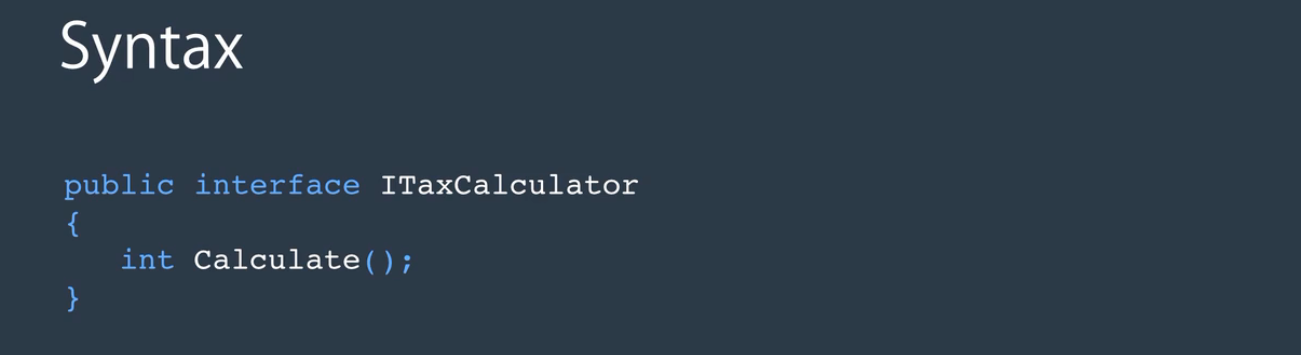
032 What is an Interface





Interface tên bắt đầu băng chữ I

Interface tạo ra một cái cấu trúc bên trong đó chứa khai bao các method và properties, khi class sử dụng cái interface đó, có thể gọi các method bên trong interface và định nghĩa lạ để sử dụng. Như vậy các method đó trong các class sẽ cùng tên, cùng cấu trúc, nhưng lại độc lập với nhau.

Vì thế ta có thể dùng nó để làm cấu trúc cho 1 class, với những method và propeties giống nhau, nhưng những class đó có cách thức khác nhau: ví dụ việc ghi log, tạo các class ghi log giống nhau nhưng dùng phương thức khác nhau bằng iterface, sau đó có thể dùng depenece injection để đưa vào class khác để thực hiện

Trong testing dùng interface tạo ra class dữ liệu để đưa vào class đang test, như vậy chỉ test độc lập class đang test chứ không ảnh hưởng đến class khác



Là một cái app mà từng component độc lập, không liên quan đến các component khác

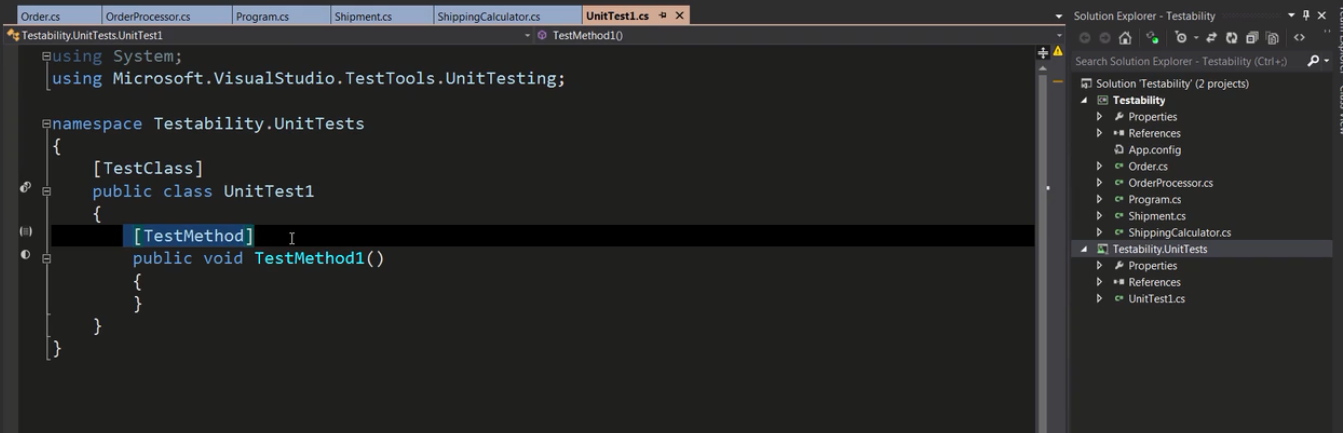
Ví dụ:

Nhà hàng có đầu bếp, và bất kỳ ai cũng có thể là đầu bếp -> component độc lâp

Nhưng nhà hàng cần john làm đầu bếp -> những ảnh hưởng lên john sẽ ảnh hưởng lên nhà hàng

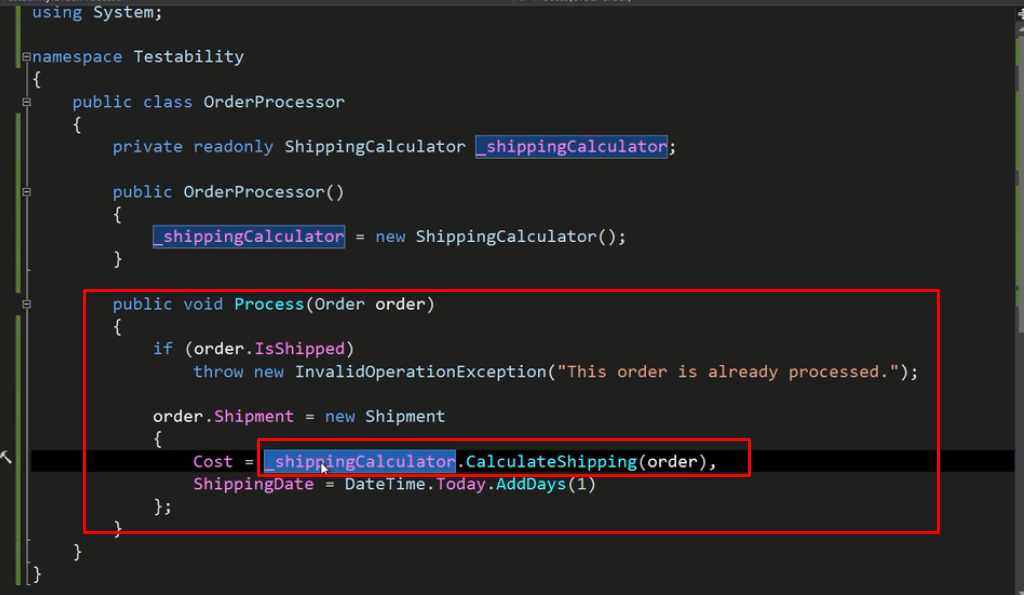
Iterface tạo ra để khi có thay đổi trên interface thì các thành phần class hoặc interface khác không bị ảnh hưởng hoặc ảnh hưởng rất ít

