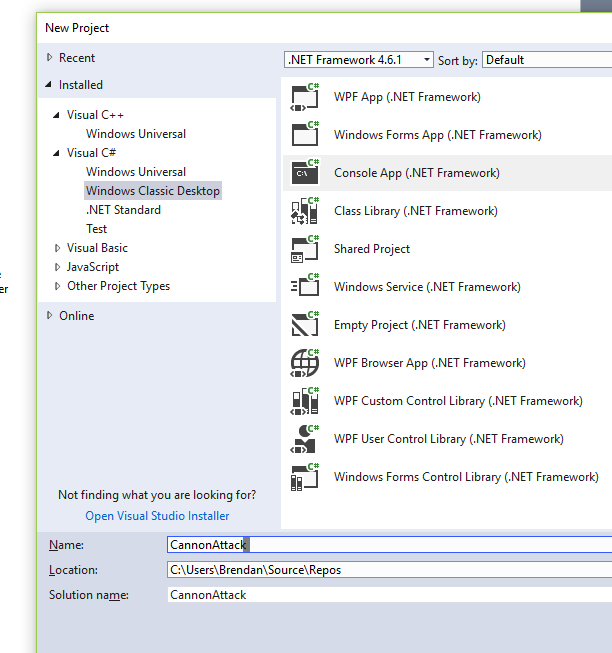
**CA Assignment – Test Driven Development (TDD)**

This assignment will consist of a facilitated workshops initially utilising Unit Testing and TTD resources identified below:

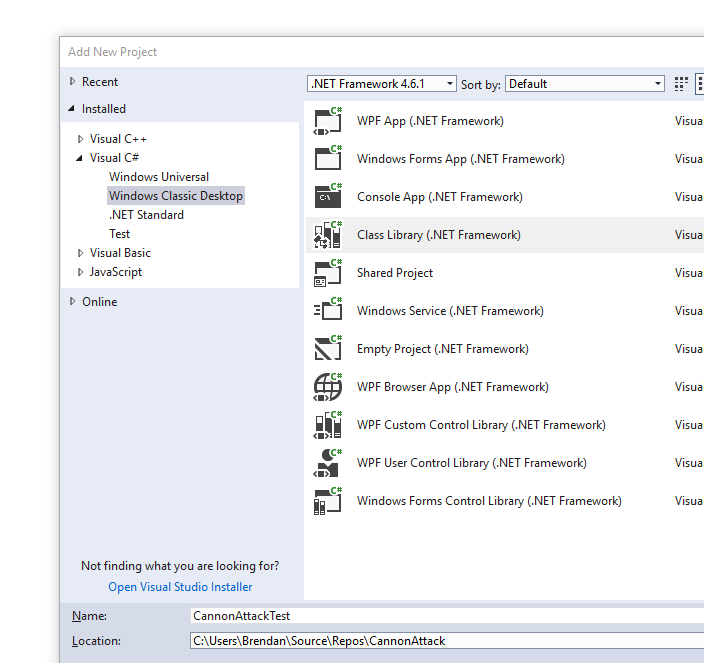
1. The Way of Testivus
2. Google Testing Blog
3. <http://www.oodesign.com/singleton-pattern.html> Information on Singleton Design pattern
4. <http://en.wikipedia.org/wiki/Thread_safety> Information on Thread Safety
5. http://www.c-sharpcorner.com/uploadfile/dommym/a-test-driven-development-tutorial-in-C-Sharp-4-0/

Assignment Description:

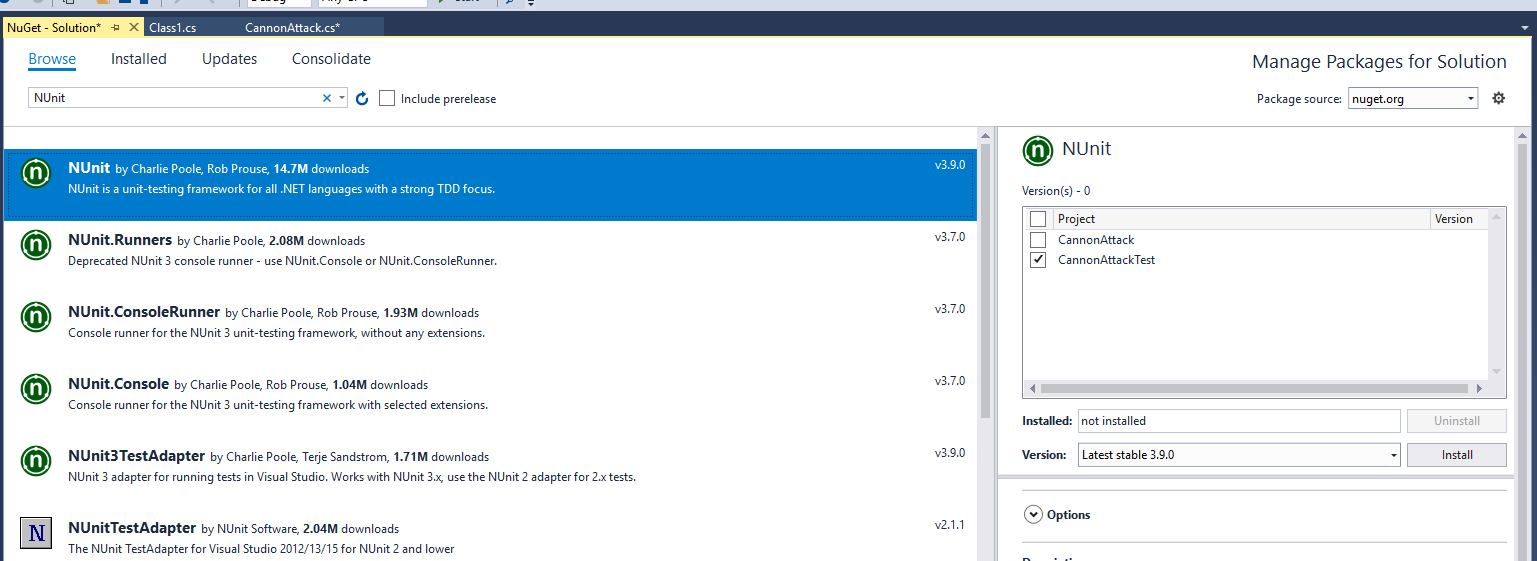
* Working individually you will work through a test driven development project over a number of iterations and produce a fully functional basic game. Each iteration will implement one or more of the requirements. You will discover how VS2017 provides TDD support through a number of it’s features such
* Create the project



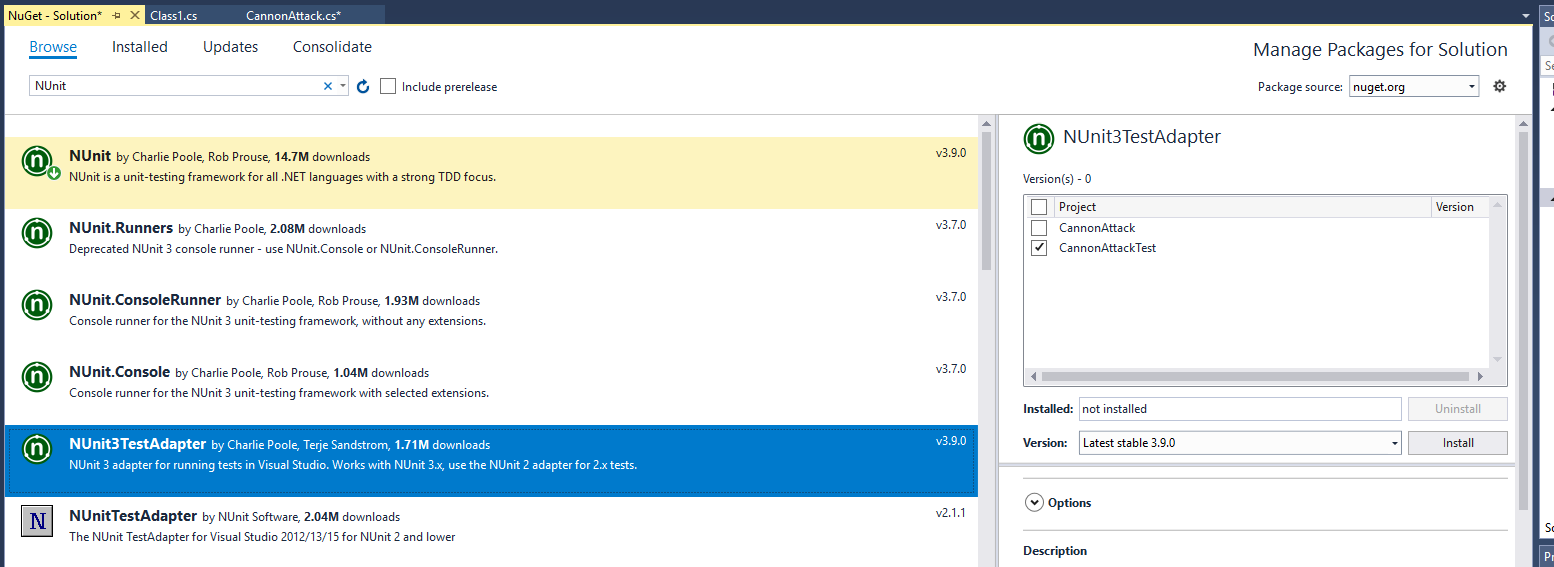
Add a Test project – Not like it is done in the Tutorial as they are using the build in MSTest unit test framework. We want to use the NUnit Test framework. So add a class library project as below – we will be using this as our test project.



