Sprint #0 Report

Instructions

Objectives

- Make decisions on the SOS software development project.
- Learn unit testing and GUI programming in the language of your choice.

Deliverables and Grading Policy

Read the "CS 449 Homework Overview" document **carefully** and make the key decisions for the software development. Use the following template to complete your report.

1. Key Decisions of the SOS Project (2 points)

Object-oriented programming language	C#
GUI library (strongly encouraged)	WinForms (Windows .NET framework)
IDE (Integrated Development Environment)	Microsoft Visual Studio
xUnit framework (e.g., JUnit for Java)	MSTest
Programming style guide (must read it carefully)	Google C# Style Guide https://google.github.io/styleguide/csharp-style.html
Project hosting site	Github https://github.com/jonathanbenson/sosgame
Other decisions if applicable	N/A

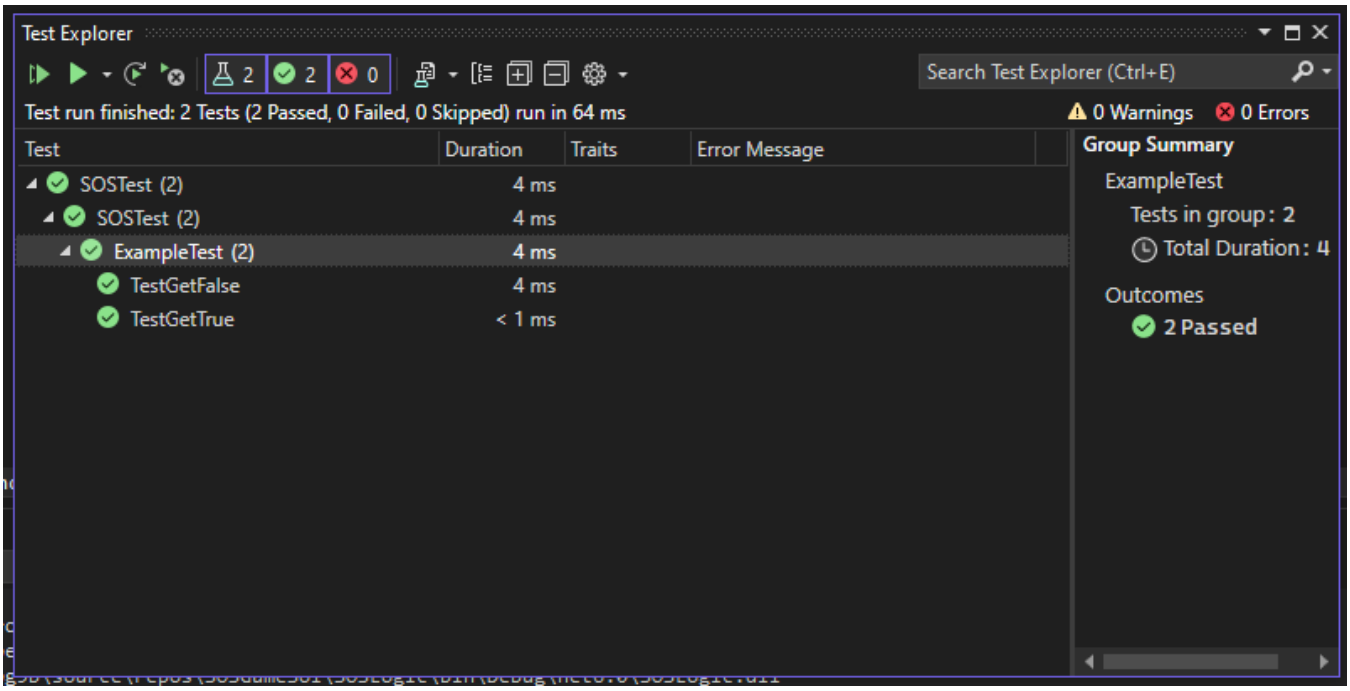
Sample programming style guides:

- Google Java Style Guide: https://google.github.io/styleguide/javaguide.html
- Google C++ Style Guide: https://google.github.io/styleguide/cppguide.html
- Google Python Style Guide: https://google.github.io/styleguide/pyguide.html