033 Interfaces and Testability

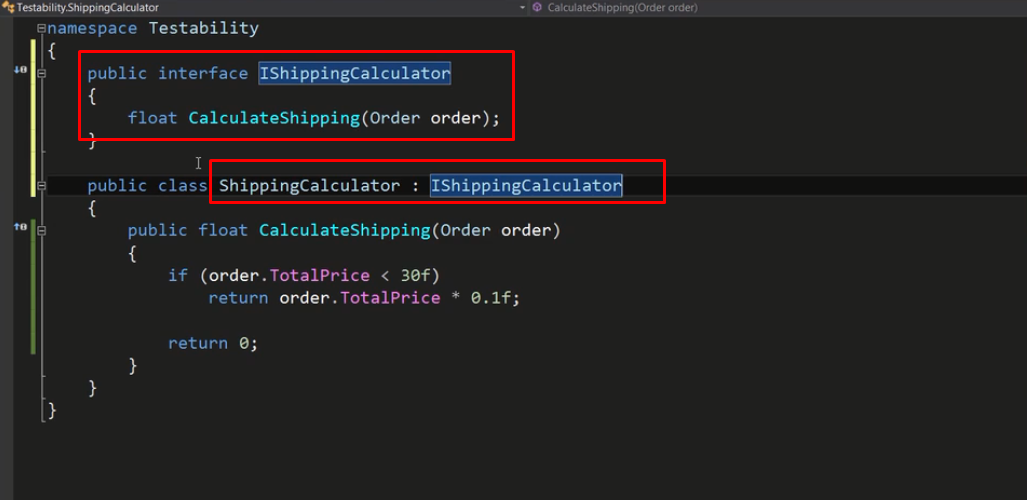


Tạo unit test

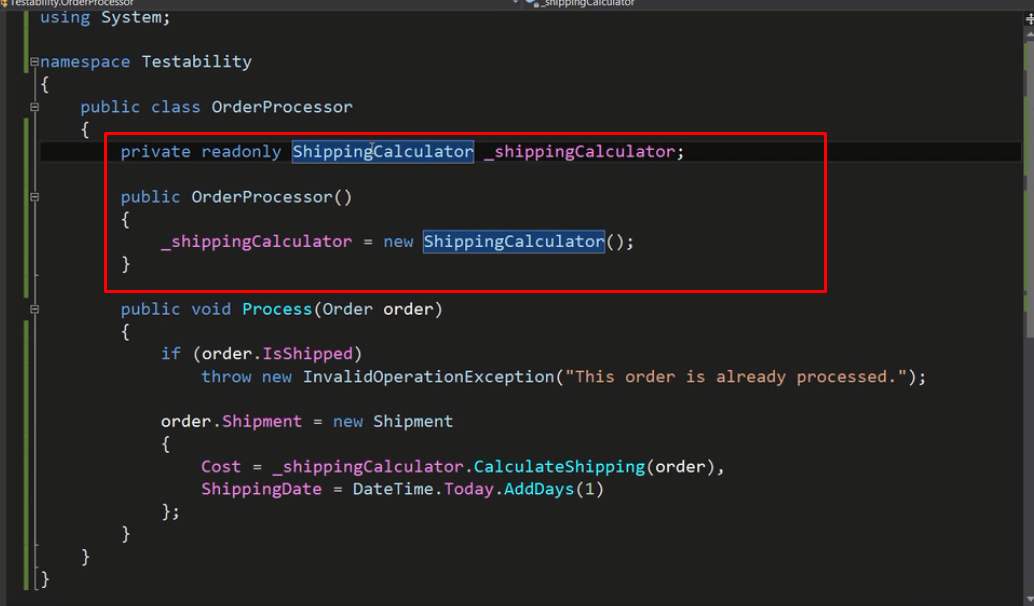
Khi chạy unit test phải isolate unit vs phần còn lại



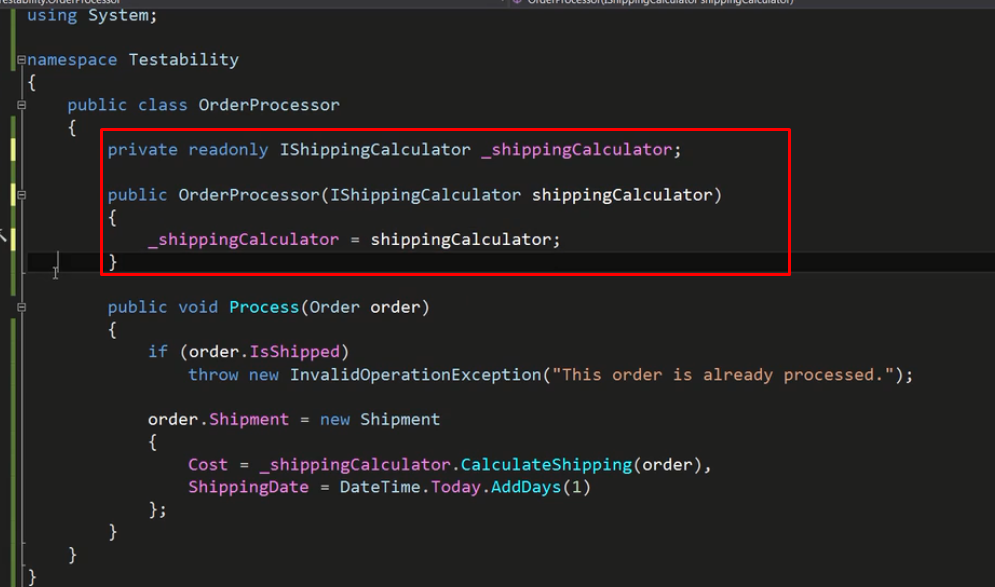
Tuy nhiên Process ở đây lại gọi \_shippingCalculator được tạo từ class ShippingCalculator ở ngoài vào trên contructor



Khai báo interface sau đó đưa vào class, đưa vào class thực hiện định nghĩa method trong interface



