Navaja Suiza 1.

1. Mi proyecto está alojado en el repositorio específico de “GITHUB” en el que he subiendo distintas versiones de mi proyecto en función de los cambios que he ido incorporando.

Aquí tenemos la multitud de commits que he hecho. A continuación explicaré en una tabla los cambios realizados en cada uno de ellos.

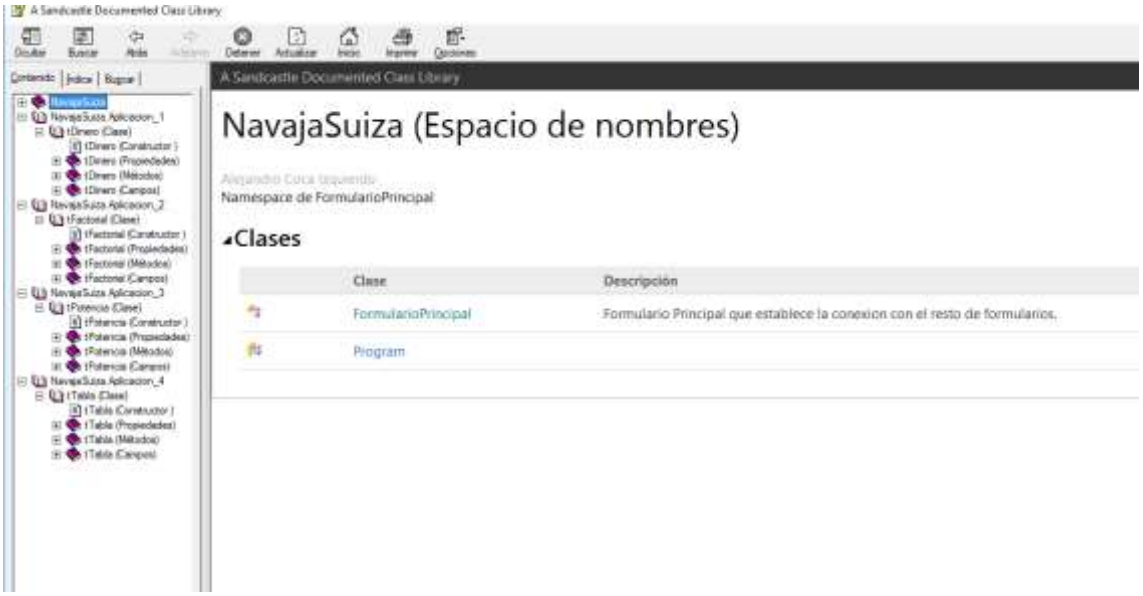
Commit	Descripción
Agregar .gitignore y .gitattributes.	Configuración del repositorio en Visual Studio.
Agregar archivos de proyecto.	Configuración del repositorio en Visual Studio.
Creación de Documentación.	Creación de la Documentación del proyecto NavajaSuiza.
He mejorado la funcionalidad de la aplicación 3 y he creado un caso de	Corrección del código de la aplicación 3 y creación de caso de

prueba de caja blanca en la aplicación 2.	prueba de caja blanca en la aplicación 2.
Caso de Prueba de caja blanca creado en la aplicación número 1.	Creación de caso de prueba de caja blanca en la aplicación número1.
Corrección de Código.	Corrección de errores en el código.
Creación de la carpeta Documentación.	He creado la carpeta documentación, he cambiado el nombre a los formularios y a algunos botones y funciones.
Creación de clases.	He creado las clases de los cuatro formularios y he modificado el código para que sea funcionarle.
Creación de un nuevo archivo ayuda	He creado un nuevo archivo ayuda y he cambiado la documentación de todo mi proyecto.
Cambios	He borrado los formularios y los he vuelto a añadir, he refactorizado, he limitado la longitud de las cajas de texto, he cambiado el nombre de las clases y he cambiado los métodos a static en dos de mis aplicaciones.
Creación de las Pruebas unitarias	He añadido a mi proyecto funciones try/catch, he añadido distintos tipos de excepciones y he creado excepciones personalizadas, además he creado las pruebas unitarias de mi proyecto.

