A3: Building Automated Unit Tests

Fresh off submitting the vending machine software to the client, your boss storms into your office. "You won't believe what I just learned about," she cries, "Automated unit tests! We've GOT to do this! NOW!" You roll your eyes: the firestorm never ends at this job.
Your job with your (up to) three-person coding team is to:

1. Translate the test scripts into unit tests with Visual Studio.

Learning Objectives

By the end of this assignment, you will:

- Conceptually group tests into sets.
- Design automated unit tests for those tests.
- Learn how to operate automated unit testing machinery within an IDE (Visual Studio).

Deliverable / Submission

You will deliver a .zip file into the Dropbox in D2L containing the test class that cover the set of test scripts from A2. Each test class should be some logical grouping of the tests from A2. Each test method should:

- have a **meaningful function name**, and
- the documentation/comment for each test method should say which test (e.g. T01) it is mimicking.

Optionally: You may include a test class called BonusTests that explores at least one interesting edge case not tested in the provided scripts (document why this test case is interesting in the comment for the test method(s)).
Tested against the solution from A2, your test cases should all pass. We will explore each of the test methods to ensure you have correctly designed the tests against the test scripts.

Steps

1. Download the A3 skeleton (link will work by beginning of lecture on 2/14)
2. Read through the documentation below
3. Add a new Unit Test Project to the solution
4. Without modifying any of the existing code from the Vending Machine project, add test classes and methods to mimic the included test scripts.
5. Submit via D2L (indicate who is on your team). Only one submission per team.

Description

In A1 and A2, the test scripts were driven by the ScriptParser, which in turn called on the VendingMachineFactory. For this assignment, both of these have been removed. Your job is to design unit tests that mimic each of these test scripts.

Each of these unit tests does two things: (1) sets up the preconditions and inputs (i.e. for that test), and (2) defines a specified expected result.
In order to complete (1), you will need to instantiate a vending machine, its logic, and then actually manipulate the hardware. To complete (2), you will need to use functions from the Assert class. Remember that the idea is that Assert functions check whether an expected is the same as an actual value.
For test scripts that were expected to pass, they should pass all the Assert functions. For test scripts that were expected to fail, they fail by raising an exception. In these cases, you should indicate that you expect this with the ExpectedException attribute that you tag your test method with (so that the method actually passes as a test).
You may assemble a team of up to three people (this means you can work independently, or with a partner, or with two partners). **Ensure that you are all from the same tutorial.**

Unit Testing in C#/Visual Studio

One of the tutorials you attended will have reviewed how to construct unit tests in Visual Studio. You can always refresh yourself on the material. Documentation on the Assert class is likely to be useful.
One function that was not covered in tutorial that might be useful is the Assert.Fail method, which immediately throws an AssertFailedException. You will likely find this useful some test cases that you expect to fail, but the underlying code does not throw an exception.

```
[TestMethod]
[ExpectedException(typeof(AssertFailedException))]
public void FailsImmediately() {
    Assert.Fail("BOOM!");
}
```

Note that the above test "passes" because we've also indicated that we *expect* this exception to be thrown using the ExpectedException attribute.
Finally, you will notice that after a while, you are likely repeating yourself across several TestMethods (e.g. to set up the test). Within a class, you can set up (i.e. initialize) all the TestMethods in that class using a method that is labeled with the TestInitialize attribute.

```
[TestInitialize]
public void InitializeAllTests() {
    // blah blah blah
}
```

To test your test cases, remember that you don't actually "Run" any code -- instead, use Test -> Run -> All Tests.

- Tutorial on Unit Testing
- Assert class documentation

Hints

1. Look at the test scripts in test-scripts/, and think about what each script is doing.
2. For each test, figure out what it is doing, and translate these "what it is doing" into what manipulations should be happening against the vending machine.
3. Finally, for each of these, figure out what the expected result is.

4. Make sure you add an instance of VendingMachineLogic to drive your VendingMachine. This will be known as "Look for Hint 4" on #A3.

Questions

Post your questions to the #A3 channel on Slack.

Grading Rubric

You code will be evaluated on whether your tests are equivalent to the test scripts, and whether they all pass against my A2 solution (they should all pass).
Make sure you have created the same number of TestMethods as test scripts you started with (21), and label clearly which test scripts each TestMethod is addressing (e.g. in a comment, or by the function name).

- 30% of the assessment will be based on whether these TestMethods pass correctly, and,
- 70% of the assessment will be based on whether the TestMethods are implemented such that they match what the original tests scripts were doing.

NOTE: Do NOT use VendingMachineFactory.

No Late Assignments

Your assignment is due at noon on the specified due date. No late assignments will be accepted.