NUNIT SETUP

COS20007

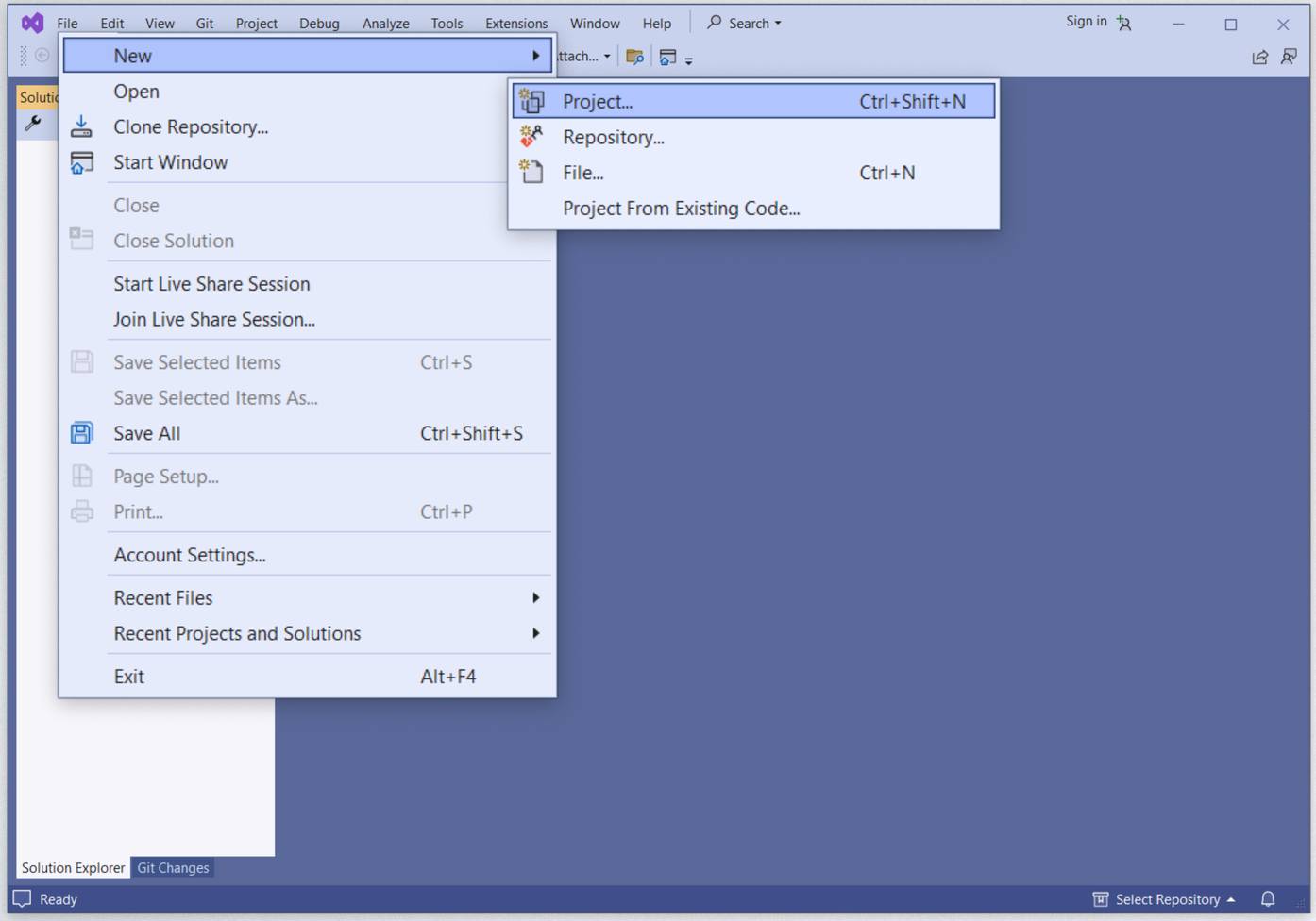
© Dr Markus Lumpe, 2024

RUNNING NUNIT TESTS

- Unit testing is a software testing method by which individual units of source code are tested to determine whether they fit together.
- NUnit is a framework to perform unit tests. It is available for many programming languages, including C#.
- An NUnit test requires the source code (or application) to be tested and the an NUnit test program that implements the desired unit tests.
- In the following, we first create a simple application (i.e., QueueApplication) and then setup an NUnit test in another application, called TestQueue, to perform a unit test on QueueApplication.

© Dr Markus Lumpe, 2024

CREATE A NEW PROJECT



1. Open Visual Studio and continue without code.

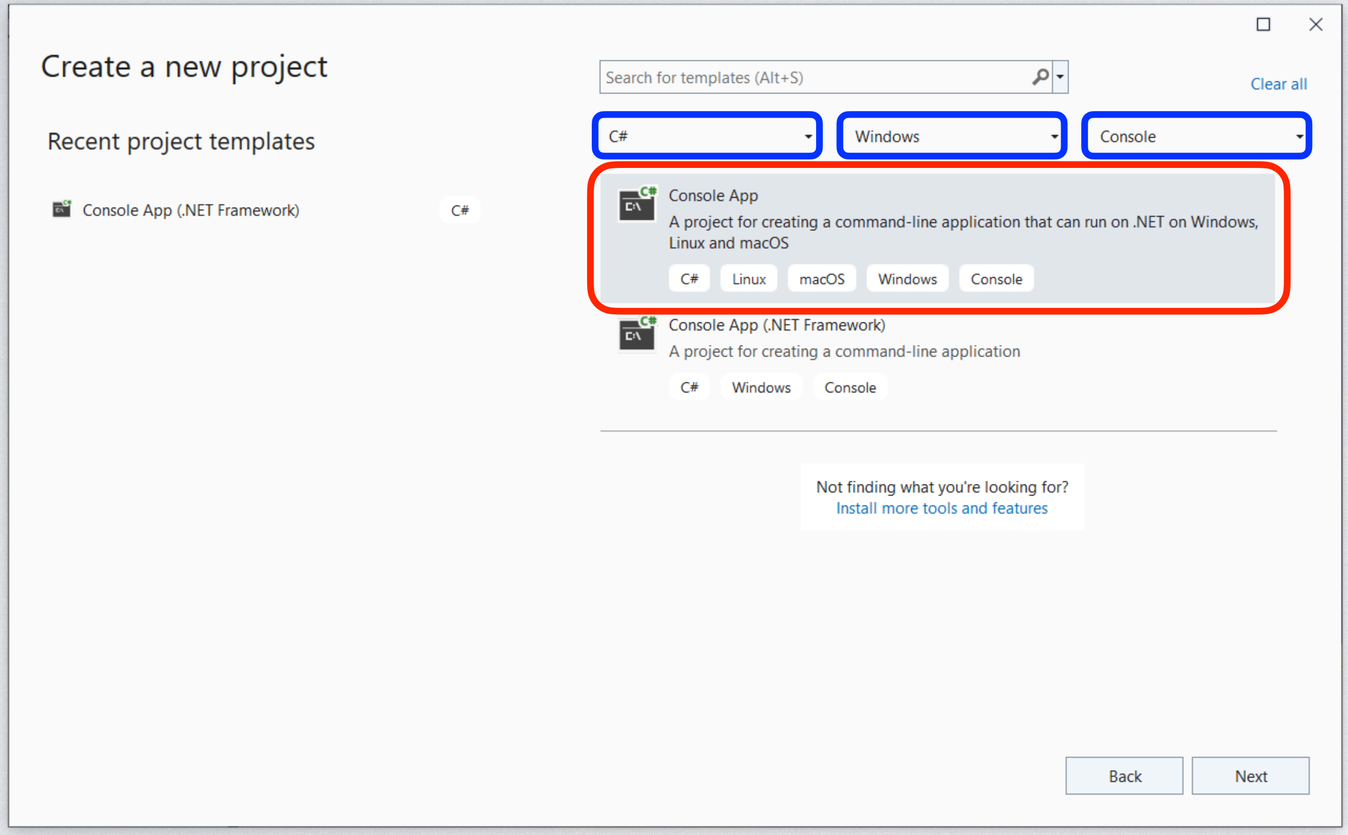
2. Select File/New

3. Finally select Project

Please note that your mouse pointer have to hover over the menus to remain visible.

© Dr Markus Lumpe, 2024

SET UP A CONSOLE APP

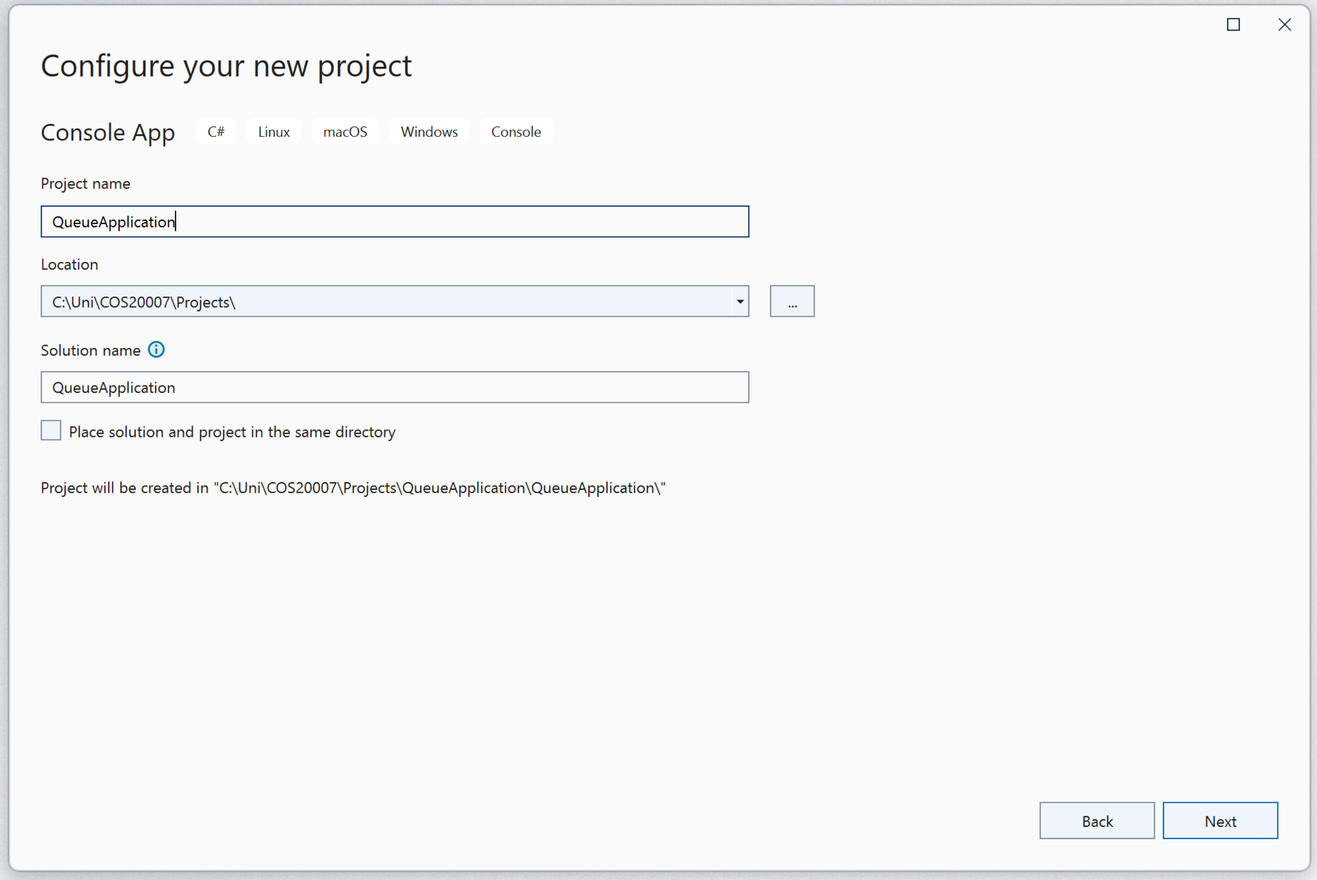


- Visual Studio offers a multitude of project types.
- In COS20007, we are working only with one project type: Console App.
- You can use the dropdown list to restrict the project type view to just the two shown on the left. Select
  - C# for Language
  - Windows for Platform
  - Console for Project Type

Please note that the project type Console App is a project template for .NET applications that can run on any CPU supporting Windows, Linux and MacOS.

