**UNIT TESTING GUIDELINE**

# Get started with unit testing

Get started and create UT project:

<https://docs.microsoft.com/vi-vn/visualstudio/test/getting-started-with-unit-testing?view=vs-2017>

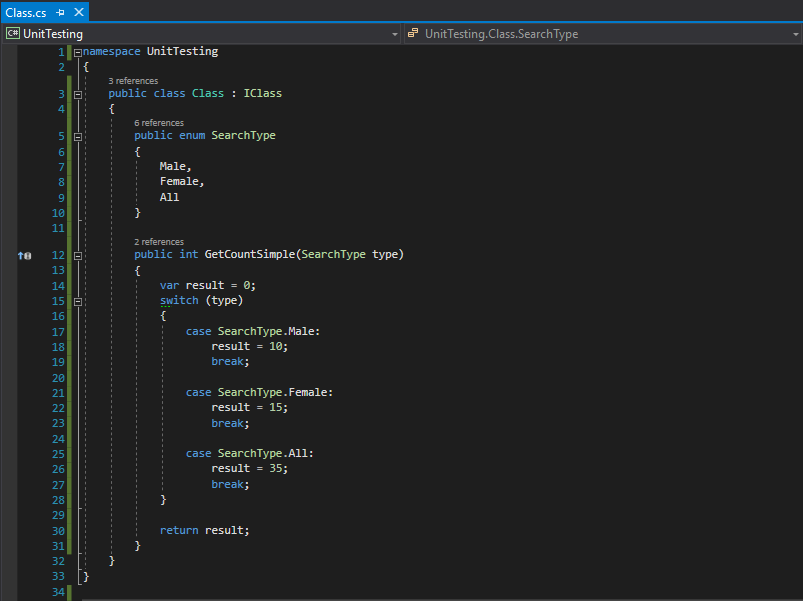
# Create unit test

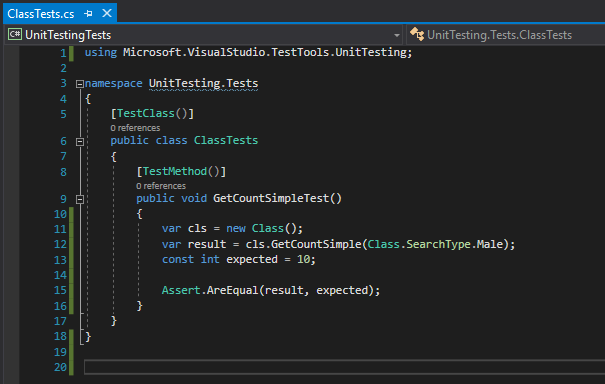
## Create unit test

Create and run UT:

<https://docs.microsoft.com/vi-vn/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2017>

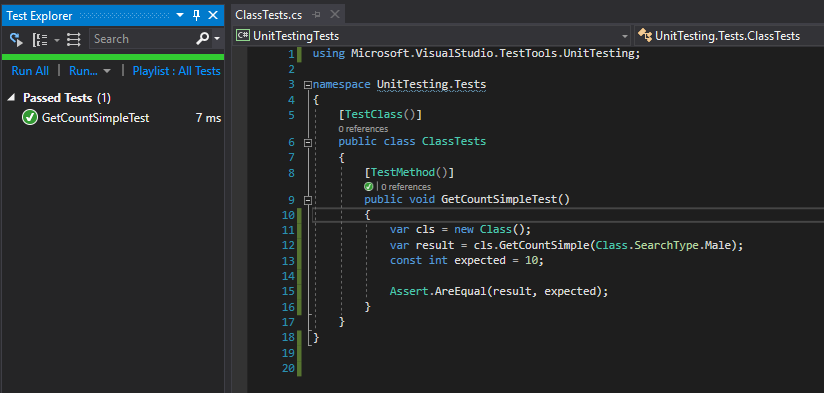
Ví dụ: cần viết Unit test cho *GetCountSimple(SearchType type)* method của *Class* object như sau:



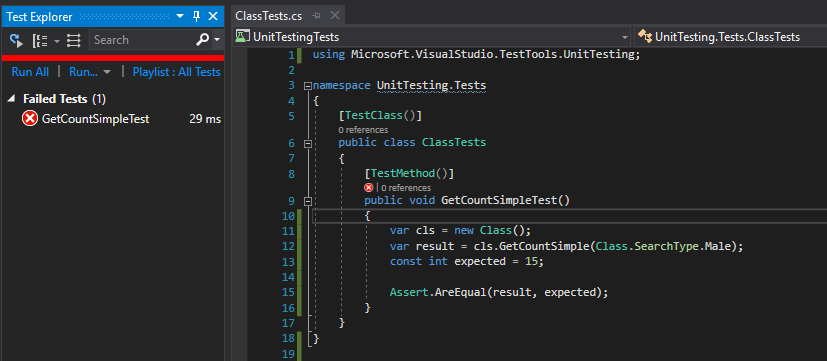


Hàm *GetCountSimpleTest()* dùng để kiểm tra giá trị khi call có giống như expected hay không.

Ở trường hợp này sẽ dùng *Assert.AreEqual(result, expected)* để kiểm tra giá trị mong đợi trả về.



Kết quả result và expected giống nhau nên test case này *Passed*.



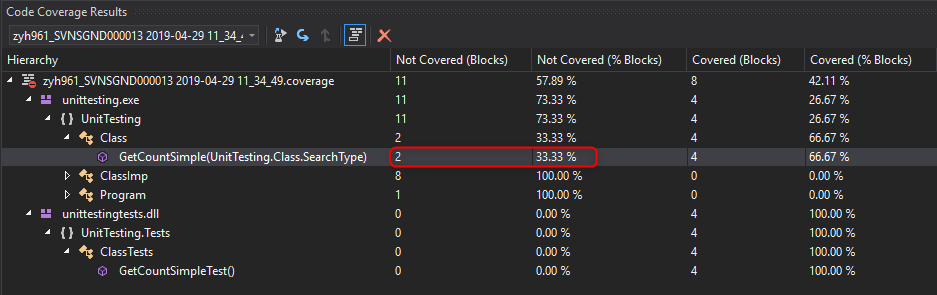
Kết quả result và expected không giống nhau nên test case này *Failed*.

