Reporte Sprint #0

Instrucciones

Objetivos

- Tomar decisiones sobre el proyecto de desarrollo de software SOS.
- Aprender pruebas unitarias y programación de GUI en el lenguaje de tu elección.

Entregables y políticas de calificación

Lean el documento "descripción del Proyecto 3S2" cuidadosamente y toma las decisiones para el desarrollo del software.

Usen el siguiente template para completar tu reporte.

1. Decisiones claves para el proyecto SOS (2 puntos)

Lenguaje de programación orientado a objetos	C#
Librería GUI (recomendable)	Windows Forms
IDE (Integrated Development Environment)	Visual Studio Community 2022
Framework xUnit (JUnit for Java por ejemplo)	MsTest (.NET Framework)
Guía de estilo de programación (debe ser leído con cuidado)	Guía de estilo de C# Microsoft (Convensiones)
Sitio de alojamiento del proyecto	https://github.com/NaoDekoNeko/CC-3S2
Otras decisiones si procede	-----

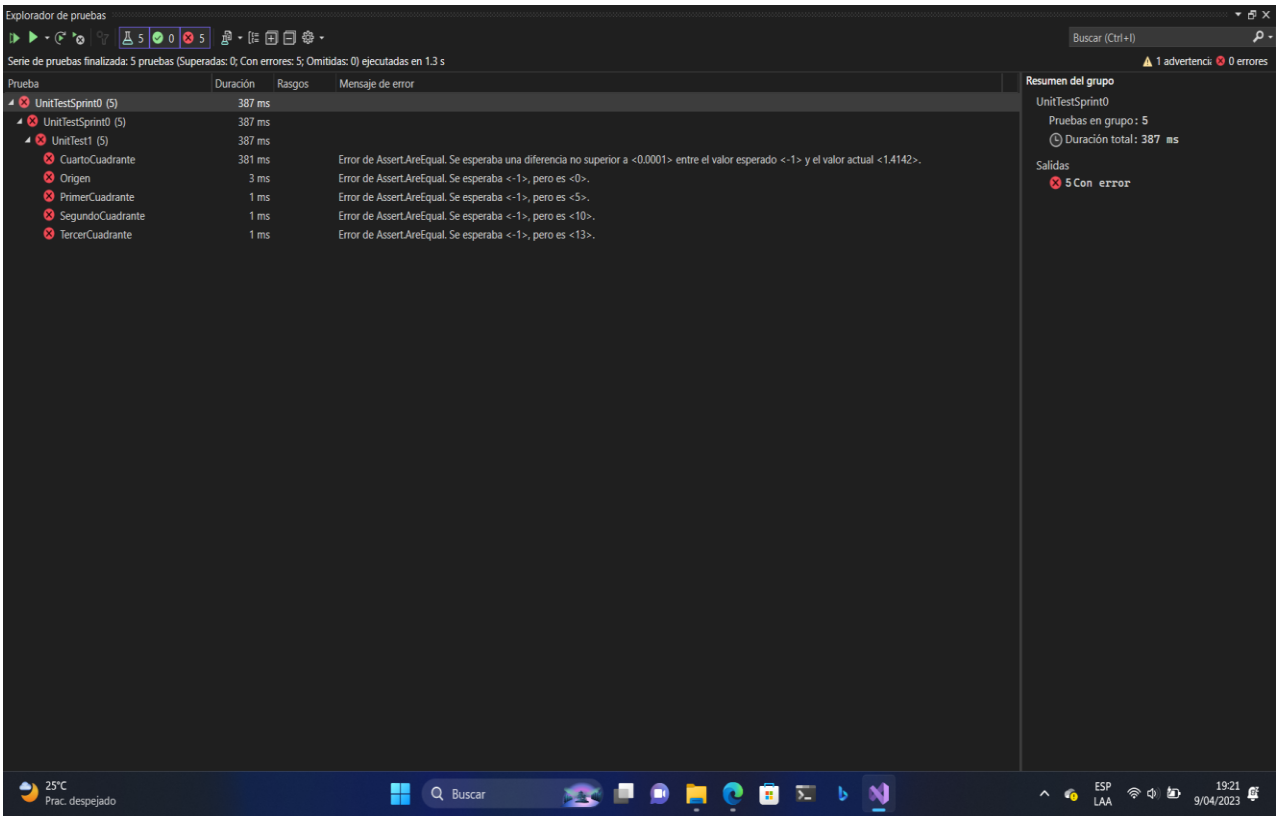
Guía de estilo de C# Microsoft (Convenciones de código de C#): Convenciones de código de C# | Microsoft Learn