© Dr Markus Lumpe, 2024

SETTING UP PROJECT NAME AND LOCATION



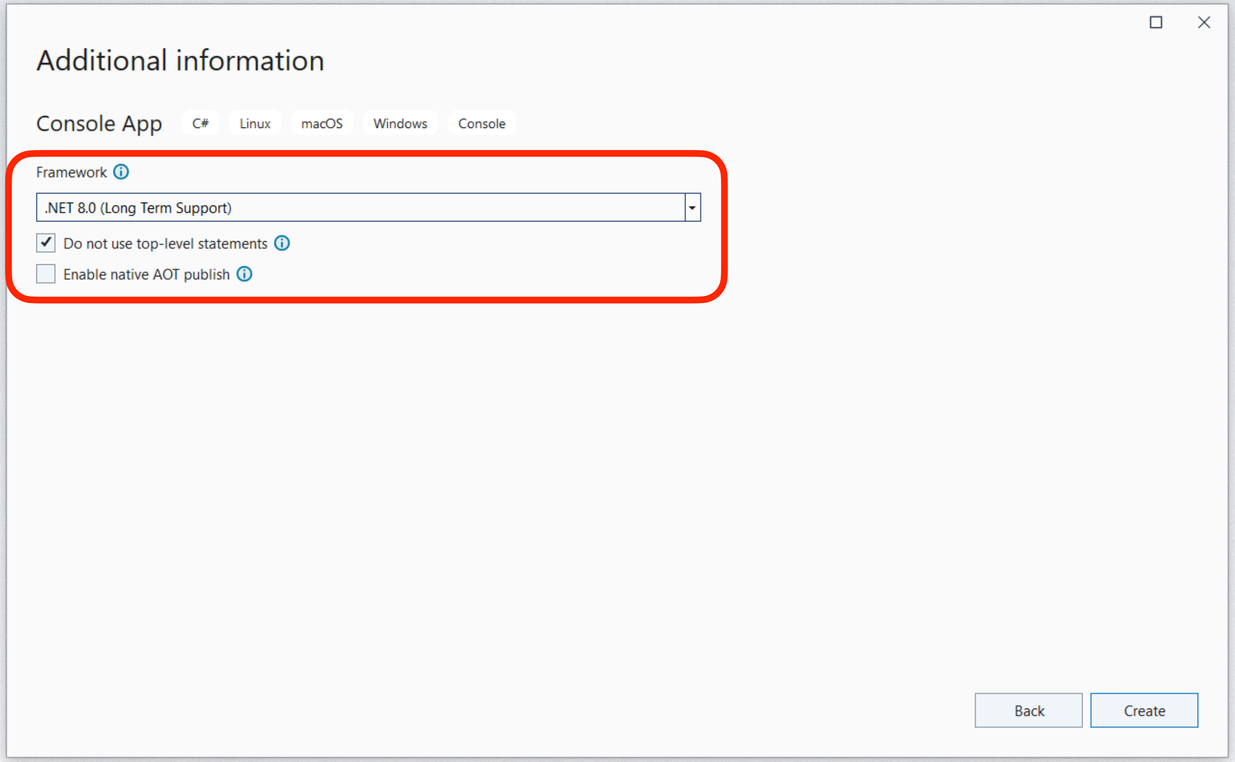
1. Select the project location. Click on … box. Never accept the Visual Studio default. Projects in this presentation a stored in the folder C:\Uni\COS20007\Projects. Choose a folder that you can easily find again. You may even consider a Cloud resource.

2. Give the project a name. Here the name is QueueApplication. Any meaningful name will do.

3. You can change the solution name also. This is useful when a solution contains several projects. Here, the solution name coincides with the project name as the solution only hosts one project.

© Dr Markus Lumpe, 2024

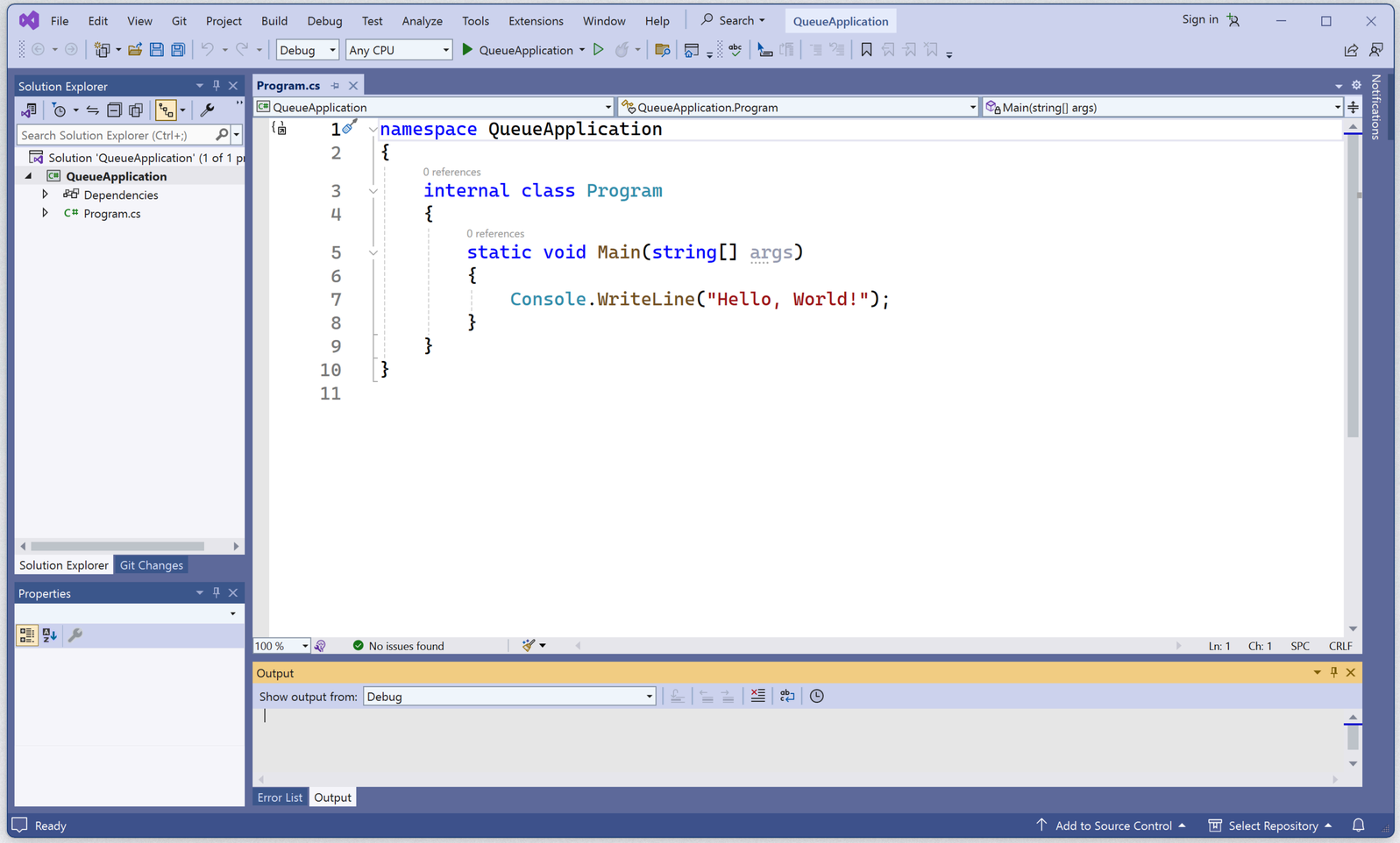
SELECT FRAMEWORK



- When creating a Console App for .NET, we need to select the .NET framework that we want to work with.
- You can safely choose the latest framework (.NET 8.0 LTS). At a minimum, COS20007 requires .NET 6.0.
- Check "Do not use top-level statements." Visual Studio creates a "Hello, World!" program this way.

© Dr Markus Lumpe, 2024

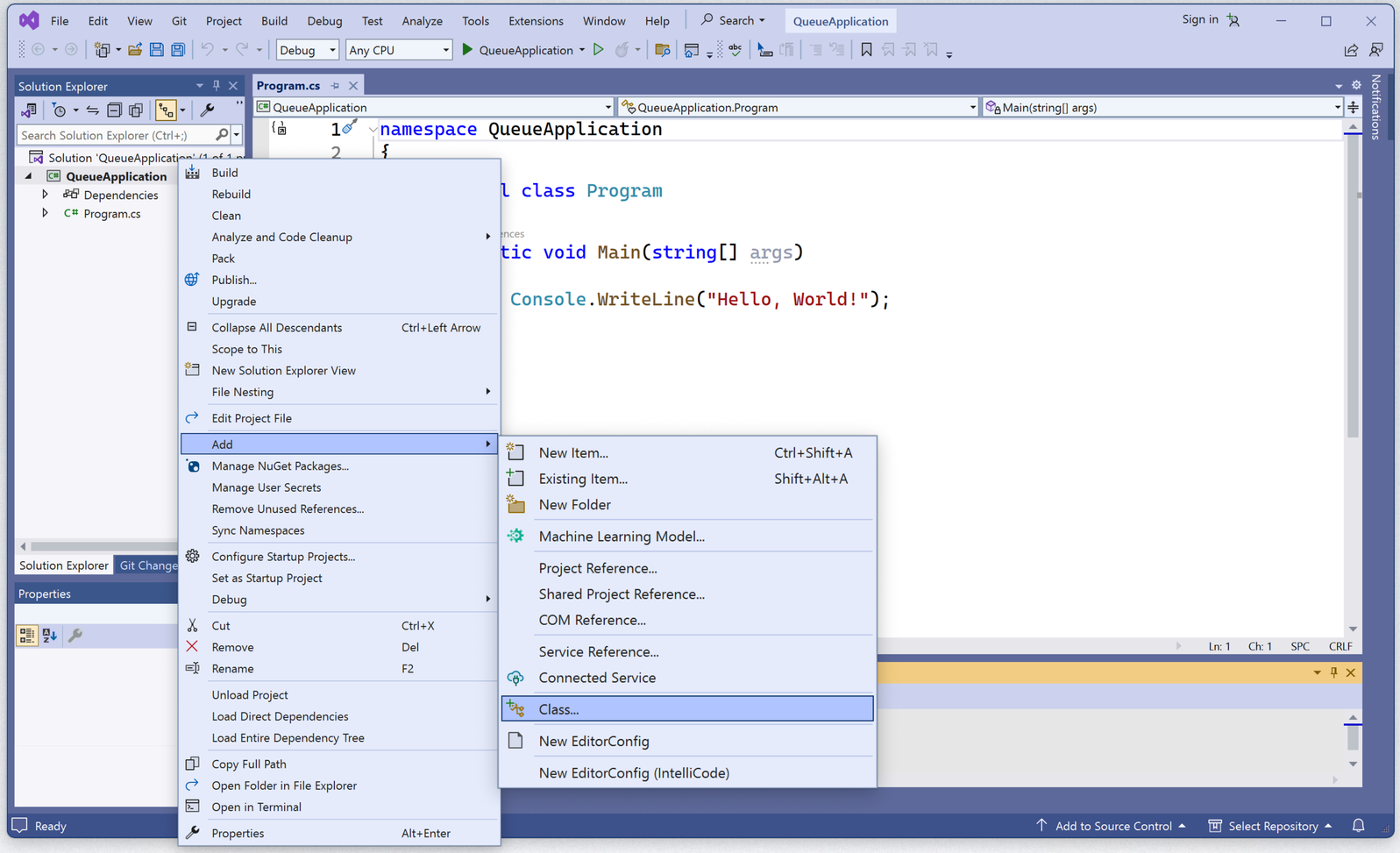
GENERATED PROJECT



- The generated project template is a fully functional C# console application.

