Updated post tutorial

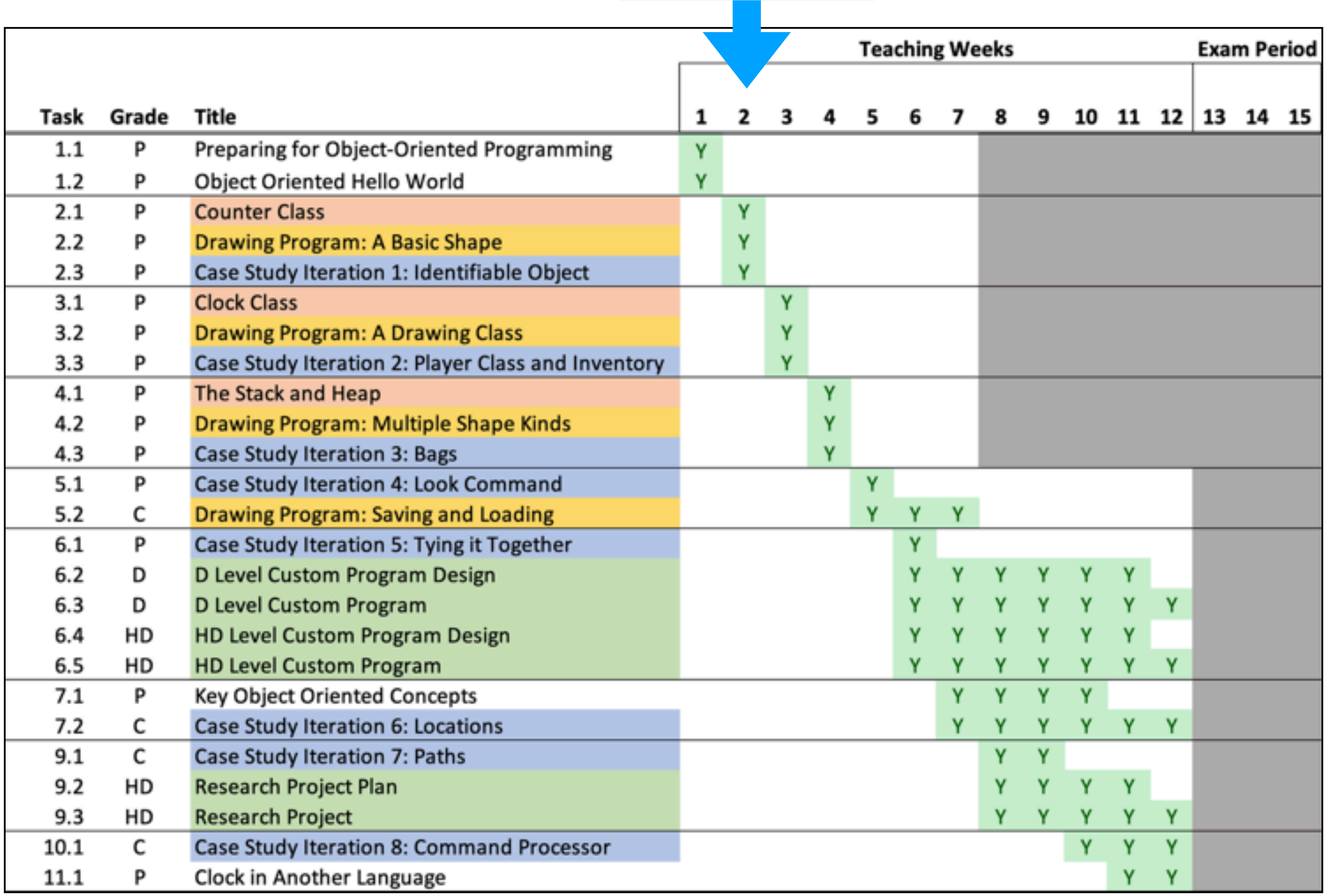
Lab 02 Tutorial Checkpoints

Ed Greenaway
Schedule

TIME	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
10:00 am					
10:30 am					OOP CL1/09
11:00 am					EN310
11:30 am	OOP HELP DESK				
12:00 pm	ATC 620				
12:30 pm	OOP HELP DESK	OOP CL1/19	OOP CL1/02		
1:00 pm	ATC 620	EN 310	EN 310		
1:30 pm					
2:00 pm					
2:30 pm	OOP HELP DESK				OOP CL1/07
3:00 pm	ATC 620				EN310
3:30 pm	OOP HELP DESK				
4:00 pm	ATC 620				

Our journey ...

we are here



Task	Grade	Title	Teaching Weeks												Exam Period		
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1.1	P	Preparing for Object-Oriented Programming	Y														
1.2	P	Object Oriented Hello World	Y														
2.1	P	Counter Class		Y													
2.2	P	Drawing Program: A Basic Shape		Y													
2.3	P	Case Study Iteration 1: Identifiable Object		Y													
3.1	P	Clock Class			Y												
3.2	P	Drawing Program: A Drawing Class			Y												
3.3	P	Case Study Iteration 2: Player Class and Inventory			Y												
4.1	P	The Stack and Heap				Y											
4.2	P	Drawing Program: Multiple Shape Kinds				Y											
4.3	P	Case Study Iteration 3: Bags				Y											
5.1	P	Case Study Iteration 4: Look Command					Y										
5.2	C	Drawing Program: Saving and Loading					Y	Y	Y								
6.1	P	Case Study Iteration 5: Tying it Together						Y									
6.2	D	D Level Custom Program Design						Y	Y	Y	Y	Y	Y				
6.3	D	D Level Custom Program						Y	Y	Y	Y	Y	Y	Y			
6.4	HD	HD Level Custom Program Design						Y	Y	Y	Y	Y	Y				
6.5	HD	HD Level Custom Program						Y	Y	Y	Y	Y	Y	Y			
7.1	P	Key Object Oriented Concepts							Y	Y	Y	Y					
7.2	C	Case Study Iteration 6: Locations							Y	Y	Y	Y	Y	Y			
9.1	C	Case Study Iteration 7: Paths								Y	Y						
9.2	HD	Research Project Plan								Y	Y	Y	Y				
9.3	HD	Research Project								Y	Y	Y	Y	Y			
10.1	C	Case Study Iteration 8: Command Processor										Y	Y	Y			
11.1	P	Clock in Another Language											Y	Y			

OOP

Graphics

Case Study

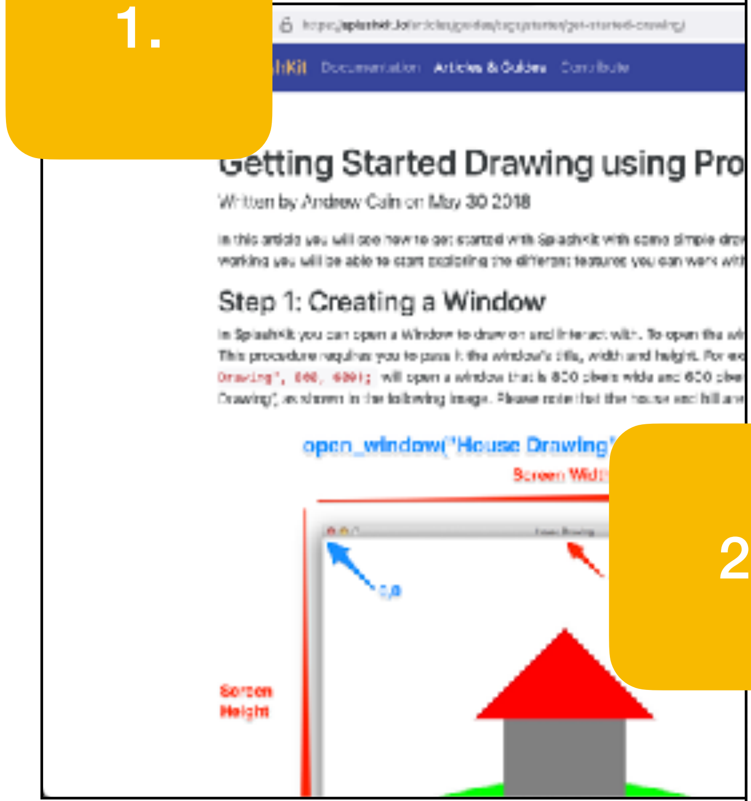
D/HD

Other

While we wait; a quick SplashKit for fun …

https://splashkit.io/guides/01-00-drawing/

1.



https://splashkit.io/articles/guides/tutorials/get-started-drawing/
lhKit
Documentation
Articles & Guides
Contribute