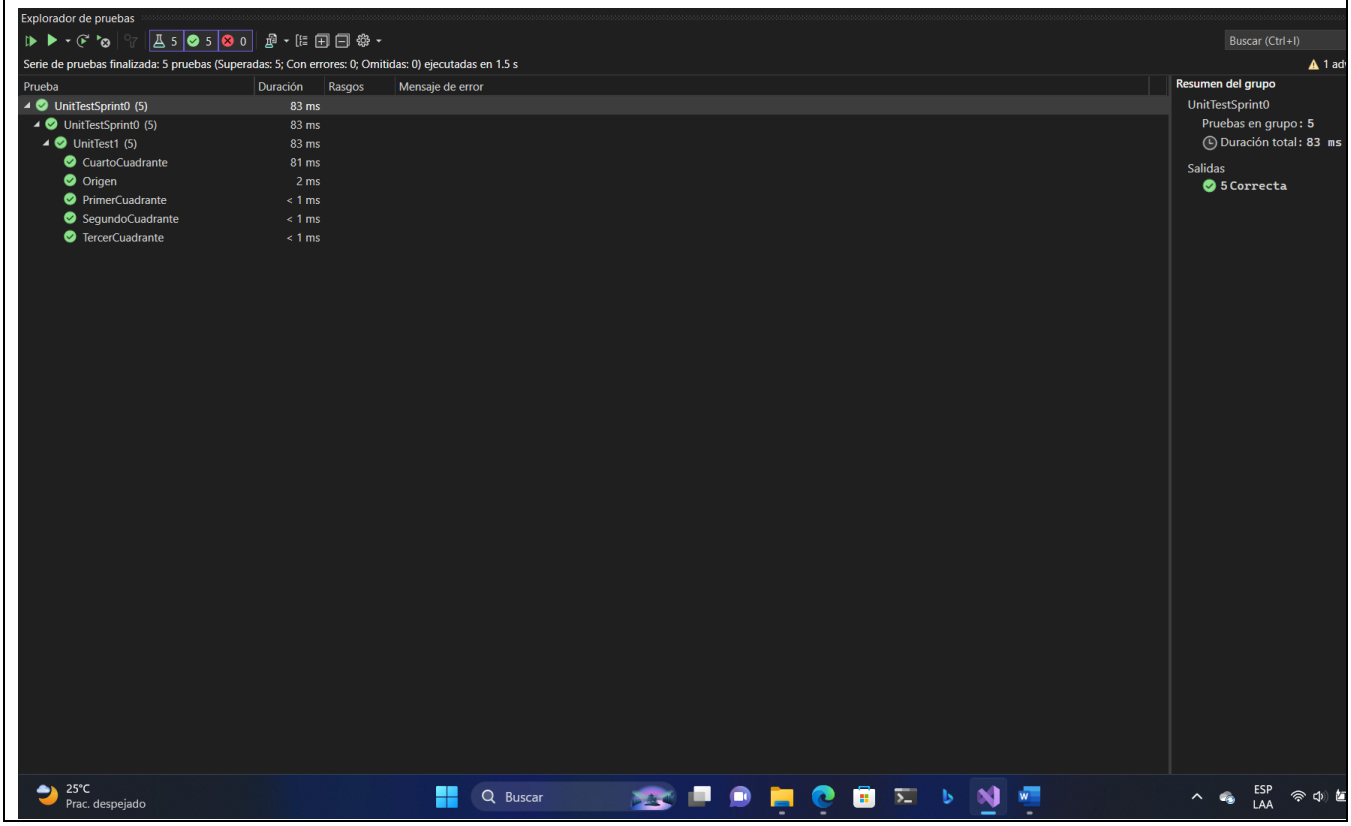
Ejemplos de guía de estilo de programación:

- Guía de estilo de Java Google: https://google.github.io/styleguide/javaguide.html
- Guía de estilo de C++ Google: https://google.github.io/styleguide/cppguide.html
- Guía de estilo Python Google: https://google.github.io/styleguide/pyguide.html

2. Pruebas unitarias (8 puntos)

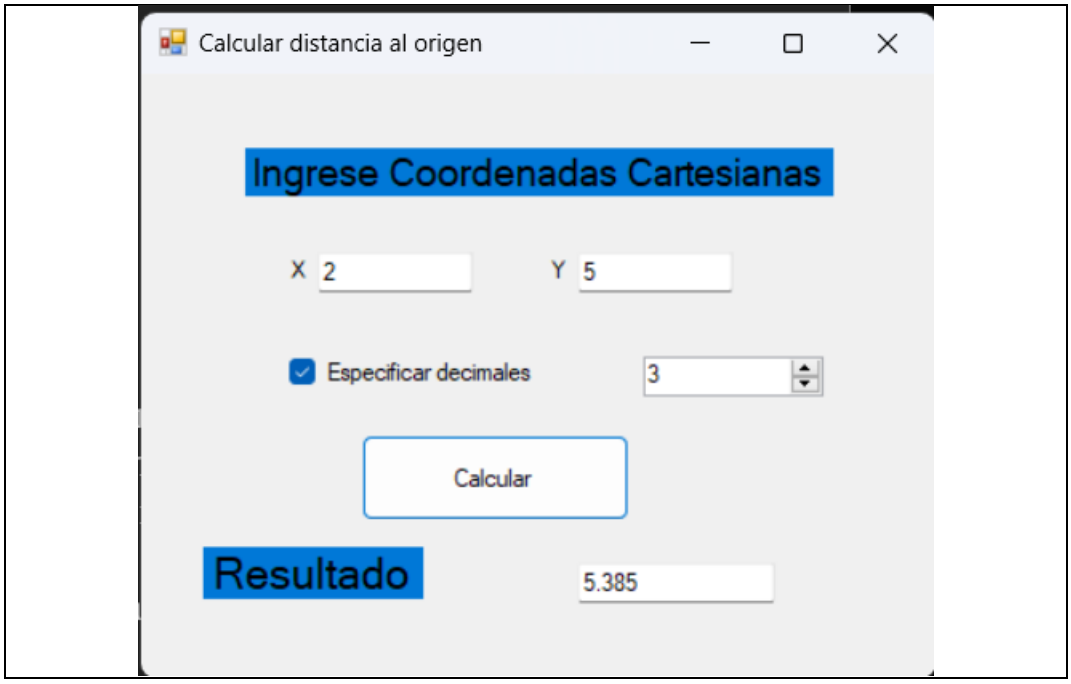
Encuentren un tutorial sobre el framework de pruebas unitarias que has elegido y escriban al menos dos pruebas xUnit de un programa que hayas escrito o encontrado en otro lugar. Adjunta aquí (1) la captura de pantalla de la ejecución de tu programa.
UnitTest1.cs


Explorador de pruebas
Serie de pruebas finalizada: 5 pruebas (Superadas: 0; Con errores: 5; Omitidas: 0) ejecutadas en 1.3 s
Prueba
Duración
Rasgos
Mensaje de error
UnitTestSprint0 (5) 387 ms
UnitTestSprint0 (5) 387 ms
UnitTest1 (5) 387 ms
CuartoCuadrante 381 ms Error de Assert.AreEqual. Se esperaba una diferencia no superior a <0.0001> entre el valor esperado <-1> y el valor actual <1.4142>.
Origen 3 ms Error de Assert.AreEqual. Se esperaba <-1>, pero es <0>.
PrimerCuadrante 1 ms Error de Assert.AreEqual. Se esperaba <-1>, pero es <5>.
SegundoCuadrante 1 ms Error de Assert.AreEqual. Se esperaba <-1>, pero es <10>.
TercerCuadrante 1 ms Error de Assert.AreEqual. Se esperaba <-1>, pero es <13>.
Resumen del grupo
UnitTestSprint0
Pruebas en grupo: 5
Duración total: 387 ms
Salidas
5 Con error
25°C
Prac. despejado
Buscar
ESP
LAA
19:21
9/04/2023