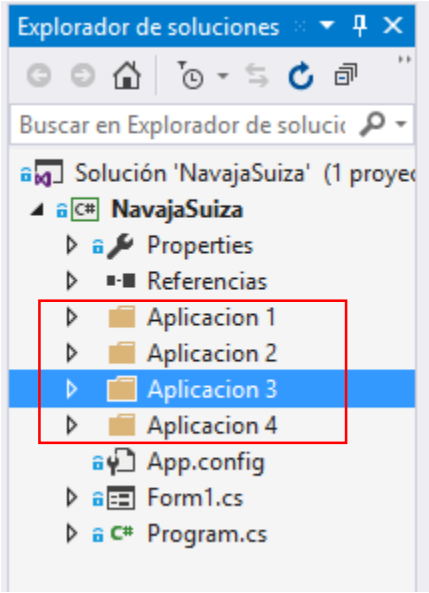
La URL del repositorio es la siguiente:

https://github.com/coca15783/NavajaSuiza

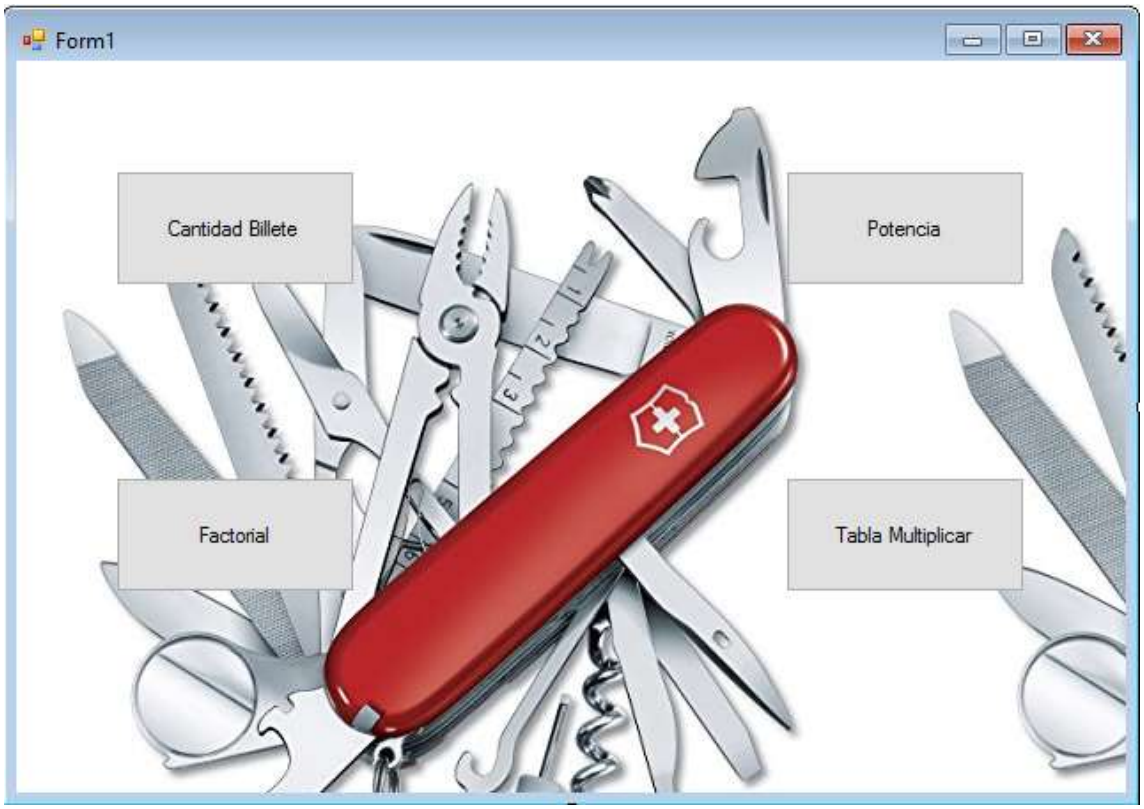
2. Una vez documentado todo el proyecto he generado un archivo de ayuda del desarrollo de la aplicación con sandcastle.



3. Las aplicaciones de mi proyecto están alojadas en distintas carpetas como muestro a continuación.



Para ejecutar estas aplicaciones, en el formulario principal he creado 4 botones, uno por cada aplicación, de forma que al pulsar los botones se ejecute una llamada al formulario de la aplicación.