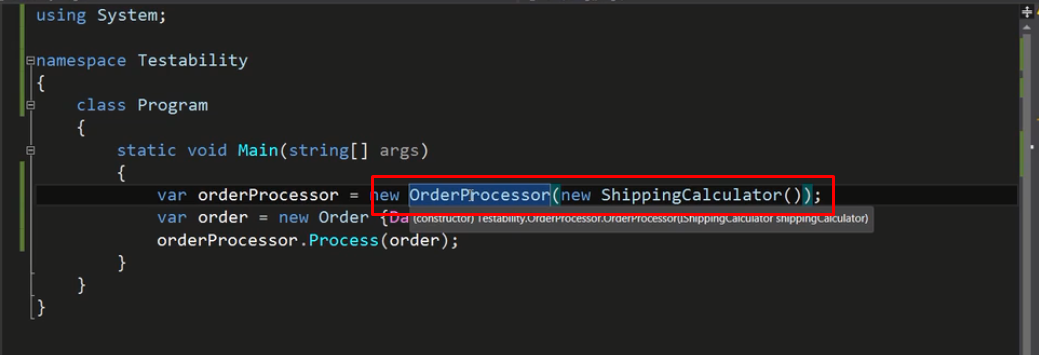
Before



After

Sử dụng method trong interface, như vậy ko liên quan đến class khác nữa -> thực hiện unit test

Như vậy class này không còn ref class ShippingCaculator nữa mà sử dụng Interface, khi đó thì nó có thể gọi bất kỳ class nào vào vị trí của ShippingCaculator khi thực hiện tạo obj

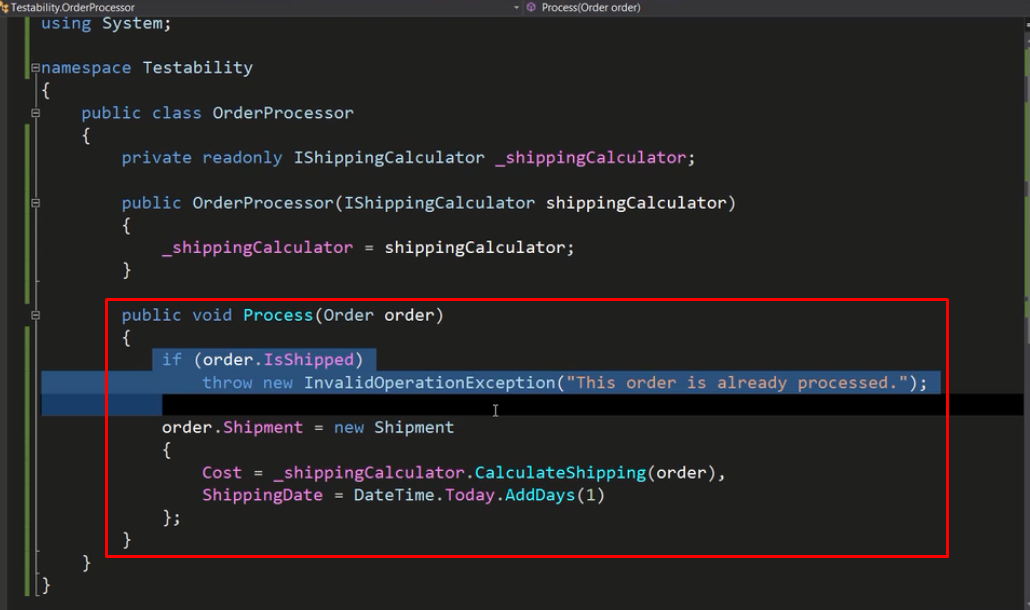


Tự hỏi là nếu như vậy thì nó sẽ liên quan đến class đưa vào

Nhưng đây là đưa vào khi chạy code bên ngoài class, còn bản thân class thì vẫn độc lập -> thuận tiện cho unit test