© Dr Markus Lumpe, 2024

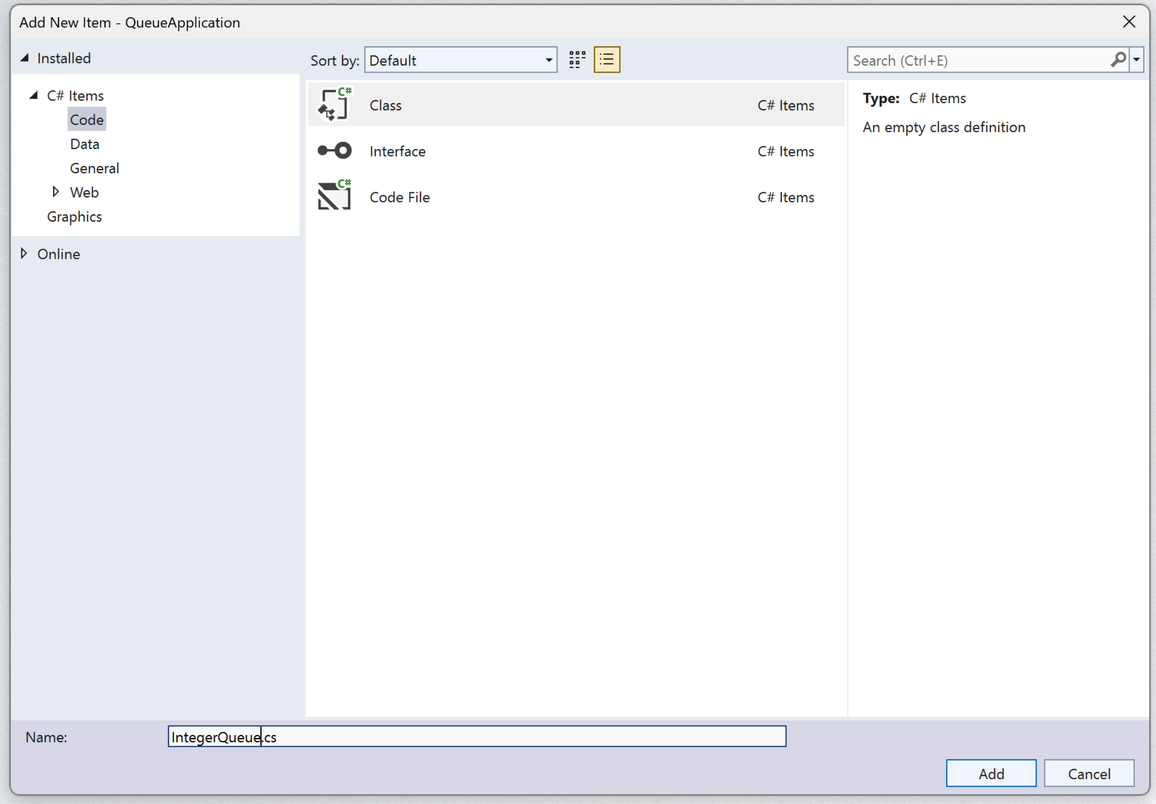
ADD CLASS INTEGERQUEUE



- Add a new class. Right-click project and choose Add/Class…

© Dr Markus Lumpe, 2024

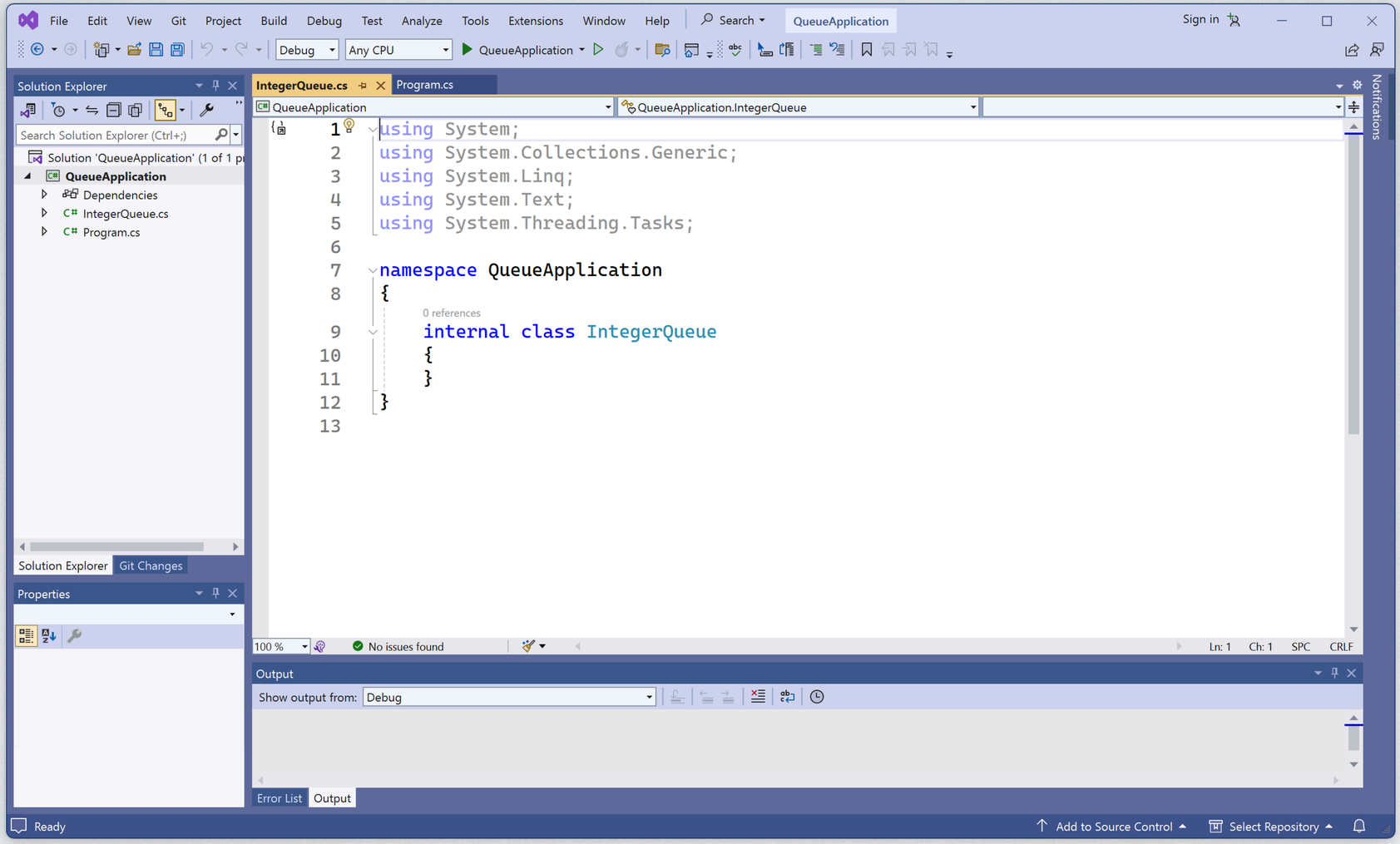
INTEGERQUEUE.CS



- Choose as C# Item Code/Class
- Name the file IntegerQueue.cs. This creates the skeleton for class IntegerQueue.

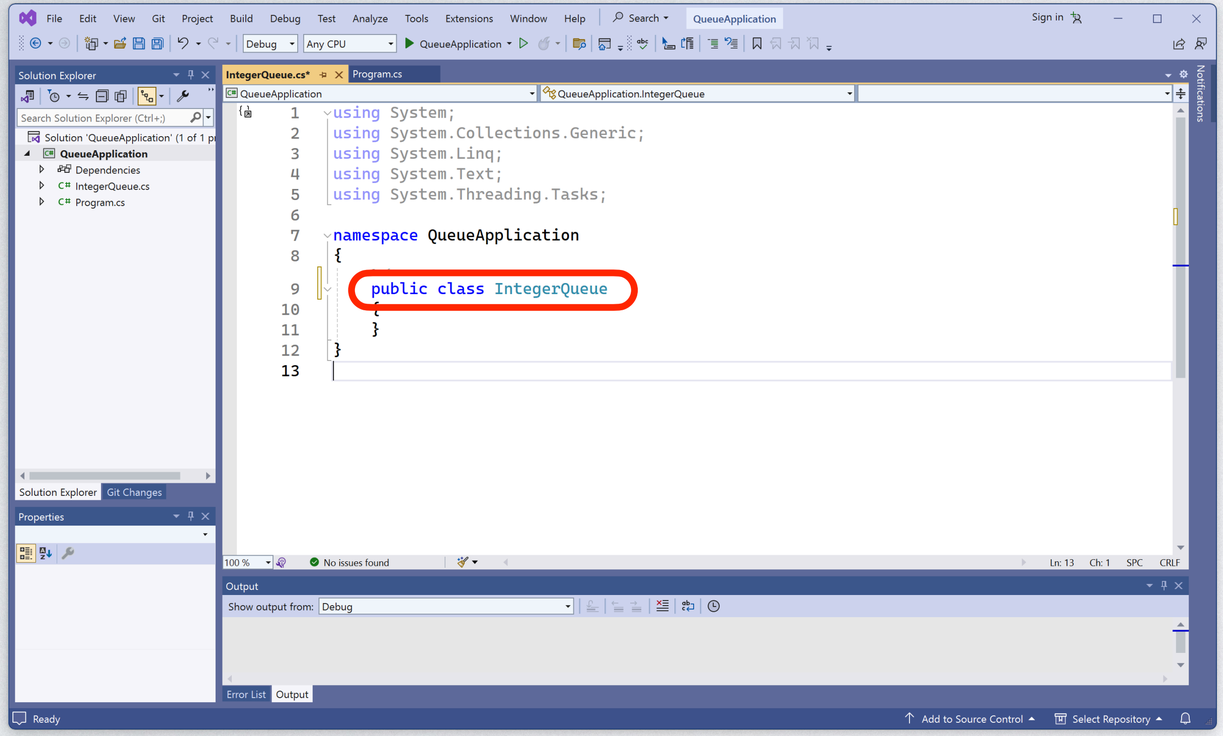
© Dr Markus Lumpe, 2024

GENERATED INTEGERQUEUE CLASS



© Dr Markus Lumpe, 2024

MAKE CLASS PUBLIC

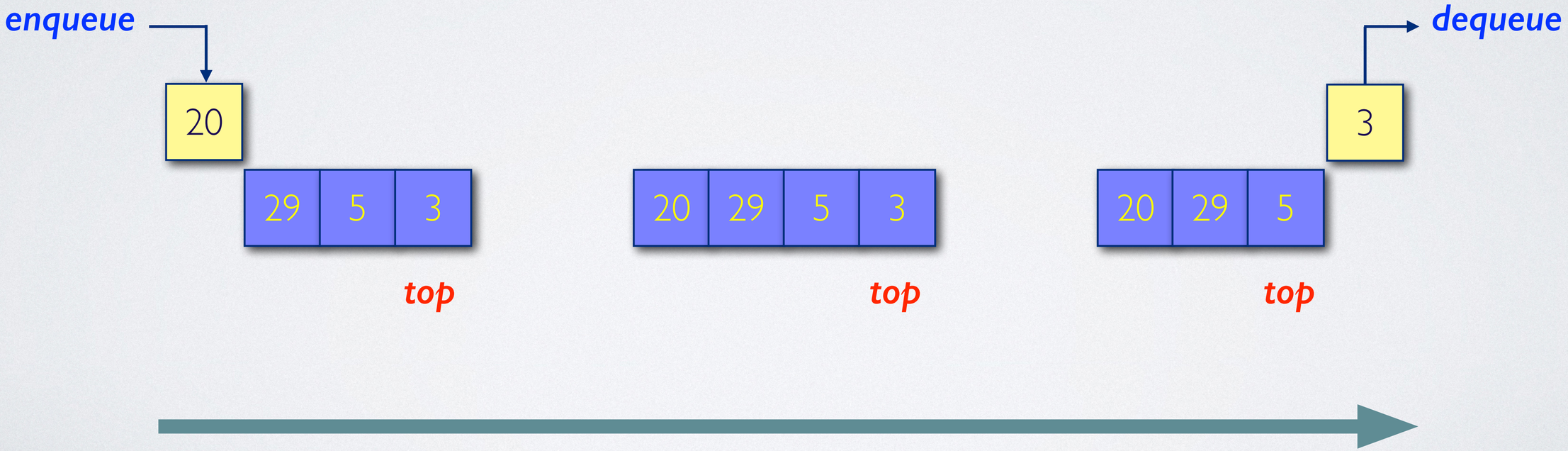


- Initially, IntegerQueue is marked internal. This means it can only be seen within the current application.
- Normally, this is fine. But we want to test the application with NUnit.
- NUnit tests are separate applications. If we want an NUnit test to see class IntegerQueue, then we need to mark class IntegerQueue public.
- Making a class public means that it becomes visual/accessible throughout the host application and any application the references the host application.

© Dr Markus Lumpe, 2024

ELEMENTARY DATA STRUCTURE QUEUE

- A queue is a dynamic set in which the element removed is always the one that has been in the set for the longest time: the queue implements a fist-in-first-out, or FIFO, policy.