Aquí tenemos el código donde hacemos las distintas llamadas.

```
        private void button2_Click(object sender, EventArgs e)
        {
            Aplicación_2.Formulario2 oFormulario = new Aplicación_2.Formulario2();
            oFormulario.ShowDialog();
        }

        /// <summary>
        /// Boton que llama a la aplicación 1
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 referencia | coca15783, Hace 3 días | 1 autor, 2 cambios
        private void button1_Click(object sender, EventArgs e)
        {
            Aplicación_1.Formulario1 oFormulario = new Aplicación_1.Formulario1();
            oFormulario.ShowDialog();
        }

        /// <summary>
        /// Boton que llama a la aplicación 3
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 referencia | coca15783, Hace 10 días | 1 autor, 1 cambio
        private void button3_Click(object sender, EventArgs e)
        {
            Aplicación_3.Formulario3 oFormulario = new Aplicación_3.Formulario3();
            oFormulario.ShowDialog();
        }
        /// <summary>
        /// Boton que llama a la aplicación 4
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 referencia | coca15783, Hace 10 días | 1 autor, 1 cambio
        private void button4_Click(object sender, EventArgs e)
        {
            Aplicación_4.Formulario4 oFormulario = new Aplicación_4.Formulario4();
            oFormulario.ShowDialog();
        }
    }
```

4. A lo largo del desarrollo de mi proyecto he utilizado una correcta política de estilos, el código está completamente tabulado, he utilizado funciones y las he llamado desde los botones, he utilizado espacios después de coma y punto y coma, también alrededor de los operadores. En los distintos bucles y if he utilizado espacios para separar la instrucción(for) de sus paréntesis, también he usado correctamente sus llaves. El código está completamente documentado además he utilizado una declaración por línea. En el convenio de nombres he utilizado el estilo PasCal.

5. Los modelos de caja blanca y caja negra de las aplicaciones actualizadas están en el proyecto dentro de la carpeta documentación.

Navaja Suiza 2.

2. Todas las actualizaciones de mi proyecto están subidas a gitHub. En la última actualización encontrarás la versión completa de mi proyecto.

3.He cambiado los nombres de las clases, variables y controles gráficos respecto a la versión de la segunda evaluación.

4. Toda la información de mi proyecto se introduce mediante cajas de texto.

```
bool resultado = Int32.TryParse(tDinero.Text, out dinero);
```

He limitado la longitud de las cajas de texto a 20 caracteres en mi proyecto.

```
    – referencias | coca15783, Hace 4 horas | 1 autor, 2 cambios
    private void textBox1_TextChanged(object sender, EventArgs e)
    {
        tDinero.MaxLength = 20;    //He limitado la longitud del textBox a 20 carácteres.
    }
}
```

Para validar la entrada de datos he utilizado la propiedad int.Tryparse que te devuelve un valor booleano en función de si el dato es entero o no, si el valor introducido en la caja de texto es entero devolverá un true por el contrario devolverá un false.

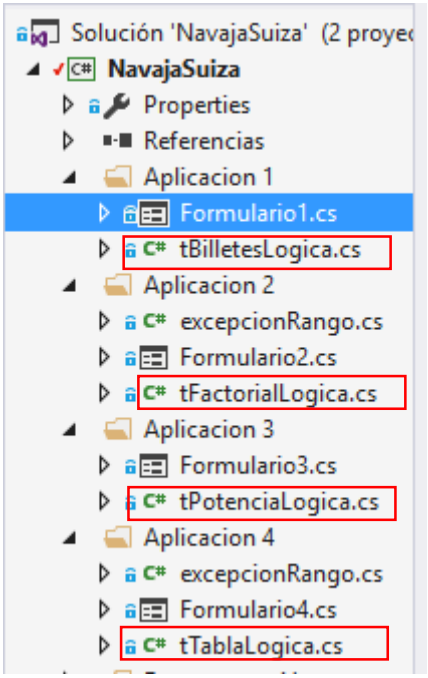
```
bool resultado = Int32.TryParse(tDinero.Text, out dinero);
```

Para comprobar si no se ha introducido ningún dato he utilizado la propiedad string.IsNullOrWhiteSpace si se cumple la siguiente propiedad se debe a que no se ha introducido ningún dato.

```
String.IsNullOrWhiteSpace((tDinero.Text)))
```

5.

Cada aplicación tendrá su clase correspondiente:



Todas las clases incluyen la palabra Lógica en su nombre.

Las cuatro clases están en el mismo namespace del subsitema.

