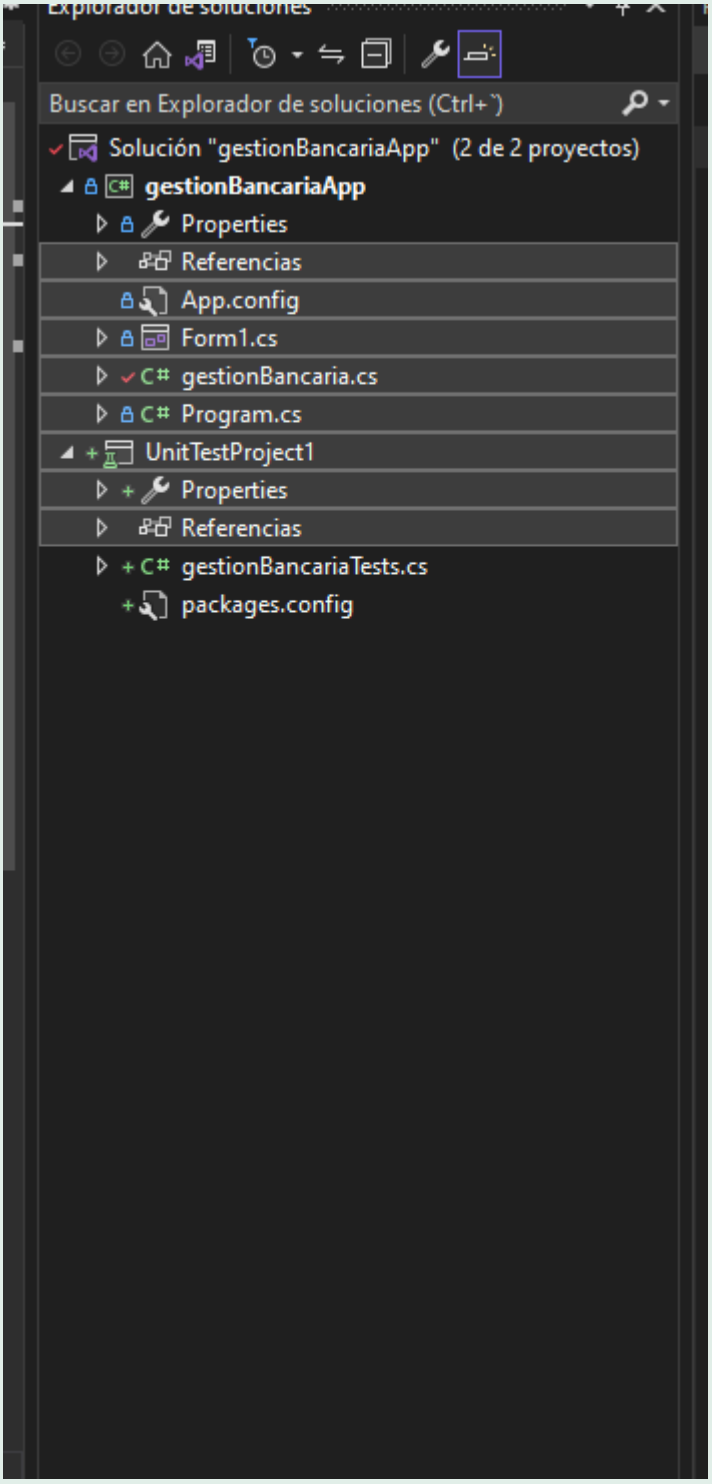
Leo Coves Guzmán 1DAW N

PRACTICA 4-3

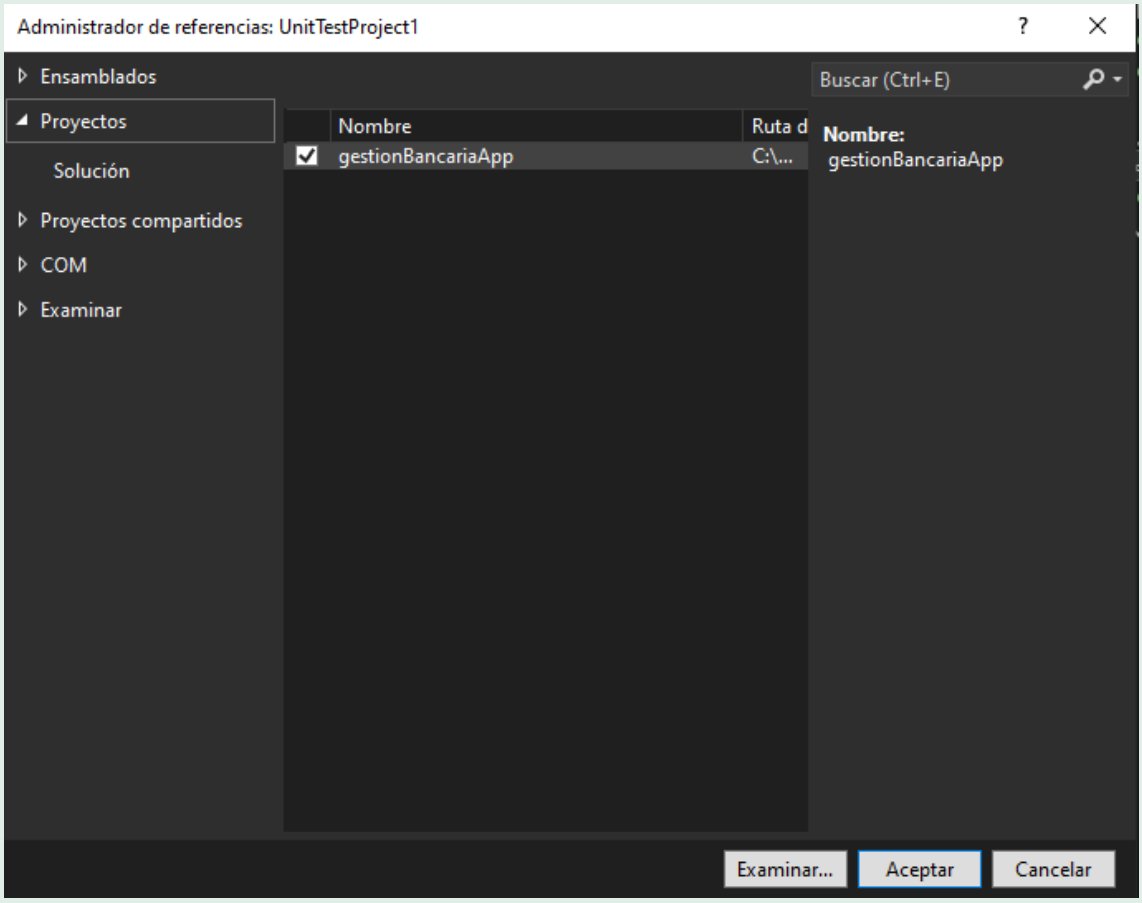
ENTORNOS DE DESARROLLO

https://github.com/leocovess/DAM_PRUEBAS_UNITARIAS_2223

Prueba unitaria ya creada