## Use code coverage

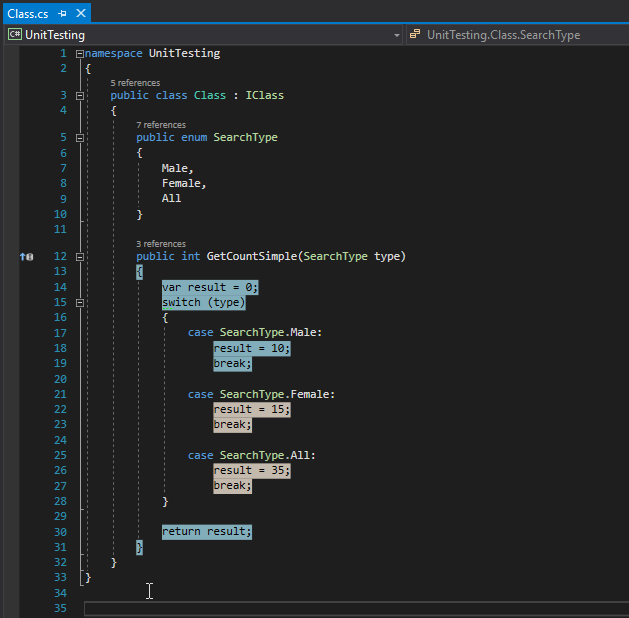
Dùng để xác định tỷ lệ source code thực tế được test bằng unit test code.

<https://docs.microsoft.com/vi-vn/visualstudio/test/using-code-coverage-to-determine-how-much-code-is-being-tested?view=vs-2017>

Ở ví dụ trên, khi run Test→Analyze Code Coverage sẽ được kết quả như sau:



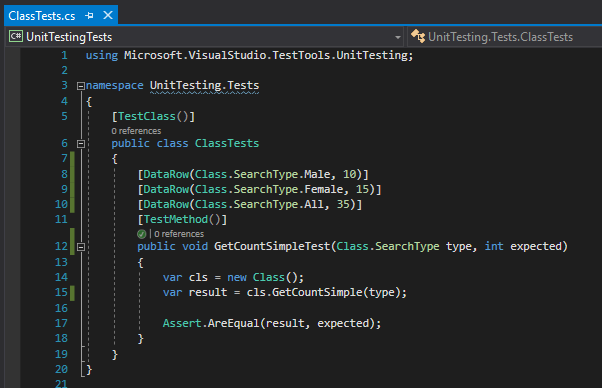
Tỷ lệ Not Covered (% Blocks) là 33.33% → khả năng là đang bị thiếu test case cho *GetCountSimple* method.



Sau khi kiểm tra lại *GetCountSimple* method, sẽ nhận thấy thiếu test case (dòng code được highlight màu cam) cho 2 trường hợp:

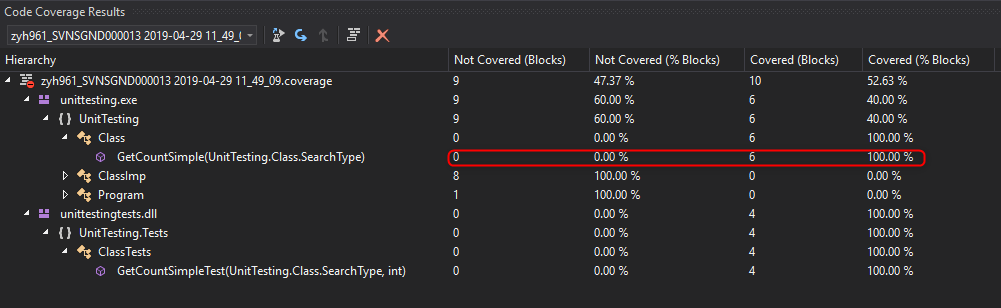
* *SearchType.Female*.
* *SearchType.All*.

Có thể viết thêm 2 test method để test cho 2 trường hợp trên hoặc modify lại *GetCountSimpleTest* (dùng *DataRow*) như sau:



Đối với trường hợp này sẽ định nghĩa *DataRow* có 2 params là SearchType và expected.

Sau khi run lại Analyze Code Coverage sẽ được kết quả như sau:





Tất cả line of code của *GetCountSimple* method đã được covered.

→ Analyze Code Coverage chỉ dùng để hỗ trợ kiểm tra tỷ lệ code coverage, nhằm xác định có thiếu test case hay không, còn **việc** **ngăn chặn bug** là **phụ thuộc** vào **chất lượng** nội dung **của** **test case**.

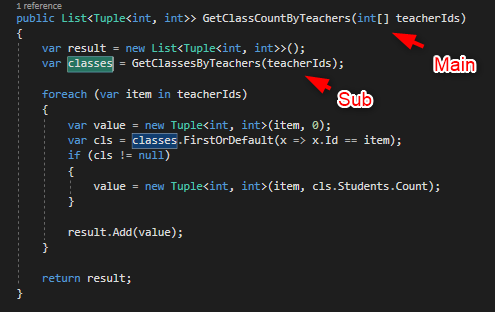
# Basic steps on unit testing creation

## Steps

Thứ tự tạo UT cho 1 method:

* Step1: Tạo unit test code verify cho các method được gọi bởi method chính.
* Step2: Giả định giá trị return của các method được gọi và verify method chính. (Giả định sub method luôn luôn execute đúng)