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

namespace NavajaSuiza.Aplicacion_1
{
    /// <summary>
    /// Proporciona las propiedades y métodos necesarios para devolver la cantidad de billetes.
    /// <remarks>Recuerde utilizar esta clase cuando necesite modificar toda la información referente a Billetes</remarks>
    /// </summary>
    19 referencias | coca15783, Hace 4 horas | 1 autor, 2 cambios
    public class tBilletesLogica
    {
        private int mDinero;

        /// <summary>
        /// Constructor de la clase tDinero.
        /// </summary>
        9 referencias | 0/8 pasando | coca15783, Hace 3 días | 1 autor, 1 cambio
        public tBilletesLogica()
        {
            mDinero = 0;
        }
```

```
namespace NavajaSuiza.Aplicacion_2
{
    /// <summary>
    /// Proporciona las propiedades y métodos necesarios para devolver el factorial de un número.
    /// <remarks>Recuerde utilizar esta clase cuando necesite modificar toda la información referente a Factorial</remarks>
    /// </summary>
    21 referencias | coca15783, Hace 4 horas | 1 autor, 2 cambios
    public class tFactorialLogica
    {
        private long mNumero;

        /// <summary>
        /// Constructor de la clase tFactorial.
        /// </summary>
        10 referencias | 0/9 pasando | coca15783, Hace 3 días | 1 autor, 1 cambio
        public tFactorialLogica()
        {
            mNumero = 0;
        }

        /// <summary>
        /// Propiedad de la clase tFactorial que obtiene el numero.
        /// <value>
        /// Numero introducido.
        /// </value>
        /// </summary>
        10 referencias | 0/9 pasando | coca15783, Hace 4 horas | 1 autor, 2 cambios
```

En las clases de la aplicación uno y dos será necesario instanciar la clase para acceder a los métodos como en la imagen siguiente:

```
tBilletesLogica Dinero = new tBilletesLogica();
```

```
namespace NavajaSuiza.Aplicacion_3
{
    /// <summary>
    /// Proporciona las propiedades y métodos necesarios para devolver la potencia de un número.
    /// <remarks>Recuerde utilizar esta clase cuando necesite modificar toda la información referente a Potencia</remarks>
    /// </summary>
    5 referencias | coca15783, Hace 4 horas | 1 autor, 2 cambios
    public static class tPotenciaLogica
    {
        ///<summary>
        ///Funcion que calcula la potencia de un número.
        ///</summary>
        ///<return>
        ///Devuelve un número que corresponde con el resultado.
        ///</return>
        5 referencias | 0/4 pasando | coca15783, Hace 3 días | 1 autor, 1 cambio
```

```
namespace NavajaSuiza.Aplicacion_4
{
    /// <summary>
    /// Proporciona las propiedades y métodos necesarios para devolver la tabla de multiplicar de un número.
    /// <remarks>Recuerde utilizar esta clase cuando necesite modificar toda la información referente a Tabla</remarks>
    /// </summary>
    8 referencias | coca15783, Hace 3 días | 1 autor, 1 cambio
    public static class tTablaLogica
    {

        ///<summary>
        ///Funcion que calcula la tabla de multiplicar de un número.
        ///</summary>
        ///<return>
        ///Devuelve un texto que corresponde con el resultado.
        ///</return>

        8 referencias | 0/7 pasando | coca15783, Hace 3 días | 1 autor, 1 cambio
```

En cambio en la aplicación tres y cuantro no será necesario ya que la clase y los métodos serán staticos por lo que con poner el nombre de la clase ya podremos acceder a sus métodos como a continuación:

```
potencia = tPotenciaLogica.potencia(basE, exponente);
```

Las cuatro clases están en el mismo namespace del subsitema.

En las cuatro aplicaciones he utilizado excepciones para comunicar errores y try/catch para recuperarlos en los formularios.

```
try
{
    if (String.IsNullOrWhiteSpace((tBase.Text)) || String.IsNullOrWhiteSpace((tPotencia.Text)))
    {
        mensaje = "Introduce un número.";
    }
    else
    {
        if (resultado1 && resultado2)
        {
            mensaje = potencia.ToString();
        }

        if(resultado1 == false && resultado2 == false)
        {
            mensaje = "Has introducido un carácter";
        }

        if (resultado1 == true && resultado2 == false)
        {
            mensaje = "Has introducido un carácter";
        }

        if (resultado1 == false && resultado2 == true)
        {
            mensaje = "Has introducido un carácter";
        }

    }
    MessageBox.Show(mensaje);
}
catch (FormatException ex)
{
    MessageBox.Show("El formato no es el correcto:" + ex.Message);
}
catch (Exception ex)
{
    MessageBox.Show("Se ha producido un error:" + ex.Message);
}
```

Así en las cuatro aplicaciones he utilizado la excepción FormatException que te devuelve un mensaje de error si el formato no es correcto y la excepción Exception que te devuelve un mensaje de error si se ha producido cualquier otro error.

No he utilizado parámetros de tipo out ya que todos mis métodos devuelven un solo resultado.

He diseñado dos excepciones personalizadas para controlar un rango de números.

Dentro de la aplicación dos he creado la siguiente clase que representa la clase de una excepción por defecto.