Now use NuGet to get the NUnit test engine and NUnit test adapter (for VS so we can use the built in VS test runner with out NUnit tests)



Now the Test Adapter

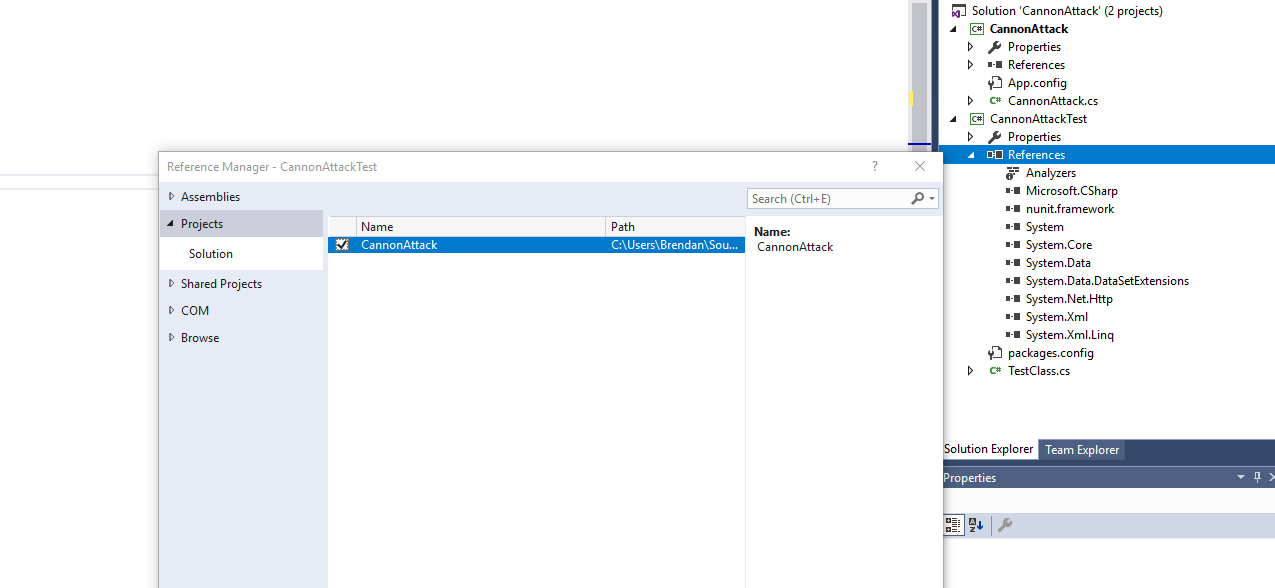


Now rename the Class1.cs file in the class library project CannonAttackTest

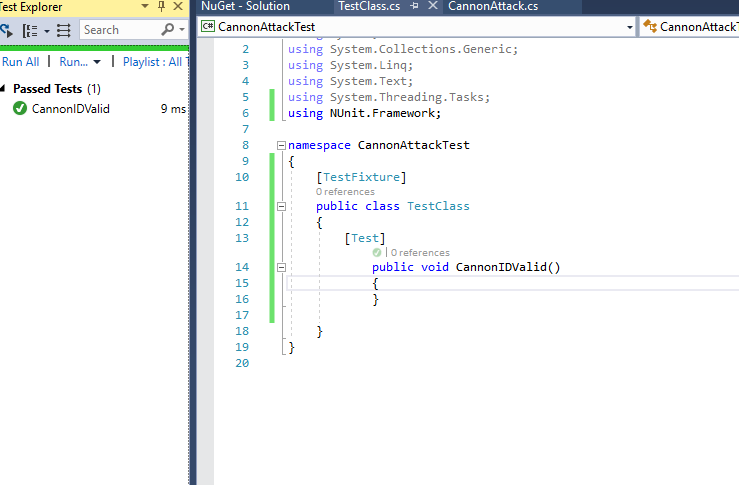
Class library to TestClass.cs.

Now add a reference into the CannonAttackTest project to the CannonAttack project (so the test project can call methos in the production CannonAttack project.

Right click References in CannonAttackTest and add a reference to CannonAttack as shown below.



Open TestClass.cs and a using statement and add the following test method and run your unit test using TestExplorer which is under Test>Windows Menu.

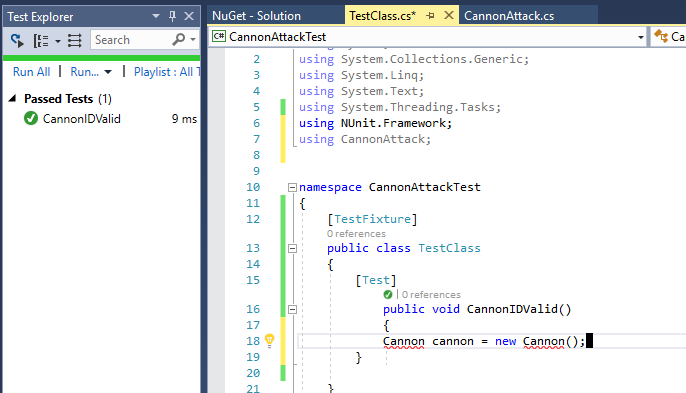


Then add the code to create the cannon in the test method

Ie add line

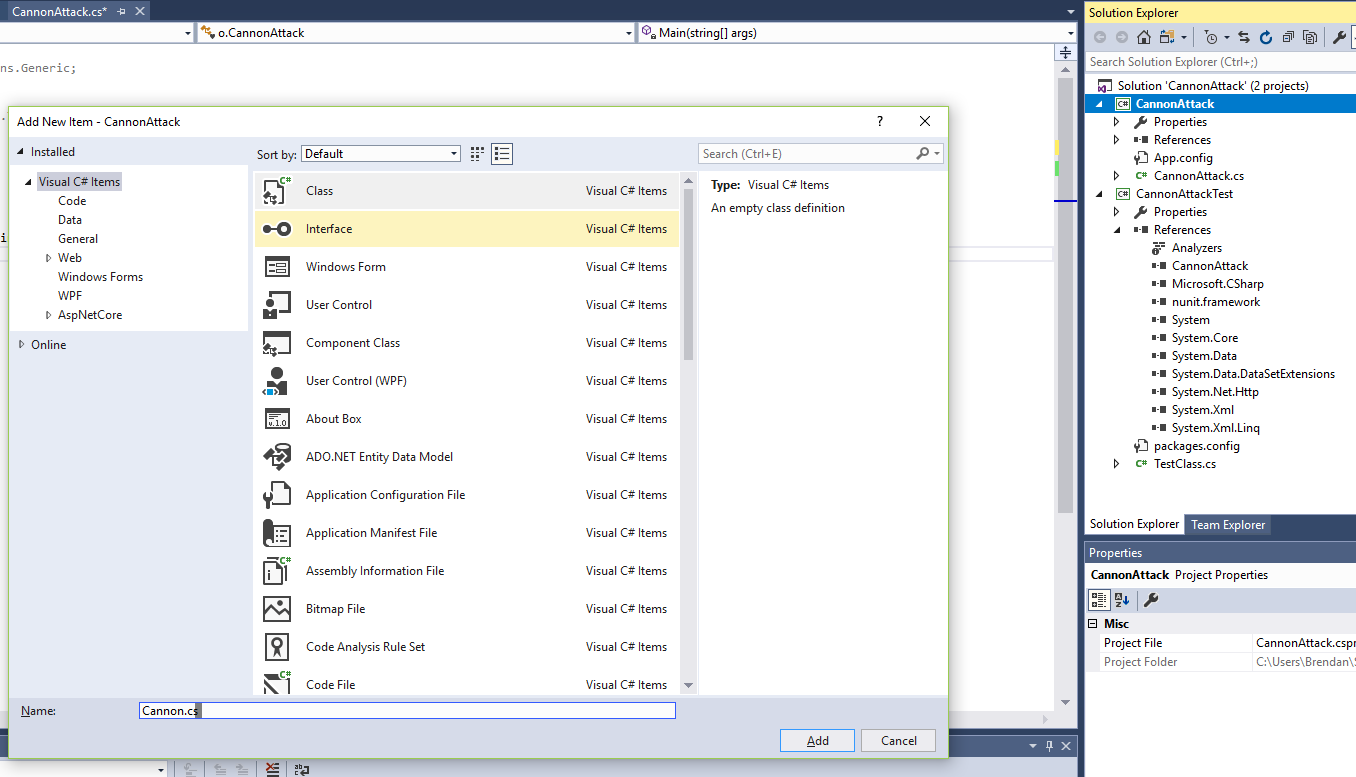
Cannon cannon = new Cannon();

And add a using statement as shown below:

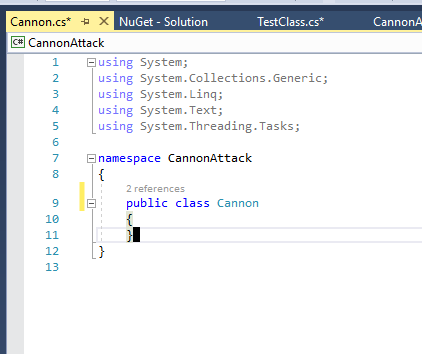


Add the cannon class to the CannonAttack project ie production code.