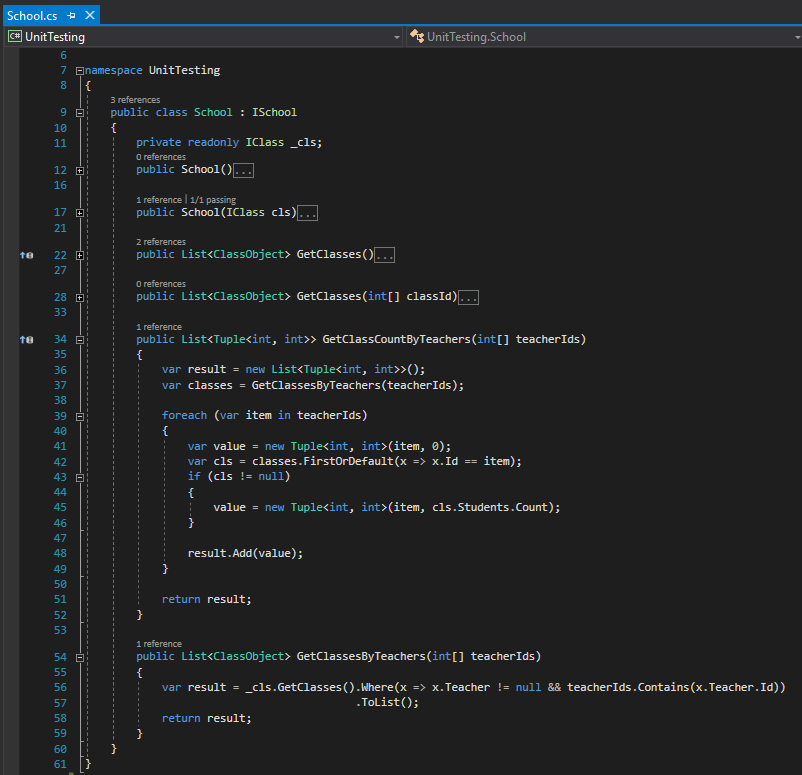
Ví dụ:

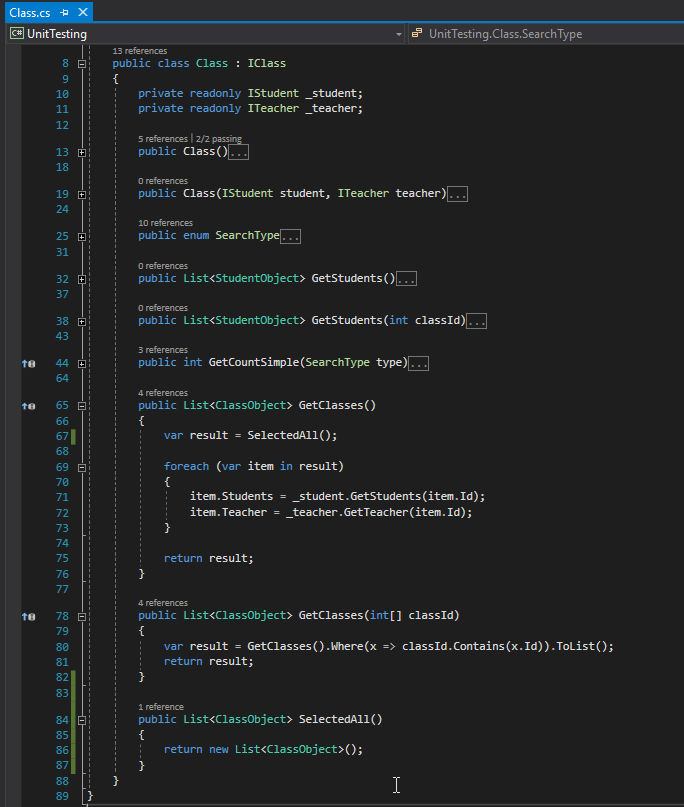


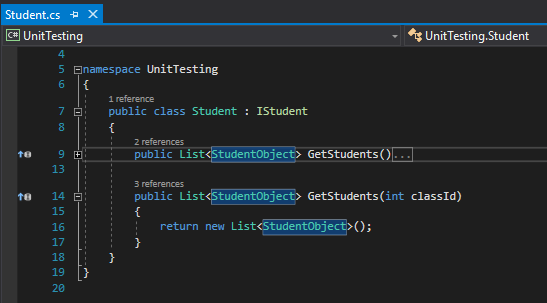
* Step1: Viết UT cho *GetClassesByTeachers* Sub method.
* Step2: Viết UT cho Main method kết hợp giả định dữ liệu của Sub method bằng 1 số phương pháp như Stub, Shim (Microsoft Fakes) và Mock (Moq library) – Tham khảo section 4.

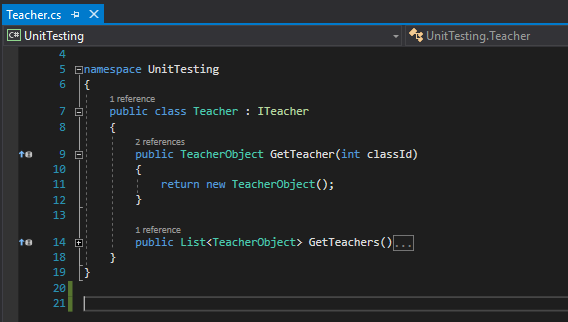
## Sample UT

Ở ví dụ trên, ta có *GetClassCountByTeachers(int[] teacherIds)* method được implement như sau:





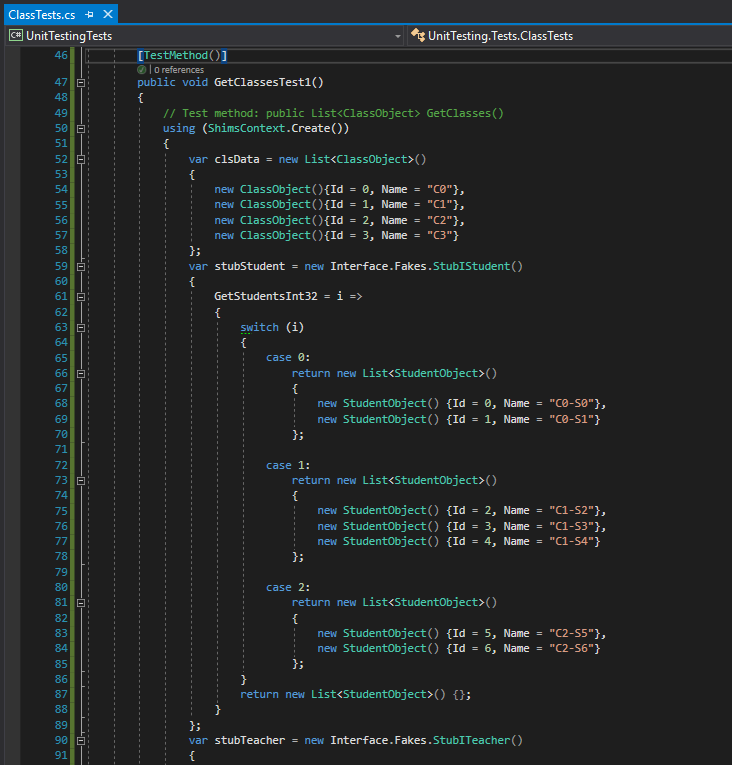


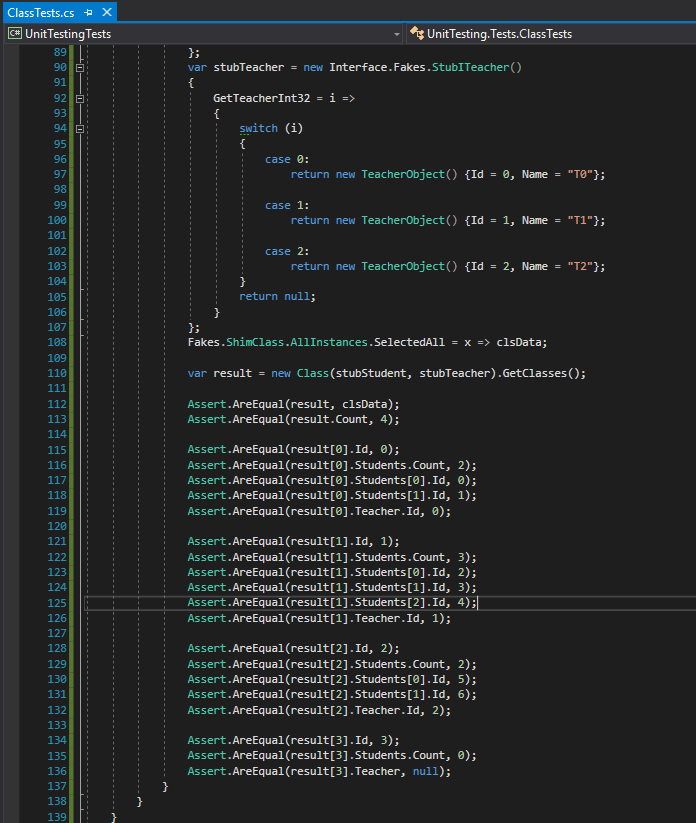


→ Cách tạo UT:

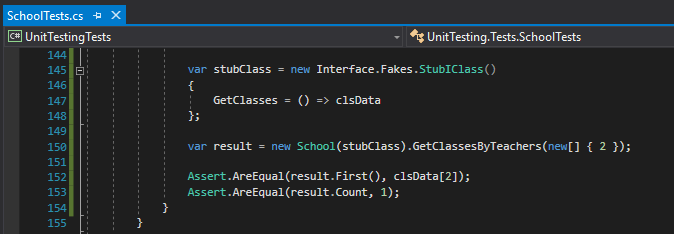
* \*Create UT cho *SelectedAll()* method của *Class* object.
* \*Create UT cho *GetStudents(int classId)* method của *Student* object.
* \*Create UT cho *GetTeacher(int classId)* method của *Teacher* object.
* Create UT cho *GetClasses()* method của *Class* object.
* Create UT cho *GetClassesByTeachers(int[] teacherIds)* method của *School* object.
* Tạo Stub/Mock cho *IClass* interface để giả định result value cho *GetClasses()* method.
* So sánh result và expected value của *GetClassCountByTeachers(int[] teacherIds)* method.

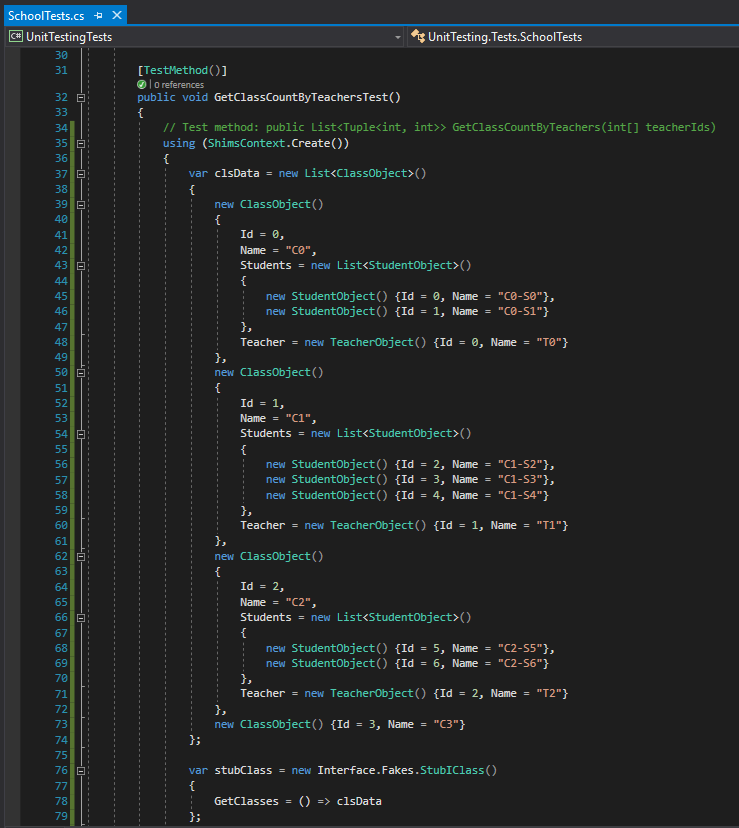
(\*) Các method này chưa được implement đầy đủ nên sẽ giả định data cho việc test.