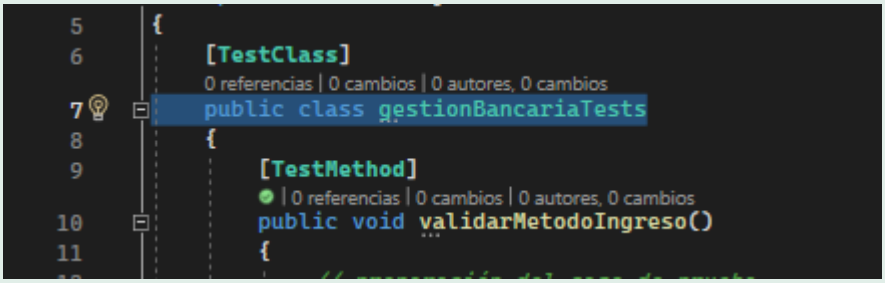


Leo Coves Guzmán 1DAW N



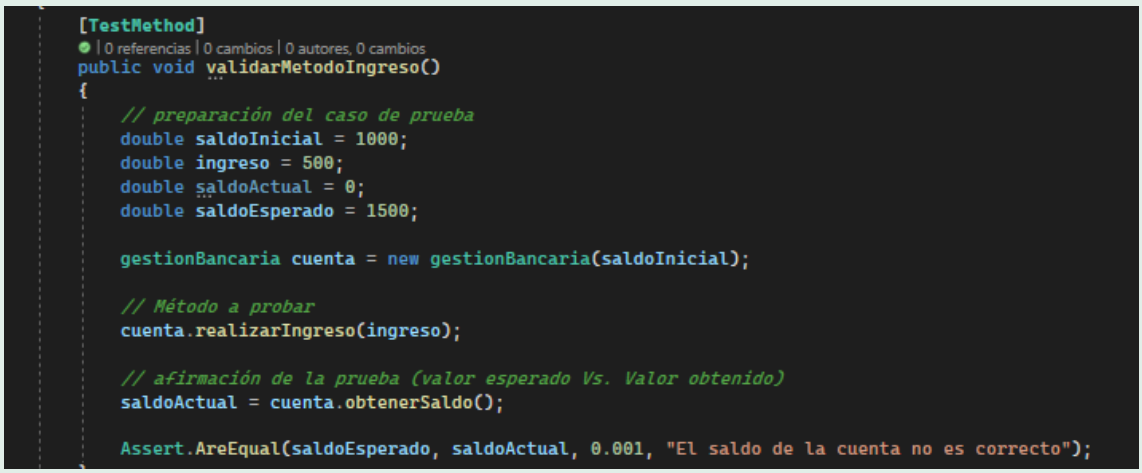
Agregamos referencia gestiónBancariaApp

4. Creamos la clase de prueba



```
{
    [TestClass]
    0 referencias | 0 cambios | 0 autores, 0 cambios
    public class gestionBancariaTests
    {
        [TestMethod]
        | 0 referencias | 0 cambios | 0 autores, 0 cambios
        public void validarMetodoIngreso()
        {
```