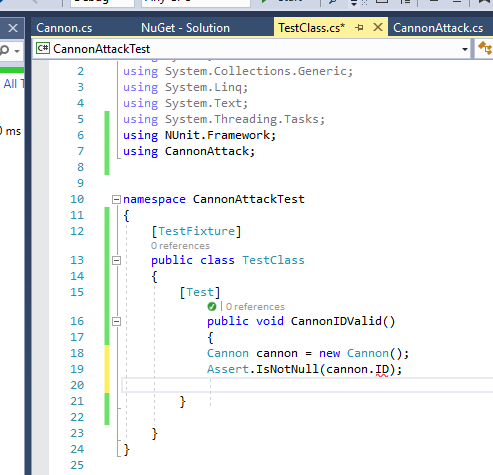
Right click CannonAttack Project and add new class as shown



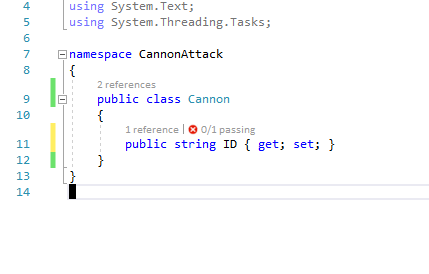
Specify the class as public



Add the code below to the TestClass



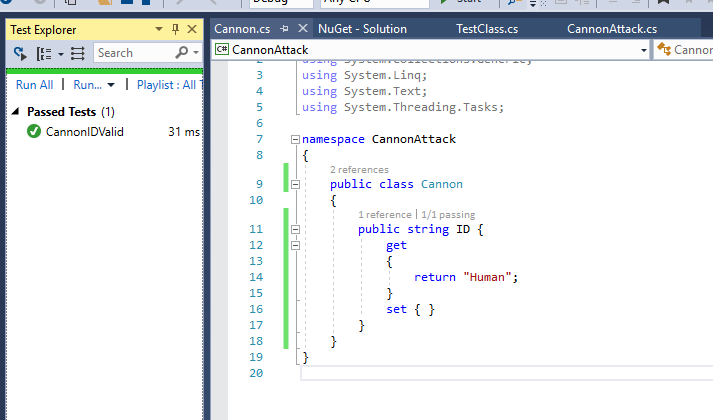
Then right click the ID and show potential fixes and add the ID property to the Cannon Class as indicated. Change the property type so that is a string and not an object as shown



Now run your test again it should fail.

Then get the ID property to return Human

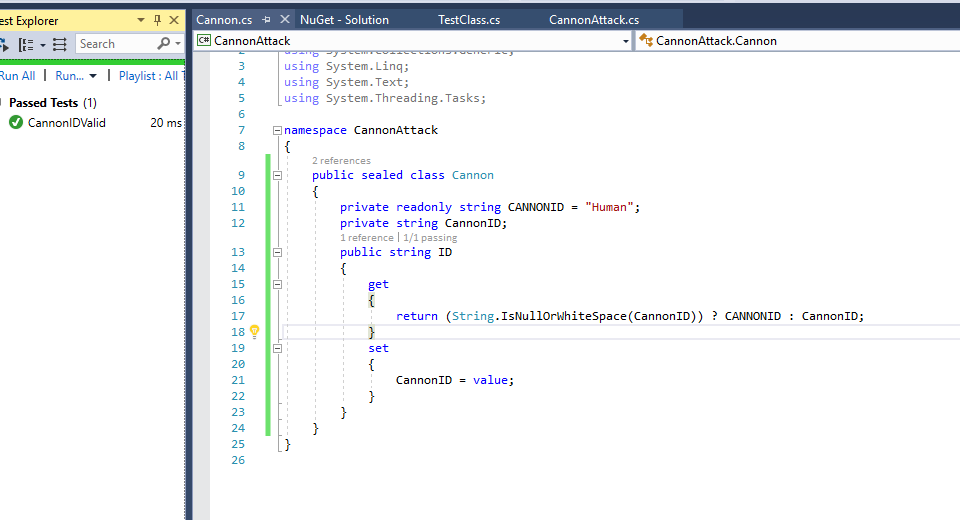
As shown



Run you r tests again, they should pass

You are doing TDD now, you wrote the test before you wrote the production code.

* We have just completed the second stage of a TDD cycle GREEN - Pass!!!!
* So the next stage is to refractor, and it would be nice to make a couple of changes to the class to clean it up, so that the class now looks like:



Check out the tutorial on why he made it sealed i.e. just above this line in the tutorial:

The refactoring is complete. Now for ITERATION 2 of the CannonAttack project.

Now onto **Iteration 2 - One Canon, and only one Cannon - Using the Singleton Pattern.**

Read what he says here but use the test method below (we are using NUnit!!):

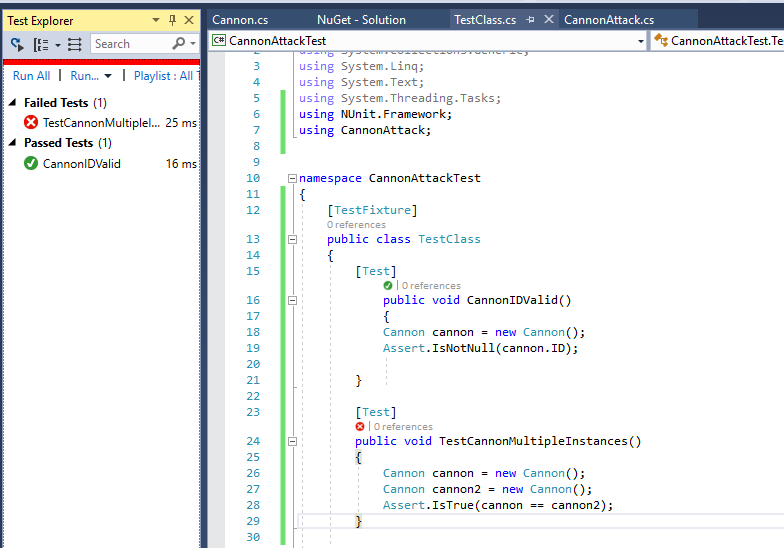
And run your test it will fail.