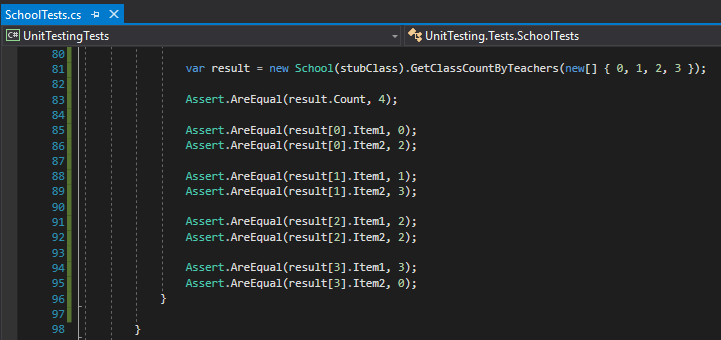












# Isolate code under test with Microsoft Fakes

## Stub (Microsoft Fakes)

Tham khảo:

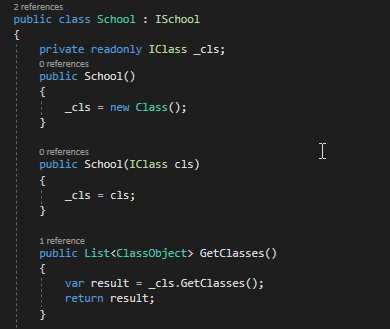
<https://docs.microsoft.com/vi-vn/visualstudio/test/using-stubs-to-isolate-parts-of-your-application-from-each-other-for-unit-testing?view=vs-2017>

Là 1 trong 2 công nghệ mà Microsoft Fakes cung cấp cho phép dễ dàng tách biệt (isolate) 1 component cần test từ 1 component khác mà nó gọi. Lợi ích là mô phỏng component khác trong trường hợp nó chưa được hoàn thiện.

Đặc điểm:

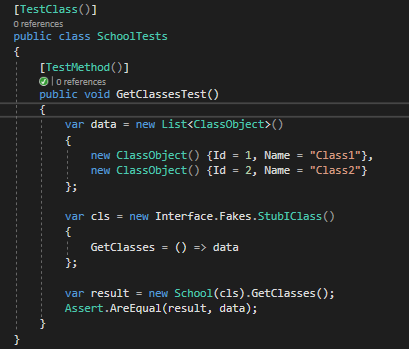
* Thay thế đối tượng thật
* Giúp việc test thực hiện nhanh hoặc giả định các component không có thật/chưa implement đầy đủ.
* Cụ thể hóa interface, cho phép các method trả về các giá trị cố định.
* Dùng kiểm tra trạng thái before/after sau khi thực hiện method.

Ví dụ:

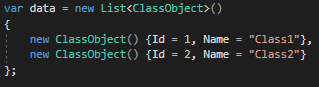


Class *School* có sử dụng *GetClasses* method của interface *IClass*.

Trường hợp interface này chưa được implement các method 1 cách đầy đủ, thì sẽ sử dụng Stub để giả định data cho việc test *GetClasses* method của class *School*.



Tạo đối tượng *StubIClass* giả định *GetClasses* method luôn có giá trị trả về là *data*.



So sánh *Assert.AreEqual(result, data)*.

## Shim (Microsoft Fakes)

Tham khảo:

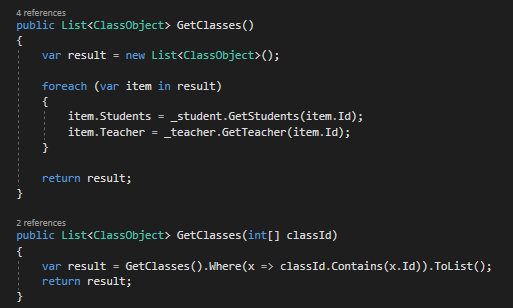
<https://docs.microsoft.com/vi-vn/visualstudio/test/using-shims-to-isolate-your-application-from-other-assemblies-for-unit-testing?view=vs-2017>

Là 1 trong 2 công nghệ mà Microsoft Fakes cung cấp cho phép giả định kết quả trả về của method.

Đặc điểm:

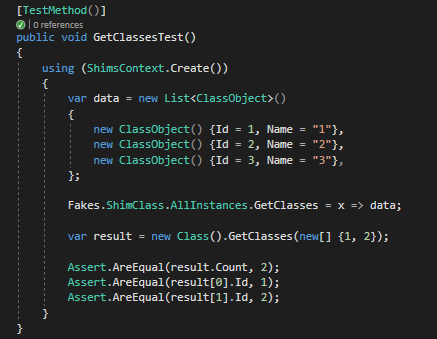
* Thay thế đối tượng thật
* Giúp việc test thực hiện nhanh hoặc giả định kết quả trả về của method.
* Có thể giả định kết quả trả về của System static method. (Ví dụ: *DateTime.Now*)
* Dùng kiểm tra trạng thái before/after sau khi thực hiện method.

Ví dụ:



Ở ví dụ này, có 2 method *GetClasses*. Công việc cần làm là viết UT cho method *GetClasses(int[] classId)*.

Giả sử method *GetClasses()* đã được test pass, nên sẽ dùng Shim để giả định kết quả trả về của method này để thực hiện UT cho method *GetClasses(int[] classId)*.



# Third-Party Unit test

## Mock (Moq library)

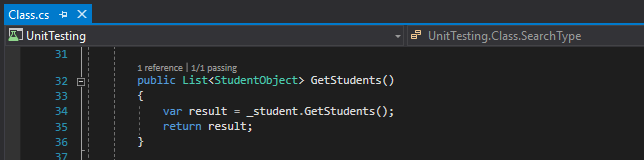
Tham khảo:

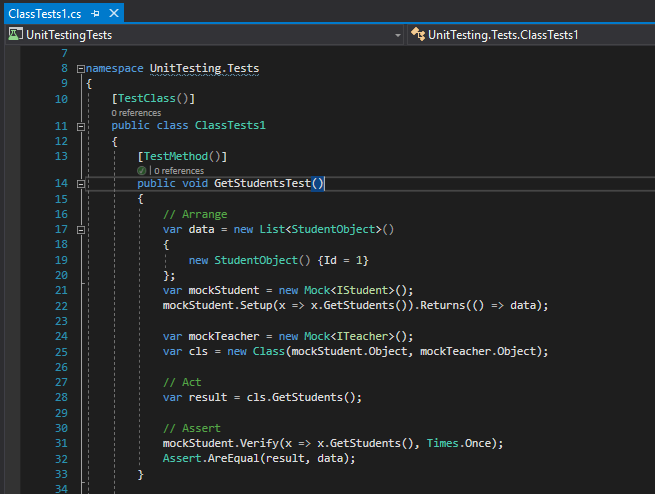
<https://www.youtube.com/watch?v=dZ2Psa_Bn2Q>

Đặc điểm:

* Thay thế đối tượng thật
* Cụ thể hóa interface nhưng chỉ ra các method nào muốn hoặc không muốn thực hiện, cách thức thực hiện, giá trị trả về và đối số mong muốn. (Ví dụ: verify số lần call của các sub method trong main method dựa vào input params)
* Kiểm tra trạng thái và hành vi (cách thức method được thực hiện – đây là điểm khác biệt lớn nhất của Stub và Mock).