Explorador de pruebas
Serie de pruebas finalizada: 5 pruebas (Superadas: 5; Con errores: 0; Omitidas: 0) ejecutadas en 1.5 s
Prueba
Duración
Rasgos
Mensaje de error
UnitTestSprint0 (5) 83 ms
UnitTestSprint0 (5) 83 ms
UnitTest1 (5) 83 ms
CuartoCuadrante 81 ms
Origen 2 ms
PrimerCuadrante < 1 ms
SegundoCuadrante < 1 ms
TercerCuadrante < 1 ms
Resumen del grupo
UnitTestSprint0
Pruebas en grupo: 5
Duración total: 83 ms
Salidas
5 Correcta
25°C
Prac. despejado
Buscar
ESP
LAA

3. Programación GUI (10 puntos)

Escriban un programa GUI en el lenguaje que hayas elegido para tu proyecto SOS. La GUI de tu programa debe incluir texto, líneas, una casilla de verificación y botones de opción. Si bien se recomienda considerar la GUI para el tablero de juego SOS, no es obligatorio. En esta tarea, cualquier programa GUI de tu propio trabajo es aceptable.
Adjunten aquí (1) la captura de pantalla de la ejecución de tu programa y (2) el código fuente de tu programa.



Form1.cs

```
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;

namespace Sprint0
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
```

```
            //extraemos las coordenadas y las convertimos a double
            double X1 = Convert.ToDouble(textBox1.Text);
            double Y1 = Convert.ToDouble(textBox2.Text);

            //instanciamos las clases
            var distancia = new Distancia();
            var coordenadas = new Coordenada(X1, Y1);
            var impri = new Imprimir();

            //variable que guarda el resultado en double
            var resultado = distancia.CalcularDistancia(coordenadas);
            //decision de cómo se mostrará (decimales a usar)
            //no se usó if inmediato (condition ? consecuente : alternativa)
            if (checkBox1.Checked)
                textBox5.Text = impri.Impresion(resultado,
Convert.ToInt16(numericUpDown1.Value));
            else
                textBox5.Text = impri.Impresion(resultado);
        }
    }
}
```

Imprimir.cs

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

namespace Sprint0
{
    public class Imprimir
    {
        //devuelve un string del numero con la cantidad de decimales seleccionados
        //devuelve 5 decimales si no se especifica
        public string Impresion(double numero, int decimales)
        {
            return numero.ToString("N"+decimales.ToString());
        }
        public string Impresion(double numero)
        {
            return numero.ToString("N5");
        }
    }
}
```

Distancia.cs

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

```
namespace Sprint0
{
    public class Distancia    {
        // Función que calcula la distancia de un punto al Origen
        public double CalcularDistancia(Coordenada punto)
        {
            return Math.Sqrt(Math.Pow(punto.getX(), 2) + Math.Pow(punto.getY(), 2));

        }
    }
}
```

UnitTest1.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using Sprint0;

namespace UnitTestSprint0
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Origen()
        {
                var distancia = new Distancia();
                var punto = new Coordenada(0, 0);
                var resultado = distancia.CalcularDistancia(punto);
                Assert.AreEqual(resultado, 0.0);

        }
        [TestMethod]
        public void PrimerCuadrante()
        {
            var distancia = new Distancia();
            var punto = new Coordenada(3, 4);
            var resultado = distancia.CalcularDistancia(punto);
            Assert.AreEqual(resultado, 5.0);
        }

        [TestMethod]
        public void SegundoCuadrante()
        {
            var distancia = new Distancia();
            var punto = new Coordenada(-6, 8);
            var resultado = distancia.CalcularDistancia(punto);
            Assert.AreEqual(resultado, 10.0);
        }

        [TestMethod]
```

```
        public void TercerCuadrante()
        {
            var distancia = new Distancia();
            var punto = new Coordenada(-12, -5);
            var resultado = distancia.CalcularDistancia(punto);
            Assert.AreEqual(resultado, 13.0);
        }

        [TestMethod]
        public void CuartoCuadrante()
        {
            var distancia = new Distancia();
            var punto = new Coordenada(1, -1);
            var resultado = distancia.CalcularDistancia(punto);
            Assert.AreEqual(resultado, 1.4142, 0.0001);
        }
    }
}
```