That is the way TDD works – write the test – it fails

Then write the production code and get the test to pass –

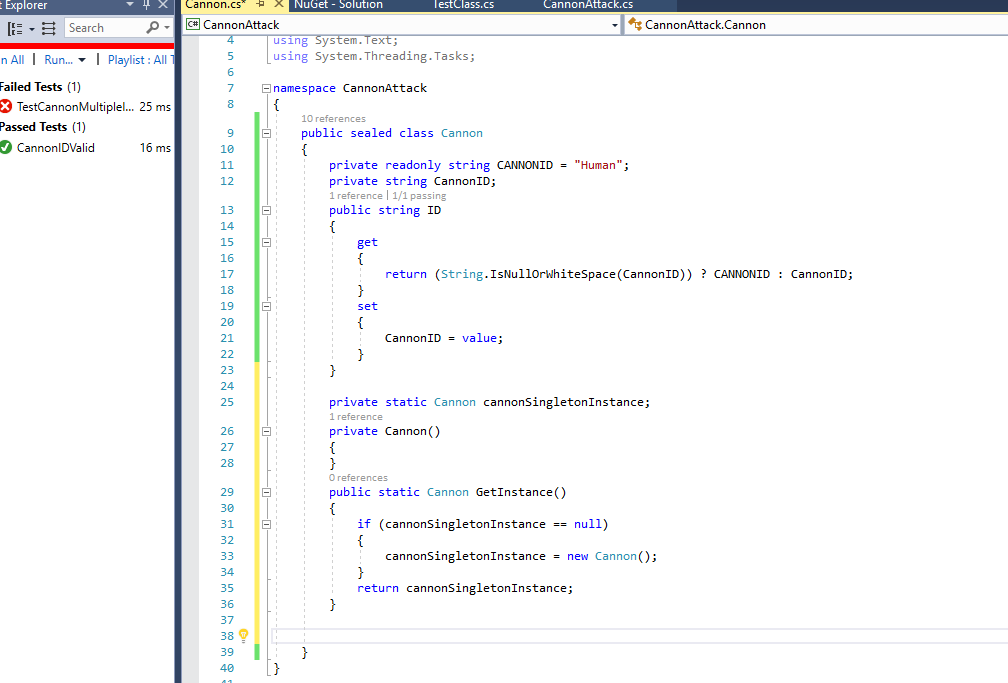
Then refactor

Ie Red – Green – Refactor

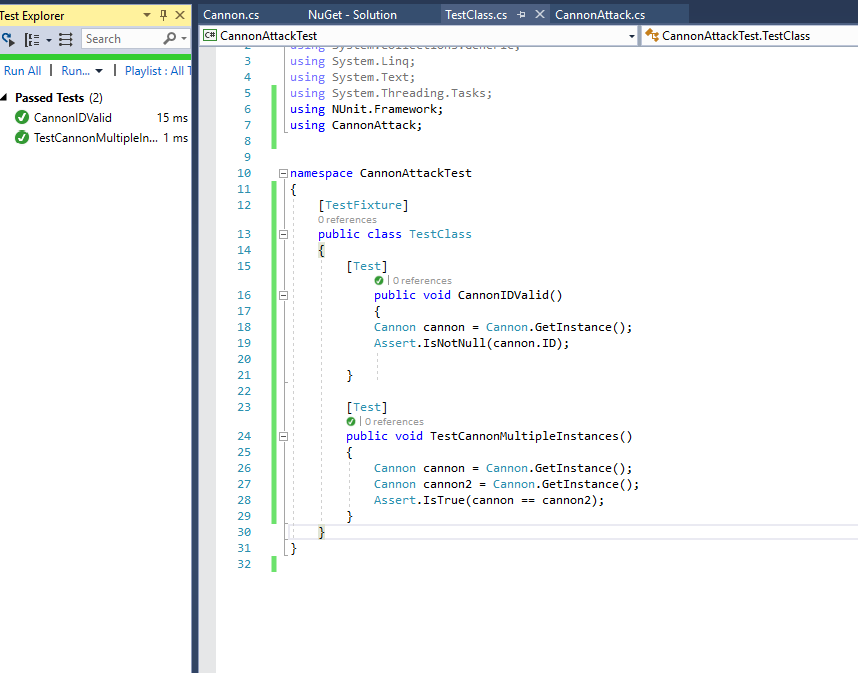


Back to tutorial – read what he says about Singleton

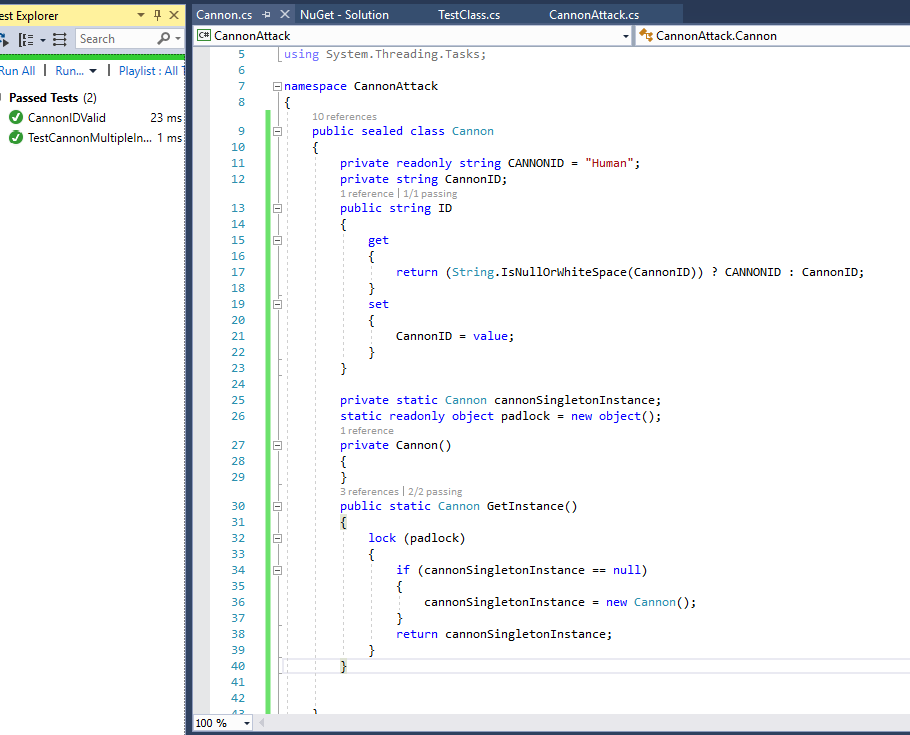
Add this code to your Cannon



Now get your test code to use the Singleton code as below and run your tests



Read the tutorial about thread safety and refactor and run your tests again as below:



Now for iteration 3 read what he says in tutorial i.e.

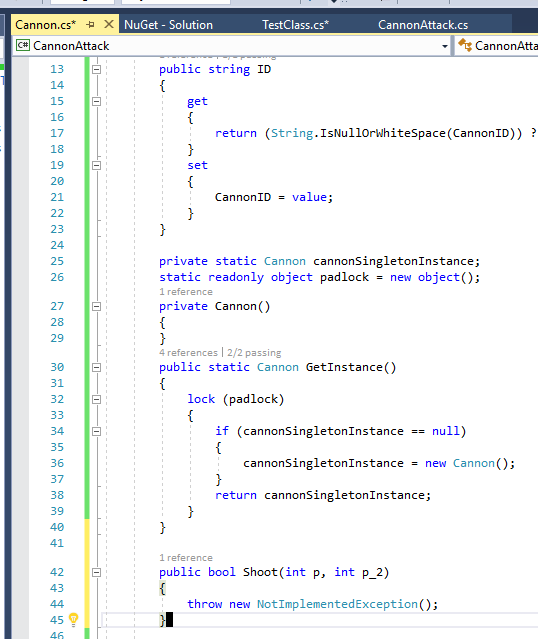
**Iteration 3 - Angling for something**

We will add another test method. This time we want to ensure that an incorrect angle (say 95 degrees) will not hit. So we need a Shoot method and a return type (lets keep it simple and make it a Boolean for now).

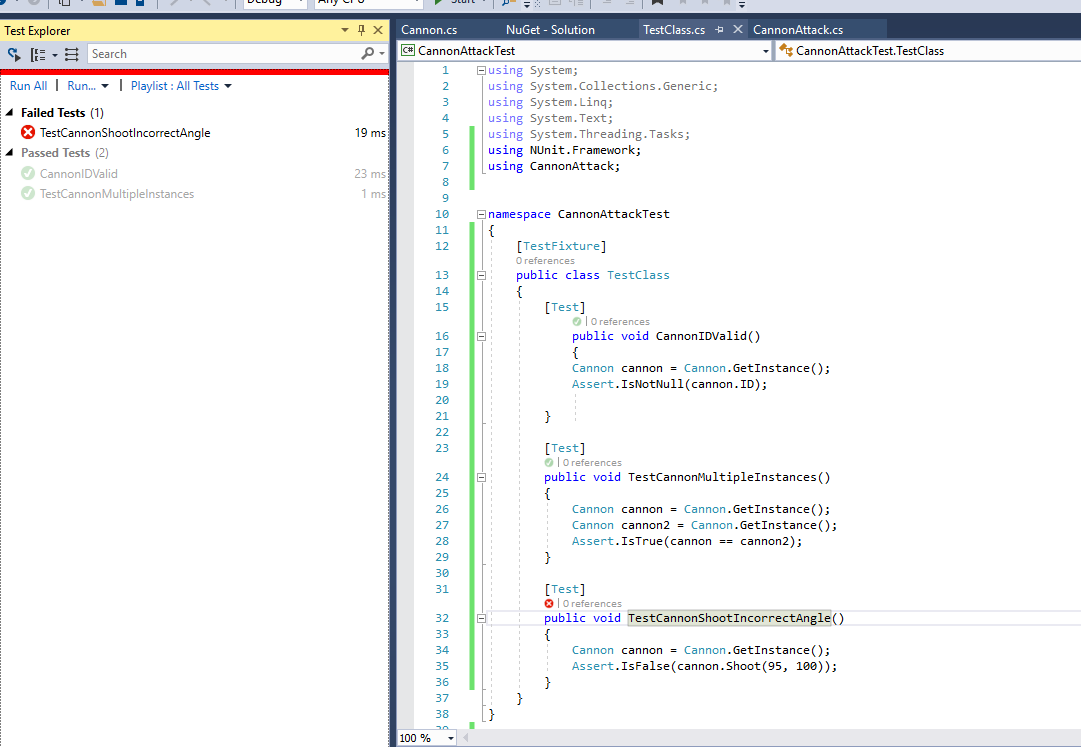
* Add the following test below the last test:

Add Shoot method to Cannon

Ie



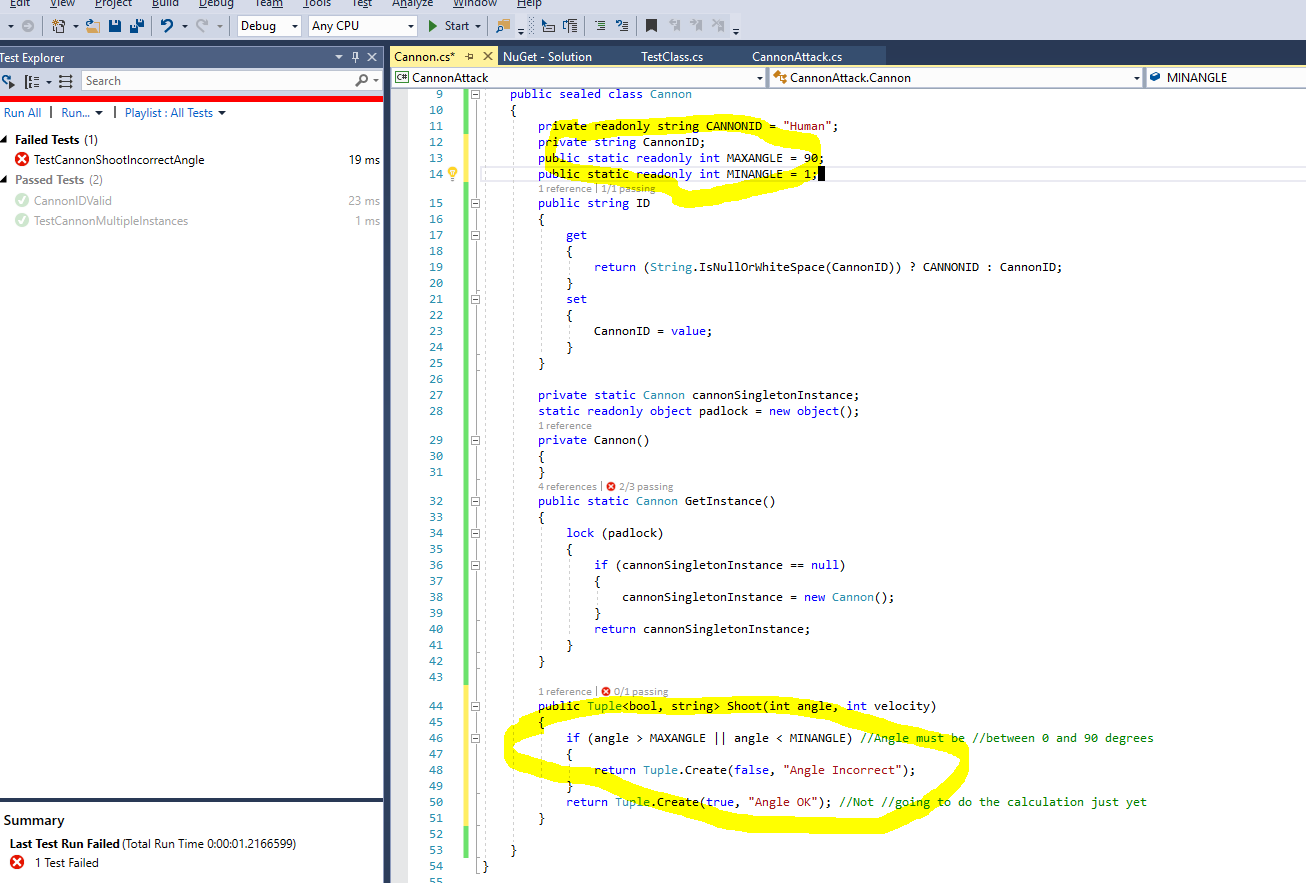
Run your tests as shown



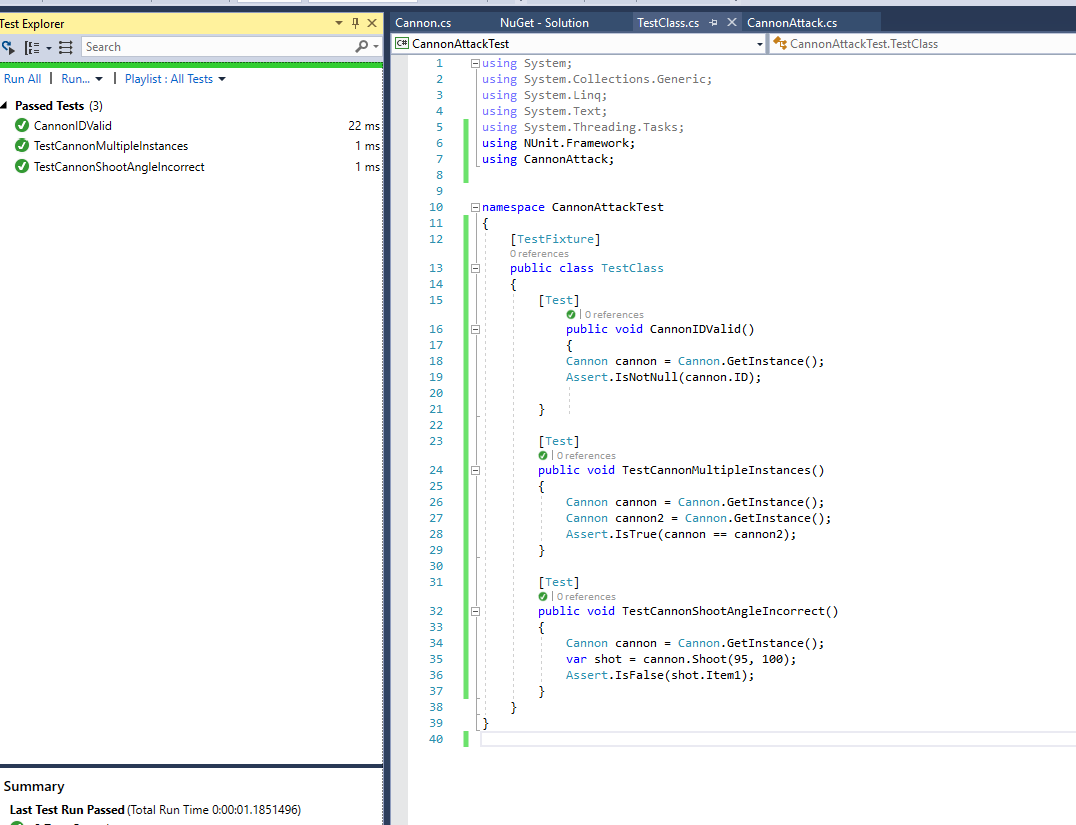
The new one fails

Now read the tutorial so you know what he is trying to do.