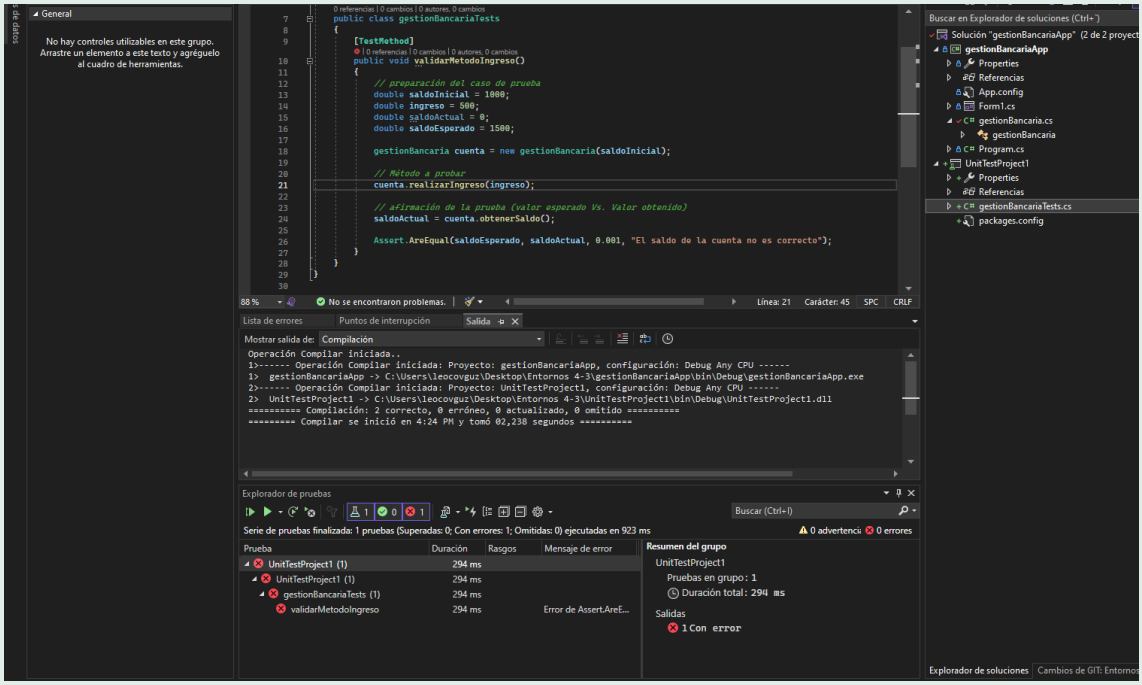
5. Creamos el primer método de prueba



```
[TestMethod]
| 0 referencias | 0 cambios | 0 autores, 0 cambios
public void validarMetodoIngreso()
{
    // preparación del caso de prueba
    double saldoInicial = 1000;
    double ingreso = 500;
    double saldoActual = 0;
    double saldoEsperado = 1500;

    gestionBancaria cuenta = new gestionBancaria(saldoInicial);

    // Método a probar
    cuenta.realizarIngreso(ingreso);

    // afirmación de la prueba (valor esperado Vs. Valor obtenido)
    saldoActual = cuenta.obtenerSaldo();

    Assert.AreEqual(saldoEsperado, saldoActual, 0.001, "El saldo de la cuenta no es correcto");
```

6.Compilar y ejecutar la prueba

En este caso nos da error y al depurar pruebas nos lleva a la función realizarIngreso()



Comprobamos que el error esta en esa función

```
47      }
48
        2 referencias | 0/1 pasando | 0 cambios | 0 autores, 0 cambios
49      public void realizarIngreso(double cantidad)
50      {
51
52          if (cantidad < 0)
53          {
54              //mostrarError(ERR_CANTIDAD_INDICADA_NEGATIVA);
55              throw new ArgumentOutOfRangeException("Cantidad negativa");
56          }
57          else
58          {
59              if (cantidad > 0)
60                  saldo -= cantidad;//
61          }
62
63      }
```

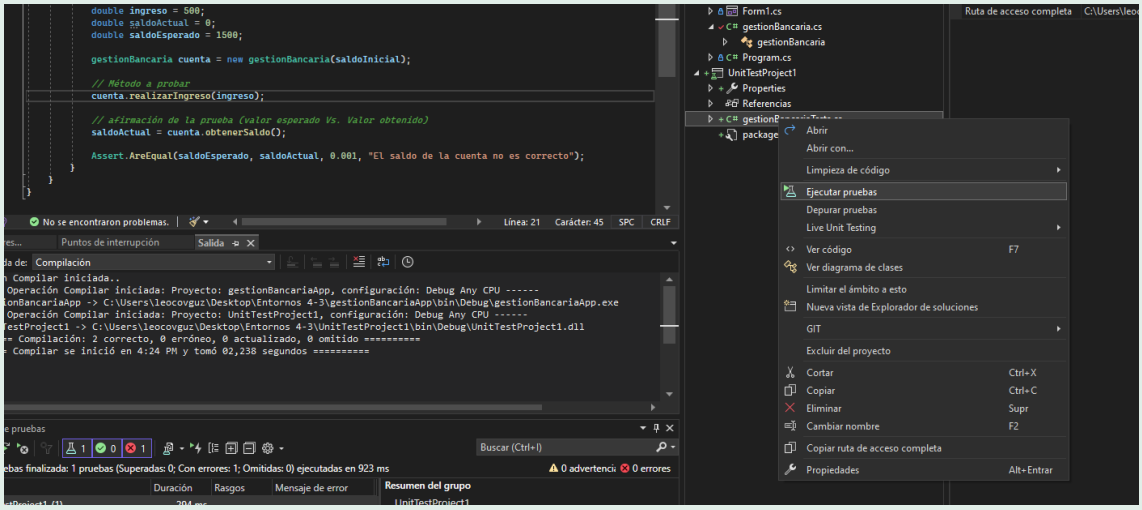
Tenia un fallo en la línea 60 en el signo negativo, pues lo cambiamos