© Dr Markus Lumpe, 2024

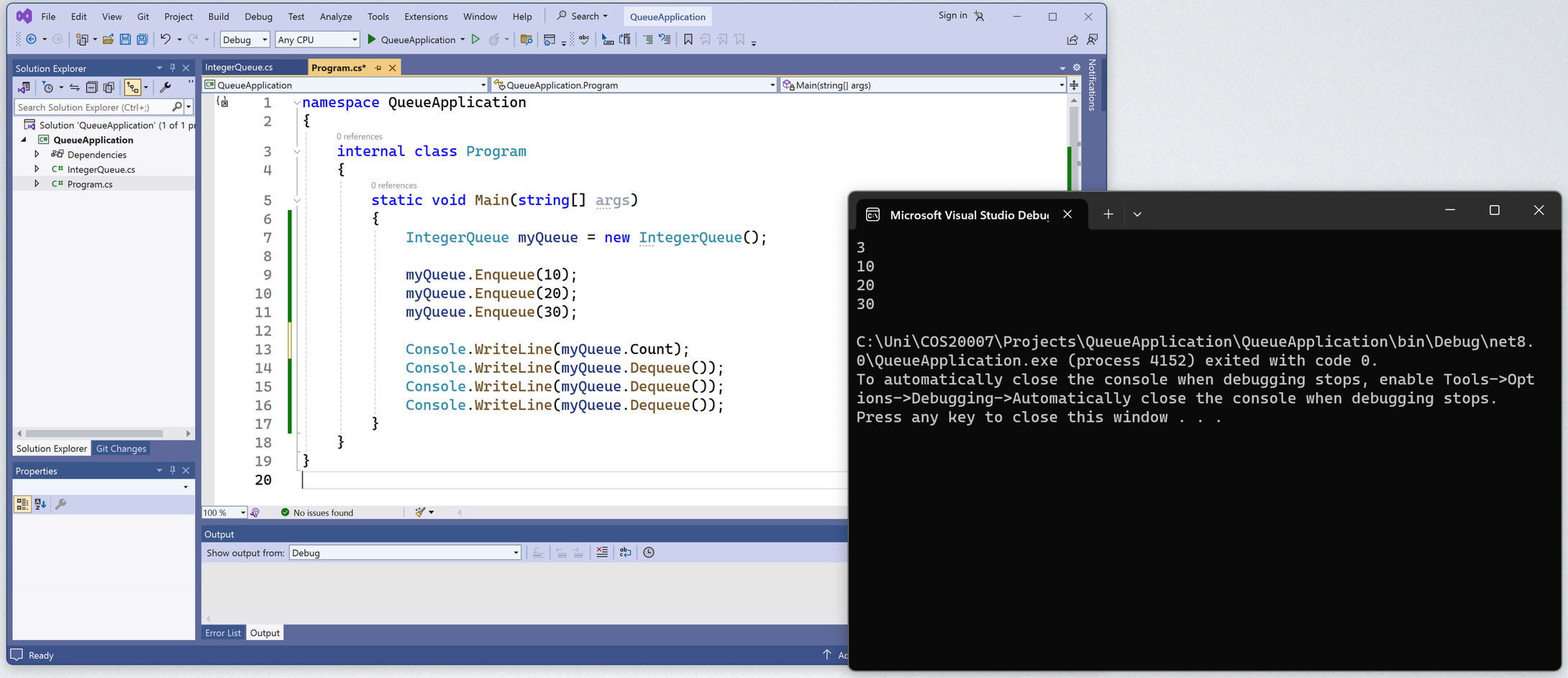
INTEGERQUEUE

```
public class IntegerQueue
{
    private List<int> _elements;

    public IntegerQueue()
    {
        _elements = new List<int>();
    }

    public int Count
    {
        get
        {
            return _elements.Count;
        }
    }

    public void Enqueue(int value)
    {
        _elements.Add(value);                // insert at end
    }

    public int Dequeue()
    {
        int result = _elements[0];           // remove at front
        _elements.RemoveAt(0);
        return result;
    }
}
```

- IntegerQueue uses a List to store elements. The list is initially empty.
- The Count property returns the number of elements in the queue. We just forward the request to list's Count property.
- Enqueue inserts an element at the end of the queue. We map this operations to list's Add method.
- Dequeue removes the first element in the queue. To achieve this, we store the first element in the list in a temporary variable result. Next, we remove the first element from the list and return result.

© Dr Markus Lumpe, 2024

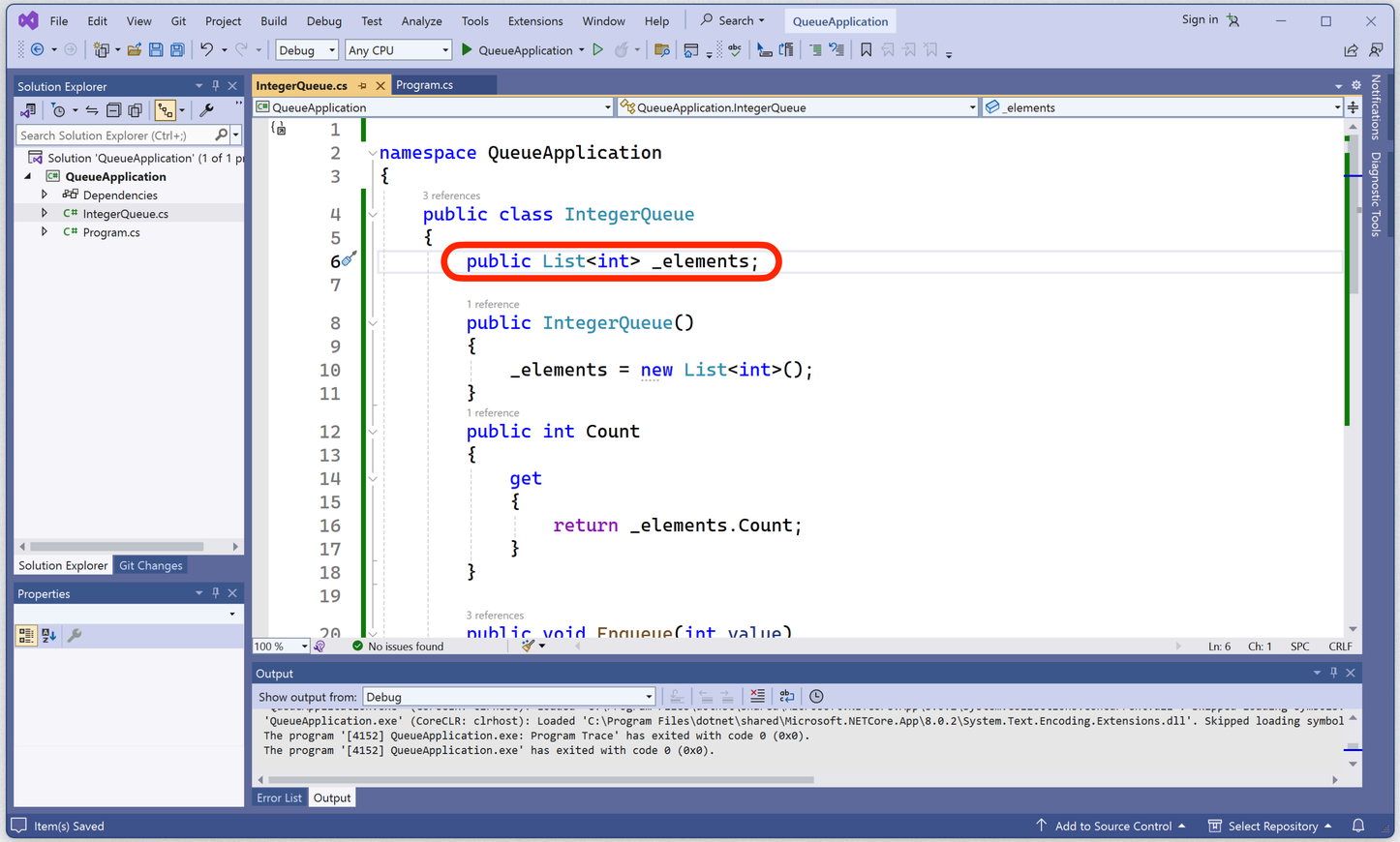
CHANGE MAIN



- In Main, we define a local variable myQueue and perform some queue operations.

© Dr Markus Lumpe, 2024

MAKE MEMBERS PUBLIC (OPTIONAL)



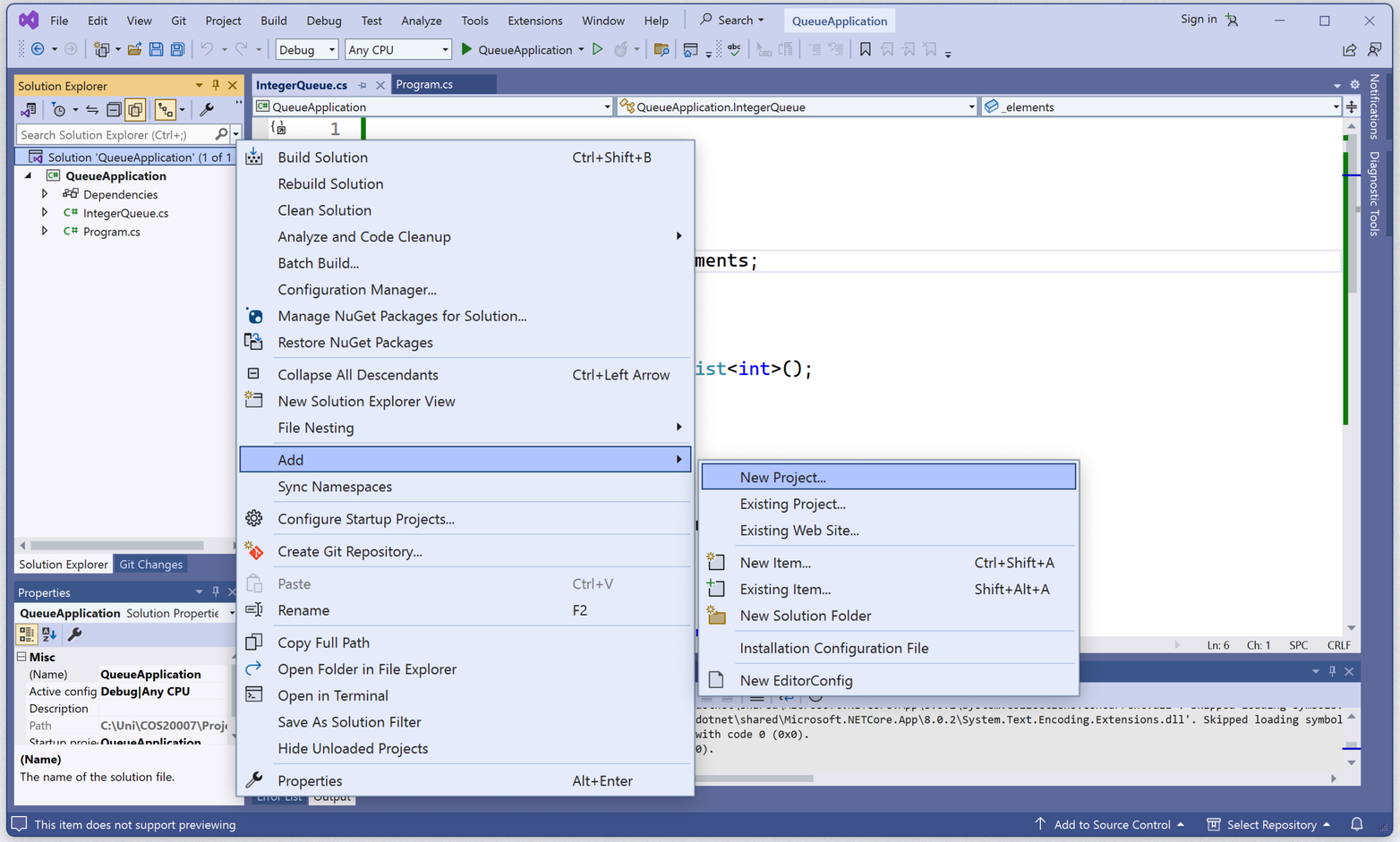
- To run NUnits tests, we may make all instance variables public. This will allow the tests to directly access the values of instance variables. Naturally, you need to revert the visibility back to private after the tests.