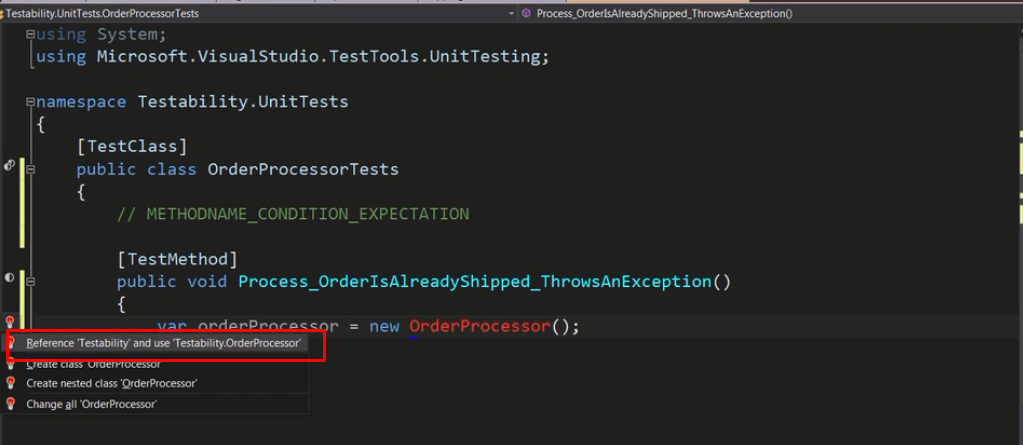
Kết luận:

Interface khởi tạo các thông số cho class hay funtion, dùng các thông số này, class không bị phụ thuộc vào class khác, mà chỉ phụ thuộc đầu vào khi tạo obj -> thuận tiện write unitest

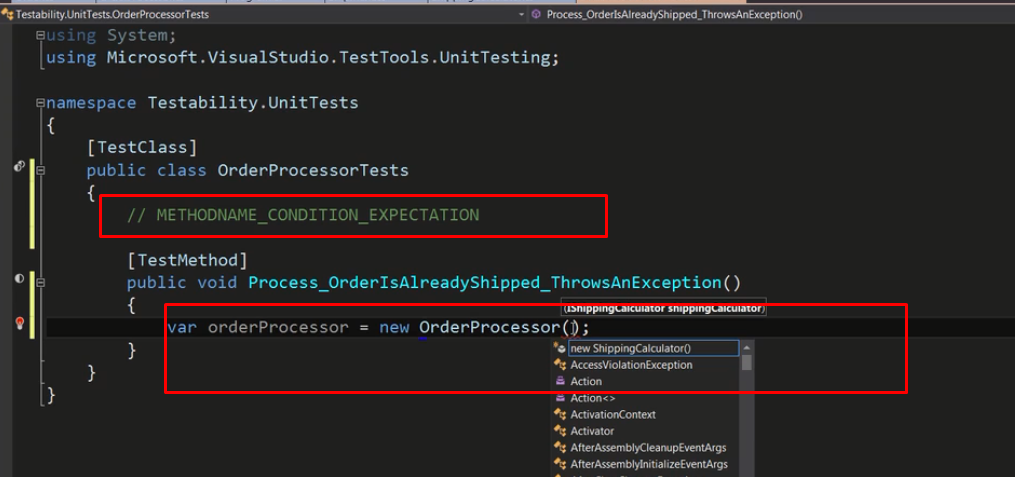


Viết thử unit test cho cái method củ kiệu ni

Viết unit test theo theo flow của code và từng flow theo điều kiện của các câu lệnh

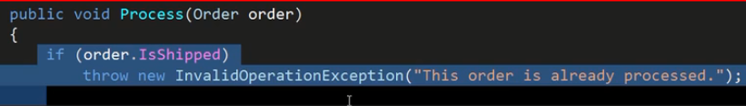


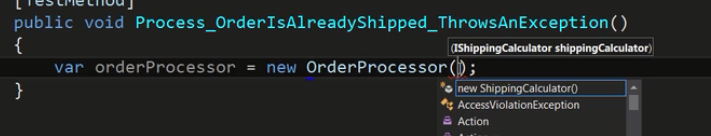
Khi tạo class từ class trong project chính



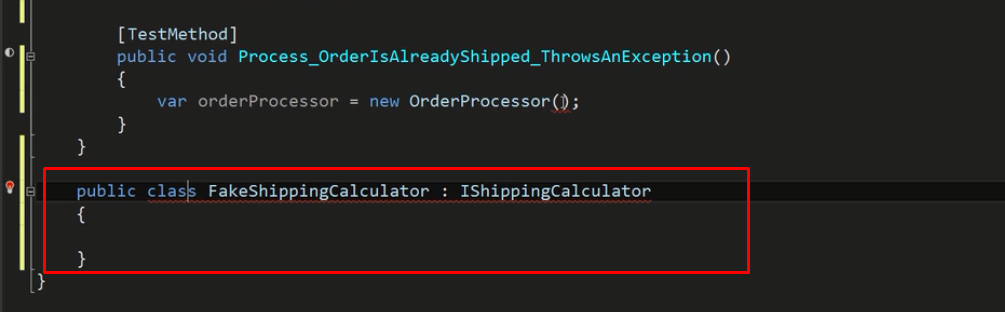
Đặt tên method test như trên

Dựa vào method cần test để tạo các method test đại diện cho testcase, dụ dụ trường hợp này đang test là test