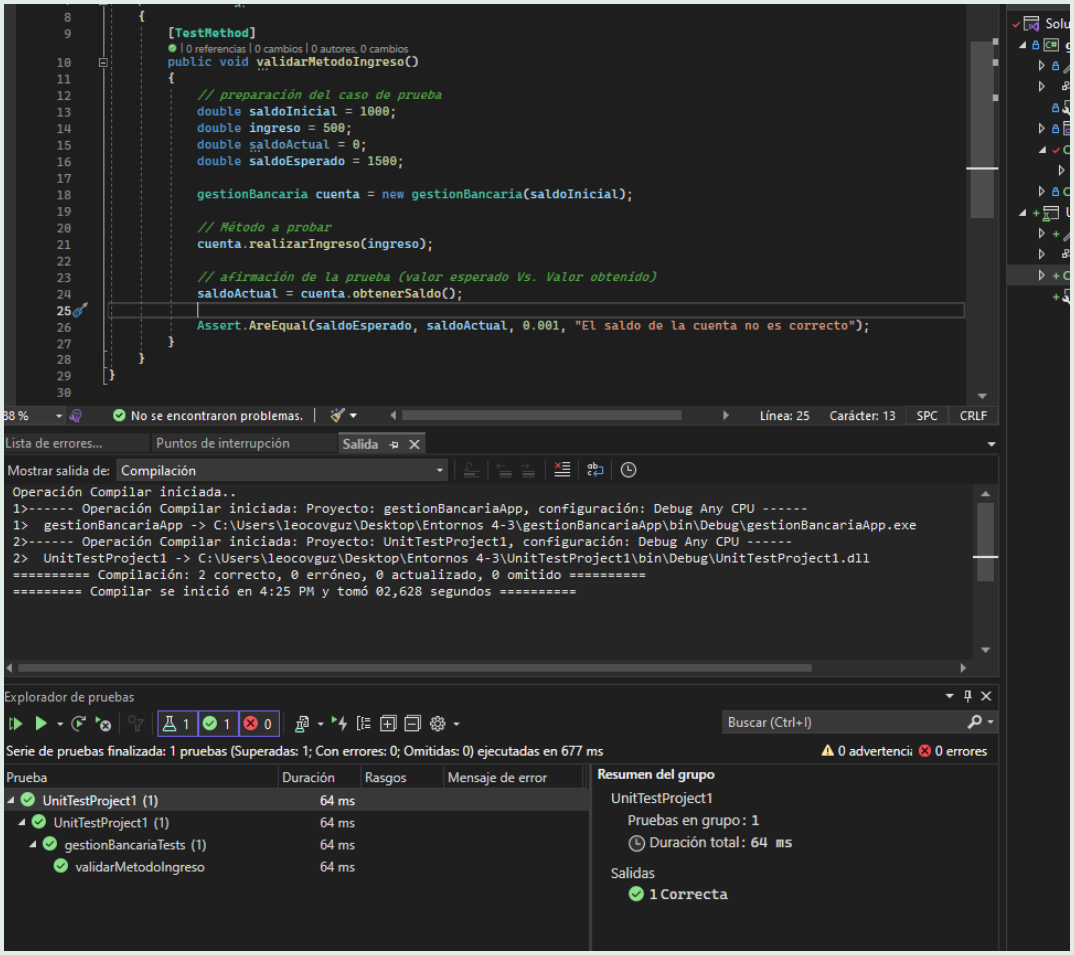
```
{

    if (cantidad < 0)
    {
        //mostrarError(ERR_CANTIDAD_INDICADA_NEGATIVA);
        throw new ArgumentOutOfRangeException("Cantidad negativa");
    }
    else
    {
        if (cantidad > 0)
            saldo += cantidad;//
    }
```