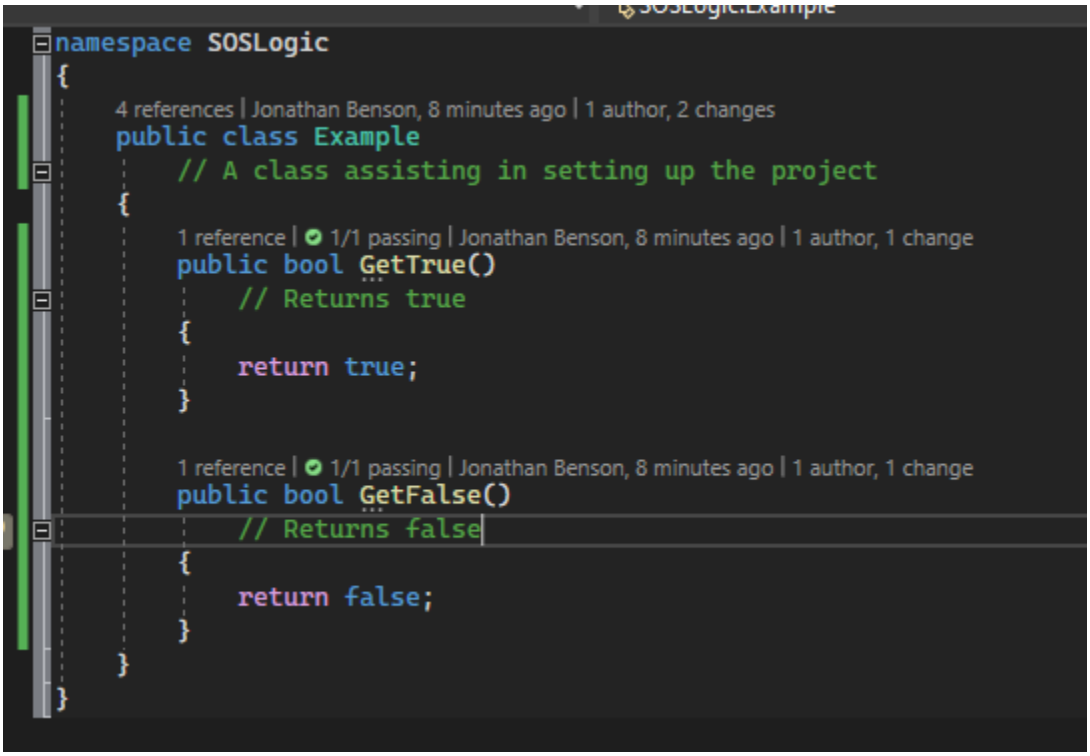
2. Unit testing (4 points)

Find a tutorial on the unit test framework you have chosen and write at least two xUnit tests of a program you have written or found elsewhere. Attach here (1) the screenshot of your program execution and (2) the source code of your program.