```
using System.Text;
using System.Threading.Tasks;

namespace NavajaSuiza.Aplicacion_2
{
    /// <summary>
    /// Clase de excepción personalizada.
    /// </summary>
    [Serializable]
    6 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
    public class excepcionRango : System.Exception
    {
        0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
        public excepcionRango() { }
        1 referencia | coca15783, Hace 5 horas | 1 autor, 1 cambio
        public excepcionRango(string message) { }
        0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
        public excepcionRango(string message, Exception innerException) { }
        0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
        public excepcionRango(SerializationInfo info, StreamingContext context) { }
    }

}
```

En el formulario de la aplicación dos he creado el siguiente método que lanza una excepción si el número es mayor que 15.

```
public void rango(long numero)
{
    if(numero>15)
    {
        throw new excepcionRango("Número fuera de rango.");
    }
}
```

En la función mostrarFactorial he ejecutado el método rango y he llamado a la excepción excepcionRango en el catch.

```
//Prueba caja blanca
try
{
    rango(numero);

    if (String.IsNullOrWhiteSpace((tNumero.Text)))
    {
        mensaje = "Introduce un número.";
    }
    else
    {

        if (numero >= 0)
        {
            if (resultado)
            {
                mensaje = res.ToString();
            }
            else
            {
                mensaje = "Has introducido un carácter.";
            }
        }
        else
        {
           mensaje = "Introduce un número positivo.";
        }
    }
    MessageBox.Show(mensaje);
}
catch (excepcionRango ex)
{
    MessageBox.Show("Número fuera de rango: " + ex.Message);
}
catch (FormatException ex)
{
    MessageBox.Show("El formato no es el correcto:" + ex.Message);
}
catch (Exception ex)
{
```

En la aplicación cuatro he creado otra excepción de rango que es exactamente igual que la mostrada anteriormente solo que cambia el rango de números en el método rango.

6. Los modelos de caja blanca y caja negra de las aplicaciones actualizadas están en el proyecto dentro de la carpeta documentación.

7.

En las pruebas unitarias tan solo vamos a poder probar valores enteros ya que los datos de entrada de mis cuatro aplicaciones son enteros por lo que

los caracteres, números, espacios en blanco y decimales quedan totalmente descartados.

Esta tabla corresponde a las pruebas de caja negra de la aplicación uno.

Caso de Prueba	Descripción.	Fecha	Datos de entrada	Resultado esperado
1	Voy a introducir un carácter minúscula.	16/05/2017	a	Mensaje
2	Voy a introducir un carácter mayúscula.	16/05/2017	A	Mensaje
3	Voy a introducir un número decimal.	16/05/2017	2.2	Error
4	Voy a introducir un número negativo.	16/05/2017	-1	Mensaje
5	Voy a introducir el 0.	16/05/2017	0	Correcto
6	Voy a introducir un número positivo.	16/05/2017	25	Correcto
7	Voy a introducir un asterisco.	16/05/2017	*	Error
8	No voy a introducir nada	16/05/2017	“”	Mensaje
9	Voy a introducir un uno.	16/05/2017	1	Correcto

Esta prueba unitaria corresponde al caso de prueba 4

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp1ValorNegativo1()
{
    tBilletesLogica Dinero = new tBilletesLogica();

    int dineroIntroducido;
    dineroIntroducido = -1;
    Dinero.Dinero = dineroIntroducido;

    int billetes10000E = 0;
    int billetes5000E = 0;
    int billetes2000E = 0;
    int billetes100E = 0;
    int billetes25E = 0;

    int billetes10000Ob = Dinero.billetes10000();
    int billetes5000Ob = Dinero.billetes5000();
    int billetes2000Ob = Dinero.billetes2000();
    int billetes100Ob = Dinero.billetes100();
    int billetes25Ob = Dinero.billetes25();


    Assert.AreEqual(billetes10000E, billetes10000Ob);
    Assert.AreEqual(billetes5000E, billetes5000Ob);
    Assert.AreEqual(billetes2000E, billetes2000Ob);
    Assert.AreEqual(billetes100E, billetes100Ob);
    Assert.AreEqual(billetes25E, billetes25Ob);
}
```