Lo sustituimos por un “+” ya que al ingresar sumas el ingreso con el saldo de la cuenta Bancaria. (Era un error intencionado según el PDF)

Volvemos a ejecutar la prueba y vemos que no da error

Click derecho en gestionBancariaTests, ejecutar pruebas y abajo saldrán el explorador de pruebas. En caso de que no salgo, pulsamos en la barra superior en Ver > click en 'Explorador de pruebas'





Leo Coves Guzmán 1DAW N

Punto 7 (clases de equivalencia y valores limites)

Clases de equivalencia → Ingreso

Caso 1: Que el ingreso sea > 0 (valor positivo) = true

```
[TestMethod]
0 referencias | 0 cambios | 0 autores, 0 cambios
public void validarMetodoIngreso()
{
    // preparación del caso de prueba
    double saldoInicial = 1000;
    double ingreso = 500;
    double saldoActual = 0;
    double saldoEsperado = 1500;

    gestionBancaria cuenta = new gestionBancaria(saldoInicial);

    // Método a probar
    cuenta.realizarIngreso(ingreso);

    // afirmación de la prueba (valor esperado Vs. Valor obtenido)
    saldoActual = cuenta.obtenerSaldo();

    Assert.AreEqual(saldoEsperado, saldoActual, 0.001, "El saldo de la cuenta no es correcto");
```

Caso 2: Que el ingreso sea < 0 (valor negativo) = false

```
0 referencias | 0 cambios | 0 autores, 0 cambios
public void validarMetodoIngresoNegativo()
{
    // preparación del caso de prueba
    double saldoInicial = 1000;
    double ingreso = -500;
    double saldoActual = 0;
    double saldoEsperado = 1500;

    gestionBancaria cuenta = new gestionBancaria(saldoInicial);

    // Método a probar
    cuenta.realizarIngreso(ingreso);

    // afirmación de la prueba (valor esperado Vs. Valor obtenido)
    saldoActual = cuenta.obtenerSaldo();

    Assert.AreEqual(saldoEsperado, saldoActual, 0.001, "El saldo de la cuenta no es correcto");
}
```

Caso 3: Que el saldo Inicial sea negativo = true

```
[TestMethod]
0 referencias | 0 cambios | 0 autores, 0 cambios
public void validarMetodoIngresoSaldoInicialNegativo()
{
    // preparación del caso de prueba
    double saldoInicial = -1000;
    double ingreso = 500;
    double saldoActual = 0;
    double saldoEsperado = -500;

    gestionBancaria cuenta = new gestionBancaria(saldoInicial);

    // Método a probar
    cuenta.realizarIngreso(ingreso);

    // afirmación de la prueba (valor esperado Vs. Valor obtenido)
    saldoActual = cuenta.obtenerSaldo();

    Assert.AreEqual(saldoEsperado, saldoActual, 0.001, "El saldo de la cuenta no es correcto");
}
```

Caso 4: Que el ingreso sea 0 = false

```
[TestMethod]
0 referencias | 0 cambios | 0 autores, 0 cambios
public void validarMetodoIngresoSiendo0()
{
    // preparación del caso de prueba
    double saldoInicial = 1000;
    double ingreso = 0;
    double saldoActual = 0;
    double saldoEsperado = 1000;

    gestionBancaria cuenta = new gestionBancaria(saldoInicial);

    // Método a probar
    cuenta.realizarIngreso(ingreso);

    // afirmación de la prueba (valor esperado Vs. Valor obtenido)
    saldoActual = cuenta.obtenerSaldo();

    Assert.AreEqual(saldoEsperado, saldoActual, 0.001, "El saldo de la cuenta no es correcto");
}
```