Replace the code in Cannon.cs



Now update your test as below and run all tests – they all pass

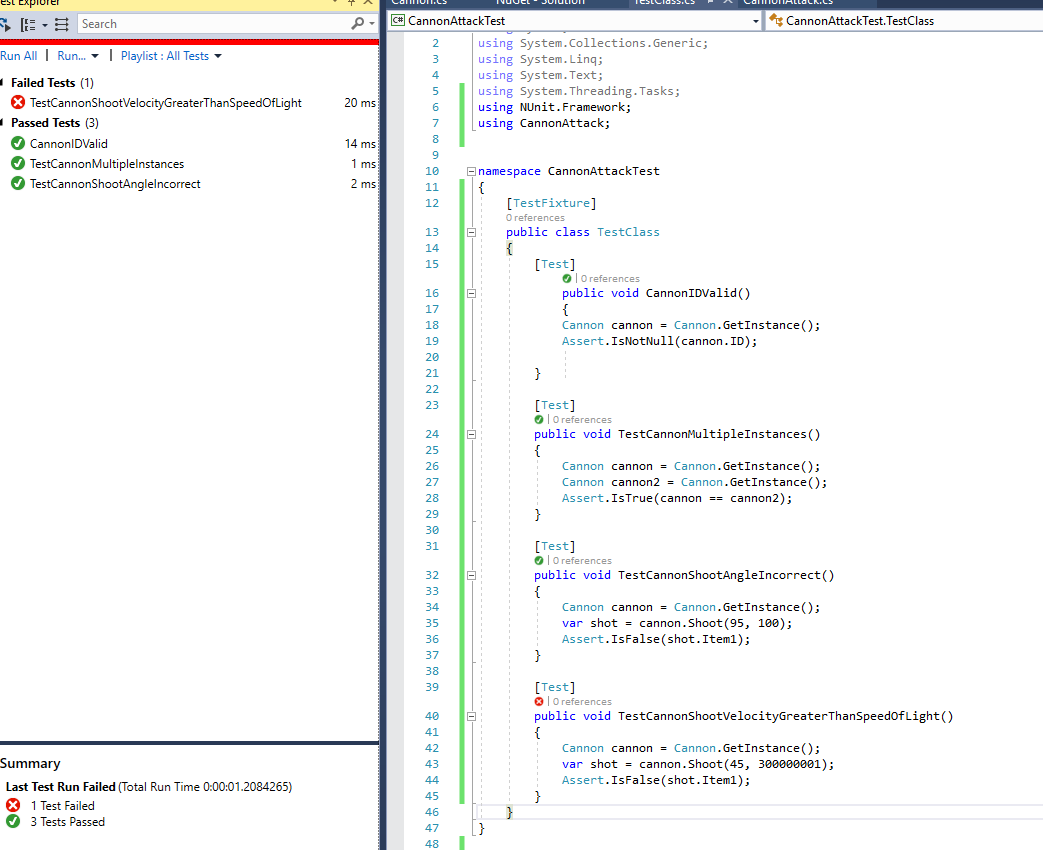


The initialisation material covered in the tutorial is for MSTest

So onto **Iteration 4 - That's a pretty fast cannonball**

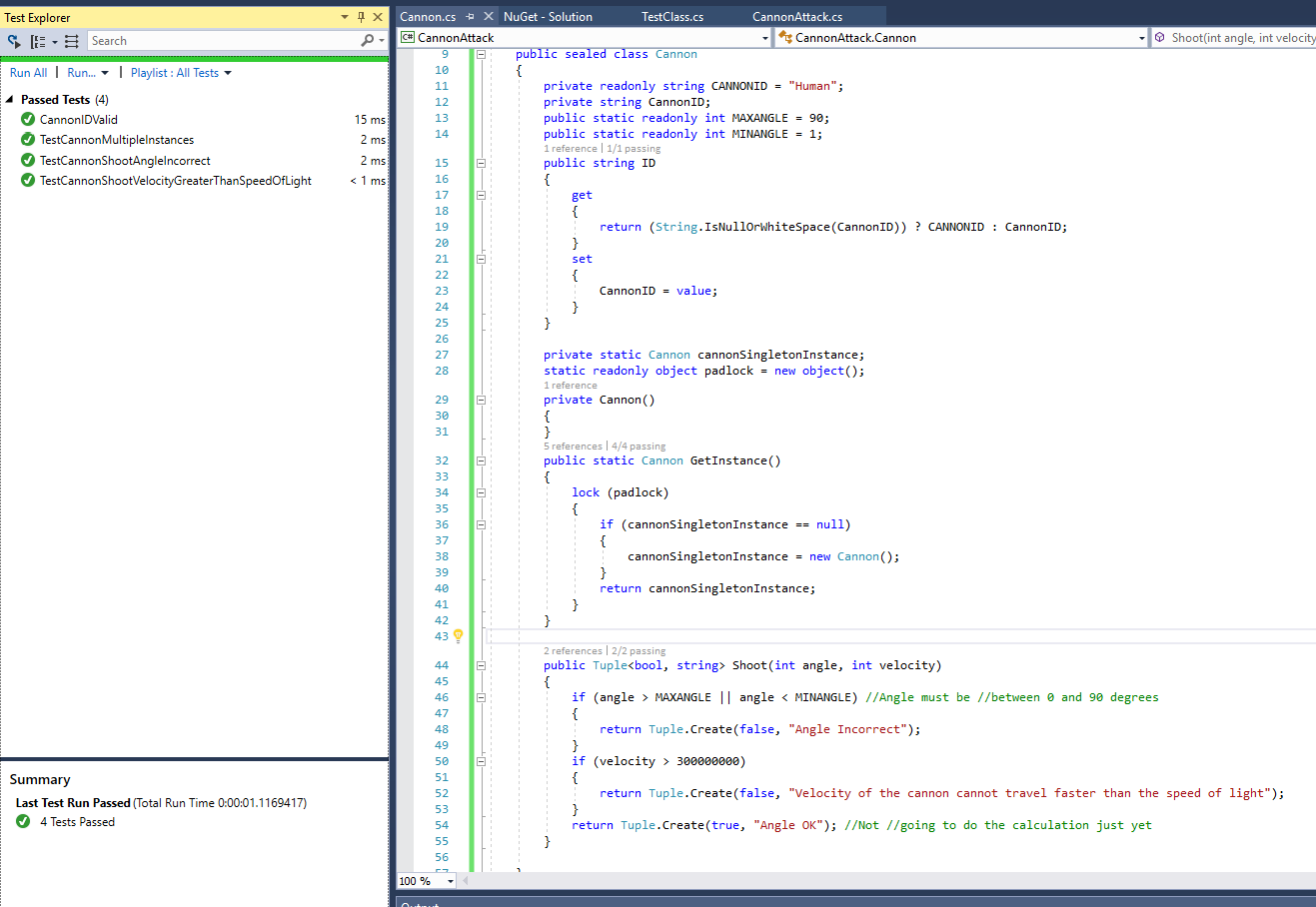
Read what this is about

Add the test as shown and run all tests



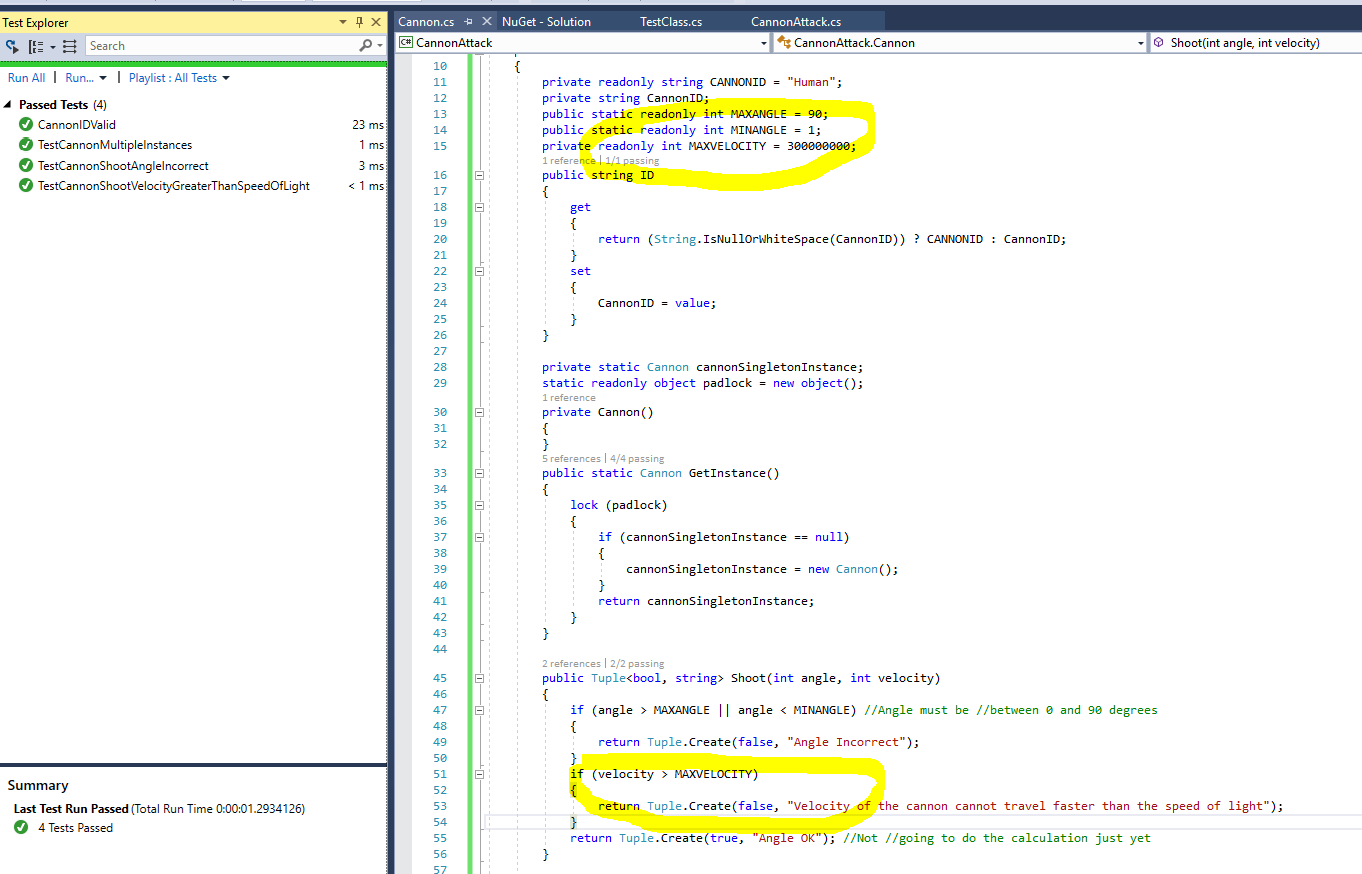
So we run the tests and it will fail RED because the validation isn't complete in the Shoot method . We add the code needed into shoot method just under the validation code for the incorrect angle:

So update Cannon as shown and run you tests again



Don’t mind the material he covers in Tutorial on Test impact view

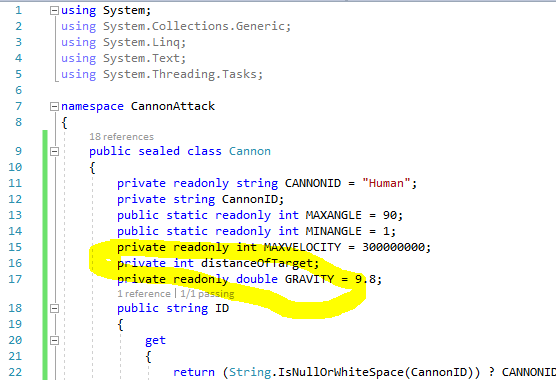
Now refactor as he states in tutorial and re run your tests



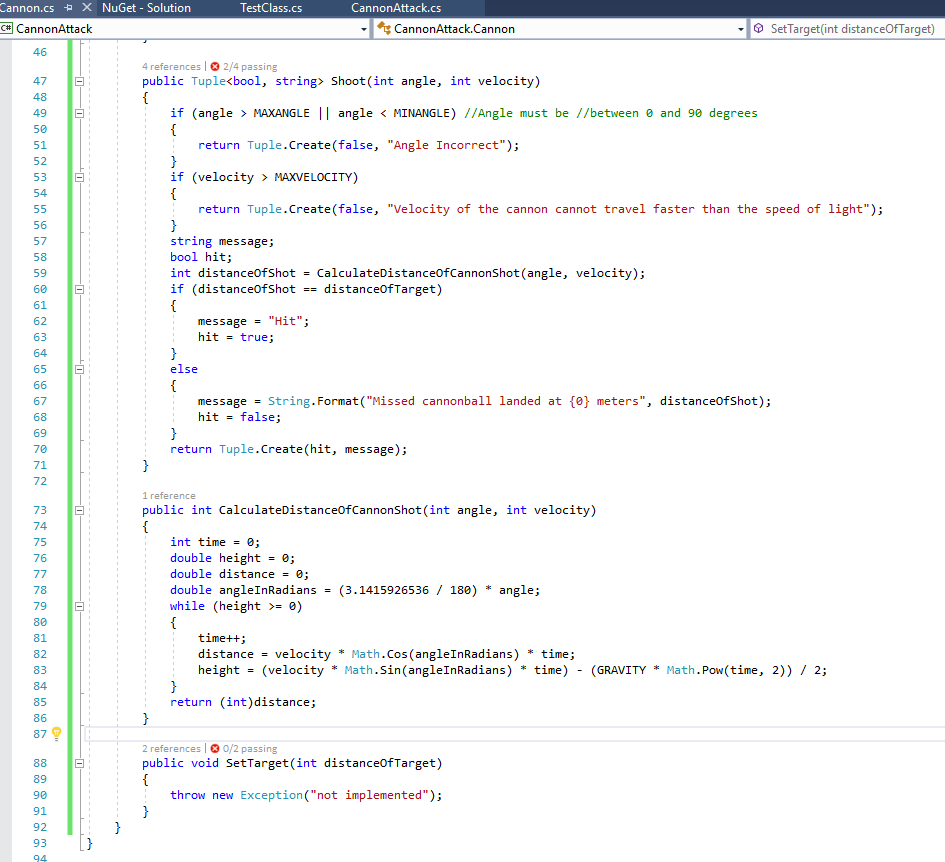
Now read tutorial for iteration 5

**Iteration 5 - Shoot, the Messenger:Shot!!!**  
  
Time for the real fun of the shoot method.etc.

