**Topic 1: Test Driven Development.**

Top of Form

Bottom of Form

**Content**

**Reading**

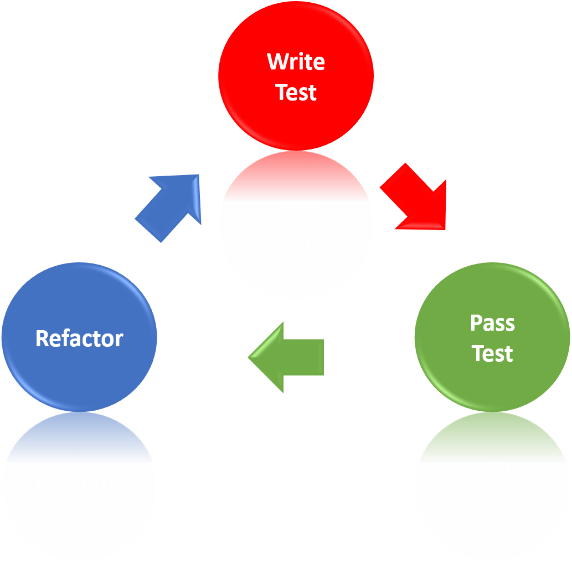
[Reading](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286661_1)

Test Driven Development --  start with the tests, then write the code!

* + Agile Data: agiledata.org
    - [Introduction to Test Driver Development (TDD)](http://agiledata.org/essays/tdd.html)
  + Agile Alliance: agilealliance.org
    - [What is Test Driven Development](https://www.agilealliance.org/glossary/tdd/)
  + Guru99: guru99.com
    - [What is Test Driven Development(TDD)? Tutorial with Example](https://www.guru99.com/test-driven-development.html)
  + freeCodeCamp: freecodecamp.org
    - [Test Driven Development: what it is, and what it is not](https://www.freecodecamp.org/news/test-driven-development-what-it-is-and-what-it-is-not-41fa6bca02a2/)

**Red, Green, Refactor**

[Red, Green, Refactor](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286661_1)

From <https://www.codecademy.com/articles/tdd-red-green-refactor>Read more there! 

Test-driven development (TDD) is an approach to software development where you write tests first, then use those tests to drive the design and development of your software application.

In this article, you will learn about a TDD approach called red, green, refactor, a framework that developers use to build a test suite, write implementation code, and optimize their codebase in short development cycles.

**RED, GREEN, REFACTOR**

The red, green, refactor approach helps developers compartmentalize their focus into three phases:

* + Red — think about what you want to develop
  + Green — think about how to make your tests pass
  + Refactor — think about how to improve your existing implementation

[**TDD Journal**](https://dmacc.blackboard.com/webapps/blackboard/content/launchLink.jsp?course_id=_102593_1&content_id=_7286666_1&mode=view)

After reading the articles, outline the process, as you understand it, in your own words for Test Driven Development. 100 word mininum.

This is 10 points.

[**TDD Method**](https://dmacc.blackboard.com/webapps/assignment/uploadAssignment?content_id=_7286667_1&course_id=_102593_1&group_id=&mode=view)

Write Unit tests for a boolean method ValidateStudentID() that accepts a string. It validates a DMACC student id (sometimes called a 900 number.)  Then implement the method to make each of the test pass.

You are not writing a full program, you are writing a method. You will need to include it in a class. If you want to test the method manually, you could add a main that asks for user input, calls the validate method and prints the results. The cases you would manually test will be the Unit Tests as listed below. Since the Unit Tests will accomplish the same testing without several runs of the program or you having to remember all the cases, they are a more efficient way of testing the method.

The unit tests should cover the expected cases of the method. Don't forget to use meaningful Unit Test names

* + Test 1. A valid 900 as input (method returns true)
  + Test 2. An invalid 900 number -- does not start with 9  (method returns false)
  + Test 3. An invalid 900 number -- second character is not zero (method returns false)
  + Test 4. An invalid 900 number -- contains letters (method returns false)
  + Test 5. An invalid 900 number -- too long (how long is your student id?)
  + Test 6. An invalid 900 number -- too short (how long is your student id?)

Write your method so all tests will pass. You should not need to change your Unit Tests unless you made an error in them. Likely, the change will be adding logic to your method.

Submit your Program.cs, UnitTest1.cs and image of your tests running. Do not zip your project.

This is 30 points