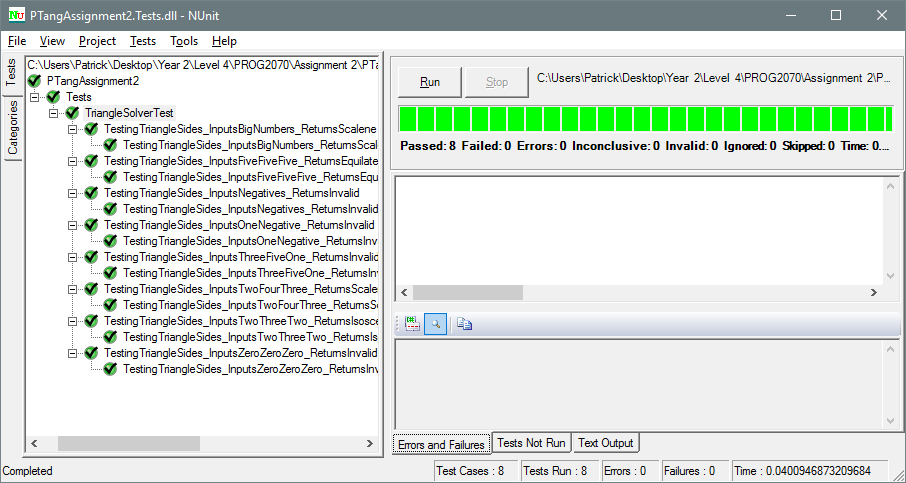
**NUnit**



**Why I Chose These Tests**

**TestingTriangleSides\_InputsFiveFiveFive\_ReturnsEquilateral()**

This was chosen to make sure that all three equal sides will pass as an equilateral triangle.

**TestingTriangleSides\_InputsTwoThreeTwo\_ReturnsIsosceles()**

This was chosen to make sure that at least two sides is equal and the third side is less than the other two combining sides equals to an isosceles triangle.

**TestingTriangleSides\_InputsTwoFourThree\_ReturnsScalene()**

This was chosen to make sure that all sides were equal to a scalene by testing if all sides were less than the next input and to make sure that the first two sides combined is not greater than the third side.

**TestingTriangleSides\_InputsThreeFiveOne\_ReturnsInvalid()**

This was chosen to make sure that it will return an invalid triangle. (1 + 3) < 5 should not equal into a scalene triangle.

**TestingTriangleSides\_InputsNegatives\_ReturnsInvalid()**

This was chosen to make sure that no inputs were negatives as they would create an invalid triangle due to its measurement.

**TestingTriangleSides\_InputsZeroZeroZero\_ReturnsInvalid()**

This was chosen to make sure that no triangle can be made with any zeroes as the input.

**TestingTriangleSides\_InputsOneNegative\_ReturnsInvalid()**

This was chosen to make sure that any of the inputs were negatives would return as an invalid triangle. The negative measurement will not create a triangle anyways.

**TestingTriangleSides\_InputsBigNumbers\_ReturnsScalene()**

This was chosen to make sure that any big number can create at least a scalene triangle and that no number above it should have an issue (if it was to create a triangle).