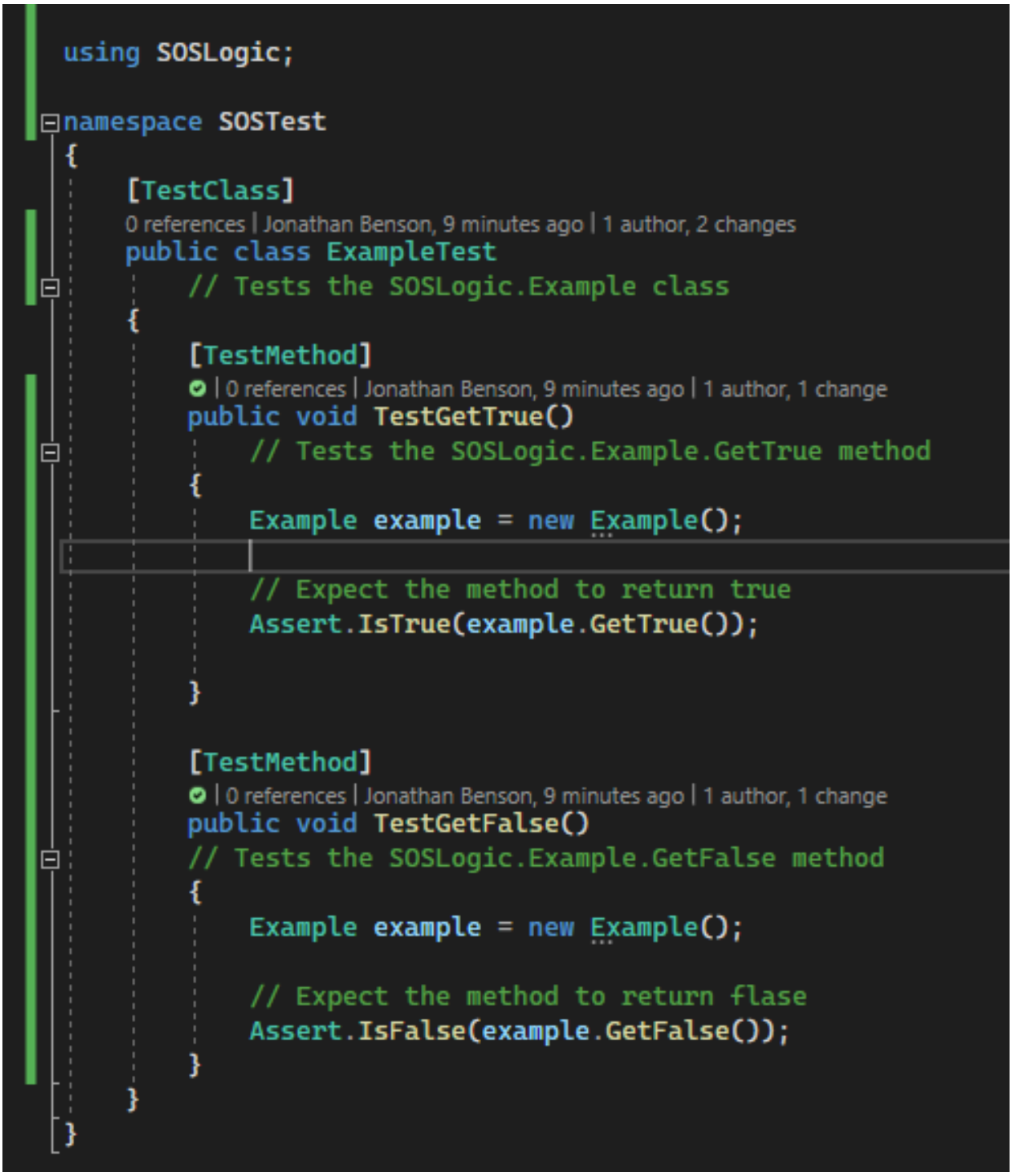
(1) Test execution using the C# MSTest unit testing framework.



(2) Here is an Example class that was written to showcase the MSTest unit testing framework.



Here is the ExampleTest class that tests the Example class. The macros above each class and method definition are interpreted by MSTest to correctly run the tests.