Caso 5: Que el ingreso sea NULL = false

Sin imagen

<u>Clases de equivalencia → Reintegro</u>

Caso 1: Que el reintegro sea < saldo Inicial = true

```
[TestMethod]
0 referencias | 0 cambios | 0 autores, 0 cambios
public void validarMetodoReintegro()
{
    // preparación del caso de prueba
    double saldoInicial = 1000;
    double reintegro = 500;
    double saldoActual = 0;
    double saldoEsperado = 500;

    gestionBancaria cuenta = new gestionBancaria(saldoInicial);

    // Método a probar
    cuenta.realizarReintegro(reintegro);

    // afirmación de la prueba (valor esperado Vs. Valor obtenido)
    saldoActual = cuenta.obtenerSaldo();

    Assert.AreEqual(saldoEsperado, saldoActual, 0.001, "El saldo de la cuenta no es correcto");
}
```

Caso 2: Que el reintegro sea > saldo Inicial = false

```
[TestMethod]
0 referencias | 0 cambios | 0 autores, 0 cambios
public void validarMetodoReintegroNegativo()
{
    // preparación del caso de prueba
    double saldoInicial = 500;
    double reintegro = 1000;
    double saldoActual = 0;
    double saldoEsperado = -500;

    gestionBancaria cuenta = new gestionBancaria(saldoInicial);

    // Método a probar
    cuenta.realizarReintegro(reintegro);

    // afirmación de la prueba (valor esperado Vs. Valor obtenido)
    saldoActual = cuenta.obtenerSaldo();

    Assert.AreEqual(saldoEsperado, saldoActual, 0.001, "El saldo de la cuenta no es correcto");
}
```

Caso 3: Que el saldo Inicial sea negativo = false

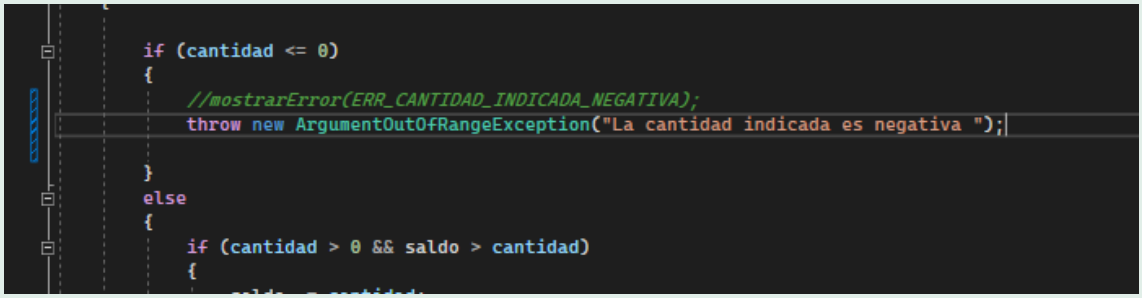
Caso 4: Que el reintegro sea negativo = La cantidad indicada es negativa