Getting Started Drawing using Pro

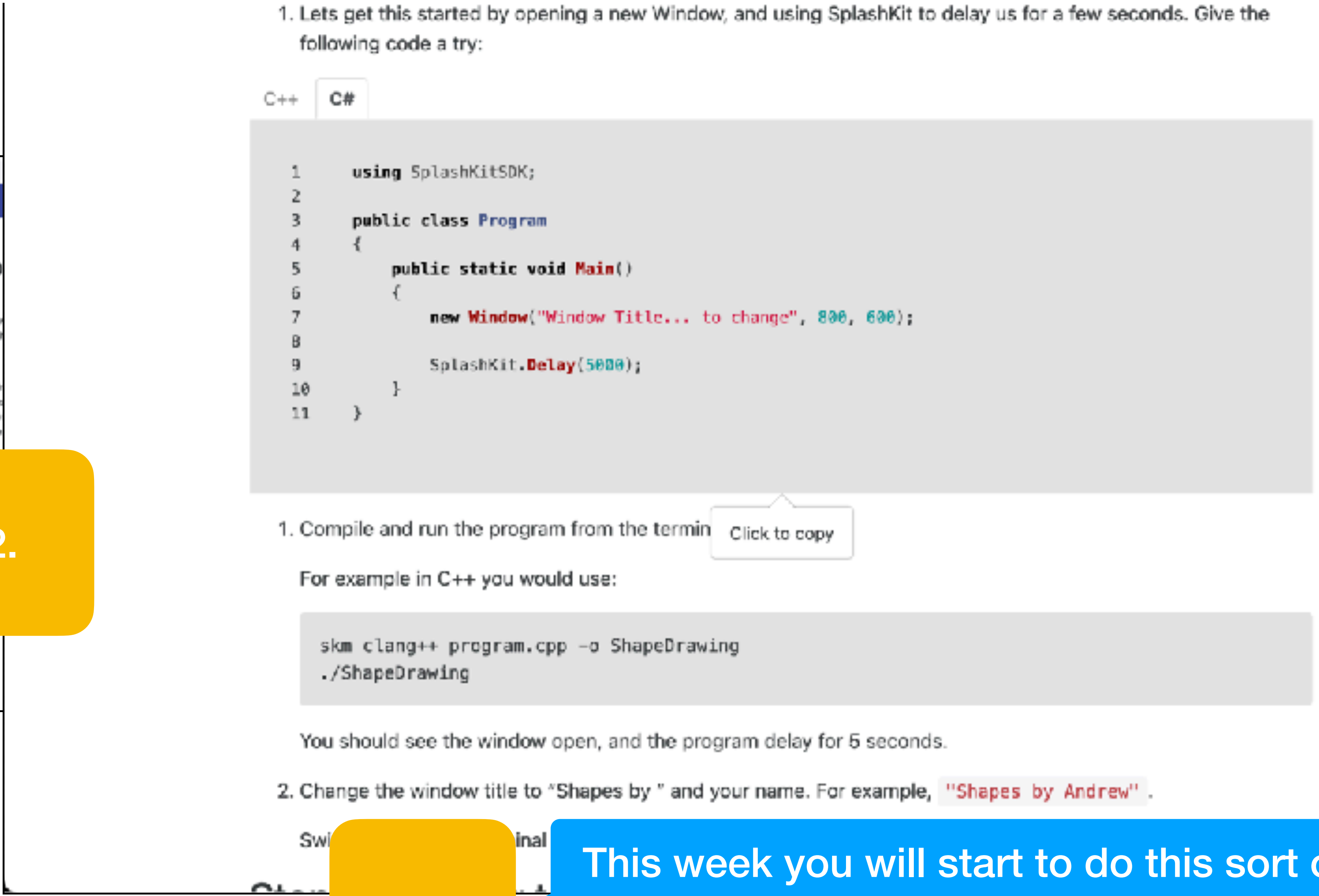
Written by Andrew Cain on May 30 2018

In this article you will see how to get started with SplashKit with some simple dra
working you will be able to start exploring the different features you can work wit

Step 1: Creating a Window

In SplashKit you can open a Window to draw on and interact with. To open the wi
This procedure requires you to pass it the window's title, width and height. For ex
Drawing", 800, 600); will open a window that is 800 pixels wide and 600 pixe
Drawing", as shown in the following image. Please note that the house and hill are

2.



1. Lets get this started by opening a new Window, and using SplashKit to delay us for a few seconds. Give the following code a try:

```
using SplashKitSDK;

public class Program
{
    public static void Main()
    {
        new Window("Window Title... to change", 800, 600);

        SplashKit.Delay(5000);
    }
}
```

1. Compile and run the program from the termin

For example in C++ you would use:

```
skm clang++ program.cpp -o ShapeDrawing
./ShapeDrawing
```

You should see the window open, and the program delay for 5 seconds.

2. Change the window title to "Shapes by " and your name. For example, "Shapes by Andrew".

Swi inal

3.

This week you will start to do this sort of thing with Objects that create a rectangle … then circles, lines …

While we wait; a quick SplashKit for fun …

https://splashkit.io/guides/01-00-drawing/

1.

Getting Started Dra

Written by Andrew Cain on May 30 2

In this article you will see how to get started working you will be able to start exploring the

Step 1: Creating a Wind

In SplashKit you can open a Window to draw This procedure requires you to pass it the wi Drawing", 800, 600); will open a window Drawing", as shown in the following image.

open_window("Hous

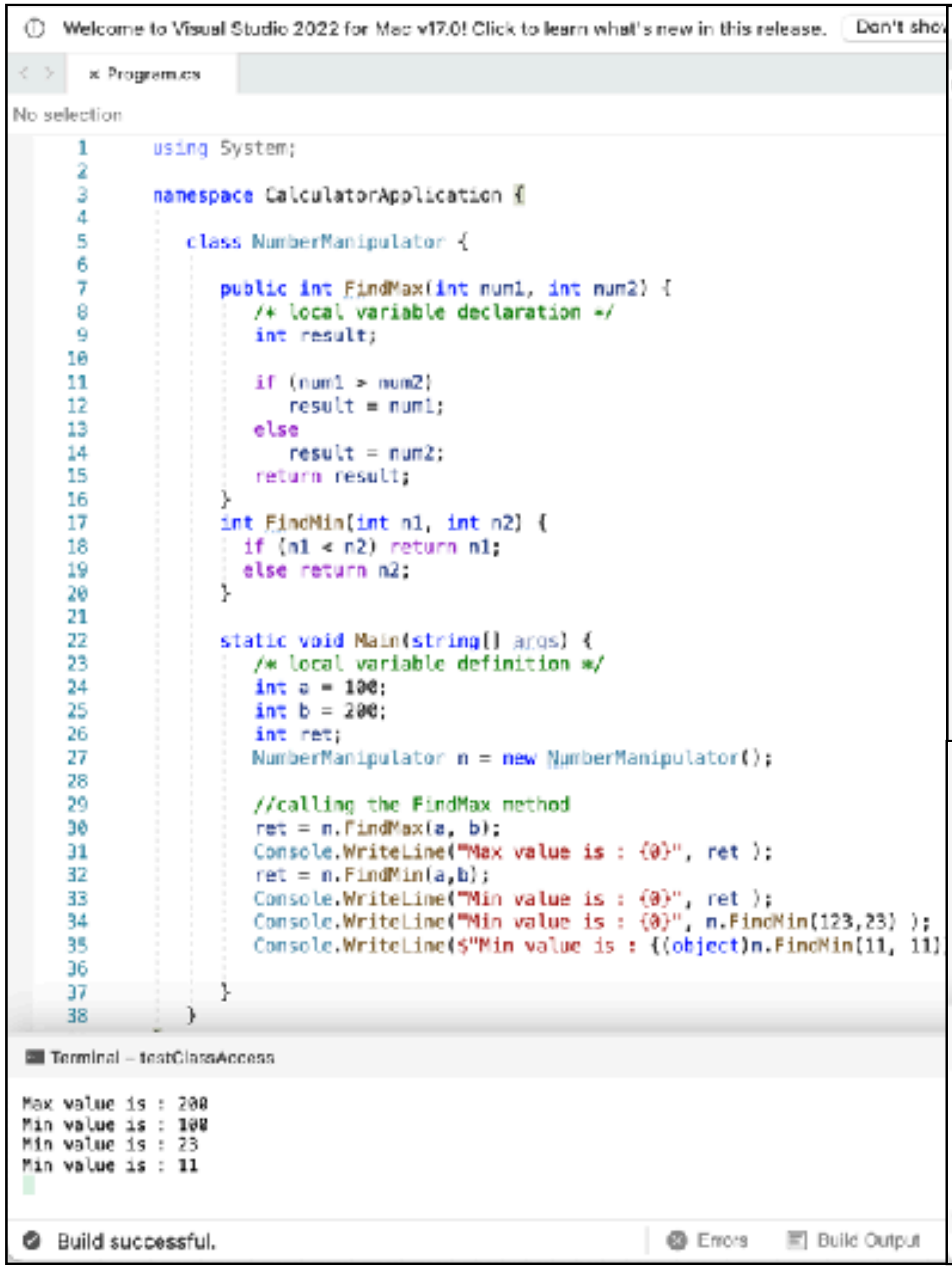
```
using SplashKitSDK;

public class Program
{
public static void Main(string[] args)
{
Window shapesWindow;
shapesWindow = new Window("Shapes by ...", 800, 600);

        shapesWindow.Clear(Color.White);
        shapesWindow.FillEllipse(Color.BrightGreen, 0, 400, 800, 400);
        shapesWindow.FillRectangle(Color.Gray, 300, 300, 200, 200);
        shapesWindow.FillTriangle(Color.Red, 250, 300, 400, 150, 550, 300);
        shapesWindow.Refresh();

        SplashKit.Delay(5000);
    }

}
```

4.

A B C D

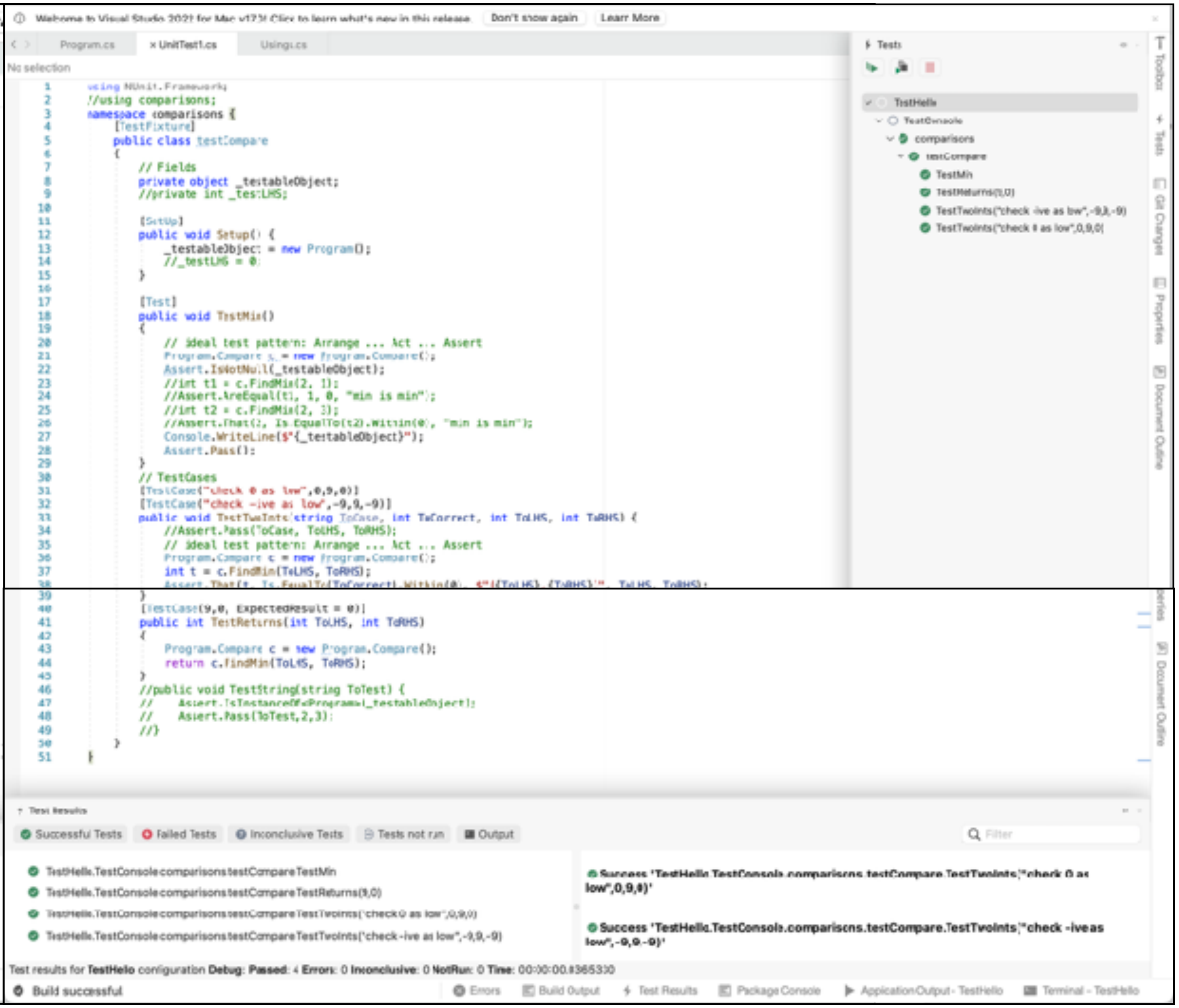
We all should have Visual Studio with C# & NUnit installed