© Dr Markus Lumpe, 2024

NUNIT SETUP WINDOWS

© Dr Markus Lumpe, 2024

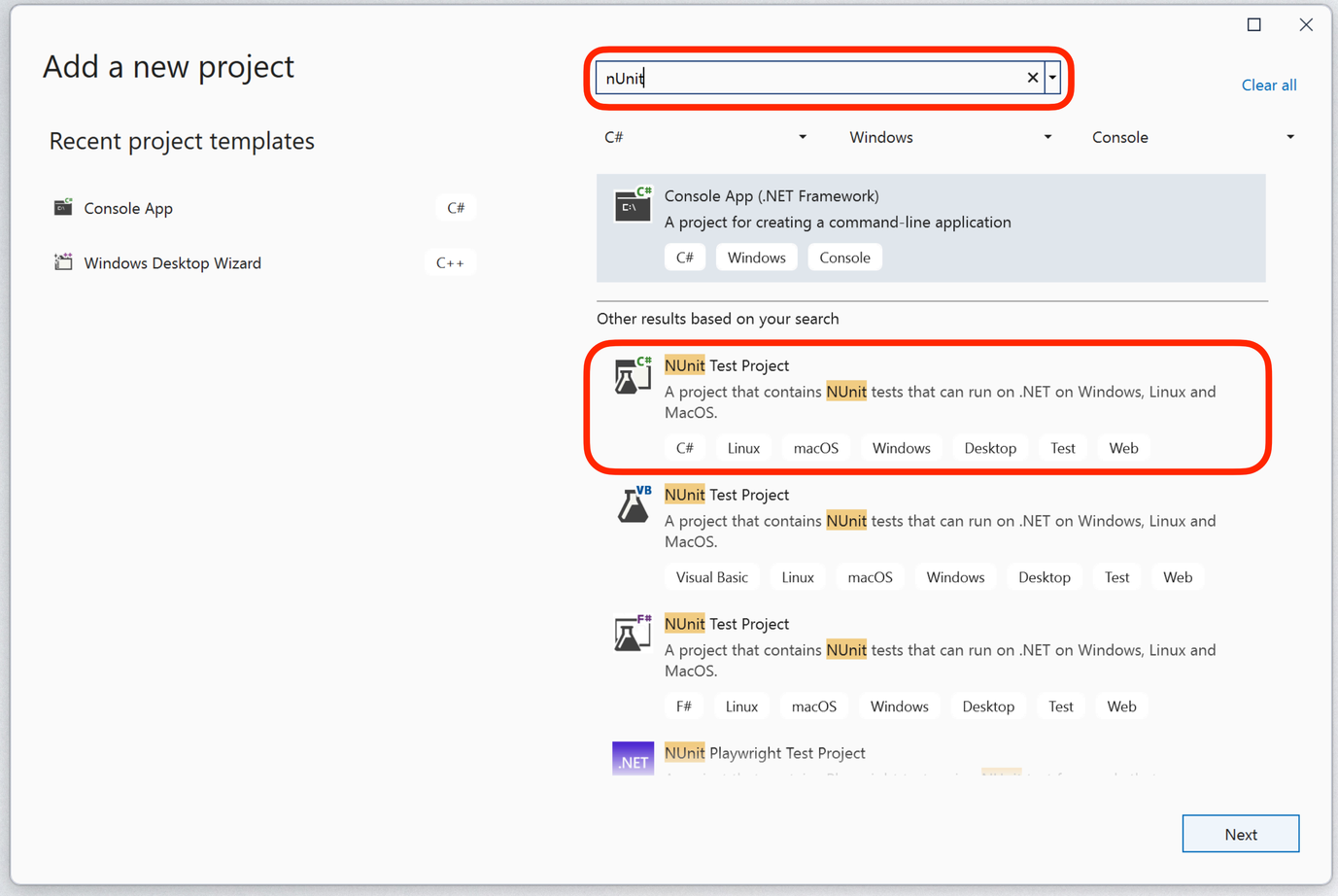
CREATE A NUNIT TEST



- Select the solution QueueApplication. Right-Click and choose Add/New Project…

© Dr Markus Lumpe, 2024

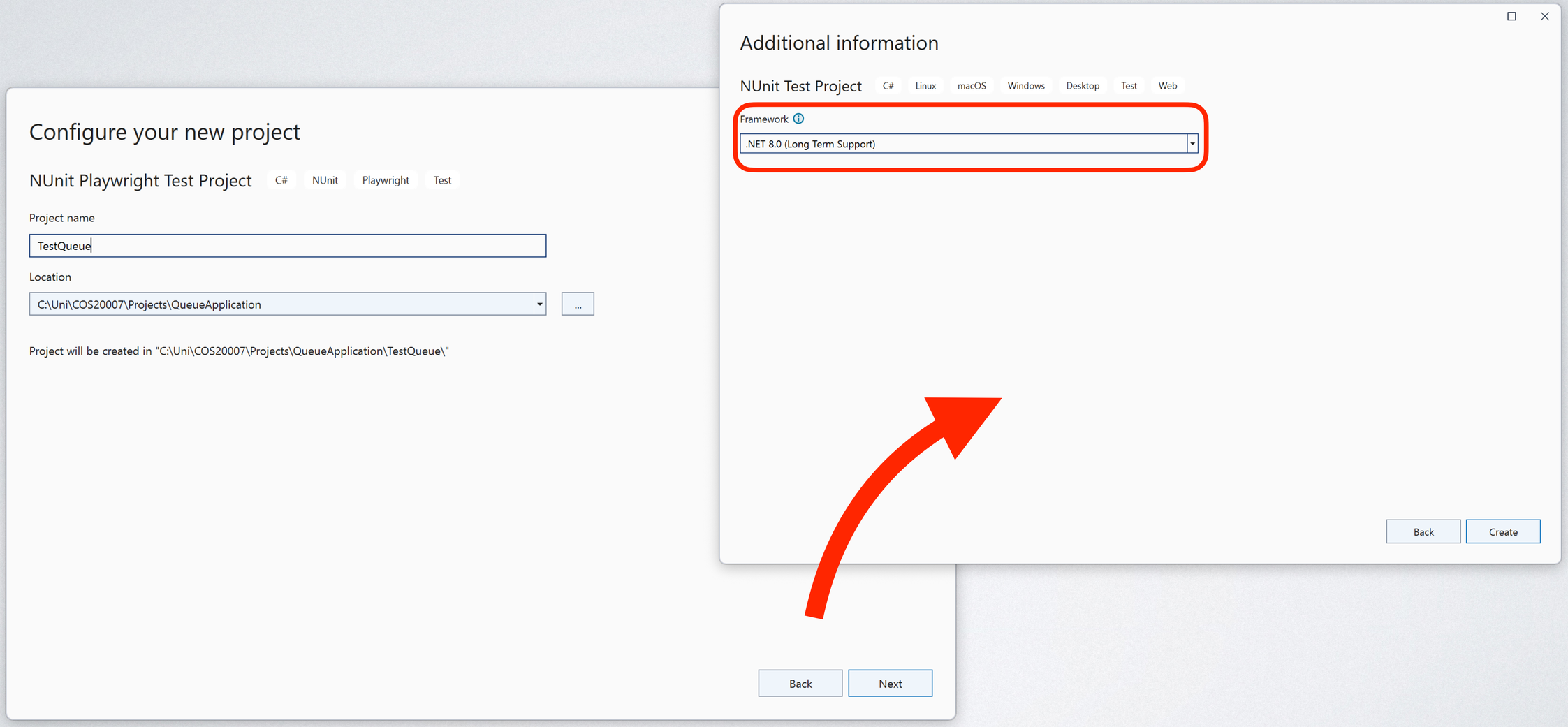
CONSOLE APP PROJECT



- Search for nUnit:
- Choose C# NUnit Test Project.