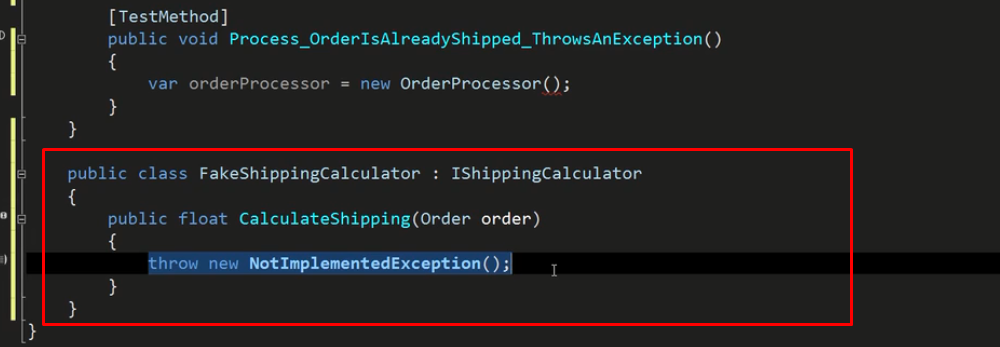


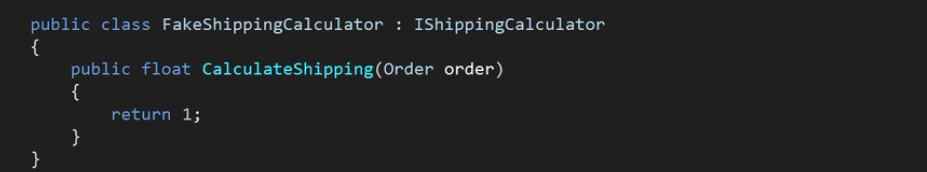
Tạo object trong trường hợp này thì dữ liệu đưa vào phải phù hợp với trường hợp, trong này dữ liệu phải cái order có IsShipped = true