C# coded tests ... versus ... **C# integrated with NUnit tests**


Welcome to Visual Studio 2022 for Mac v17.0! Click to learn what's new in this release.
Don't sho
Program.cs
No selection
Terminal – testClassAccess
Build successful.
Errors
Build Output

```
using System;

namespace CalculatorApplication {

   class NumberManipulator {

      public int FindMax(int num1, int num2) {
         /* local variable declaration */
         int result;

         if (num1 > num2)
            result = num1;
         else
            result = num2;
         return result;
      }
      int FindMin(int n1, int n2) {
        if (n1 < n2) return n1;
        else return n2;
      }

      static void Main(string[] args) {
         /* local variable definition */
         int a = 100;
         int b = 200;
         int ret;
         NumberManipulator n = new NumberManipulator();

         //calling the FindMax method
         ret = n.FindMax(a, b);
         Console.WriteLine("Max value is : {0}", ret );
         ret = n.FindMin(a,b);
         Console.WriteLine("Min value is : {0}", ret );
         Console.WriteLine("Min value is : {0}", n.FindMin(123,23) );
         Console.WriteLine($"Min value is : {(object)n.FindMin(11, 11]

      }
   }
```

```
Max value is : 200
Min value is : 100
Min value is : 23
Min value is : 11
```



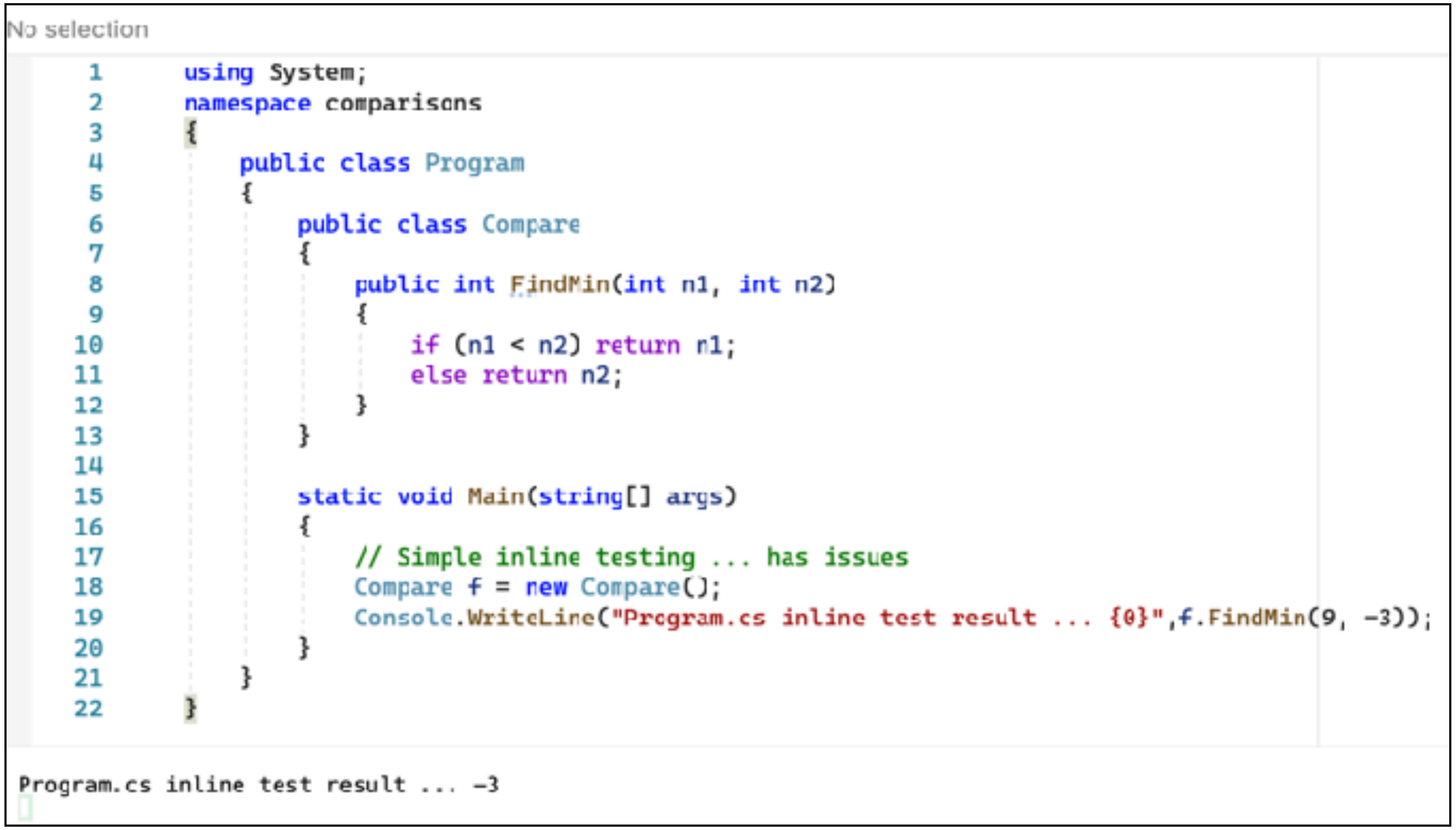
```
using NUnit.Framework;
//using comparisons;
namespace comparisons {
    [TestFixture]
    public class testCompare
    {
        // Fields
        private object _testableObject;
        //private int _testLHS;

        [SetUp]
        public void Setup() {
            _testableObject = new Program();
            //_testLHS = 0;
        }

        [Test]
        public void TestMin()
        {
            // ideal test pattern: Arrange ... Act ... Assert
            Program.Compare c = new Program.Compare();
            Assert.IsNotNull(_testableObject);
            //int t1 = c.FindMin(2, 1);
            //Assert.AreEqual(t1, 1, 0, "min is min");
            //int t2 = c.FindMin(2, 3);
            //Assert.That(2, Is.EqualTo(t2).Within(0), "min is min");
            Console.WriteLine($"{_testableObject}");
            Assert.Pass();
        }
        // TestCases
        [TestCase("check 0 as low",0,9,0)]
        [TestCase("check -ive as low",-9,9,-9)]
        public void TestTwoInts(string ToCase, int ToCorrect, int ToLHS, int ToRHS) {
            //Assert.Pass(ToCase, ToLHS, ToRHS);
            // ideal test pattern: Arrange ... Act ... Assert
            Program.Compare c = new Program.Compare();
            int t = c.FindMin(ToLHS, ToRHS);
            Assert.That(t, Is.EqualTo(ToCorrect).Within(0), $"{ToLHS} {ToRHS}'", ToLHS, ToRHS);
        }
        [TestCase(9,0, ExpectedResult = 0)]
        public int TestReturns(int ToLHS, int ToRHS)
        {
            Program.Compare c = new Program.Compare();
            return c.FindMin(ToLHS, ToRHS);
        }
        //public void TestString(string ToTest) {
        //    Assert.IsInstanceOf<Program>(_testableObject);
        //    Assert.Pass(ToTest,2,3);
        //}
    }
}
```

```
TestHello.TestConsole.comparisons.testCompare.TestMin
TestHello.TestConsole.comparisons.testCompare.TestReturns(9,0)
TestHello.TestConsole.comparisons.testCompare.TestTwoInts("check 0 as low",0,9,0)
TestHello.TestConsole.comparisons.testCompare.TestTwoInts("check -ive as low",-9,9,-9)

Success 'TestHello.TestConsole.comparisons.testCompare.TestTwoInts("check 0 as low",0,9,0)'

Success 'TestHello.TestConsole.comparisons.testCompare.TestTwoInts("check -ive as low",-9,9,-9)'

Test results for TestHello configuration Debug: Passed: 4 Errors: 0 Inconclusive: 0 NotRun: 0 Time: 00:00:00.8365330
Build successful
```