Update Cannon to have

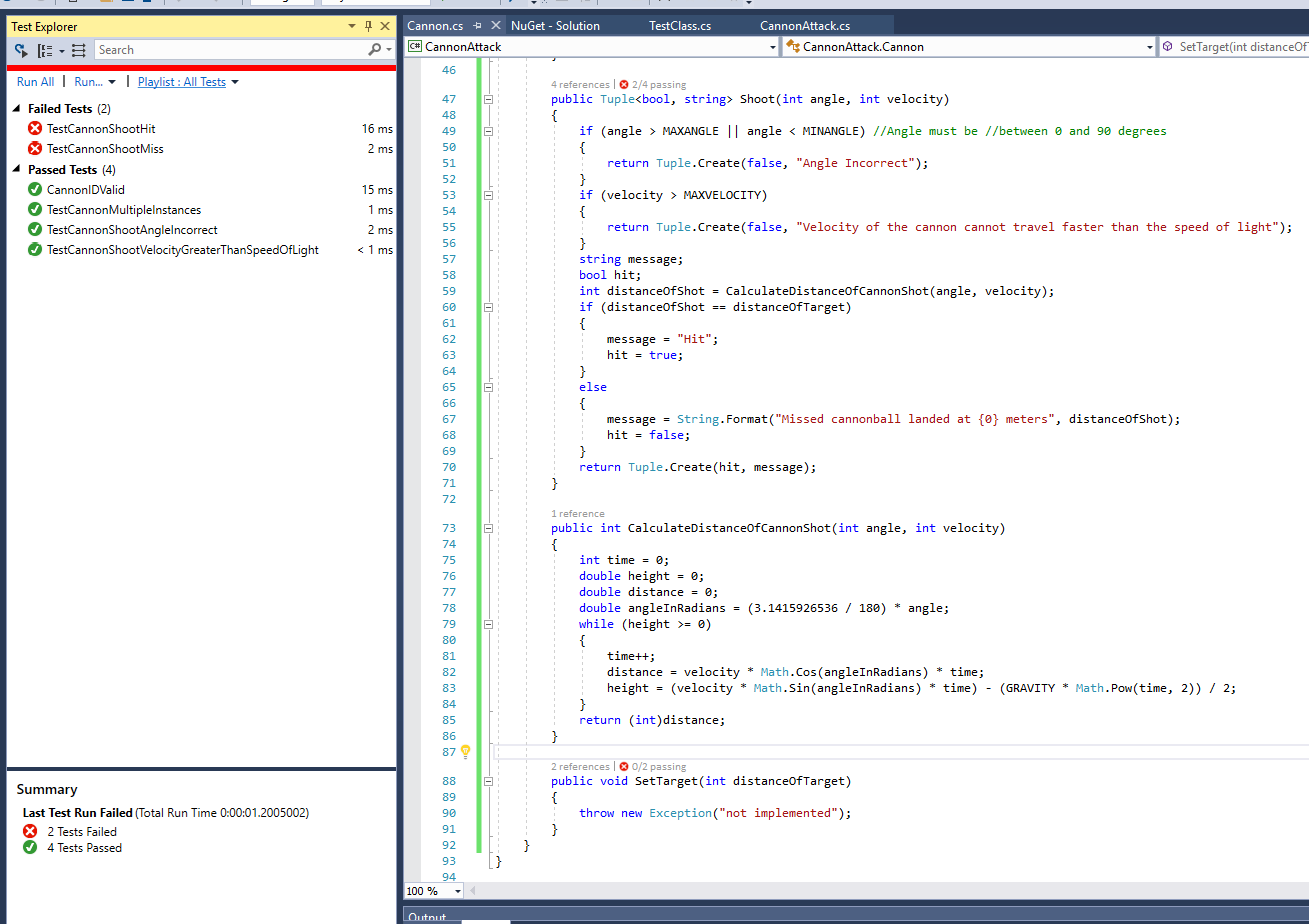


And

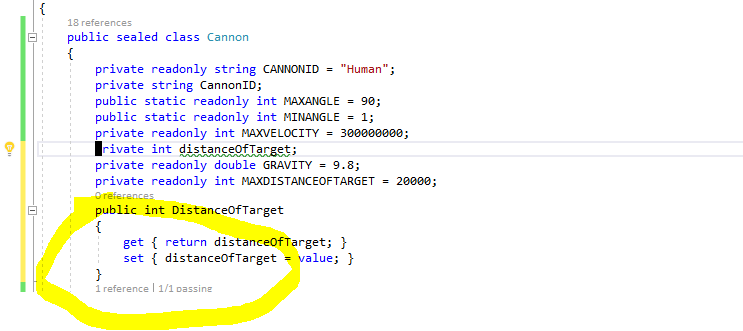


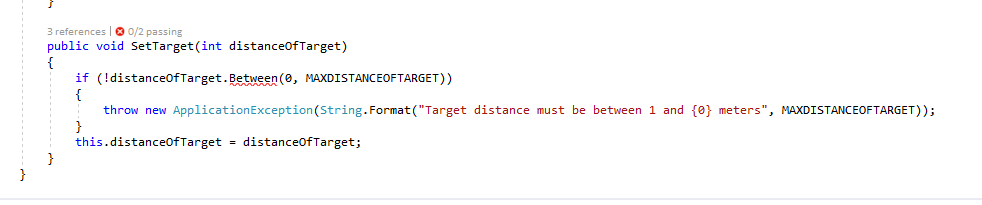
Run your tests

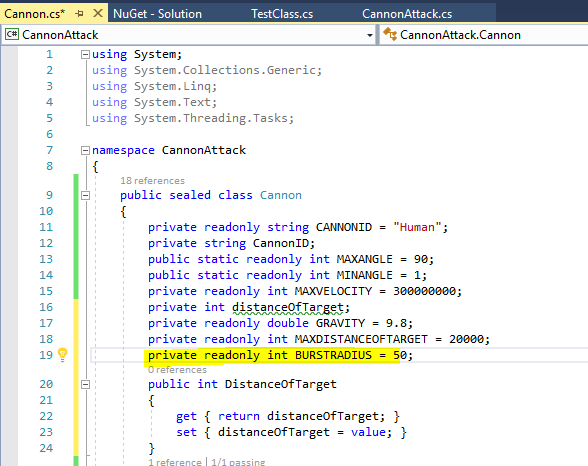
Update Set Target method

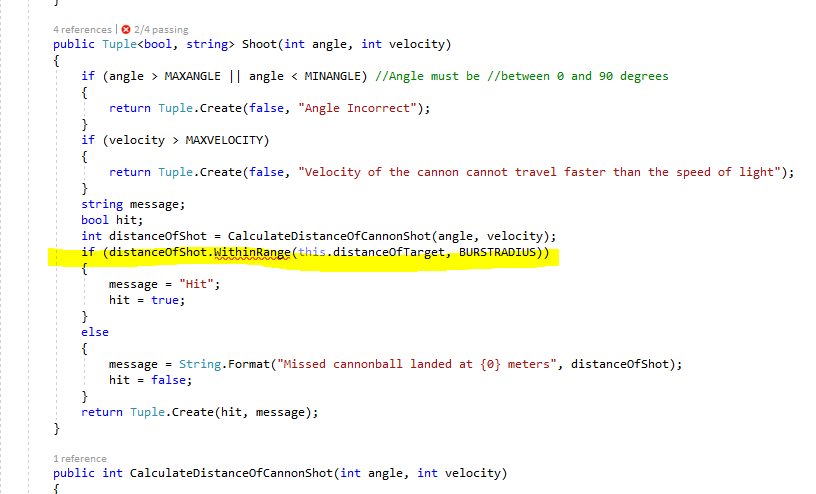


Update Cannon

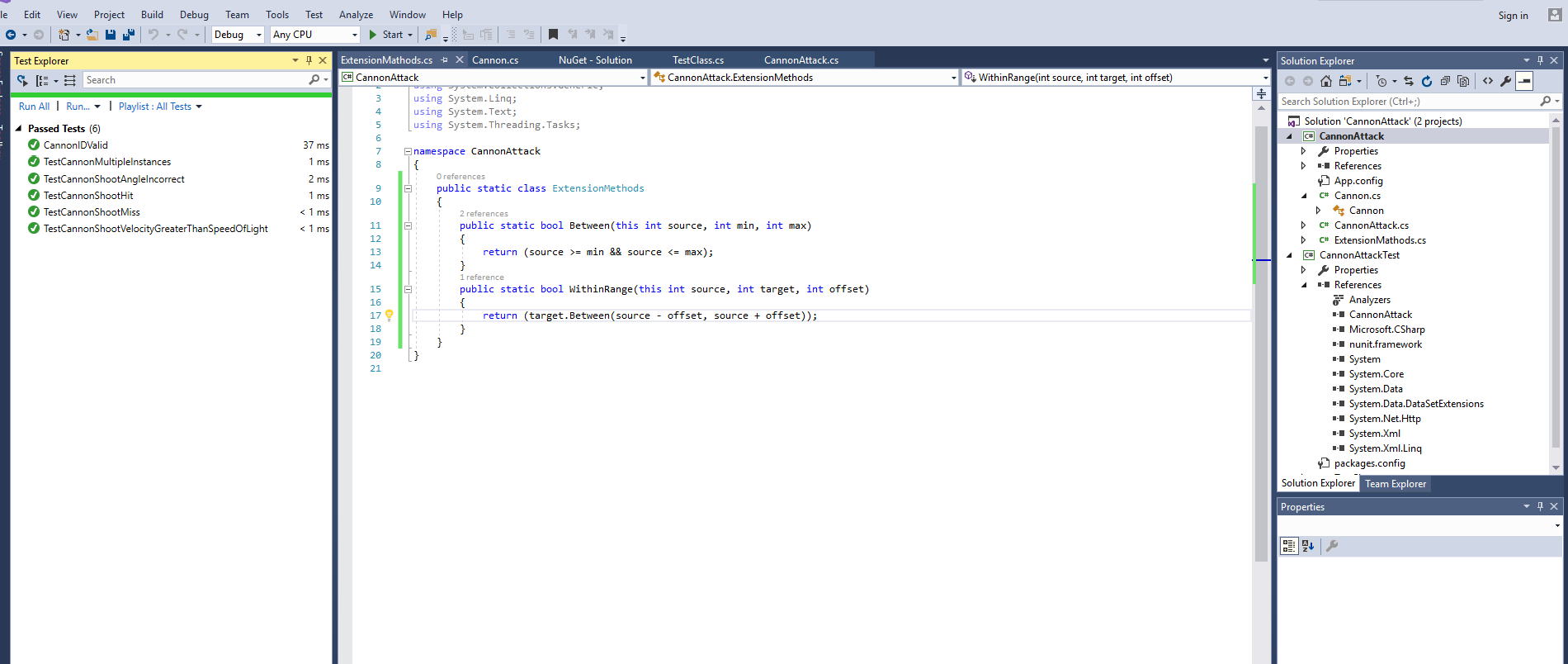








And add the Extension methods and run all tests again



**Iteration 6 - Counting Shots not Crows.**

Etc.

Test Driven Development (TDD) in Visual Studio 2010 (VS2010)

CannonAttack is a simple text based game in which a player enters an angle and velocity of a cannonball to hit a target at a given distance. The game uses a basic formula for calculating the trajectory of the cannonball (will be given) and the player keeps taking turns at shooting at the target until it has been hit.

Game Requirements and Specifications

1. Windows Console Application;
2. Player identified by an id, default is set to a constant "Human";
3. Single player only, no multi-play
4. Allow player to set Angle and Speed of the Cannon Ball to Shoot at a Target;
5. Target Distance is simply the distance of the Cannon to Target, and is created randomly by default but can be overridden;
6. Angle and Speed needs to be validated (specifically not greater than 90 degrees and Speed not greater than speed of light);
7. Max distance for target is 20000 meters;
8. Base the algorithm for the calculation of the cannons trajectory upon the following C# code (distance and height is meters and velocity is meters per second):  
   distance = velocity \* Math.Cos(angleInRadians) \* time;  
   height = (velocity \* Math.Sin(angleInRadians) \* time) - (GRAVITY \* Math.Pow(time, 2)) / 2;
9. A hit occurs if the cannon is within 50m of the target;
10. Display number of shots for a hit
11. Game text will be similar to following:   
    Welcome to Cannon Attack  
    Target Distance:12621m  
    Please Enter Angle:40  
    Please Enter Speed:350  
    Missed cannonball landed at 12333m  
    Please Enter Angle:45  
    Please Enter Speed:350  
    Hit - 2 Shots  
    Would you like to play again (Y/N)  
    Y  
    Target Distance:2078m  
    Please Enter Angle:45  
    Please Enter Speed:100  
    Missed cannonball landed at 1060m  
    Please Enter Angle:45  
    Please Enter Speed:170  
    Missed cannonball landed at 3005m  
    Please Enter Angle:45  
    Please Enter Speed:140  
    Hit - 3 shots   
    Would you like to play again (Y/N)  
    N  
    Thanks for playing CannonAttack