Esta prueba unitaria corresponde al caso de prueba 5

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp1ValorCero1()
{
    tBilletesLogica Dinero = new tBilletesLogica();

    int dineroIntroducido;
    dineroIntroducido = 0;
    Dinero.Dinero = dineroIntroducido;

    int billetes10000E = 0;
    int billetes5000E = 0;
    int billetes2000E = 0;
    int billetes100E = 0;
    int billetes25E = 0;

    int billetes10000Ob = Dinero.billetes10000();
    int billetes5000Ob = Dinero.billetes5000();
    int billetes2000Ob = Dinero.billetes2000();
    int billetes100Ob = Dinero.billetes100();
    int billetes25Ob = Dinero.billetes25();


    Assert.AreEqual(billetes10000E, billetes10000Ob);
    Assert.AreEqual(billetes5000E, billetes5000Ob);
    Assert.AreEqual(billetes2000E, billetes2000Ob);
    Assert.AreEqual(billetes100E, billetes100Ob);
    Assert.AreEqual(billetes25E, billetes25Ob);
}
```

Esta prueba unitaria corresponde al caso de prueba 6

```
[TestMethod]
| 0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp1Valor25Uno()
{
    tBilletesLogica Dinero = new tBilletesLogica();

    int dineroIntroducido;
    dineroIntroducido = 25;
    Dinero.Dinero = dineroIntroducido;

    int billetes10000E = 0;
    int billetes5000E = 0;
    int billetes2000E = 0;
    int billetes100E = 0;
    int billetes25E = 1;

    int billetes10000Ob = Dinero.billetes10000();
    int billetes5000Ob = Dinero.billetes5000();
    int billetes2000Ob = Dinero.billetes2000();
    int billetes100Ob = Dinero.billetes100();
    int billetes25Ob = Dinero.billetes25();


    Assert.AreEqual(billetes10000E, billetes10000Ob);
    Assert.AreEqual(billetes5000E, billetes5000Ob);
    Assert.AreEqual(billetes2000E, billetes2000Ob);
    Assert.AreEqual(billetes100E, billetes100Ob);
    Assert.AreEqual(billetes25E, billetes25Ob);
}
```

Esta prueba unitaria corresponde con el caso de prueba 9

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp1Valor1Uno()
{
    tBilletesLogica Dinero = new tBilletesLogica();

    int dineroIntroducido;
    dineroIntroducido = 1;
    Dinero.Dinero = dineroIntroducido;

    int billetes10000E = 0;
    int billetes5000E = 0;
    int billetes2000E = 0;
    int billetes100E = 0;
    int billetes25E = 0;

    int billetes10000Ob = Dinero.billetes10000();
    int billetes5000Ob = Dinero.billetes5000();
    int billetes2000Ob = Dinero.billetes2000();
    int billetes100Ob = Dinero.billetes100();
    int billetes25Ob = Dinero.billetes25();


    Assert.AreEqual(billetes10000E, billetes10000Ob);
    Assert.AreEqual(billetes5000E, billetes5000Ob);
    Assert.AreEqual(billetes2000E, billetes2000Ob);
    Assert.AreEqual(billetes100E, billetes100Ob);
    Assert.AreEqual(billetes25E, billetes25Ob);
}
```

Esta tabla corresponde a las pruebas de caja negra de la aplicación dos.

Caso de Prueba	Descripción.	Fecha	Datos de entrada	Resultado esperado
1	Voy a introducir un carácter minúscula.	16/05/2017	a	Mensaje
2	Voy a introducir un carácter mayúscula.	16/05/2017	A	Mensaje
3	Voy a introducir un número decimal.	16/05/2017	2.2	Error
4	Voy a introducir un número negativo.	16/05/2017	-1	Mensaje
5	Voy a introducir el 0.	16/05/2017	0	Correcto
6	Voy a introducir un número positivo.	16/05/2017	25	Error
7	Voy a introducir un asterisco.	16/05/2017	*	Error

8	No voy a introducir nada	16/05/2017	“”	Mensaje
9	Voy a introducir un uno.	16/05/2017	1	Correcto
10	Voy a introducir un catorce.	16/05/2017	14	Correcto
11	Voy a introducir un 15	16/05/2017	15	Correcto
12	Voy a introducir un 16	16/05/2017	16	Error