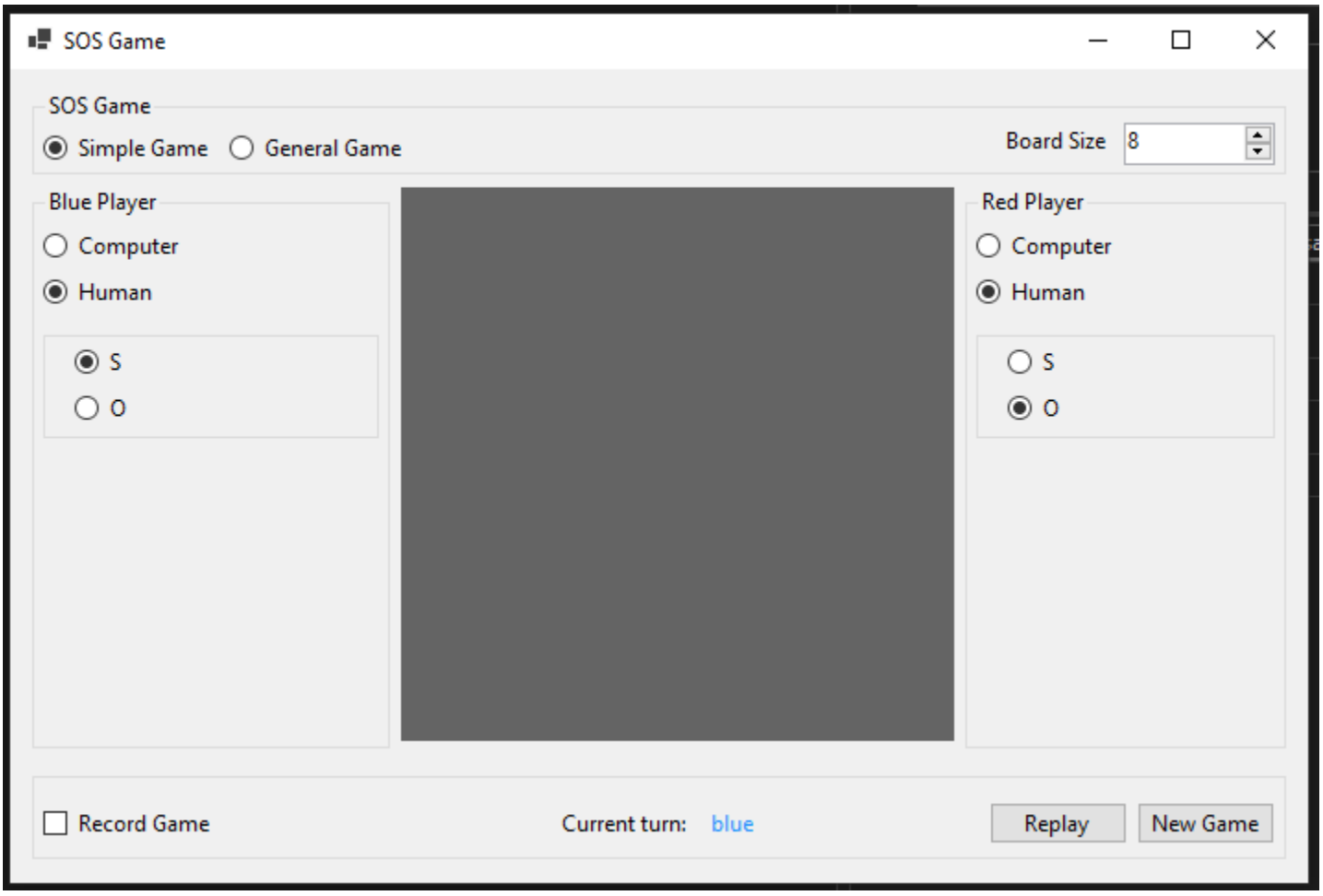
```
using SOSLogic;

namespace SOSTest
{
    [TestClass]
    0 references | Jonathan Benson, 9 minutes ago | 1 author, 2 changes
    public class ExampleTest
        // Tests the SOSLogic.Example class
    {
        [TestMethod]
        ✓ | 0 references | Jonathan Benson, 9 minutes ago | 1 author, 1 change
        public void TestGetTrue()
            // Tests the SOSLogic.Example.GetTrue method
        {
            Example example = new Example();

            // Expect the method to return true
            Assert.IsTrue(example.GetTrue());

        }

        [TestMethod]
        ✓ | 0 references | Jonathan Benson, 9 minutes ago | 1 author, 1 change
        public void TestGetFalse()
        // Tests the SOSLogic.Example.GetFalse method
        {
            Example example = new Example();

            // Expect the method to return flase
            Assert.IsFalse(example.GetFalse());
        }
    }
}
```

