# *Web Programming VI (420-H60-HR)*

# *Lab 04a –Test Driven Development Using xUnit*

Date due: **Friday, September 13, 1:00pm (Part A)**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will have:

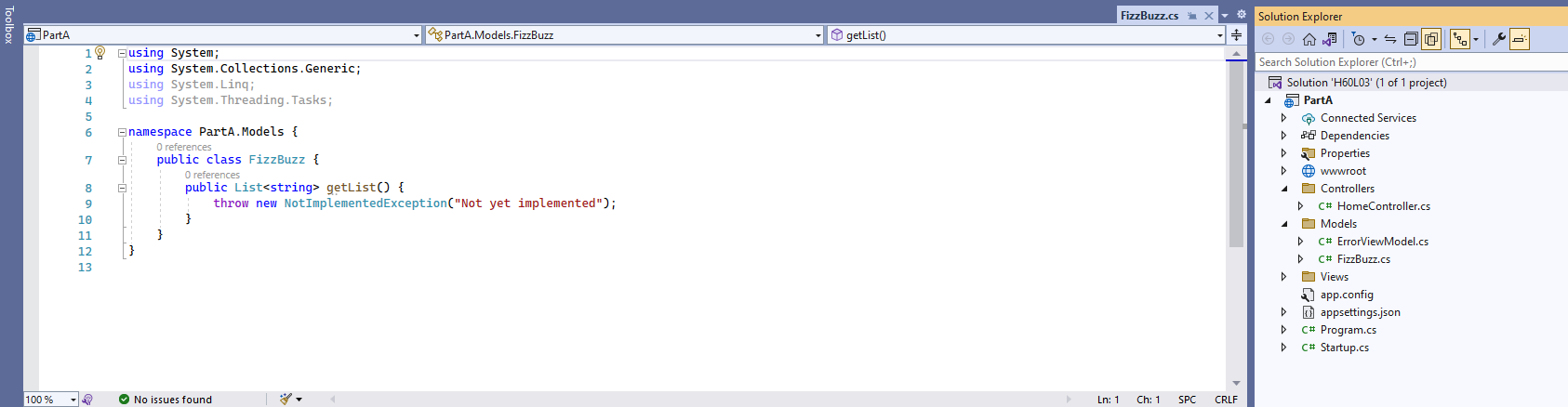
* Use xUnit with TDD to develop and test a project

Lab Setup

1. For the tests, use the naming format used in the class kata to name your namespace (Class being tested), test class (method being tested), and test methods: WhatsBeingTested\_ExpectedResult\_OtherInfo as the naming standard for your test cases. For example: ValidInputs\_ReturnCorrectResult\_Test01. Yes, the underscores are a pain, but with such long method names they are important for readability.

Part A – Fizz Buzz Testing

1. Create a solution called H60L03 with a MVC project called PartA. Create a Model class FizzBuzz. Your setup, and the signature for the FizzBuzz.getList() should look like this:



1. Create an xUnit test project in the solution. You are to test the FizzBuzz, getList() method.
2. Create test cases for each rule in turn and then implement the code for the method. For each step copy the method, comment out the first one and add the code to the second (copied) method to implement the code to make the failing test pass. I will want to see all the methods in turn (all commented out except the last one).
3. You are going to create tests for the following rules for the getList method of the Fizzbuuzz class. Remember to implement (a) test case(s) for each, run them to fail, implement only the minimal code to make them pass and then move on. The rules are:

* The method returns a list of numbers from 1 to 100
* When a number is a multiple of three return "Fizz" instead of the number,
* When a number is a multiple of five return "Buzz" instead of the number.
* When a number is a multiple of both three and five return "FizzBuzz" instead of the number.
* Add an integer parameter which is the upper limit to return. When a parameter is passed, the numbers from 1 to that number are returned. The default value for the parameter is 100.
* Add a second integer parameter which is the lower limit to return. When a parameter is passed, the numbers from that number to the upper limit are returned. The default value for the lower limit is 1
* Add a rule so that if the second parameter (lower limit) is greater than or equal to the first parameter (upper limit), return an ArgumentOutOfRangeException exception.

Note: any test methods that become obsolete with added functionality should be kept, but commented out for the sake of review.

1. Remember to keep the Singularity of Purpose Principle in mind. Be modular, use other (private) methods when refactoring the code. Richard hates gratuitous code duplication.
2. After you have it working add a call from the HomeController to display the list to a view (no formatting required as long as it can be read) with some sample ranges.

**Marking**

Marking is done in-lab.

|  |  |
| --- | --- |
| **Part A - FizzBuzz Testing** |  |
| 1st test - Returns list 1 to 100 | 4 |
| 2nd test - Fizz for 3 multiples | 4 |
| 3rd test - Buzz for 5 multiples | 3 |
| 4th test - FizzBuzz for 3&5 | 3 |
| 5th test - Upper limit with default | 4 |
| 6th test - Lower limit with default | 3 |
| 7th test - Argument out of Range exception | 3 |
| View samples | 4 |
| Refactoring and following process, 100% code coverage | 6 |
|  |  |
| **Part B - DI/MOQ** |  |
| Movie Project Controller cleanup (no dB context) | 4 |
| Test project setup, simple test - real DB | 6 |
| simple test - Moq DB | 6 |
| Moq test - average ratings | 6 |
| Handed in properly | 4 |
| **Total** | 60 |

**To submit**

When you have completed the lab, push your work in your github classroom repository.