Ví dụ 1:





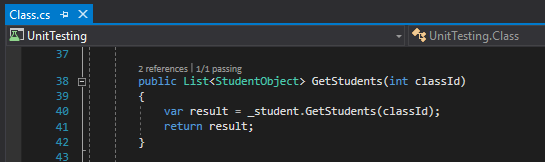
Cú pháp AAA (Arrange, Act, Assert) là cách tổ chức UT: Arrange - tạo các phụ thuộc cần cho việc test, Act – thực hiện test, Assert – so sánh kết quả mong muốn.

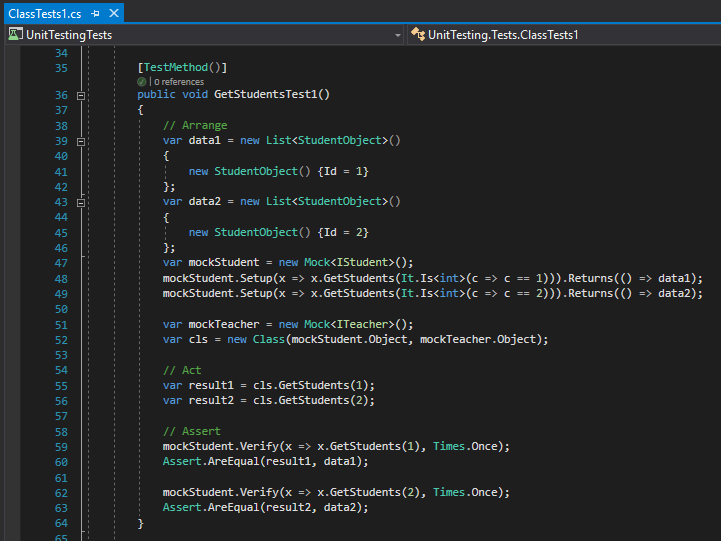
Ở ví dụ này, sẽ sử dụng mock object cho đối tượng *IStudent* và *ITeacher*. Để lấy instance của mock object, sử dụng thuộc tính *Object*.

Hàm *Setup* cho phép định nghĩa method được call và giá trị trả về của nó. Ở ví dụ này, method *GetStudents()* được định nghĩa giá trị trả về luôn là *data*.

Hàm *Verify* cho phép xác nhận method được call trên mock object. Ở ví dụ này, mục đích việc verify dùng để kiểm tra method *GetStudents()* chỉ được execute duy nhất 1 lần.

Ví dụ 2:





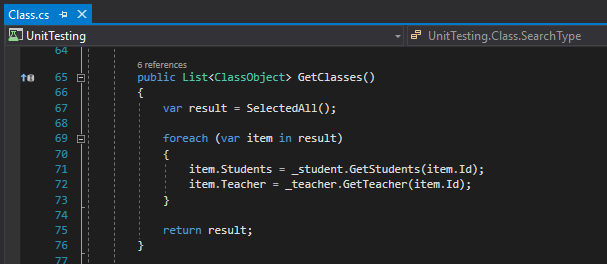
Ở ví dụ này, cũng tương tự Ví dụ 1, sẽ sử dụng mock object cho đối tượng *IStudent* và *ITeacher*.

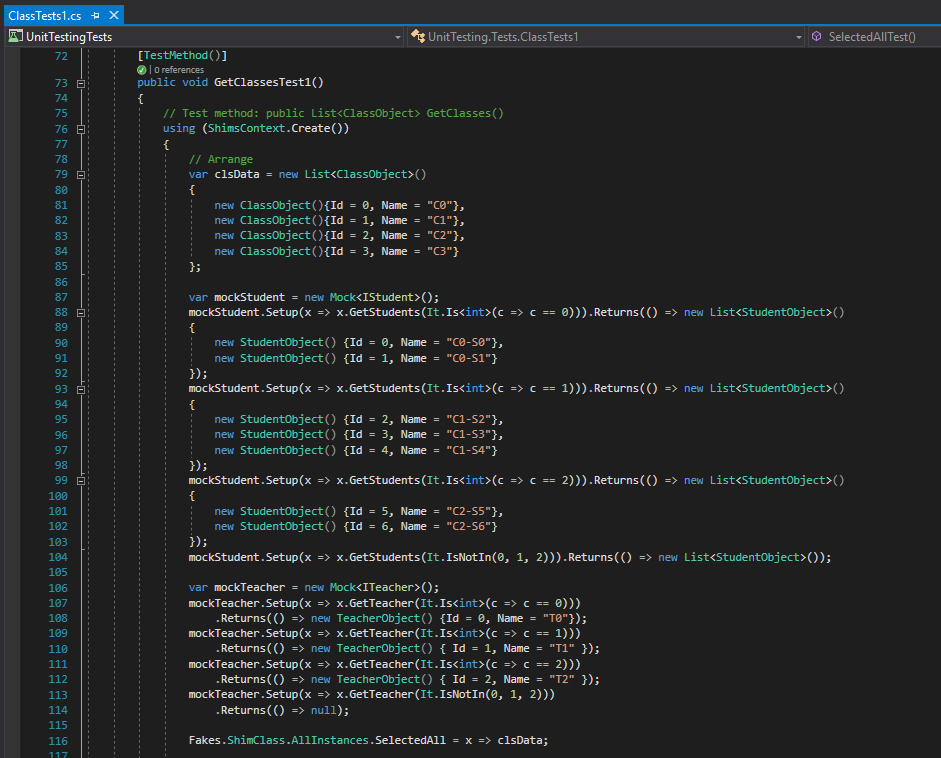
Instance *It* cung cấp 1 số method dùng để định nghĩa input param cho test method.