3. GUI programming (4 points)

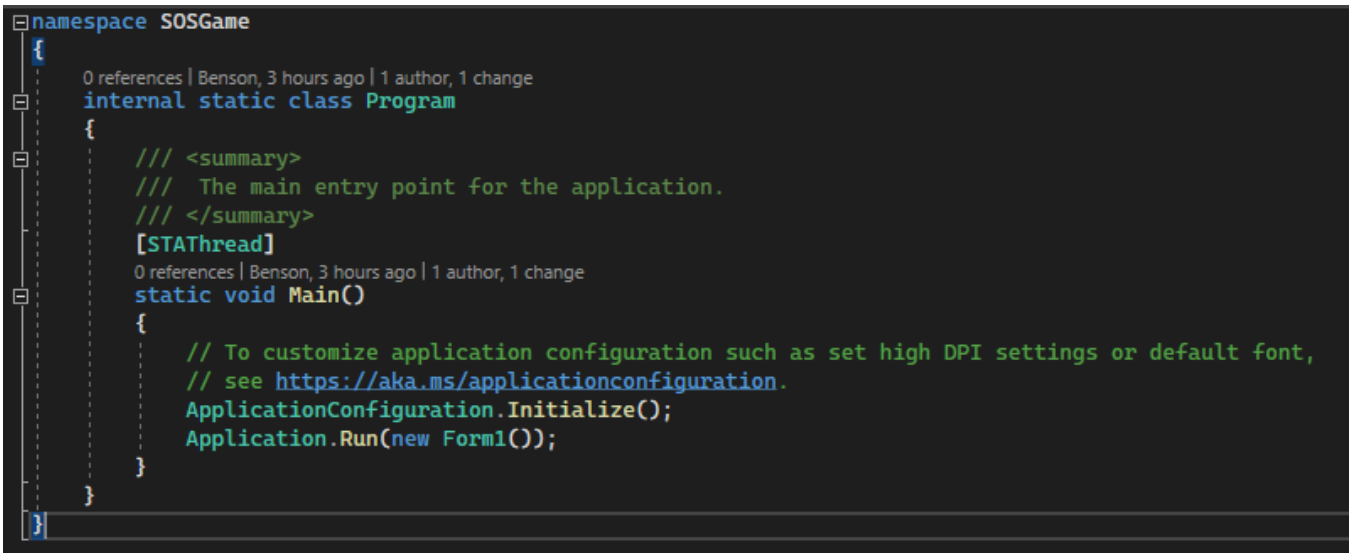
Write a GUI program in the language you have chosen for your SOS project. The GUI of your program must include text, lines, a check box, and radio buttons. While you are recommended to consider the GUI for the SOS game board, it is not required. In this assignment, any GUI program of your own work is acceptable.

Attach here (1) the screenshot of your program execution and (2) the source code of your program.

(1) Below is the GUI of the SOS game. The 300x300 pixel box in the center is where the game board will be rendered.



(2) Below is the entry point for the program (main function). This is boilerplate code generated by Visual Studio upon creation of the project.



```
namespace SOSGame
{
    0 references | Benson, 3 hours ago | 1 author, 1 change
    internal static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        0 references | Benson, 3 hours ago | 1 author, 1 change
        static void Main()
        {
            // To customize application configuration such as set high DPI settings or default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new Form1());
        }
    }
}
```

Here is the class that handles the functionality of the main GUI window. This code is also boilerplate generated by Visual Studio when components are dragged onto the window in the editor.

```
namespace SOSGame
{
    3 references | Jonathan Benson, 34 minutes ago | 2 authors, 3 changes
    partial class Form1
    {
        /// <summary>
        ///  Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary> Clean up any resources being used.
        0 references | Benson, 3 hours ago | 1 author, 1 change
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        Windows Form Designer generated code

        private Label label1;
        private GroupBox _topGroupBox;
        private Label _boardSizeLabel;
        private NumericUpDown _boardSizeNum;
        private RadioButton _generalGameRadio;
        private RadioButton _simpleGameRadio;
        private CheckBox _recordGameCheckBox;
        private Label _currentTurnLabel;
        private Label _currentTurn;
        private Button _newGameButton;
        private Button _replayButton;
        private GroupBox groupBox2;
        private GroupBox _bluePlayerGroupBox;
        private RadioButton _bluePlayerORadio;
        private RadioButton _bluePlayerSRadio;
        private RadioButton _bluePlayerHumanRadio;
        private RadioButton _bluePlayerComputerRadio;
        private GroupBox _redPlayerGroupBox;
        private RadioButton _redPlayerHumanRadio;
        private RadioButton _redPlayerComputerRadio;
        private GroupBox groupBox5;
        private GroupBox groupBox6;
        private RadioButton _redPlayerSRadio;
        private RadioButton _redPlayerORadio;
        private PictureBox pictureBox1;
    }
}
```