Caso 5: Que el reintegro sea 0 = false

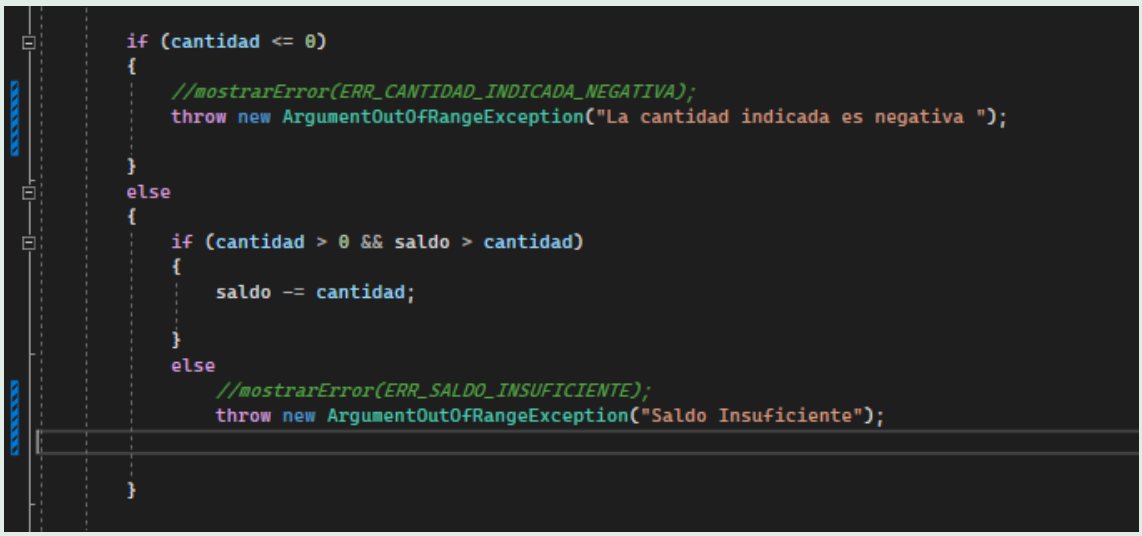
Caso 6: Que el reintegro sea NULL = false

Valores Limite en el Ingreso es que el valor sea un numero positivo y en el Reintegro que el valor sea menor que el saldo Inicial que hay dentro de la cuenta bancaria

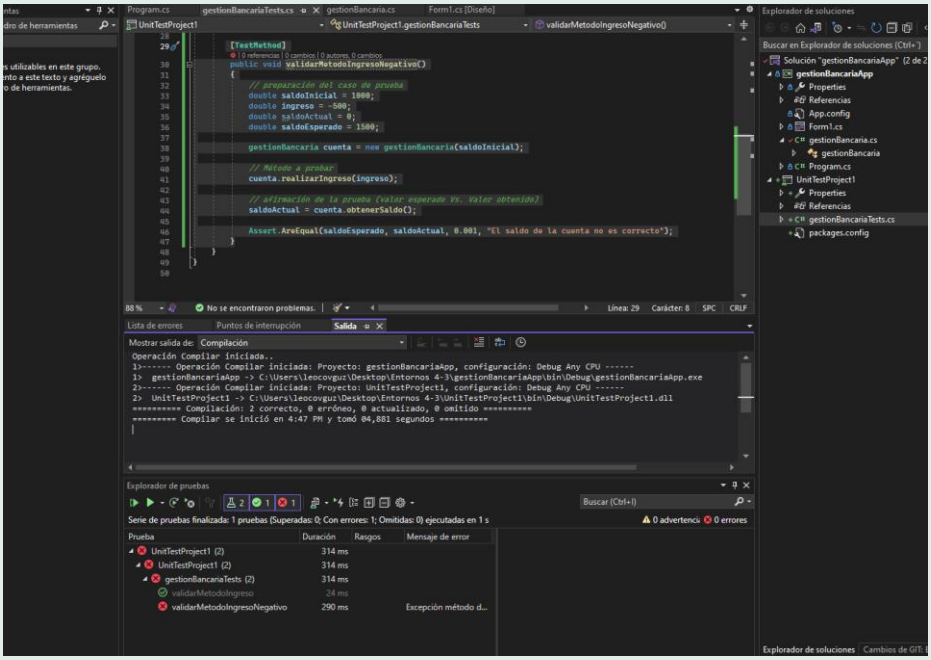
Punto 8 Pruebas que esperan excepciones



Excepciones cambiadas correctamente por ArgumentOutOfRangeException



Leo Coves Guzmán 1DAW N



```
[TestMethod]
[ExpectedException(typeof(ArgumentOutOfRangeException))]
0 referencias | 0 cambios | 0 autores, 0 cambios
public void validarMetodoIngresoNegativo()
```

Colocamos esa excepción para que se pueda realizar el que sea negativo el ingreso