Laura Carratalá Castillo

Política de estilos para el desarrollo del proyecto Navaja Suiza

A continuación se expondrá una breve explicación de las normas de estilo que se han seguido para desarrollar las aplicaciones que forman parte del proyecto Navaja Suiza. Nuestro trabajo consta de 4 aplicaciones sencillas y debidamente compartimentadas en métodos, para hacer que sea más clara su comprensión. En ningún caso se han excedido los 4 niveles de anidamiento.

1. Declaración de variables:

Se ha realizado una declaración por línea, inicializando en la medida de lo posible durante la declaración. Se han empleado nombres autoexplicativos y fáciles de entender y siguiendo el convenio de nombres de estilo **caMel** (la primera letra de cada palabra es minúscula y las iniciales siguientes son en mayúscula)**.**

Para los bucles, se usan contadores locales de tipo i, j, etc, y se declaran e inicializan en el mismo bucle.

```
string cadenaTexto = "";

for (int i = 1; i <= numeroIntroducido; i++)
{
    if (i % 3 == 0 || i % 5 == 0)
    {
```

2. Declaración de constantes:

La declaración de la única constante empleada en las aplicaciones 3 y 4 (kTamanyo) se ha hecho siguiendo el estilo **caMel** y no utilizando caracteres problemáticos como la ñ.

```
const int kTamanyo = 10;
```

3. Nombres de métodos:

Los métodos se nombran al estilo **PasCal** (la primera letra de cada palabra es mayúscula y las iniciales siguientes son todas mayúsculas). Para nombrarlos se usan verbos indicativos de la función que se realiza en el método.

```
/// <returns>Cadena con los numeros del vector.</returns>
public string MostrarVector(int[] vectorNumeros)
{
    string cadenaTexto = "";

    for (int i = 0; i < KTamanyo; i++)
    {
        cadenaTexto = cadenaTexto + vectorNumeros[i] + ", ";
    }
    return cadenaTexto;
}
```

4. Nombres de clases:

Las clases de los formularios se han renombrado al estilo **PasCal**, usando sustantivos entendibles e identificativos de cada aplicación (FrmMain, FrmAplicacion1, etc).

```
public partial class FrmAplicacion3 : Form
```

5. Namespaces:

Los espacios de nombres no están duplicados. Tenemos un namespace general que abarca todas las aplicaciones llamado “NavajaSuiza” y después cada una de las aplicaciones tiene su propio namespace identificativo. Así, la aplicación 1 tiene como namespace “NavajaSuiza.Aplicación_1”, la aplicación 2 “NavajaSuiza.Aplicación_2”, la aplicación 3 “NavajaSuiza.Aplicación_3” y la aplicación 4 “NavajaSuiza.Aplicación_4”.

```
namespace NavajaSuiza
```

```
namespace NavajaSuiza.Aplicación_1
```

6. Indentación, llaves, espacios en blanco y líneas en blanco:

Se ha cuidado la indentación del código, no usando espacios y dividiendo las líneas demasiado largas a partir de comas, aunque en algún caso se ha tenido que dividir alguna línea con paréntesis.

```
for (int i = 0; i < kTamanyo; i++)
{
    do
    {
        elementoValido = int.TryParse(InputBox("Introduzca el elemento: " + i),
            out numeroIntroducido);
```

Se han respetado las llaves, no dejando ningún bucle sin ellas y sin incluir ningún comentario o instrucción en ninguna de ellas.

Se han dejado también espacios en blanco después de coma o punto y coma y alrededor de operadores.

También se han usado líneas en blanco para separar declaraciones de variables del resto del código y dentro de los métodos para separar las diferentes secciones lógicas.

```
for (int i = 1; i <= numeroIntroducido; i++)
{
    if (i % 3 == 0 || i % 5 == 0)
    {
        cadenaTexto = cadenaTexto + i + ", ";
    }
}
return cadenaTexto;
```

7. Comentarios:

Se han usado comentarios en todo el proyecto para explicar lo que hacen los métodos, clases, etc. y así realizar la documentación, utilizando todas las etiquetas pertinentes (summary, remarks, param, returns, etc.).

```
/// <summary>
/// Función que verifica si un número está comprendido entre el 1 y el 100.
/// de ser así, se considera válido.
/// </summary>
/// <remarks>Si el número no es válido, saltará un mensaje para que
/// se introduzca otro número.</remarks>
/// <param name="numeroIntroducido">Número que introduce el usuario.</param>
/// <returns>Si es válido o no.</returns>
public bool ValidarNumero(int numeroIntroducido)
{
    bool numeroValido = false;

    if (numeroIntroducido >= 1 && numeroIntroducido <= 100)
    {
        numeroValido = true;
    }
    return numeroValido;
}
```