Tipo de error y descripción: No cumple con las reglas de nombrado

Método: crear

# Línea(s): 8

Tipo de error y descripción: Se usa una variable no definida en el contexto

Método: crear

# Línea(s): 9

Tipo de error y descripción: Error de sintaxis, GetLength está escrito de manera incorrecta.

Método: ---

# Línea(s): ---

Tipo de error y descripción: La clase no sigue el mismo formato de indentación que los otros archivos.

Método: crear

# Línea(s): 8-9

Tipo de error y descripción: Avoid mental mapping

**Archivo: Program.cs**

Método: ---

# Línea(s): 5-10

Tipo de error y descripción: Don't have journal comments

Método: Main

# Línea(s): 14-17

Tipo de error y descripción: Dejar código comentado.

Método: Main

# Línea(s): ---

Tipo de error y descripción: No existe manejo de errores.

Método:

# Línea(s):

Tipo de error y descripción:

**Archivo: Roman.cs**

Método: Roman

# Línea(s): 16-22, 26-40

Tipo de error y descripción: Indentación incorrecta en multiples líneas.

Método: Combine

# Línea(s): 45

Tipo de error y descripción: int value se puede declarar dentro de la línea 46.

Método: To

# Línea(s): 55

Tipo de error y descripción: Nombre significativo del método.

Método:

# Línea(s):

Tipo de error y descripción:

Método:

# Línea(s):

Tipo de error y descripción:

## Soluciones a problemas:

1.

Archivo: BosqueEncantado.cs

Método: BosqueEncantado

# Línea(s): 24

Tipo de error y descripción: Use consistent capitalization

```
24 int_decision = (int)numeroAleatorio;
```

No se presenta un uso consistente en las mayúsculas, importante para mantener un código legible.

```
24 intDecision = (int)numeroAleatorio;
```

Se cambia el nombre de la variable al formato camelcase para una mayor legibilidad.

2.

Archivo: Hangman.cs

Método: PlayHangman

# Línea(s): 16

Tipo de error y descripción: Don't leave commented out code in your codebase

Línea de código antiguo comentado

```
14 Random randGen = new Random();
15 var idx = randGen.Next(0, 61333);
16 //var idx = randGen.Next(0, 10);
17 string mysteryWord = new string("");
```

Dejar líneas de código comentado innecesario dificulta la legibilidad del código.

```
14 Random randGen = new Random();
15 var idx = randGen.Next(0, 61333);
16 string mysteryWord = new string("");
```

Se eliminó la línea de comentario innecesario, se usa el control de versiones si se desea ver esta otra línea.



3.

Archivo: Hangman.cs

Método: PlayHangman

# Línea(s): 37-40

Tipo de error y descripción: Don't ignore caught errors

Se atrapa el error, pero no se hace nada con el mismo.

```
37         catch (FormatException e)
38         {
39             playerGuess = ' ';
40         }
41         for (int j = 0; j < mysteryWord.Length; j++)
```

Al atrapar el error, pero no hacer nada con el mismo se crea un error “fantasma”.

```
37         catch (FormatException e)
38         {
39             throw new FormatException("Unexpected format",e);
40         }
```

Se maneja el error y se hace un throw para que sea manejado por el padre. Así podemos saber que está fallando en caso de error.

4.