A

Testing is goodness!
However a test harness is better …

C# coded tests



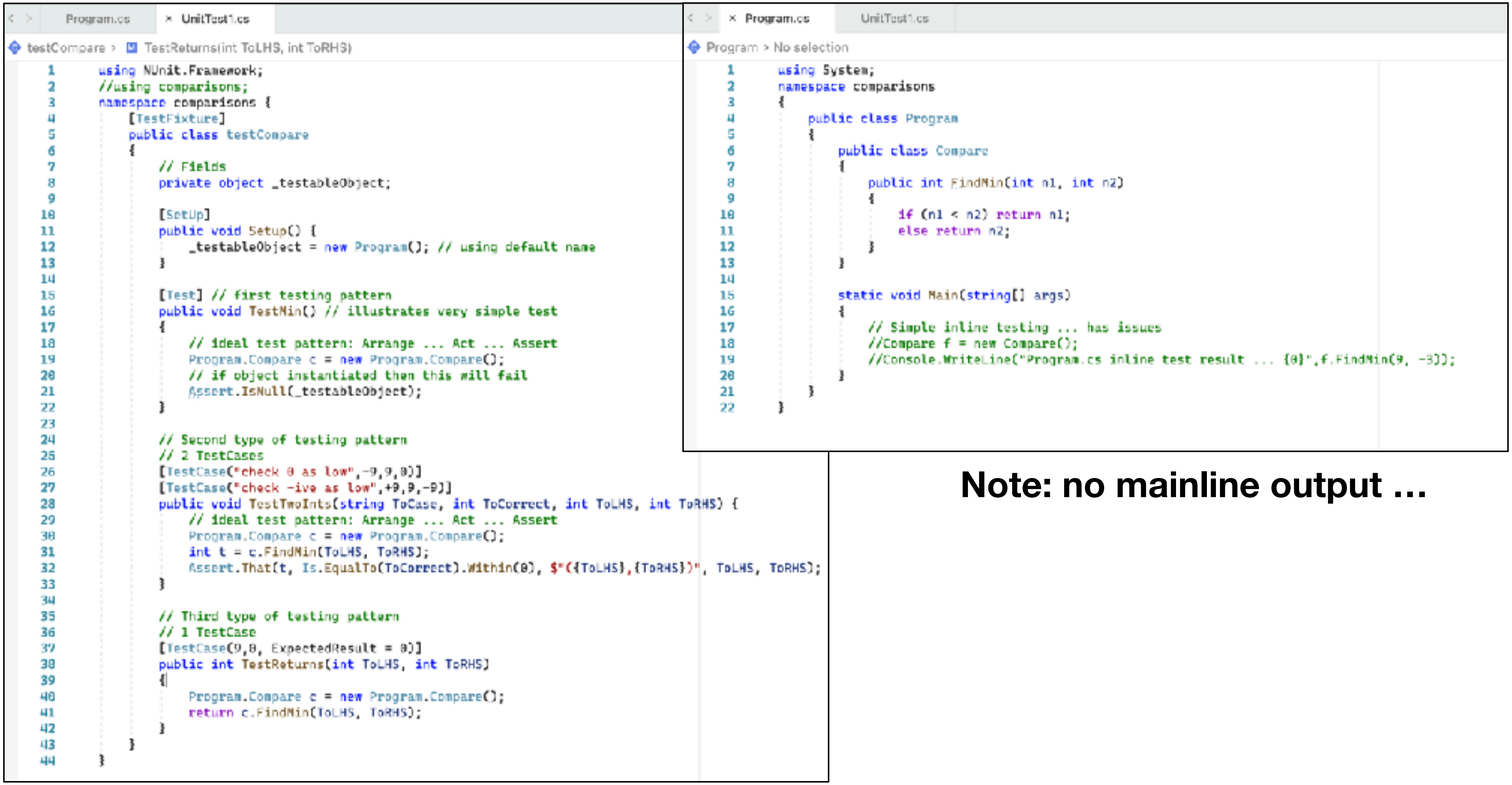
```
using System;
namespace comparisons
{
    public class Program
    {
        public class Compare
        {
            public int FindMin(int n1, int n2)
            {
                if (n1 < n2) return n1;
                else return n2;
            }
        }

        static void Main(string[] args)
        {
            // Simple inline testing ... has issues
            Compare f = new Compare();
            Console.WriteLine("Program.cs inline test result ... {0}",f.FindMin(9, -3));
        }
    }
}
```

```
Program.cs inline test result ... -3
```

B

Testing is goodness!
However a test harness is better.

NUnit harness and unit test markups run exercise the Program.cs' object(s)/class(es)



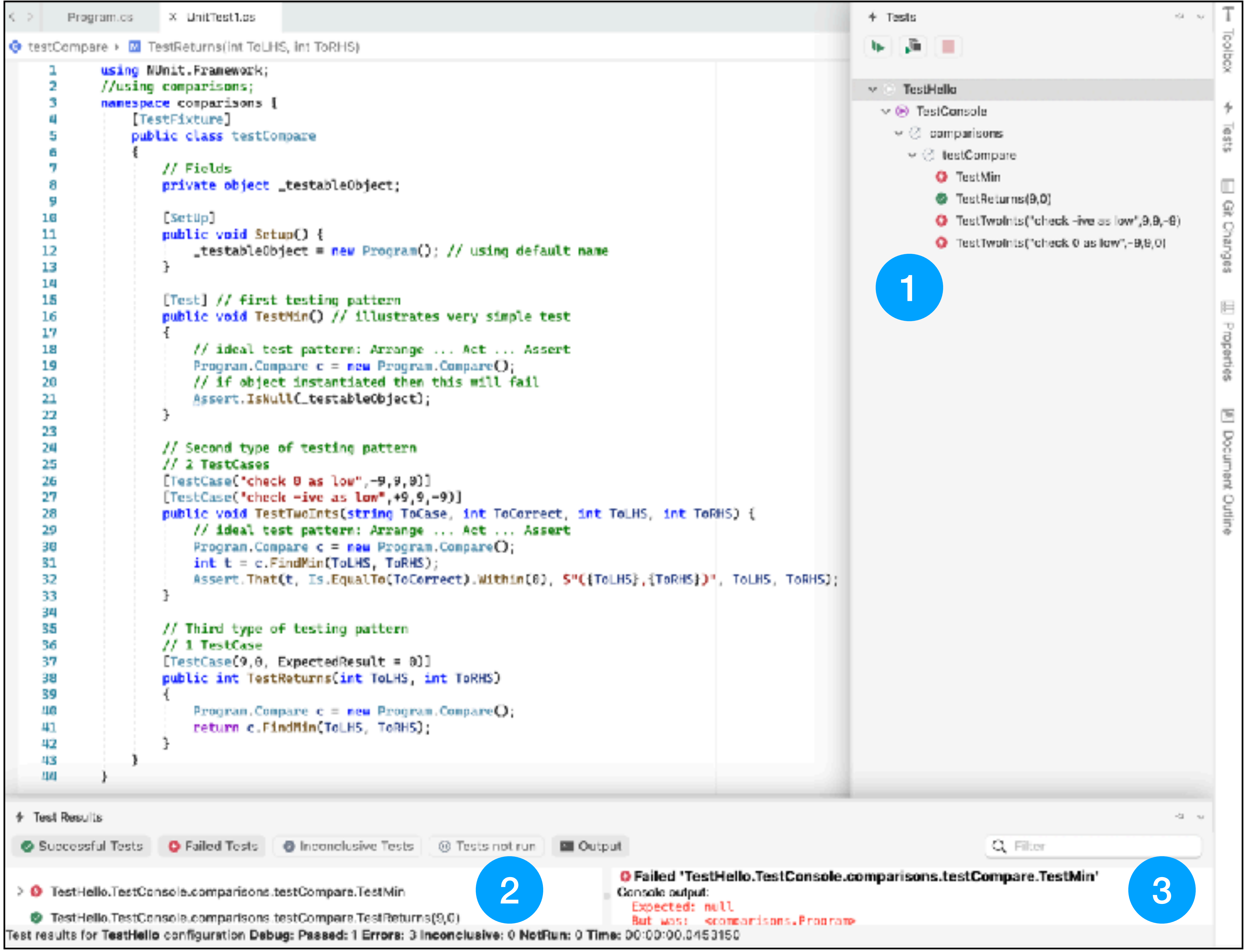
```csharp
using NUnit.Framework;
//using comparisons;
namespace comparisons {
    [TestFixture]
    public class testCompare
    {
        // Fields
        private object _testableObject;

        [SetUp]
        public void Setup() {
            _testableObject = new Program(); // using default name
        }

        [Test] // first testing pattern
        public void TestMin() // illustrates very simple test
        {
            // ideal test pattern: Arrange ... Act ... Assert
            Program.Compare c = new Program.Compare();
            // if object instantiated then this will fail
            Assert.IsNull(_testableObject);
        }

        // Second type of testing pattern
        // 2 TestCases
        [TestCase("check 0 as low",-9,9,0)]
        [TestCase("check -ive as low",+9,9,-9)]
        public void TestTwoInts(string ToCase, int ToCorrect, int ToLHS, int ToRHS) {
            // ideal test pattern: Arrange ... Act ... Assert
            Program.Compare c = new Program.Compare();
            int t = c.FindMin(ToLHS, ToRHS);
            Assert.That(t, Is.EqualTo(ToCorrect).Within(0), $"({ToLHS},{ToRHS})", ToLHS, ToRHS);
        }

        // Third type of testing pattern
        // 1 TestCase
        [TestCase(9,0, ExpectedResult = 0)]
        public int TestReturns(int ToLHS, int ToRHS)
        {
            Program.Compare c = new Program.Compare();
            return c.FindMin(ToLHS, ToRHS);
        }
    }
}
```

```csharp
using System;
namespace comparisons
{
    public class Program
    {
        public class Compare
        {
            public int FindMin(int n1, int n2)
            {
                if (n1 < n2) return n1;
                else return n2;
            }
        }

        static void Main(string[] args)
        {
            // Simple inline testing ... has issues
            //Compare f = new Compare();
            //Console.WriteLine("Program.cs inline test result ... {0}",f.FindMin(9, -3));
        }
    }
}
```

Note: no mainline output …

C

Testing is goodness!
The test harness reports the results:



```
using NUnit.Framework;
//using comparisons;
namespace comparisons {
    [TestFixture]
    public class testCompare
    {
        // Fields
        private object _testableObject;

        [SetUp]
        public void Setup() {
            _testableObject = new Program(); // using default name
        }

        [Test] // first testing pattern
        public void TestMin() // illustrates very simple test
        {
            // ideal test pattern: Arrange ... Act ... Assert
            Program.Compare c = new Program.Compare();
            // if object instantiated then this will fail
            Assert.IsNull(_testableObject);
        }

        // Second type of testing pattern
        // 2 TestCases
        [TestCase("check 0 as low",-9,9,0)]
        [TestCase("check -ive as low",+9,9,-9)]
        public void TestTwoInts(string ToCase, int ToCorrect, int ToLHS, int ToRHS) {
            // ideal test pattern: Arrange ... Act ... Assert
            Program.Compare c = new Program.Compare();
            int t = c.FindMin(ToLHS, ToRHS);
            Assert.That(t, Is.EqualTo(ToCorrect).Within(0), $"({ToLHS},{ToRHS})", ToLHS, ToRHS);
        }

        // Third type of testing pattern
        // 1 TestCase
        [TestCase(9,0, ExpectedResult = 0)]
        public int TestReturns(int ToLHS, int ToRHS)
        {
            Program.Compare c = new Program.Compare();
            return c.FindMin(ToLHS, ToRHS);
        }
    }
}
```

```
TestHello
  TestConsole
    comparisons
      testCompare
        TestMin
        TestReturns(9,0)
        TestTwoInts("check -ive as low",9,9,-9)
        TestTwoInts("check 0 as low",-9,9,0)
```

```
> TestHello.TestConsole.comparisons.testCompare.TestMin
  TestHello.TestConsole.comparisons.testCompare.TestReturns(9,0)
```

```
Failed 'TestHello.TestConsole.comparisons.testCompare.TestMin'
Console output:
  Expected: null
  But was:  <comparisons.Program>
```