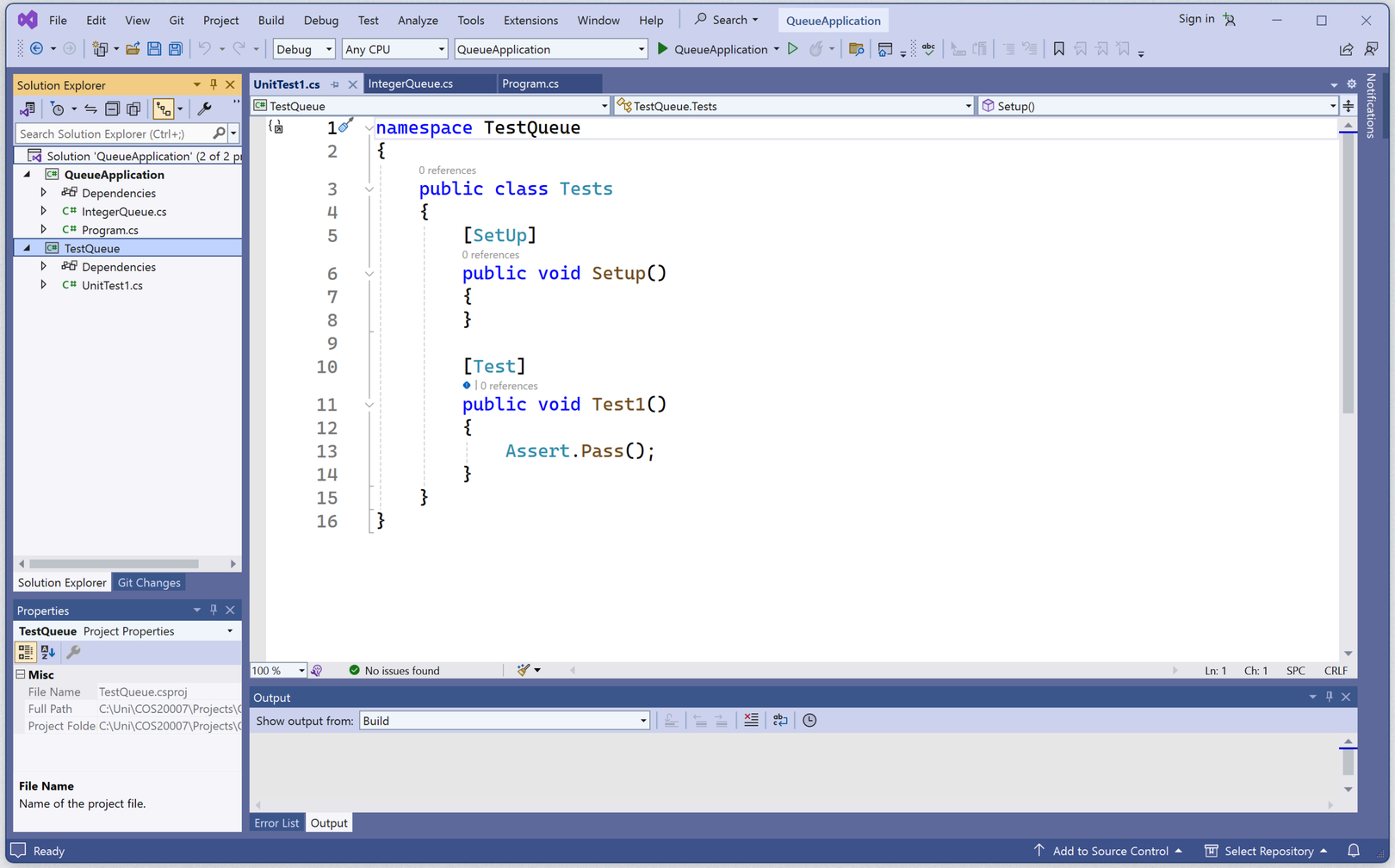
© Dr Markus Lumpe, 2024

PROJECT NAME: TESTQUEUE



© Dr Markus Lumpe, 2024

TEST SKELETON

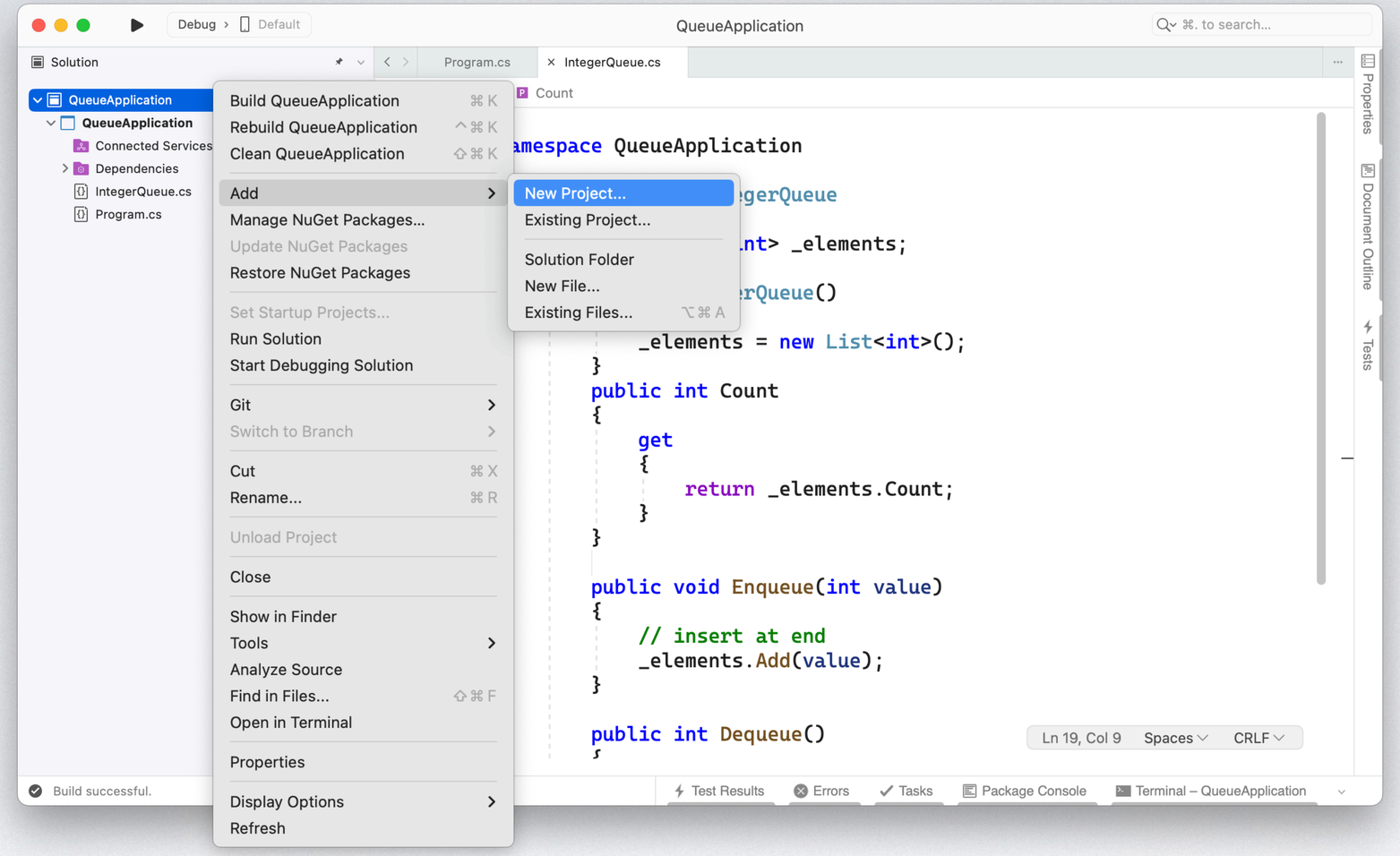


© Dr Markus Lumpe, 2024

NUNIT SETUP MACOS

© Dr Markus Lumpe, 2024

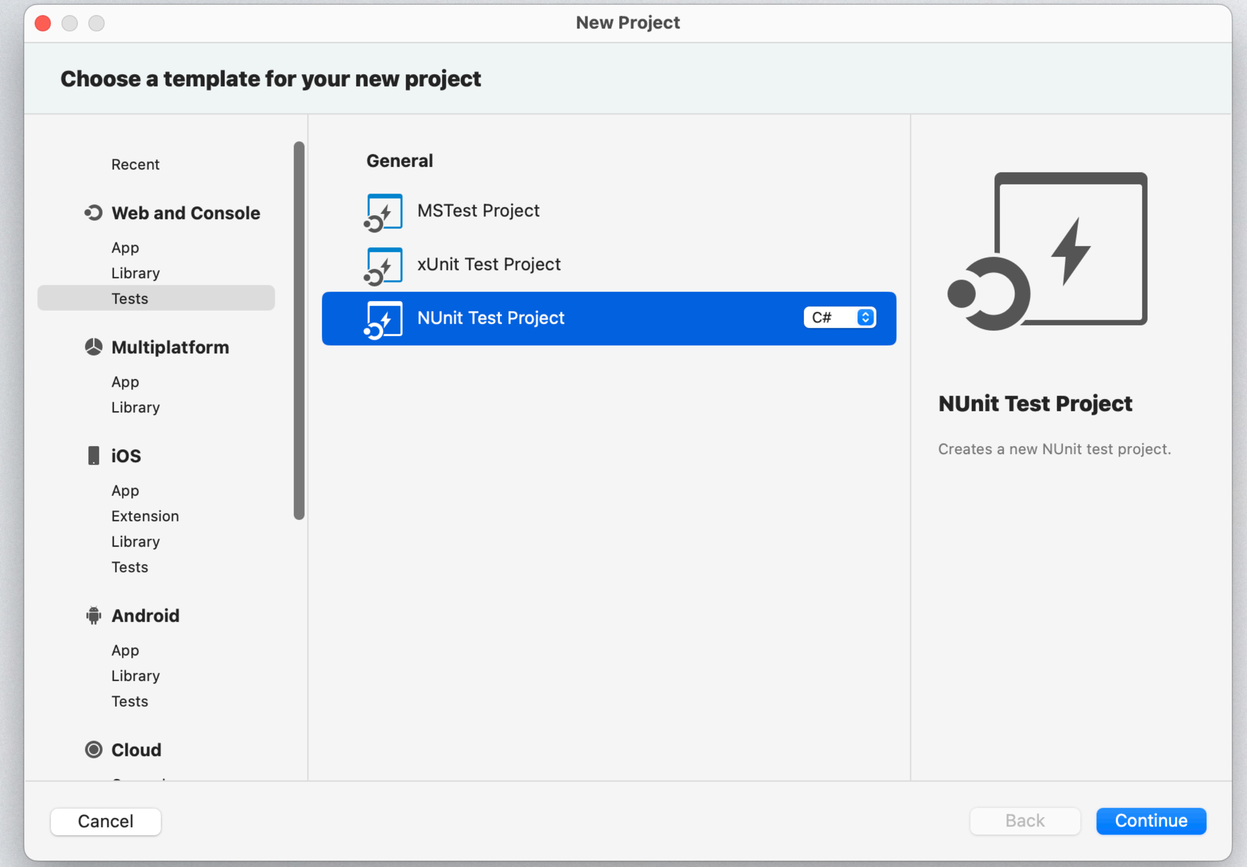
CREATE A NUNIT TEST



- Select the solution QueueApplication. Right-Click and choose Add/New Project…

© Dr Markus Lumpe, 2024

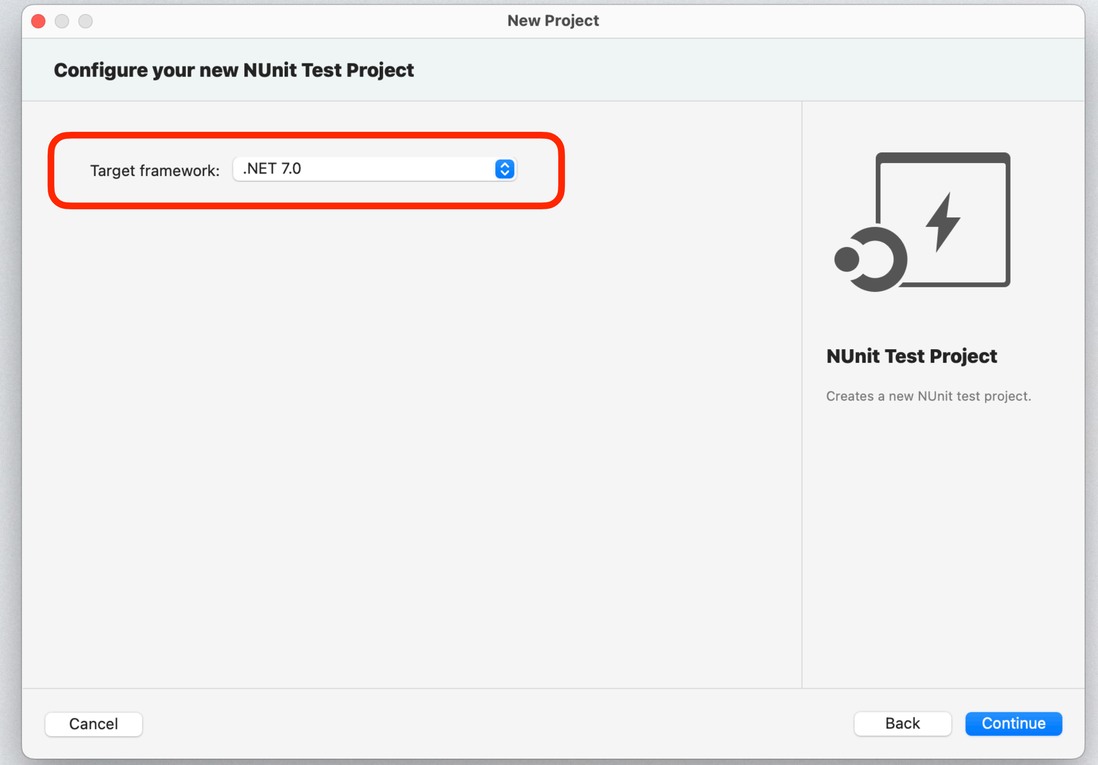
NUNIT TEST PROJECT



- Under Web and Console, select Tests.
- As project type choose NUnit Test Project.

© Dr Markus Lumpe, 2024

SELECT FRAMEWORK

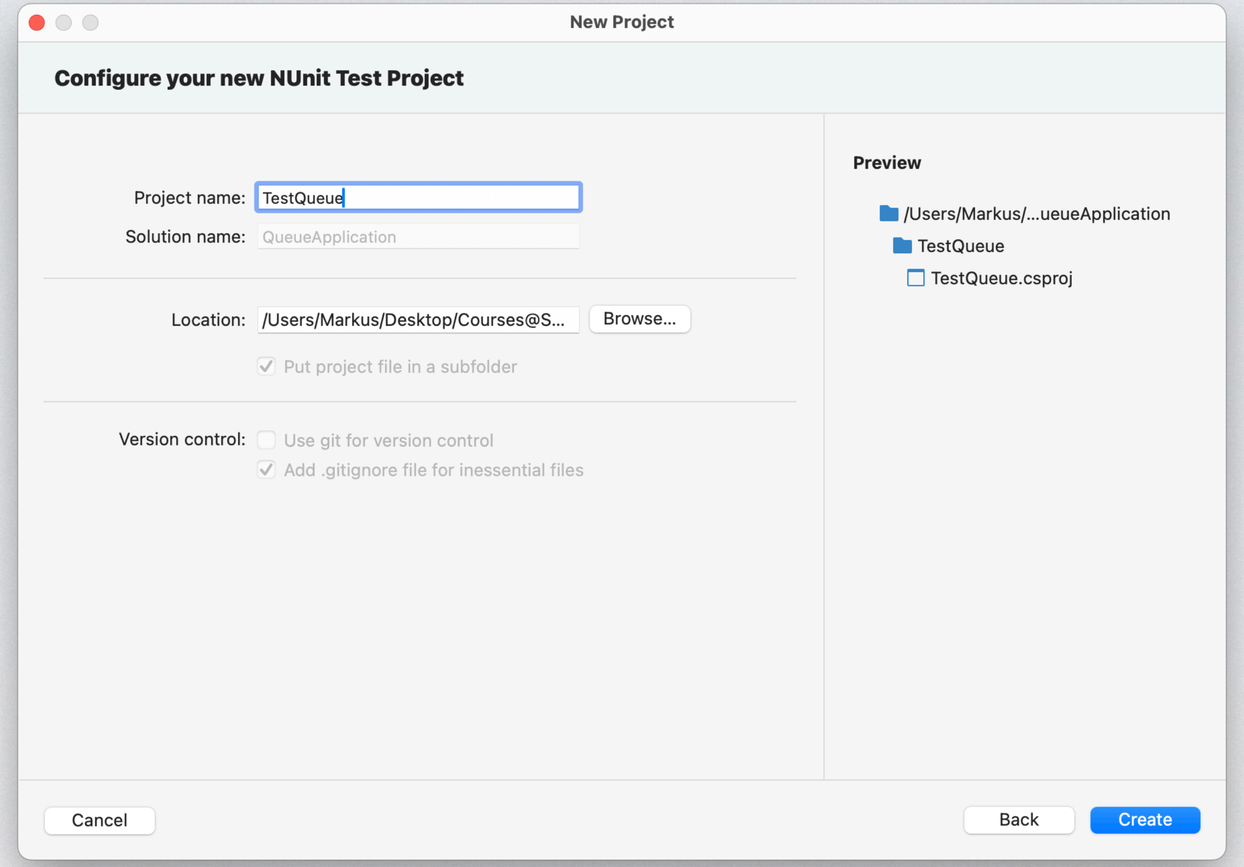


- When creating an NUnit Test Project for .NET, we need to select the .NET framework that we want to work with.
- You can safely choose the latest framework (.NET 7.0*). At a minimum, COS20007 requires .NET 6.0.

* At present, Visual Studio for Mac only supports .NET 7.0 even if a .NET 8.0 framework is available on your system.