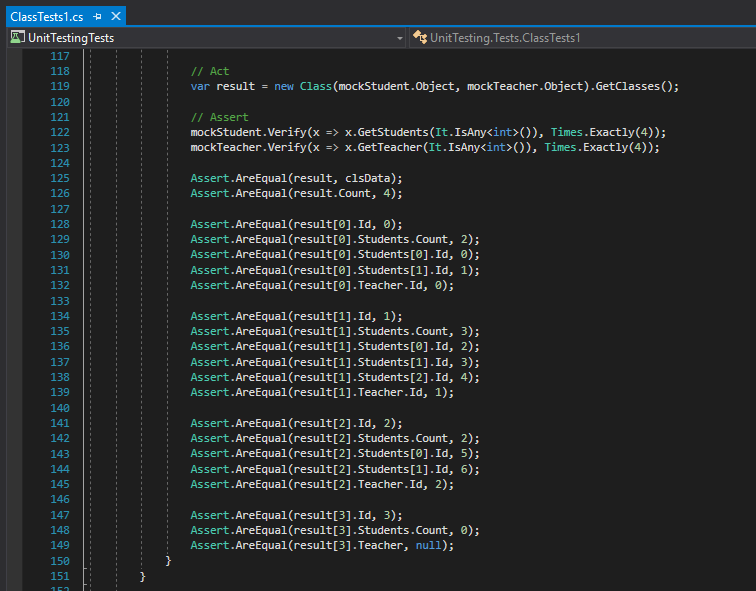
Hàm *Setup* định nghĩa method *GetStudents()* nếu input param là 1 thì sẽ có return value là *data1*. Tương tự nếu input param là 2 thì sẽ có return value là *data2*.

Hàm *Verify* cho phép xác nhận method *GetStudents()* được call 1 lần cho trường hợp input param là 1 và 1 lần cho trường hợp input param là 2.

Ví dụ 3:







Ở ví dụ này, sẽ demo sự kết hợp dùng **Shim (Microsoft Fakes)** và **Mock**.

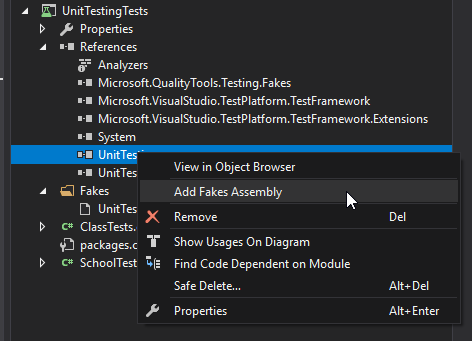
* Shim dùng để giả định data cho method *SelectedAll()*.
* Mock dùng kiểm tra hành vi của test method.

Với data được giả định *clsData*, hàm *Verify* xác nhận method *GetStudents()* được call 4 lần và method *GetTeacher()* được call 4 lần.

# Other notices

## Add reference for Microsoft Fakes

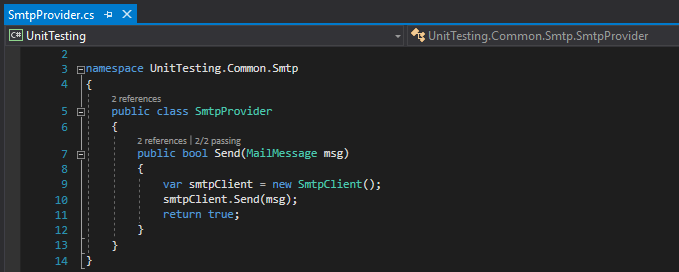
Unit test project → References → Chọn *“Add Fakes Assembly”* cho reference cần Fakes.



## Wrapper class

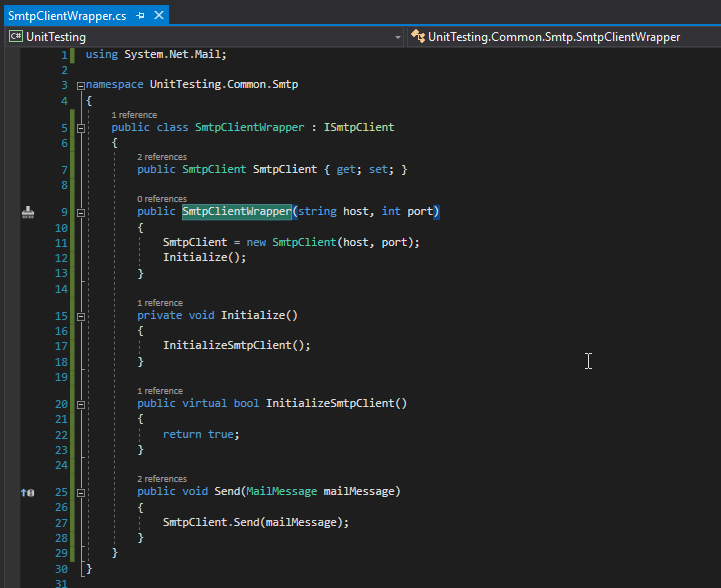
Có thể dùng wrapper class cho trường hợp cần ngăn chặn xử lý của system khi execute unit test.

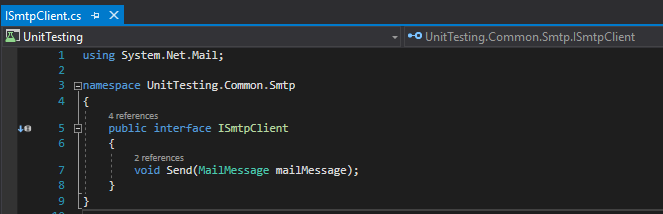
Ví dụ: dùng wrapper để ngăn chặn việc gửi mail khi execute test code nếu dùng *SmtpClient* class.

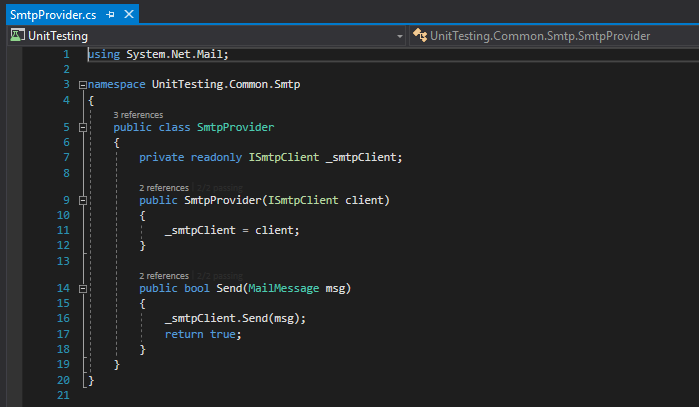


Ở cách dùng này, nếu mỗi lần execute UT cho *Send(MailMessage msg)* method, email sẽ được send qua mailbox.