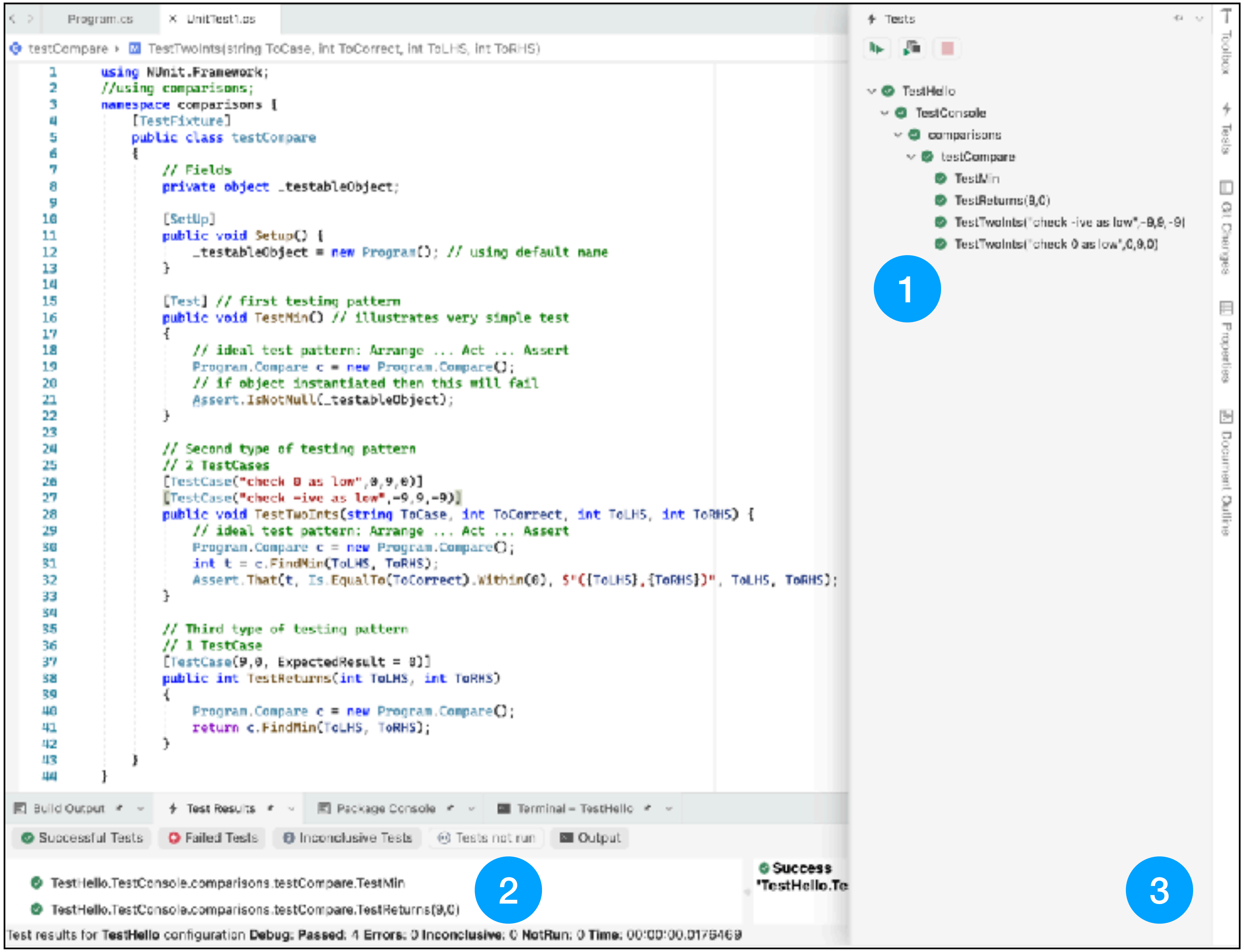
```
Test results for TestHello configuration Debug: Passed: 1 Errors: 3 Inconclusive: 0 NotRun: 0 Time: 00:00:00.0453150
```

Results are reported in three locations

In this run we deliberately caused 3 failures by tests with incorrect assertions

D

Testing is goodness!
This is a clean run for all assertions



```csharp
using NUnit.Framework;
//using comparisons;
namespace comparisons {
    [TestFixture]
    public class testCompare
    {
        // Fields
        private object _testableObject;

        [SetUp]
        public void Setup() {
            _testableObject = new Program(); // using default name
        }

        [Test] // first testing pattern
        public void TestMin() // illustrates very simple test
        {
            // ideal test pattern: Arrange ... Act ... Assert
            Program.Compare c = new Program.Compare();
            // if object instantiated then this will fail
            Assert.IsNotNull(_testableObject);
        }

        // Second type of testing pattern
        // 2 TestCases
        [TestCase("check 0 as low",0,9,0)]
        [TestCase("check -ive as low",-9,9,-9)]
        public void TestTwoInts(string ToCase, int ToCorrect, int ToLHS, int ToRHS) {
            // ideal test pattern: Arrange ... Act ... Assert
            Program.Compare c = new Program.Compare();
            int t = c.FindMin(ToLHS, ToRHS);
            Assert.That(t, Is.EqualTo(ToCorrect).Within(0), $"({ToLHS},{ToRHS})", ToLHS, ToRHS);
        }

        // Third type of testing pattern
        // 1 TestCase
        [TestCase(9,0, ExpectedResult = 0)]
        public int TestReturns(int ToLHS, int ToRHS)
        {
            Program.Compare c = new Program.Compare();
            return c.FindMin(ToLHS, ToRHS);
        }
    }
}
```

```
TestHello
  TestConsole
    comparisons
      testCompare
        TestMin
        TestReturns(9,0)
        TestTwoInts("check -ive as low",-9,9,-9)
        TestTwoInts("check 0 as low",0,9,0)
```

```
TestHello.TestConsole.comparisons.testCompare.TestMin
TestHello.TestConsole.comparisons.testCompare.TestReturns(9,0)
Test results for TestHello configuration Debug: Passed: 4 Errors: 0 Inconclusive: 0 NotRun: 0 Time: 00:00:00.0175469
```

```
Success
"TestHello.Te
```

Results are reported in three locations

Notice the pattern in the commentary …

1st we **Arrange**

2nd we **Act**

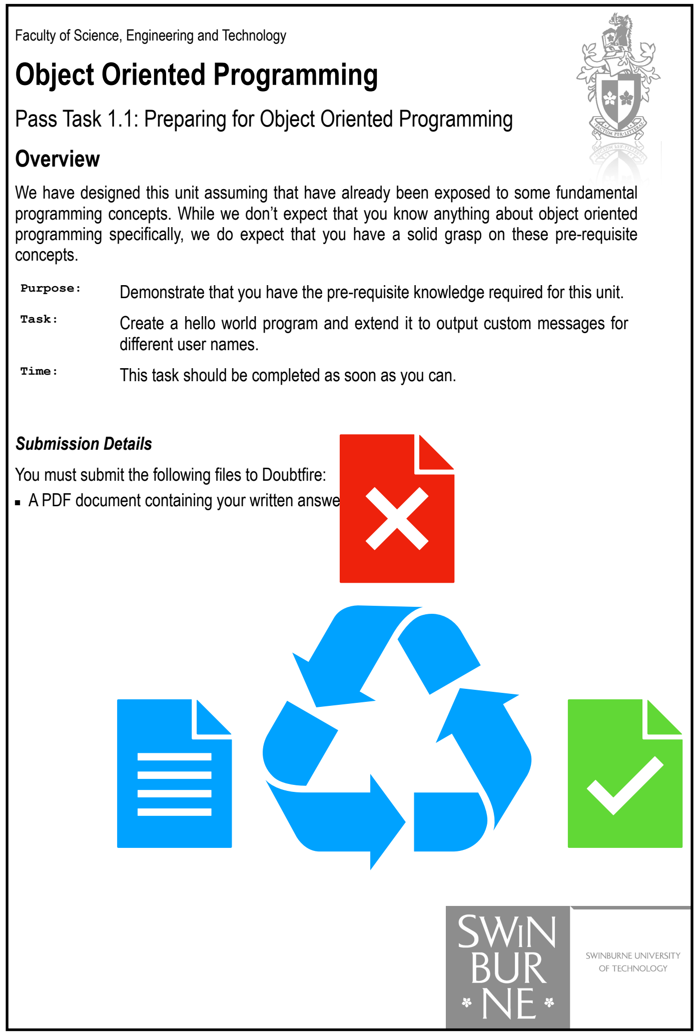
3rd we **Assert**

E!

Your early SwinAdventure submissions will use this testing technique

Many are submitting their first tasks …

1.1P



Faculty of Science, Engineering and Technology

Object Oriented Programming

Pass Task 1.1: Preparing for Object Oriented Programming

Overview

We have designed this unit assuming that have already been exposed to some fundamental programming concepts. While we don't expect that you know anything about object oriented programming specifically, we do expect that you have a solid grasp on these pre-requisite concepts.

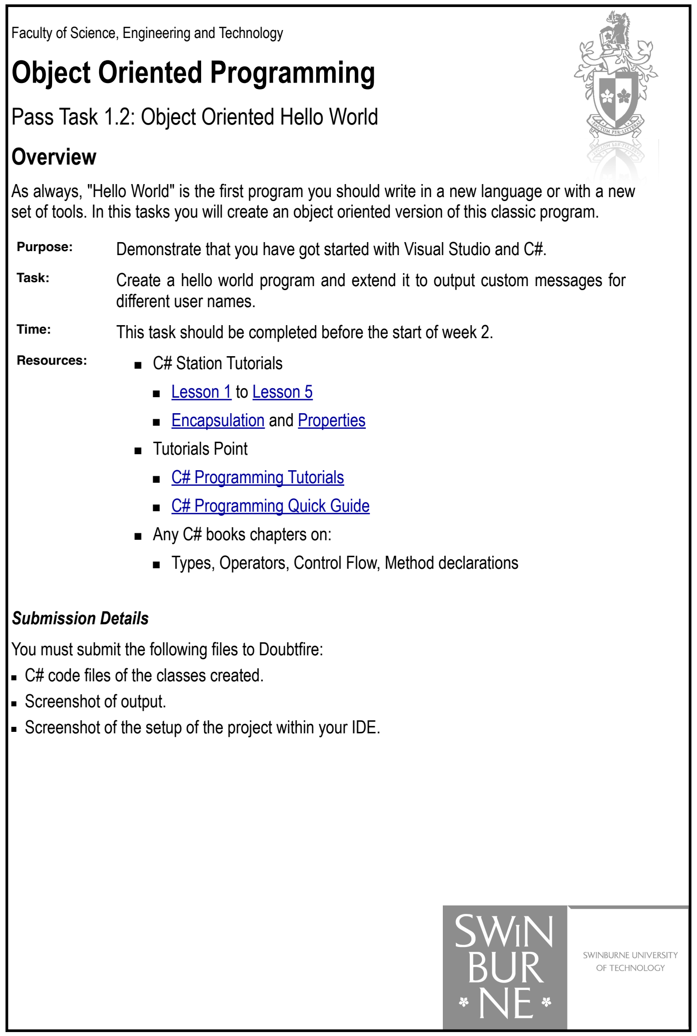
Purpose:	Demonstrate that you have the pre-requisite knowledge required for this unit.
Task:	Create a hello world program and extend it to output custom messages for different user names.
Time:	This task should be completed as soon as you can.