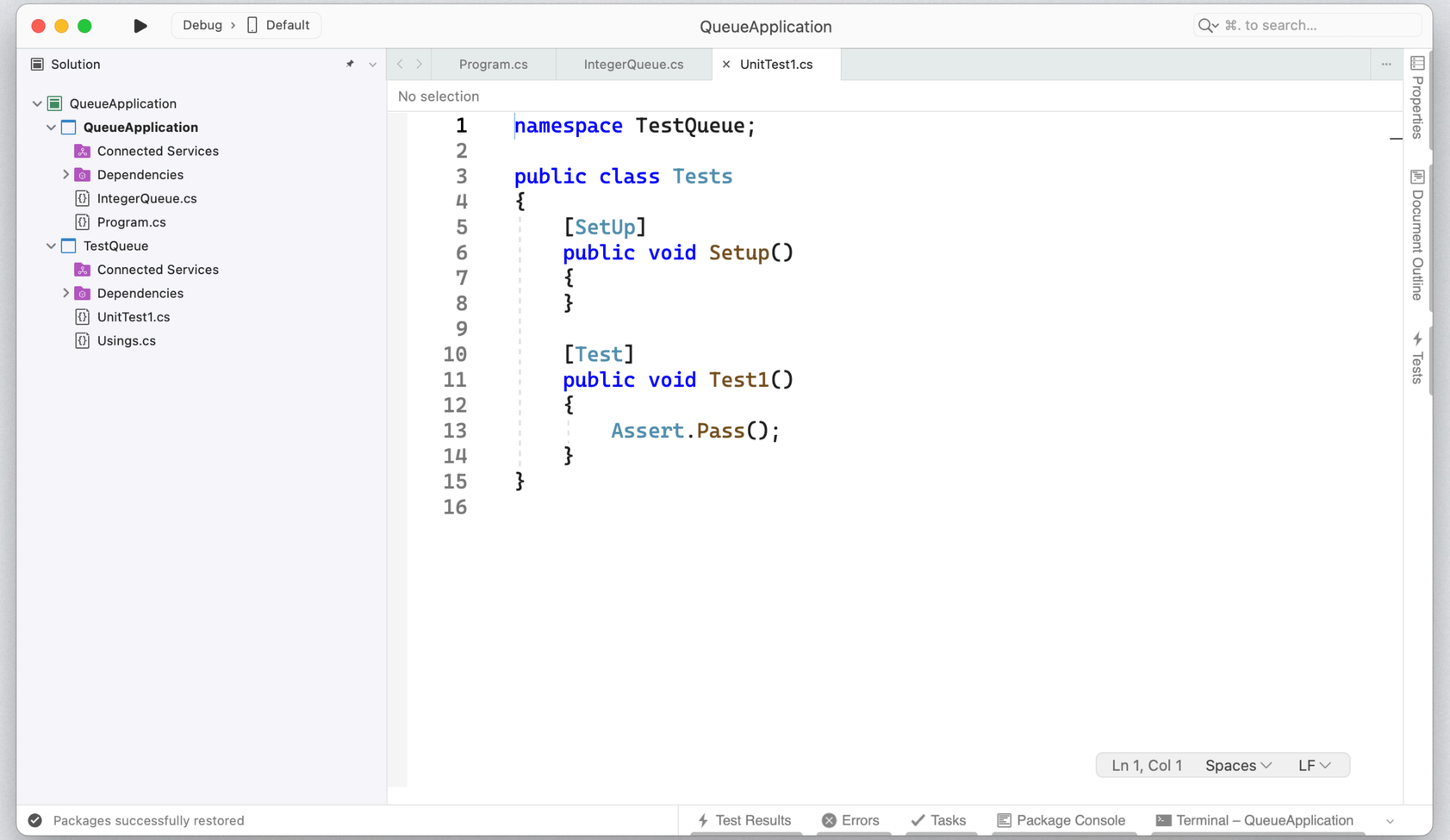
© Dr Markus Lumpe, 2024

PROJECT NAME: TESTQUEUE



© Dr Markus Lumpe, 2024

TEST SKELETON

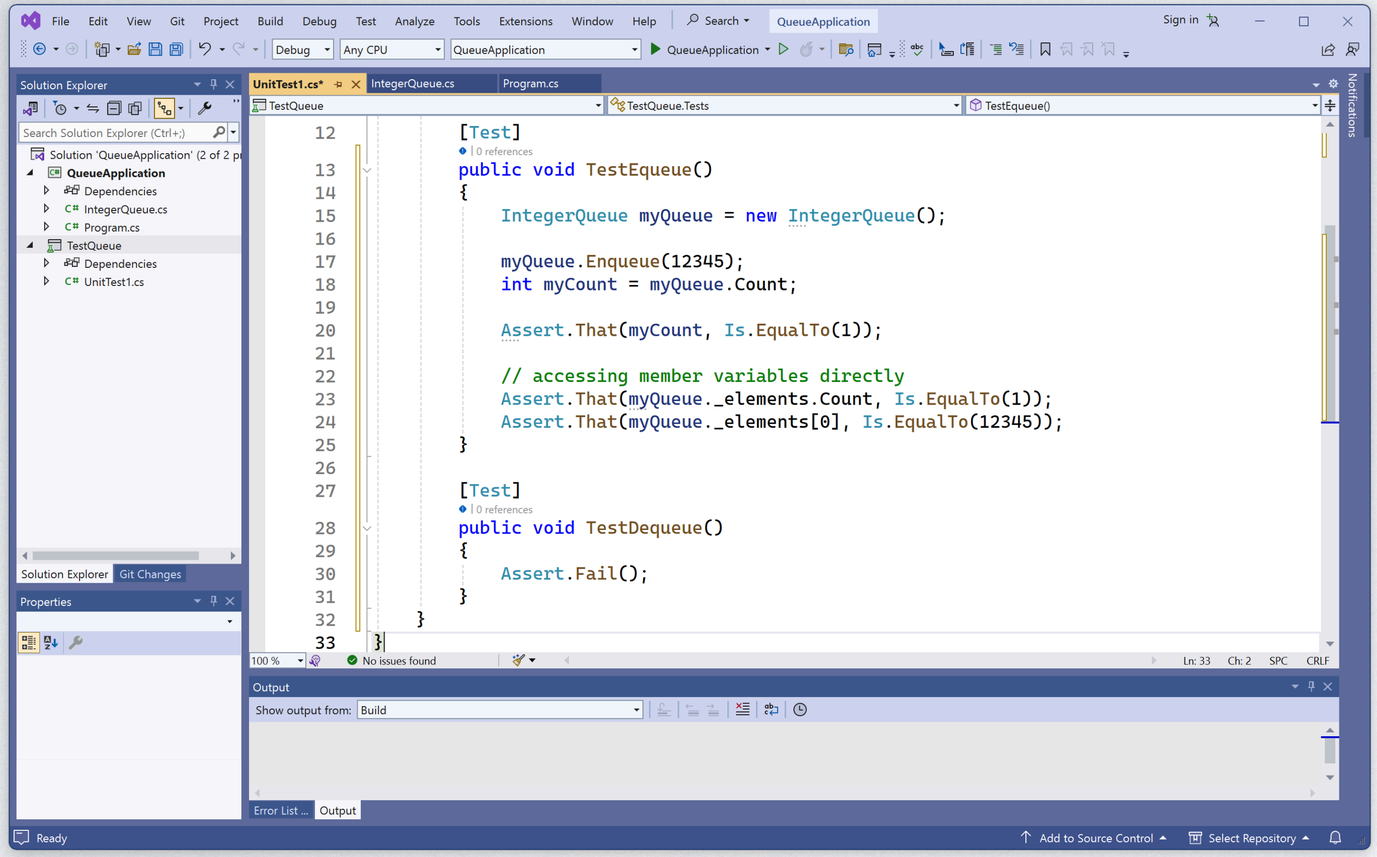


© Dr Markus Lumpe, 2024

DEFINE TESTS

© Dr Markus Lumpe, 2024

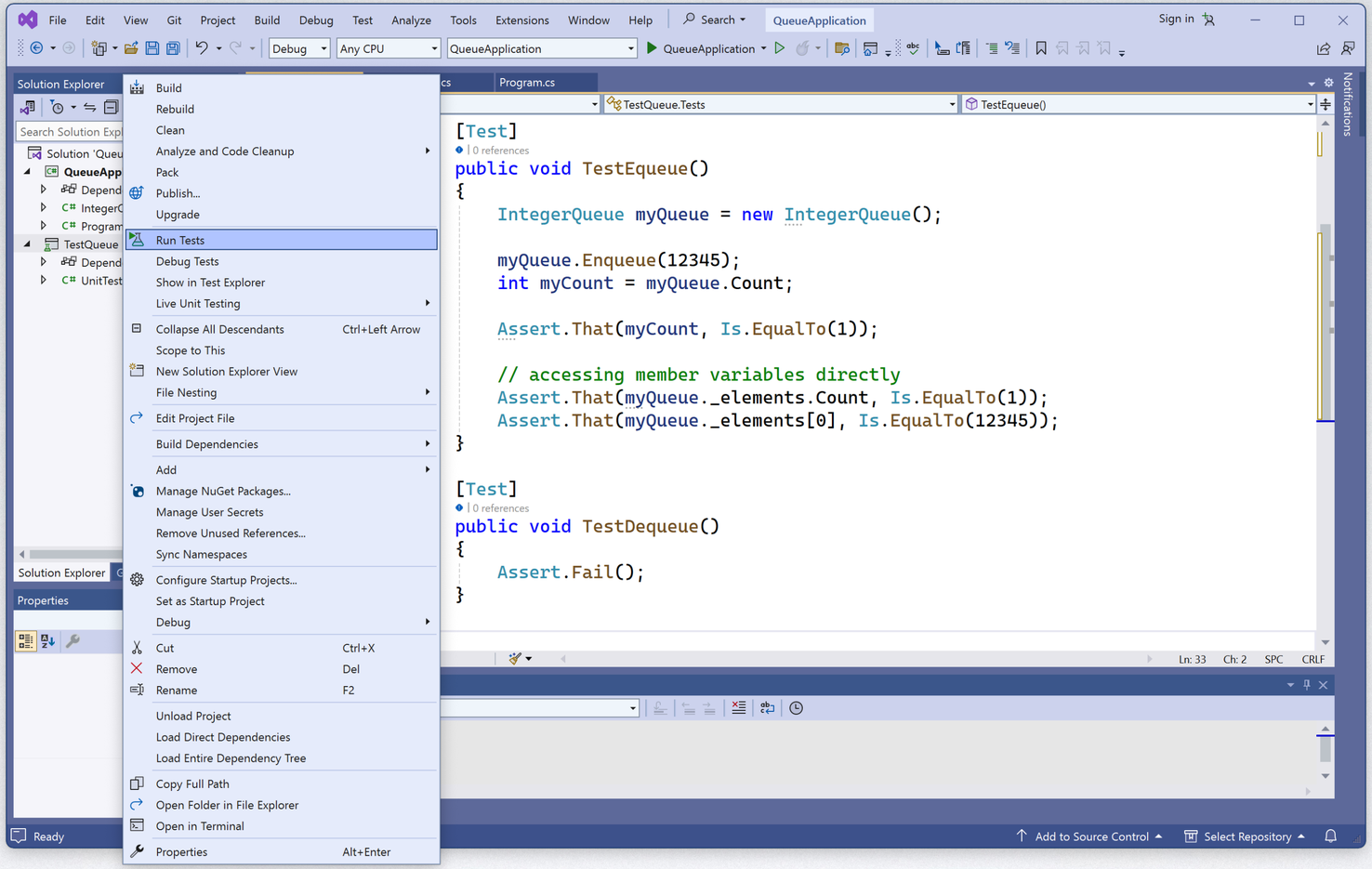
CREATE A TEST FOR ENQUEUE



- Add QueueApplication as a project reference:
  - ***Windows***: right-click TestQueue, choose Add/Project Reference…, tick QueueApplication
  - ***macOs***: right-click TestQueue, choose Add/Reference…, select tab All, and select QueueApplication).
- Add using QueueApplication; at the top of the NUnit source file (e.g., UnitTest1.cs)
- Implement a test for member Enqueue. You can decide on the level of detail.
- Note, the test for Dequeue is designed to fail.

© Dr Markus Lumpe, 2024

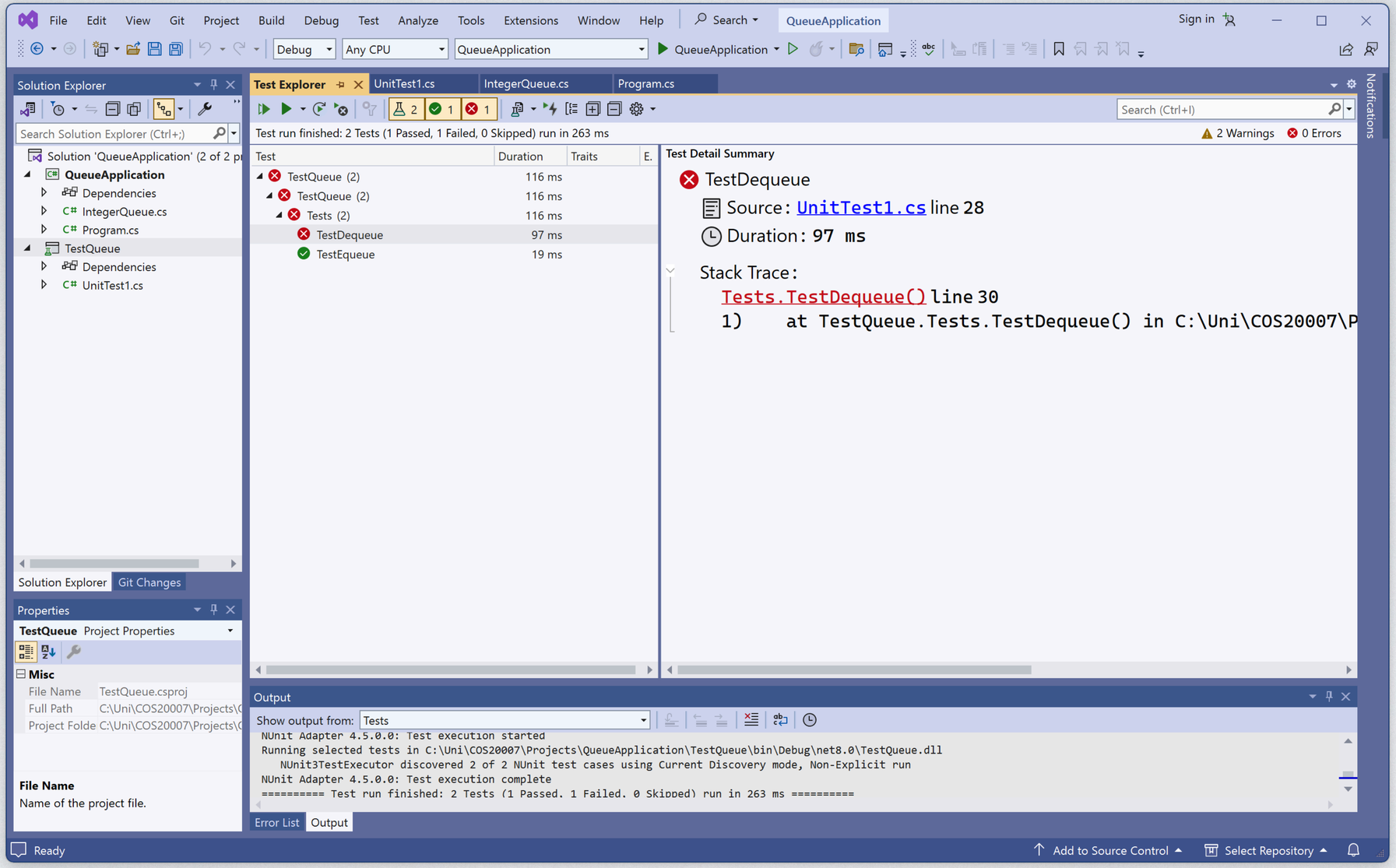
RUN TEST - WINDOWS



- Select Run Tests.