Esta prueba unitaria corresponde con el caso de prueba 4

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp2FactorialValorNegativoUno()
{
    tFactorialLogica Factorial = new tFactorialLogica();

    long numeroIntroducido;
    numeroIntroducido = -1;
    Factorial.Numero = numeroIntroducido;

    long resultadoEsperado;
    long resultadoObtenido;

    resultadoObtenido = Factorial.factorial();

    resultadoEsperado = 1;

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 5

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp2FactorialValorCeroUno()
{
    tFactorialLogica Factorial = new tFactorialLogica();

    long numeroIntroducido;
    numeroIntroducido = 0;
    Factorial.Numero = numeroIntroducido;

    long resultadoEsperado;
    long resultadoObtenido;

    resultadoObtenido = Factorial.factorial();

    resultadoEsperado = 1;

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 9

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp2FactorialValorUno()
{
    tFactorialLogica Factorial = new tFactorialLogica();

    long numeroIntroducido;
    numeroIntroducido = 1;
    Factorial.Numero = numeroIntroducido;

    long resultadoEsperado;
    long resultadoObtenido;

    resultadoObtenido = Factorial.factorial();

    resultadoEsperado = 1;

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 10

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp2FactorialValor14()
{
    tFactorialLogica Factorial = new tFactorialLogica();

    long numeroIntroducido;
    numeroIntroducido = 14;
    Factorial.Numero = numeroIntroducido;

    long resultadoEsperado;
    long resultadoObtenido;

    resultadoObtenido = Factorial.factorial();

    resultadoEsperado = 87178291200;

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 11

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp2FactorialValor15()
{
    tFactorialLogica Factorial = new tFactorialLogica();

    long numeroIntroducido;
    numeroIntroducido = 15;
    Factorial.Numero = numeroIntroducido;

    long resultadoEsperado;
    long resultadoObtenido;

    resultadoObtenido = Factorial.factorial();

    resultadoEsperado = 1307674368000;

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 12

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp2FactorialValor16()
{
    tFactorialLogica Factorial = new tFactorialLogica();
    Formulario2 Formulario = new Formulario2();

    long numeroIntroducido;
    numeroIntroducido = 16;
    Factorial.Numero = numeroIntroducido;

    long resultadoEsperado;
    long resultadoObtenido;

    resultadoObtenido = Factorial.factorial();
    Formulario.rango(numeroIntroducido);

    resultadoEsperado = 20922789888000;

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta tabla corresponde a las pruebas de caja negra de la aplicación 3,

Caso de Prueba	Descripción.	Fecha	Datos de entrada	Resultado esperado
1	Voy a introducir un carácter minúscula.	16/05/2017	Base: a	Mensaje
2	Voy a introducir un carácter mayúscula.	16/05/2017	Base: A	Mensaje
3	Voy a introducir un número decimal.	16/05/2017	2.2	Error
4	Voy a introducir una base positiva y un exponente positivo.	16/05/2017	Base: 0 Exponente: 0	Correcto
5	Voy a introducir una base negativa y un exponente negativo.	16/05/2017	Base : -1 Exponente : -1	Correcto
6	Voy a introducir una base positiva y un exponente negativo.	16/05/2017	Base : 0 Exponente : -1	Correcto

7	Voy a introducir una base negativa y un exponente positivo.	16/05/2017	Base : -1 Exponente : 0	Correcto
8	Voy a introducir un asterisco.	16/05/2017	*	Error
9	No voy a introducir nada	16/05/2017	“”	Mensaje
10	Voy a introducir un número como base y un carácter como exponente.	16/05/2017	Base: 1 Exponente: d	Mensaje
11	Voy a introducir un carácter como base y un número como exponente.	16/05/2017	Base: d Exponente: 1	Mensaje
12	Voy a introducir un carácter como base y un carácter como exponente.	16/05/2017	Base: d Exponente: d	Mensaje

Esta prueba unitaria corresponde con el caso de prueba 4

```
[TestClass]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public class PruebasUnitariasApp3
{
    [TestMethod]
    0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
    public void PruebaApp3PotenciaBaseyExponentePositivo()
    {

        int baseIntroducido;
        baseIntroducido = 0;

        int exponenteIntroducido;
        exponenteIntroducido = 0;

        long resultadoEsperado;
        long resultadoObtenido;

        resultadoObtenido = tPotenciaLogica.potencia(baseIntroducido,exponenteIntroducido);

        resultadoEsperado = 0;

        Assert.AreEqual(resultadoEsperado, resultadoObtenido);

    }
```