Submission Details

You must submit the following files to Doubtfire:

- A PDF document containing your written answe

1.2P



Faculty of Science, Engineering and Technology

Object Oriented Programming

Pass Task 1.2: Object Oriented Hello World

Overview

As always, "Hello World" is the first program you should write in a new language or with a new set of tools. In this tasks you will create an object oriented version of this classic program.

Purpose: Demonstrate that you have got started with Visual Studio and C#.

Task: Create a hello world program and extend it to output custom messages for different user names.

Time: This task should be completed before the start of week 2.

Resources:

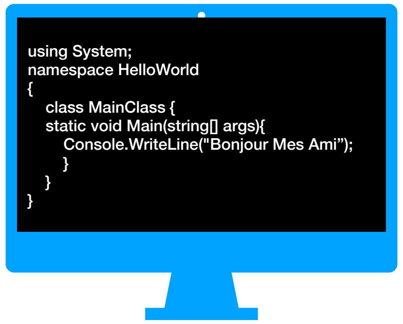
- C# Station Tutorials
  - Lesson 1 to Lesson 5
  - Encapsulation and Properties
- Tutorials Point
  - C# Programming Tutorials
  - C# Programming Quick Guide
- Any C# books chapters on:
  - Types, Operators, Control Flow, Method declarations

Submission Details

You must submit the following files to Doubtfire:

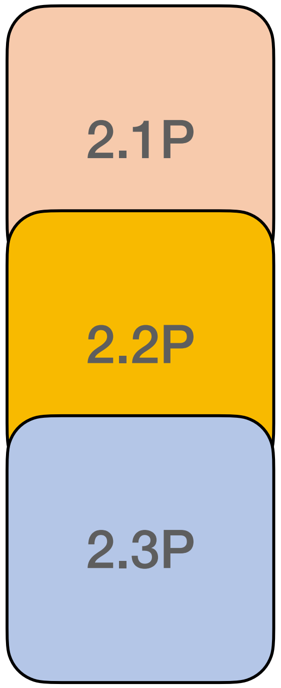
- C# code files of the classes created.
- Screenshot of output.
- Screenshot of the setup of the project within your IDE.

Today's 1:1s



```
using System;
namespace HelloWorld
{
    class MainClass {
    static void Main(string[] args){
        Console.WriteLine("Bonjour Mes Ami");
        }
    }
}
```

The week ahead



Let's review the process …

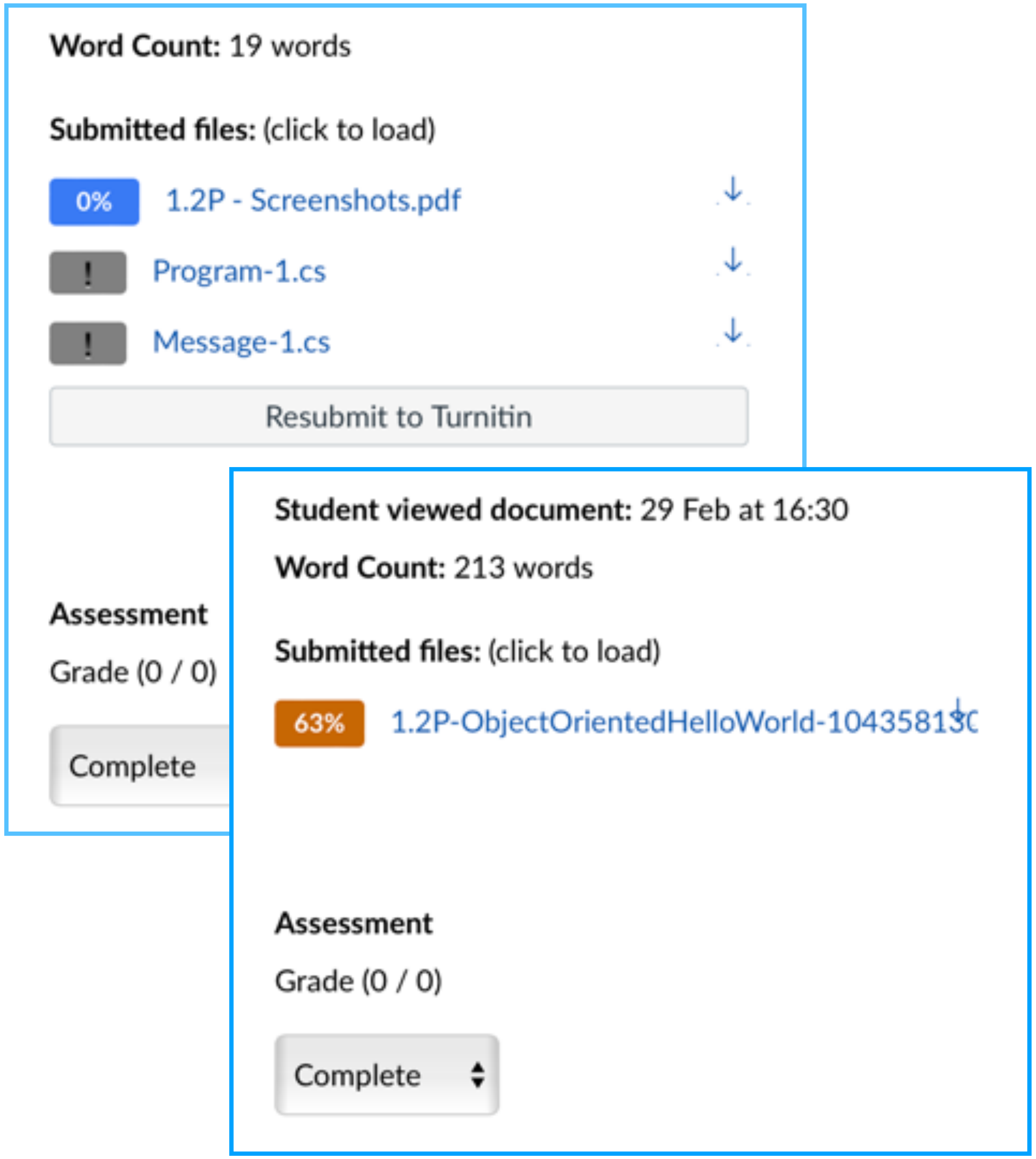
Submission Process 1/2

Sometimes you need to submit multiple files … including screenshots which can not be submitted individually as their filetypes are not accepted.

And neither do we accept .zip, .tar, .rar, et cetera.

So sometimes you will need to assemble a PDF file from the component parts:

- Code module(s) as formatted text
- Unit test module(s) as formatted text
- UML drawing
- Interaction diagrams
- Conceptual diagrams
- Evidence:
  - Screenshots e.g. breakpoints
  - SwinAdventure if no NUnit (sometimes)
  - ShapeKit Window screenshot(s)



Word Count: 19 words

Submitted files: (click to load)

0% 1.2P - Screenshots.pdf

! Program-1.cs

! Message-1.cs

Resubmit to Turnitin

Assessment

Grade (0 / 0)

Complete

Student viewed document: 29 Feb at 16:30

Word Count: 213 words

Submitted files: (click to load)

63% 1.2P-ObjectOrientedHelloWorld-10435813(

Assessment

Grade (0 / 0)