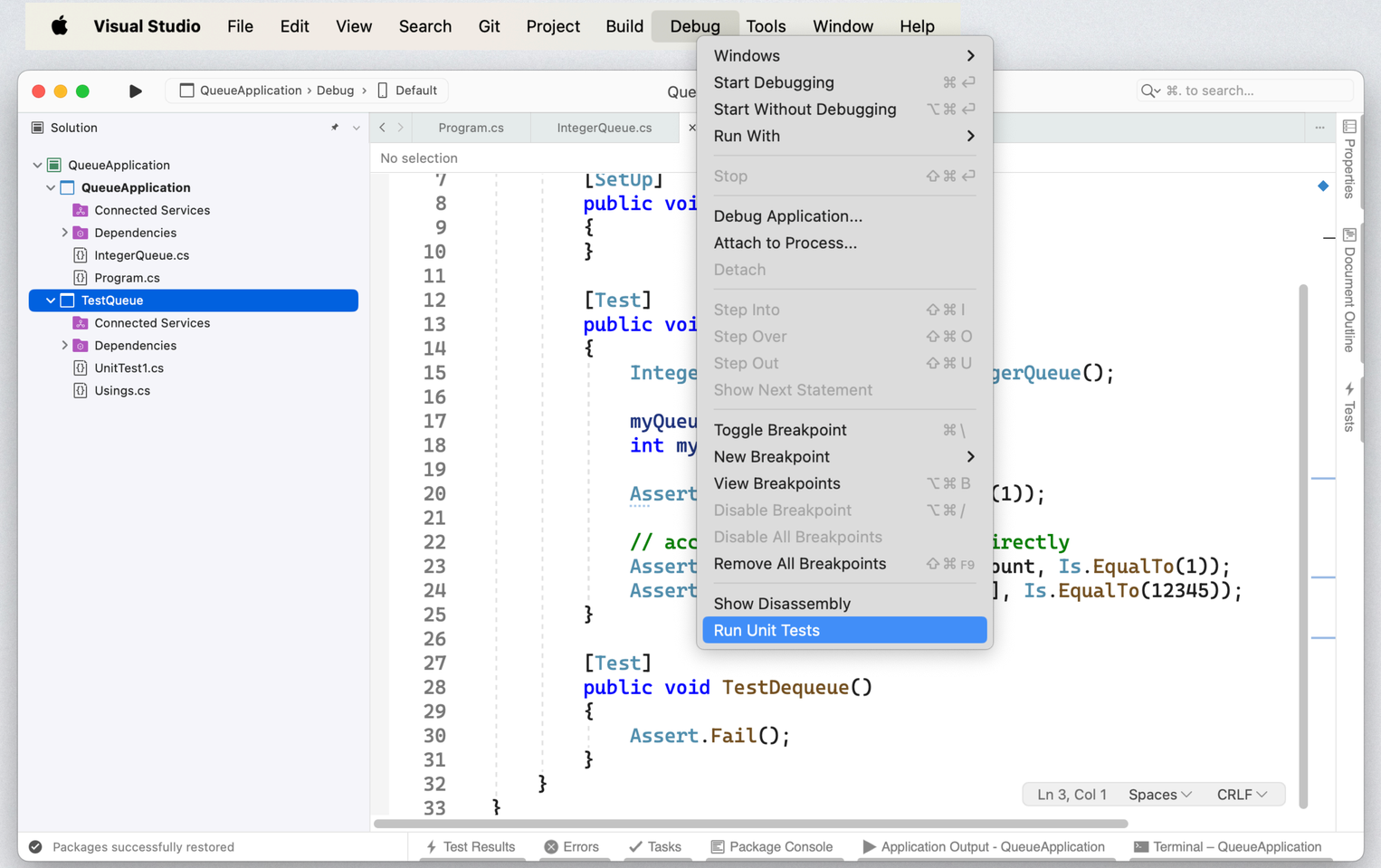
© Dr Markus Lumpe, 2024

INSPECT TEST RESULTS - WINDOWS



© Dr Markus Lumpe, 2024

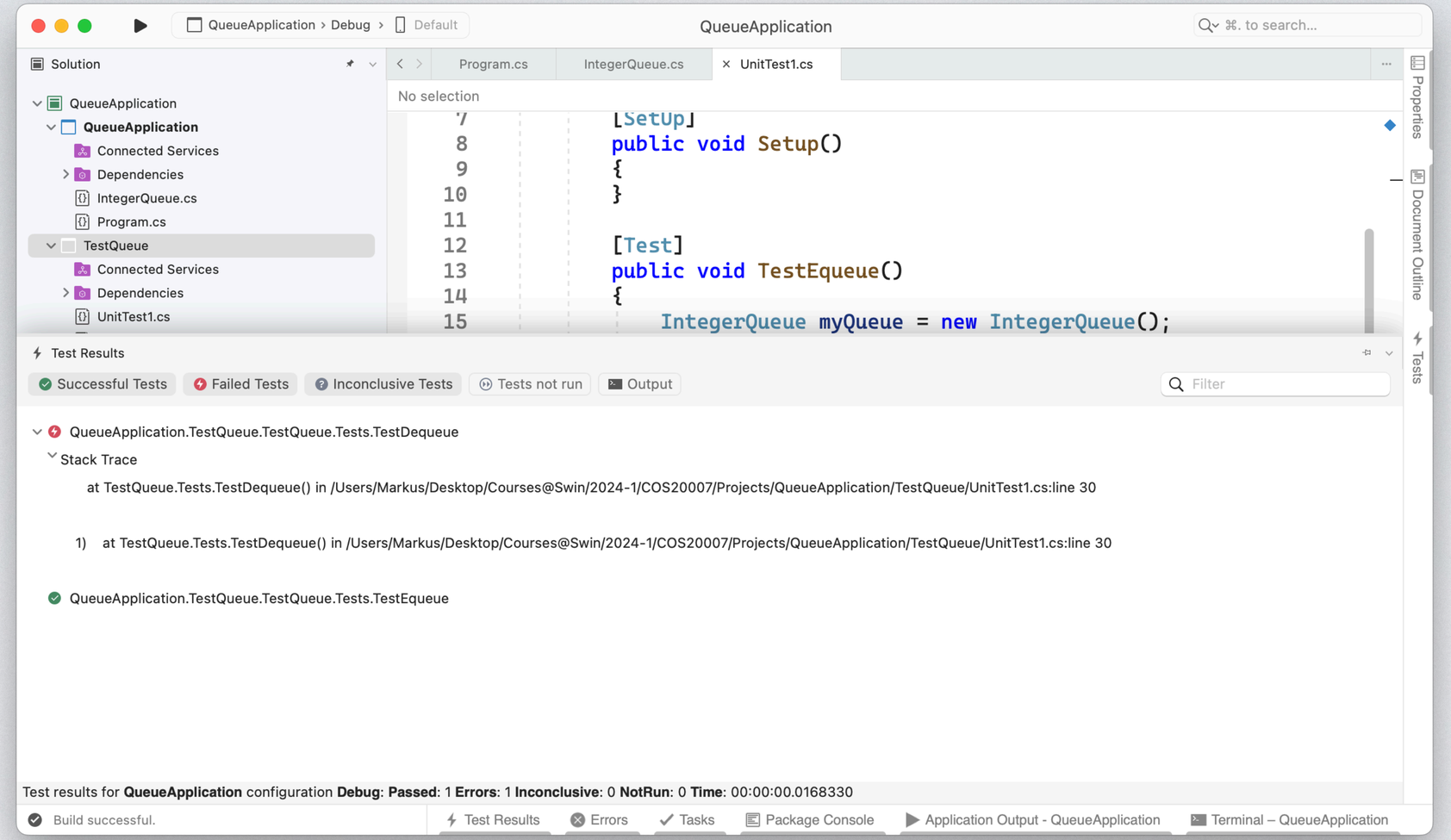
RUN TEST - MAC



- Select Debug/Run Unit Tests.

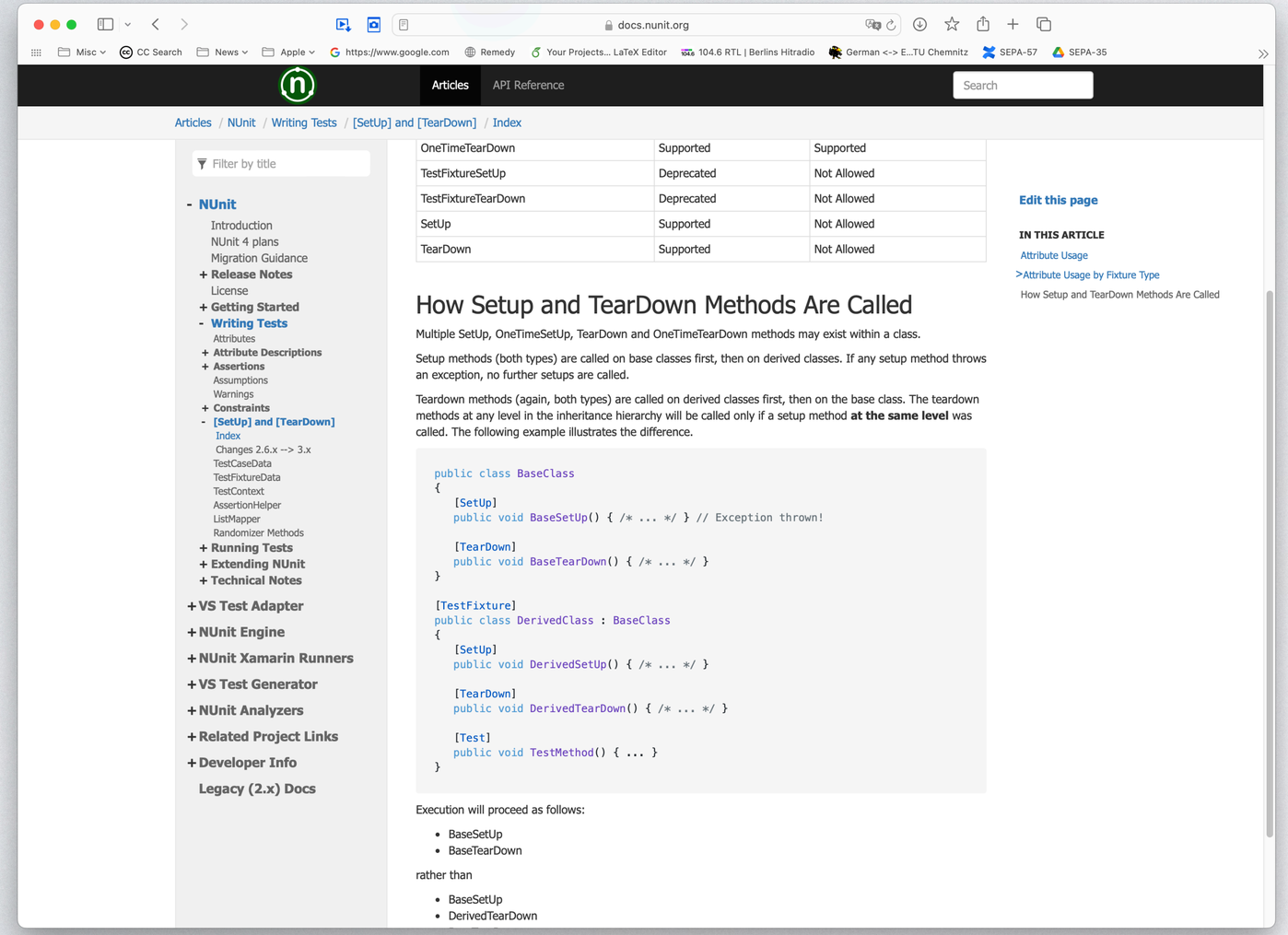
© Dr Markus Lumpe, 2024

INSPECT TEST RESULTS - MAC



© Dr Markus Lumpe, 2024

NUNIT DOCUMENTATION: NUNIT



- NUnit comes with a detailed user documentation.
- Take a moment and browse.
- There is a multitude of test options available.

© Dr Markus Lumpe, 2024

HAPPY TESTING

© Dr Markus Lumpe, 2024