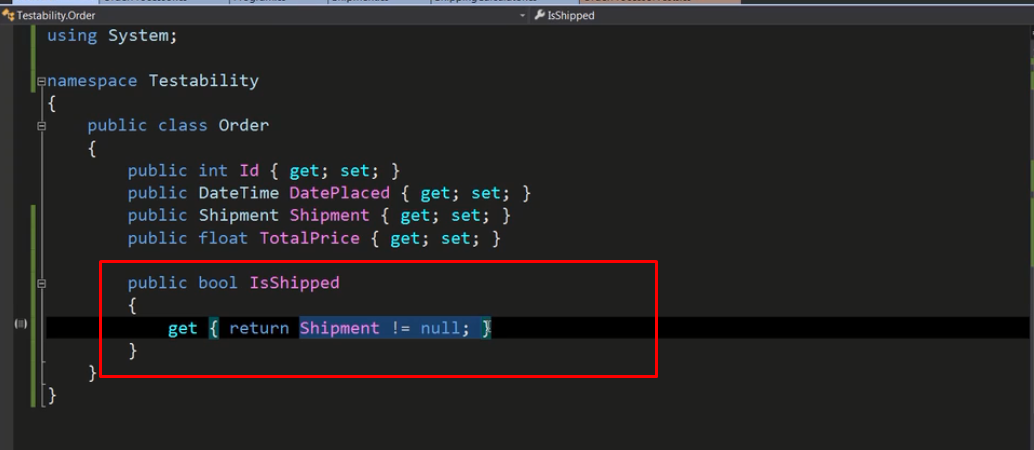
Tạo fake obj để đưa vào

ở đây màu đỏ vì thiết code, alt+enter để nó tạo ra

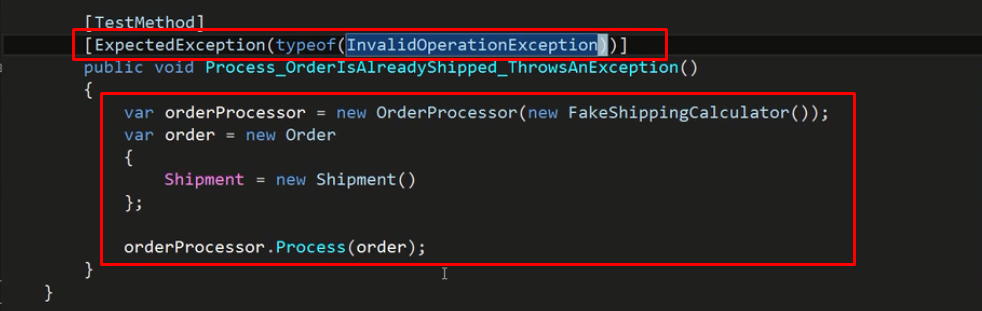




Sửa thành ntn



Order shipped là khi biến shipment return true



Cái thứ nhất là kết quả mong muốn của test

Ơ dưới

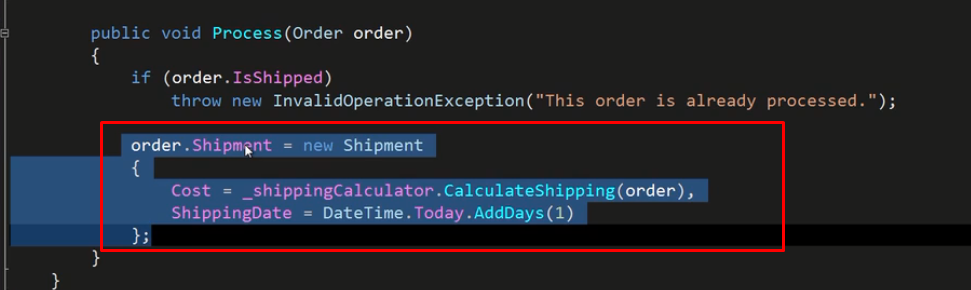
Tạo ra obj orderProcesser

Tạo order mới, với thông số shipment -> shipment not null -> isShipped = true : phù hợp với testcase

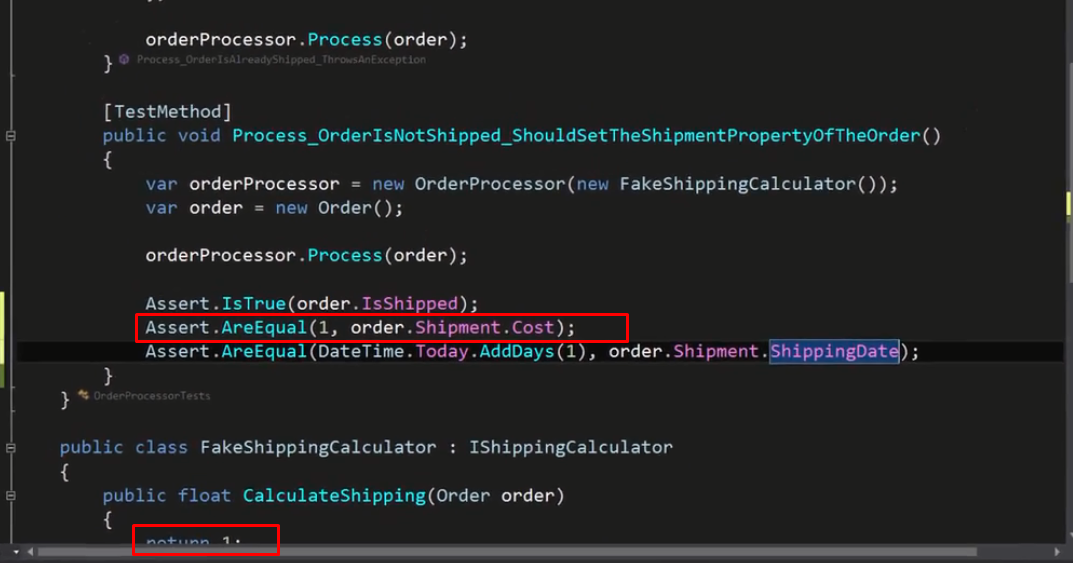
Đưa order vừa tạo vào orderProcessor.Process(method cần test)

Run

Trên vscode chạy dotnet test



Tiếp tục test case if trên false code nhảy xuống đây





Số 1 ở đây vì cái order được tạo ra ở dưới luôn return 1