Archivo: Pantalla.cs

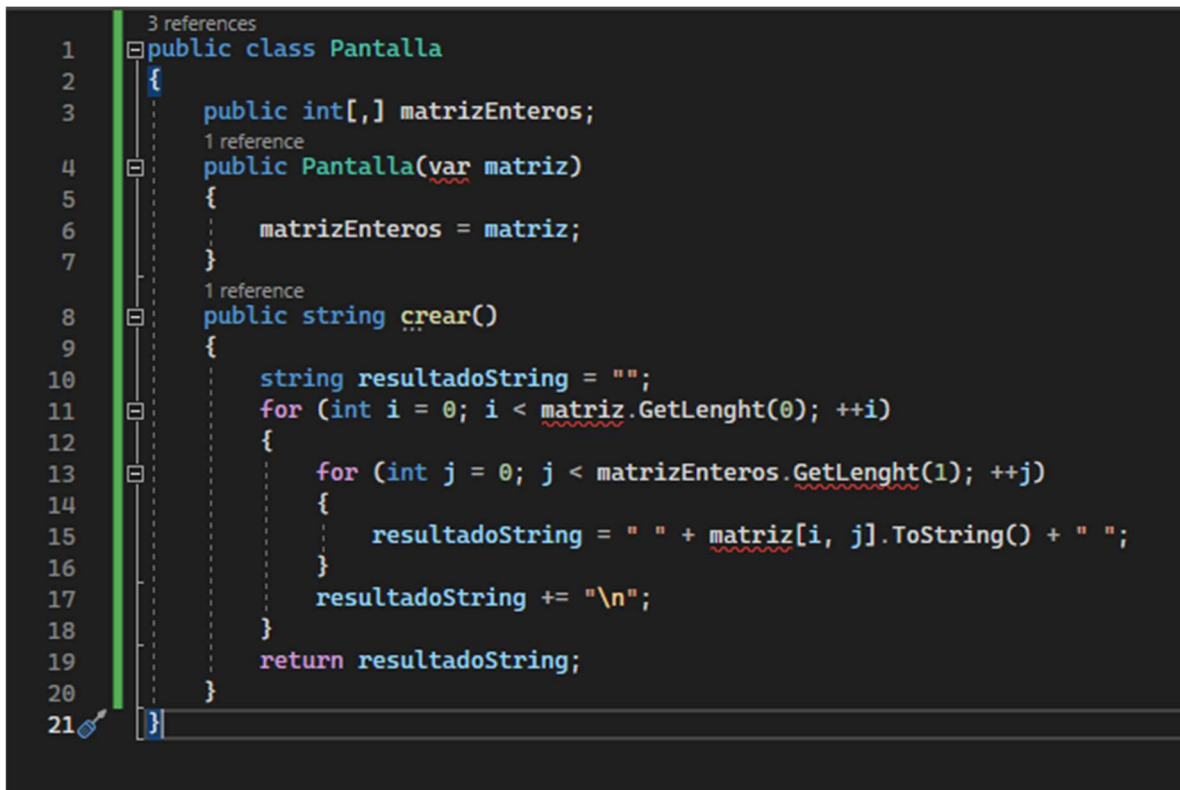
Método: ---

# Línea(s): ---

Tipo de error y descripción: La clase no sigue el mismo formato de indentación que los otros archivos.

```
1 public class Pantalla{
2     public int [,] matrizEnteros;
3     public Pantalla(var matriz){
4         matrizEnteros=matriz;
5     }
6     public string crear(){
7         string resultadoString="";
8         for (int i=0; i<matriz.GetLenght(0);++i){
9             for(int j=0;j<matrizEnteros.GetLenght(1);++j){
10                 resultadoString+=" "+matriz[i,j].ToString()+ " ";
11             }
12             resultadoString+="\n";
13         }
14         return resultadoString;
15     }
16 }
```

Se dificulta la legibilidad del código al cambiar de indentación y no seguir un formato.



```
1 public class Pantalla
2 {
3     public int[,] matrizEnteros;
4     public Pantalla(var matriz)
5     {
6         matrizEnteros = matriz;
7     }
8     public string crear()
9     {
10        string resultadoString = "";
11        for (int i = 0; i < matriz.GetLength(0); ++i)
12        {
13            for (int j = 0; j < matrizEnteros.GetLength(1); ++j)
14            {
15                resultadoString = " " + matriz[i, j].ToString() + " ";
16            }
17            resultadoString += "\n";
18        }
19        return resultadoString;
20    }
21 }
```

The screenshot shows a code editor with a dark background. On the left, a vertical line marks line numbers 1 through 21. The code is for a C# class named 'Pantalla'. It has a public field 'matrizEnteros' of type 'int[,]'. There are two methods: 'Pantalla(var matriz)' and 'crear()'. The indentation is inconsistent: the first method's body is indented 4 spaces, while the second method's body is indented 2 spaces. The 'crear()' method contains two nested 'for' loops. The first loop iterates over 'matriz' and the second over 'matrizEnteros'. The code is not formatted with consistent indentation, making it harder to read.

El código es mas legible al indentar de una manera adecuada y que siga los estándares de los otros archivos.

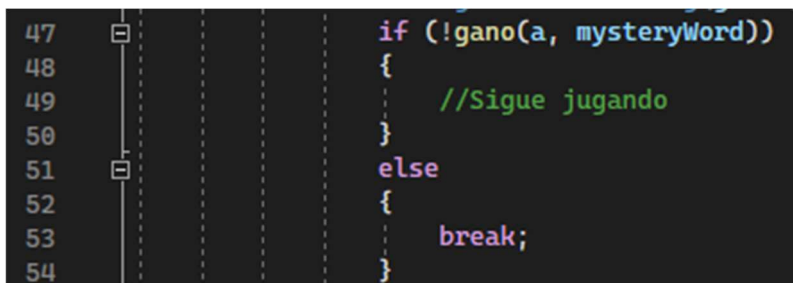
5.

Archivo: Hangman.cs

Método: PlayHangman

# Línea(s): 47-51

Tipo de error y descripción: "else" innecesario, se hace más complejo de leer el código sin necesidad.



```
47 if (!gano(a, mysteryWord))
48 {
49     //Sigue jugando
50 }
51 else
52 {
53     break;
54 }
```

The screenshot shows a code editor with a dark background. On the left, a vertical line marks line numbers 47 through 54. The code is an 'if-else' block. The 'if' condition is '!gano(a, mysteryWord)'. The 'if' block contains a single line of code: '//Sigue jugando'. The 'else' block contains a single line of code: 'break;'. The code is not formatted with consistent indentation, making it harder to read.

Se aumenta la complejidad del código sin necesidad aparte se revisa por un statement negativo sin necesidad.

```
47  if (gano(a, mysteryWord))
48  {
49      break;
50  }
```

Se elimina el else innecesario y así se reduce el código y se aumenta la legibilidad.

Link repo:

[https://github.com/BryanVillegas/LabsIngeSoft\\_B78383](https://github.com/BryanVillegas/LabsIngeSoft_B78383)

Conclusión:

No sabía acerca de todas las cosas que se podían tomar en cuenta dentro del CleanCode, reglas, etc. Tampoco sabía que existía el Git que ayuda con el CleanCode, lo cual es útil.

En ocasiones reconocer algunos errores se dificultó, porque normalmente los cometo al programar.

Se me hizo fácil resolver ciertos errores, ya fuera por experiencia o gracias a herramientas como Visual Studio.

Tardé aprox. 3 horas y media con el lab.