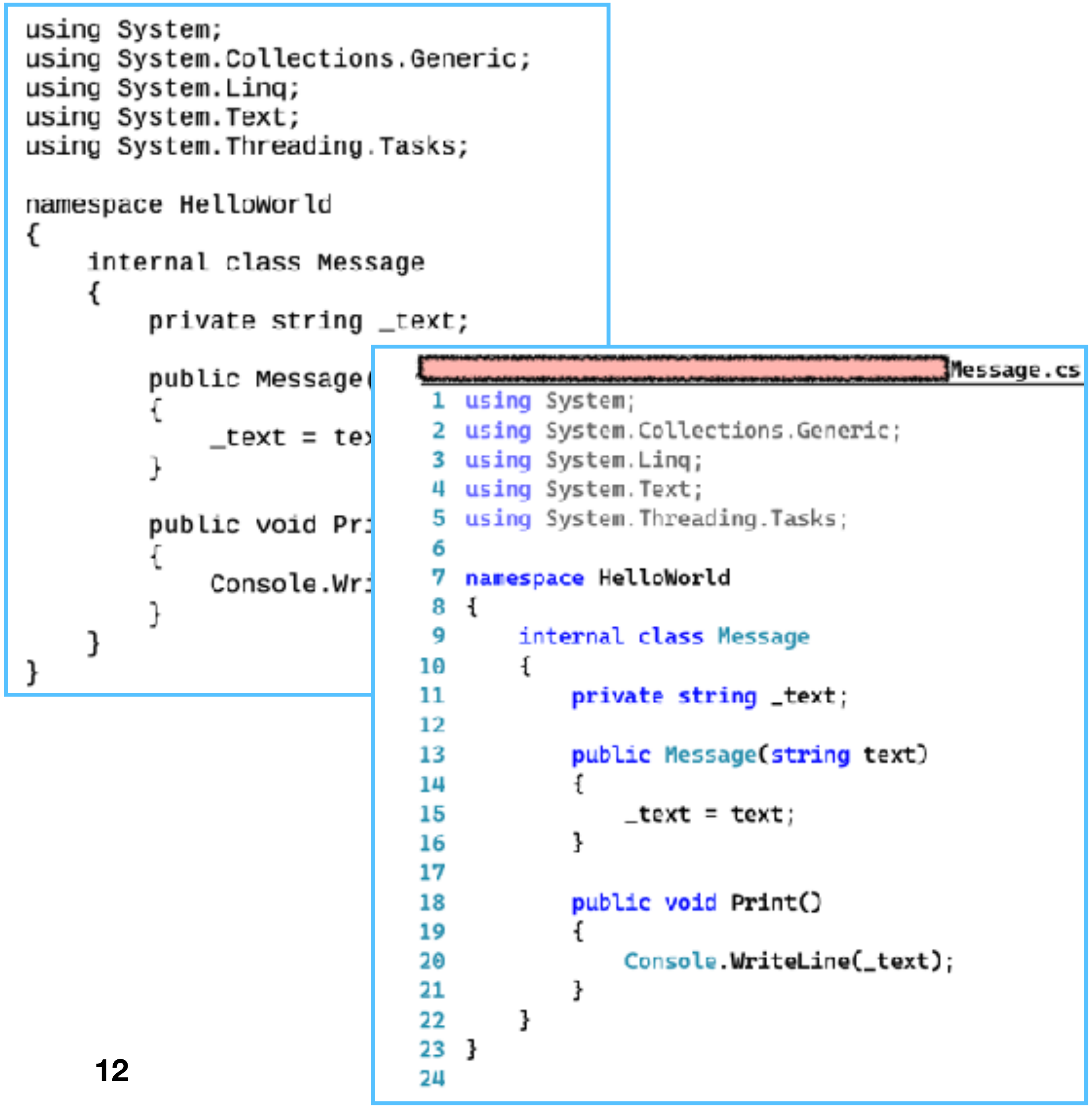
Complete

Submission Process 2/2

The preference is for you to include the source code(s) as a .pdf that holds the printout(s) of your source code file(s).

This should have two benefits:

- a better looking format of the code; and
- a sequence of labelled modules that can be scrolled through - soon some modules will exceed a single page



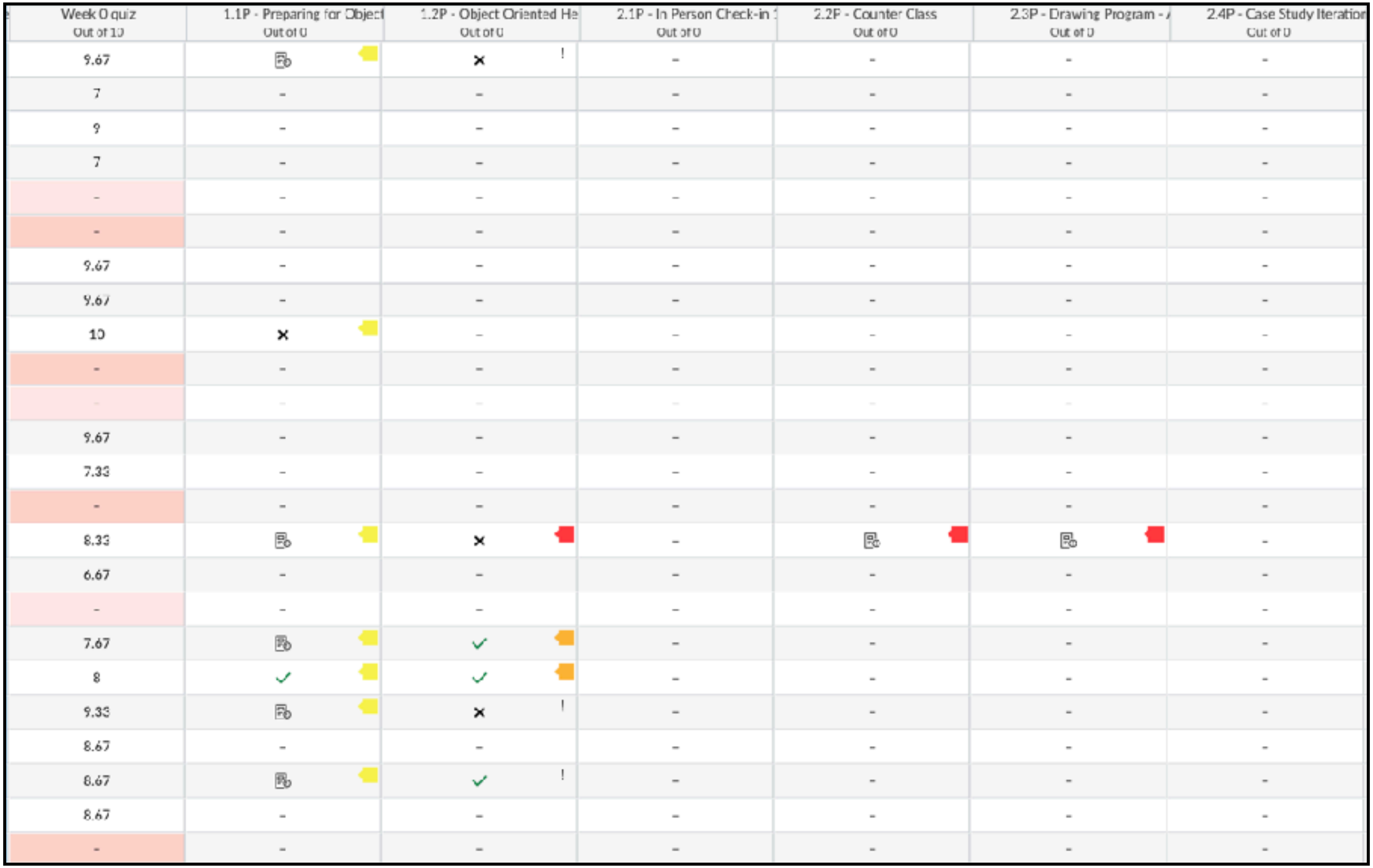
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloWorld
{
    internal class Message
    {
        private string _text;

        public Message(
        {
            _text = te
        }

        public void Pr
        {
            Console.Wr
        }
    }
}
```

[redacted] Message.cs

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

namespace HelloWorld
{
    internal class Message
    {
        private string _text;

        public Message(string text)
        {
            _text = text;
        }

        public void Print()
        {
            Console.WriteLine(_text);
        }
    }
}
```

So how are <u>we</u> doing ?

TBH <u>we</u> all could do better !



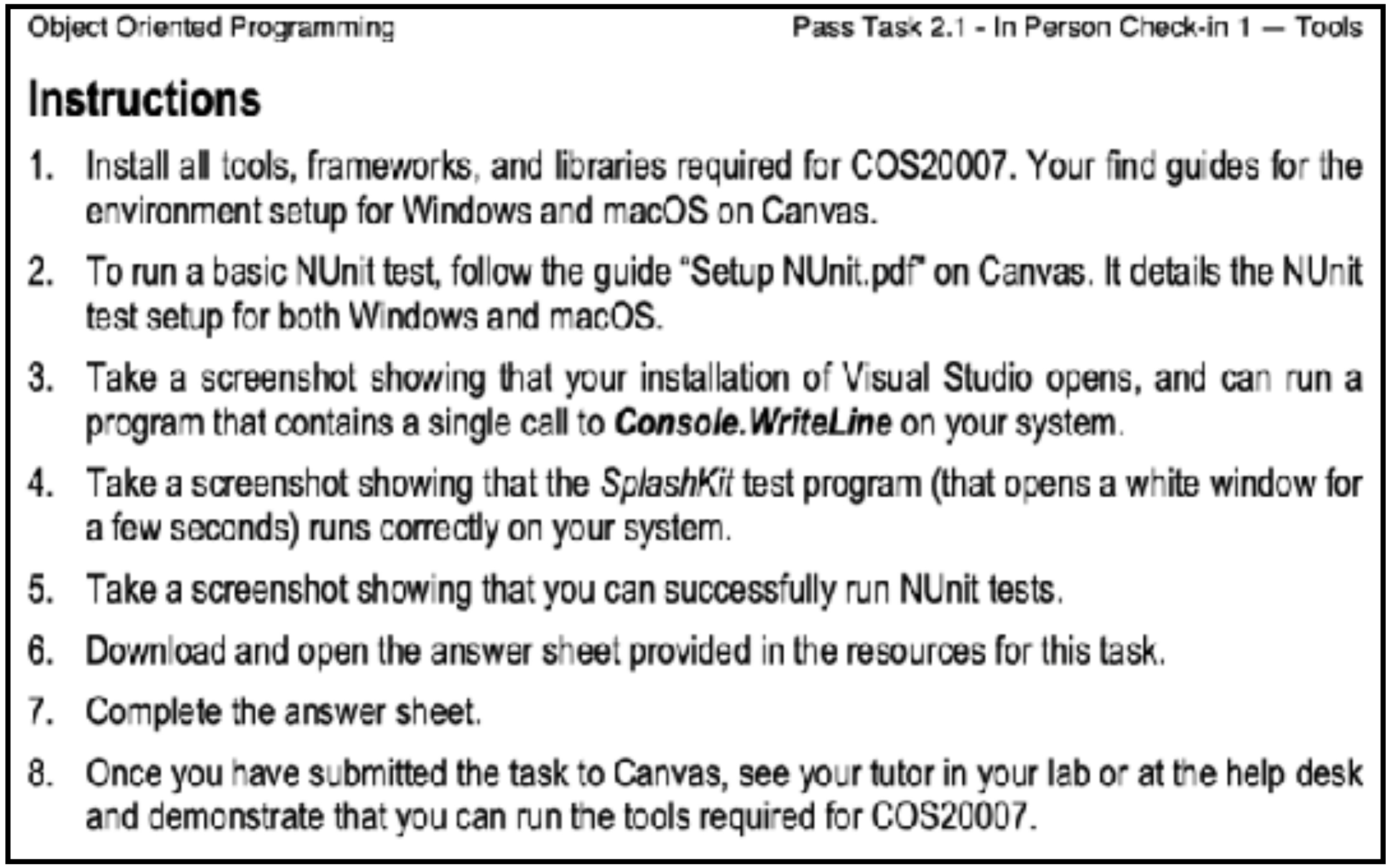
Week 0 quiz Out of 10	1.1P - Preparing for Object Out of 0	1.2P - Object Oriented He Out of 0	2.1P - In Person Check-in 1 Out of 0	2.2P - Counter Class Out of 0	2.3P - Drawing Program - A Out of 0	2.4P - Case Study Iteration Out of 0
9.67	📄 🟨	× !	-	-	-	-
7	-	-	-	-	-	-
9	-	-	-	-	-	-
7	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
9.67	-	-	-	-	-	-
9.67	-	-	-	-	-	-
10	× 🟨	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
9.67	-	-	-	-	-	-
7.33	-	-	-	-	-	-
-	-	-	-	-	-	-
8.33	📄 🟨	× 🟥	-	📄 🟥	📄 🟥	-
6.67	-	-	-	-	-	-
-	-	-	-	-	-	-
7.67	📄 🟨	✓ 🟧	-	-	-	-
8	✓ 🟨	✓ 🟧	-	-	-	-
9.33	📄 🟨	× !	-	-	-	-
8.67	-	-	-	-	-	-
8.67	📄 🟨	✓ !	-	-	-	-
8.67	-	-	-	-	-	-
-	-	-	-	-	-	-