Esta prueba unitaria corresponde con el caso de prueba 5

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp3BaseyExponenteNegativo()
{

    int baseIntroducido;
    baseIntroducido = -1;

    int exponenteIntroducido;
    exponenteIntroducido = -1;

    long resultadoEsperado;
    long resultadoObtenido;

    resultadoObtenido = tPotenciaLogica.potencia(baseIntroducido, exponenteIntroducido);

    resultadoEsperado = 1;

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 6

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp3PotenciaBasePositivayExponenteNegativo()
{

    int baseIntroducido;
    baseIntroducido = 0;

    int exponenteIntroducido;
    exponenteIntroducido = -1;

    long resultadoEsperado;
    long resultadoObtenido;

    resultadoObtenido = tPotenciaLogica.potencia(baseIntroducido, exponenteIntroducido);

    resultadoEsperado = 0;

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);


}
```

Esta prueba unitaria corresponde con el caso de prueba 7

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp3PotenciaBaseNegativayExponentePositivo()
{

    int baseIntroducido;
    baseIntroducido = -1;

    int exponenteIntroducido;
    exponenteIntroducido = 0;

    long resultadoEsperado;
    long resultadoObtenido;

    resultadoObtenido = tPotenciaLogica.potencia(baseIntroducido, exponenteIntroducido);

    resultadoEsperado = 0;

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);


}
```

Esta tabla corresponde a las pruebas de caja negra de la aplicación 4.

Caso de Prueba	Descripción.	Fecha	Datos de entrada	Resultado esperado
1	Voy a introducir un carácter minúscula.	16/05/2017	a	Mensaje
2	Voy a introducir un carácter mayúscula.	16/05/2017	A	Mensaje
3	Voy a introducir un número decimal.	16/05/2017	2.2	Error
4	Voy a introducir un número negativo.	16/05/2017	-1	Error
5	Voy a introducir el 0.	16/05/2017	0	Correcto
6	Voy a introducir un número positivo.	16/05/2017	25	Correcto
7	Voy a introducir un asterisco.	16/05/2017	*	Error
8	No voy a introducir nada	16/05/2017	“”	Mensaje
9	Voy a introducir un uno.	16/05/2017	1	Correcto
10	Voy a introducir un catorce.	16/05/2017	99	Correcto
11	Voy a introducir un 15	16/05/2017	100	Correcto

12	Voy a introducir un 16	16/05/2017	101	Error

Esta prueba unitaria corresponde con el caso de prueba 5

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp4TablaCero()
{

    int numeroIntroducido;
    numeroIntroducido = 0;

    string resultadoEsperado;
    string resultadoObtenido;

    resultadoObtenido = tTablaLogica.tabla(numeroIntroducido);

    resultadoEsperado = "";

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 9

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp4TablaUno()
{

    int numeroIntroducido;
    numeroIntroducido = 1;

    string resultadoEsperado;
    string resultadoObtenido;

    resultadoObtenido = tTablaLogica.tabla(numeroIntroducido);

    resultadoEsperado = "";

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 10

```
[TestMethod]
| 0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp4Tabla99()
{

    int numeroIntroducido;
    numeroIntroducido = 99;

    string resultadoEsperado;
    string resultadoObtenido;

    resultadoObtenido = tTablaLogica.tabla(numeroIntroducido);

    resultadoEsperado = "";

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 11

```
[TestMethod]
| 0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp4Tabla100()
{

    int numeroIntroducido;
    numeroIntroducido = 100;

    string resultadoEsperado;
    string resultadoObtenido;

    resultadoObtenido = tTablaLogica.tabla(numeroIntroducido);

    resultadoEsperado = "";

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 12

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp4FueradeRango1()
{

    int numeroIntroducido;
    numeroIntroducido = 101;

    string resultadoEsperado;
    string resultadoObtenido;

    resultadoObtenido = tTablaLogica.tabla(numeroIntroducido);
    Formulario4.rango(numeroIntroducido);

    resultadoEsperado = "";

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```

Esta prueba unitaria corresponde con el caso de prueba 4

```
[TestMethod]
0 referencias | coca15783, Hace 5 horas | 1 autor, 1 cambio
public void PruebaApp4FueradeRango2()
{

    int numeroIntroducido;
    numeroIntroducido = -1;

    string resultadoEsperado;
    string resultadoObtenido;

    resultadoObtenido = tTablaLogica.tabla(numeroIntroducido);
    Formulario4.rango(numeroIntroducido);

    resultadoEsperado = "";

    Assert.AreEqual(resultadoEsperado, resultadoObtenido);

}
```