***→ Sử dụng Wrapper class:***





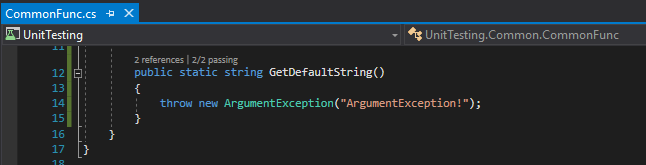




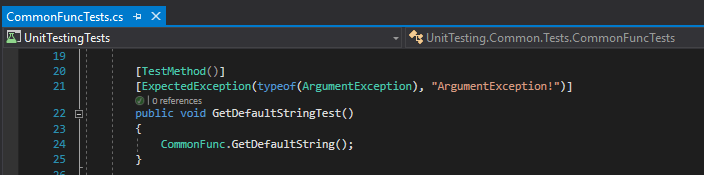
Ở cách dùng này, UT sẽ implement đối tượng *ISmtpClient* dưới dạng Stub/Mock. Do đó sẽ ngăn chặn việc send mail không mong muốn.

## Assert to verify an exception has been thrown

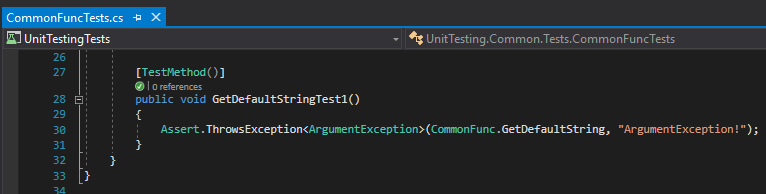
Cách để Assert exception thrown



Cách 1:



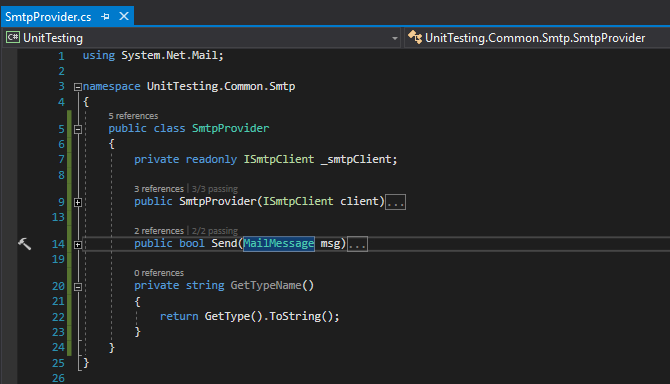
Cách 2:

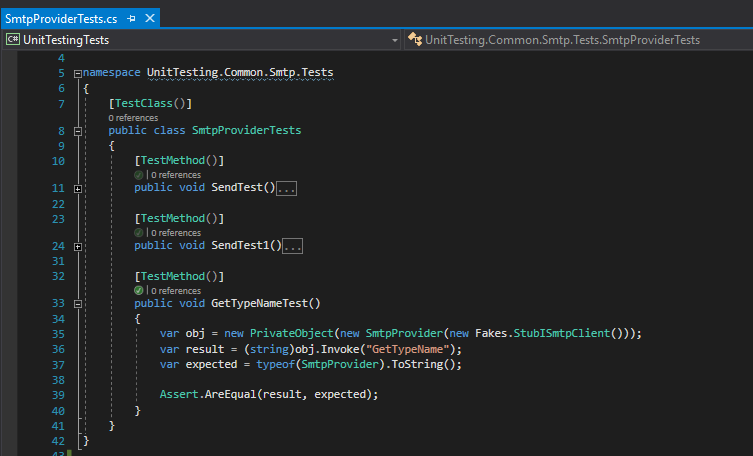


## Write UT for *private methods*

Sử dụng *PrivateObject* để viết UT code cho private methods.

Ví dụ: viết UT cho *GetTypeName()* method như sau:

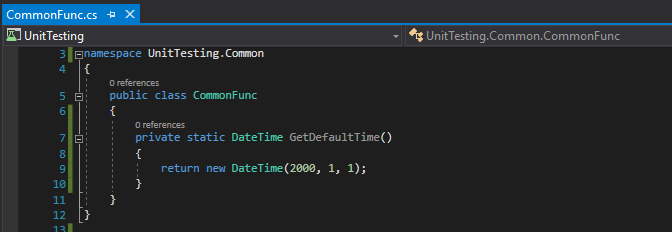


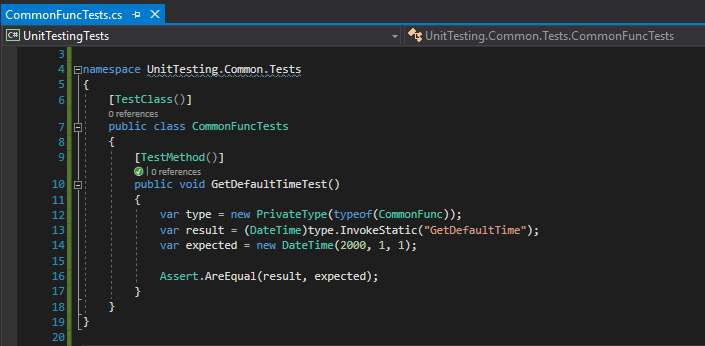


## Write UT for *private static methods*

Sử dụng *PrivateType* để viết UT code cho private static methods.

Ví dụ: viết UT cho *GetTypeName()* method như sau:





## Use a configuration file to define a data source

Tham khảo:

<https://docs.microsoft.com/vi-vn/visualstudio/test/walkthrough-using-a-configuration-file-to-define-a-data-source?view=vs-2017>

## Unit tests for generic methods

Tham khảo:

<https://docs.microsoft.com/vi-vn/visualstudio/test/unit-tests-for-generic-methods?view=vs-2017>

# Attachment