NB the class list’s sequence has been randomised to protect the innocent

So let us start ticking off those in person checkins … Task 2.1P

One by one, please come forward for your interviews

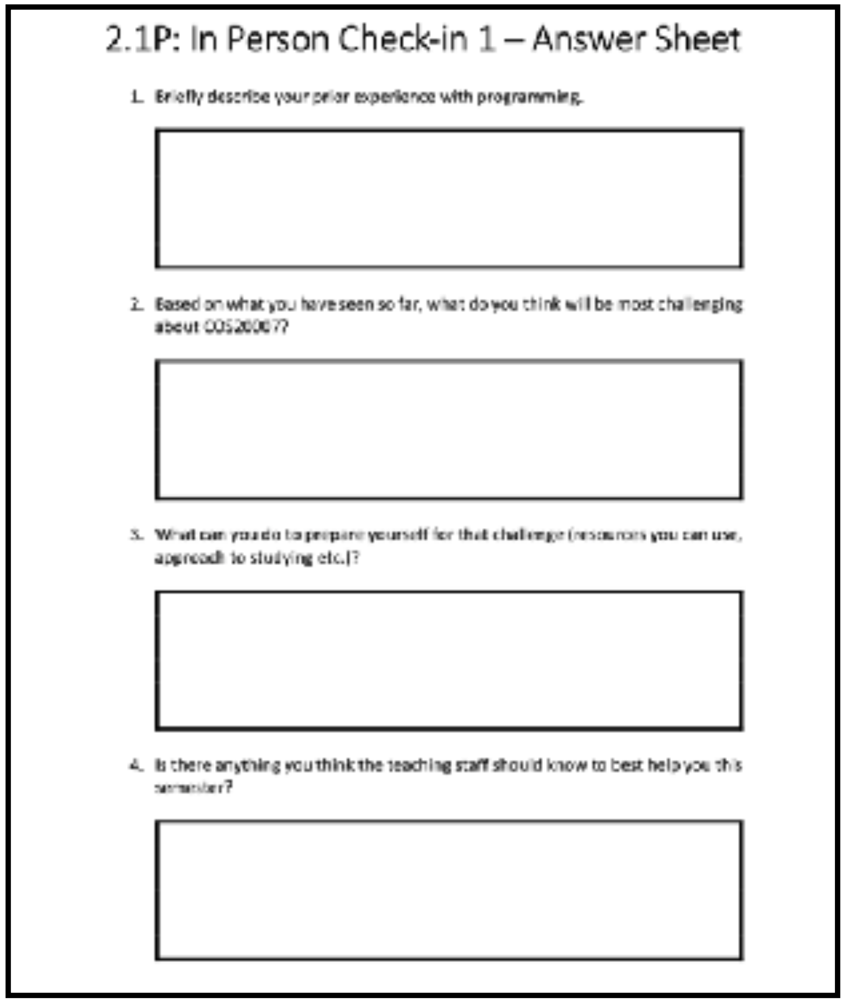
5 minutes each



Object Oriented Programming — Pass Task 2.1 - In Person Check-in 1 — Tools

Instructions

1. Install all tools, frameworks, and libraries required for COS20007. Your find guides for the environment setup for Windows and macOS on Canvas.
2. To run a basic NUnit test, follow the guide "Setup NUnit.pdf" on Canvas. It details the NUnit test setup for both Windows and macOS.
3. Take a screenshot showing that your installation of Visual Studio opens, and can run a program that contains a single call to ***Console.WriteLine*** on your system.
4. Take a screenshot showing that the *SplashKit* test program (that opens a white window for a few seconds) runs correctly on your system.
5. Take a screenshot showing that you can successfully run NUnit tests.
6. Download and open the answer sheet provided in the resources for this task.
7. Complete the answer sheet.
8. Once you have submitted the task to Canvas, see your tutor in your lab or at the help desk and demonstrate that you can run the tools required for COS20007.



2.1P: In Person Check-in 1 – Answer Sheet

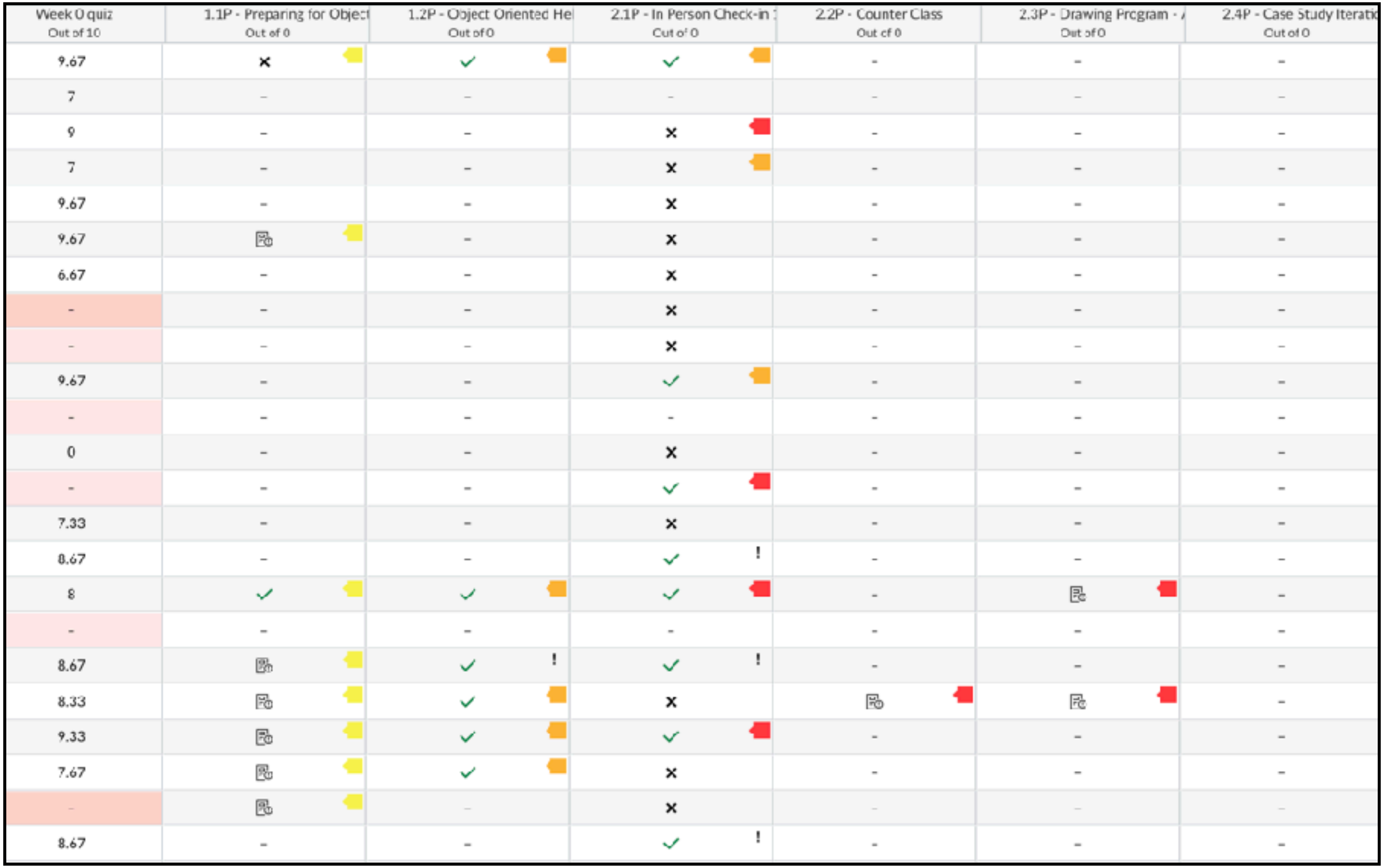
1. Briefly describe your prior experience with programming.

2. Based on what you have seen so far, what do you think will be most challenging about COS20007?

3. What can you do to prepare yourself for that challenge (resources you can use, approach to studying etc.)?

4. Is there anything you think the teaching staff should know to best help you this semester?

And after the tutorial, passageway interviews, and sessional office cleanup we are all looking so much better!



Week 0 quiz Out of 10	1.1P - Preparing for Objec Out of 0	1.2P - Object Oriented He Out of 0	2.1P - In Person Check-in Out of 0	2.2P - Counter Class Out of 0	2.3P - Drawing Program - Out of 0	2.4P - Case Study Iteratio Out of 0
9.67	× 🟨	✓ 🟧	✓ 🟧	-	-	-
7	-	-	-	-	-	-
9	-	-	× 🟥	-	-	-
7	-	-	× 🟧	-	-	-
9.67	-	-	×	-	-	-
9.67	🗎 🟨	-	×	-	-	-
6.67	-	-	×	-	-	-
-	-	-	×	-	-	-
-	-	-	×	-	-	-
9.67	-	-	✓ 🟧	-	-	-
-	-	-	-	-	-	-
0	-	-	×	-	-	-
-	-	-	✓ 🟥	-	-	-
7.33	-	-	×	-	-	-
8.67	-	-	✓ !	-	-	-
8	✓ 🟨	✓ 🟧	✓ 🟥	-	🗎 🟥	-
-	-	-	-	-	-	-
8.67	🗎 🟨	✓ !	✓ !	-	-	-
8.33	🗎 🟨	✓ 🟧	×	🗎 🟥	🗎 🟥	-
9.33	🗎 🟨	✓ 🟧	✓ 🟥	-	-	-
7.67	🗎 🟨	✓ 🟧	×	-	-	-
-	🗎 🟨	-	×	-	-	-
8.67	-	-	✓ !	-	-	-

NB the class list's sequence has been randomised yet again to protect the innocent