Test doubles

Isn’t everything a mock?

Robson da Silva (sc.robson@gmail.com)

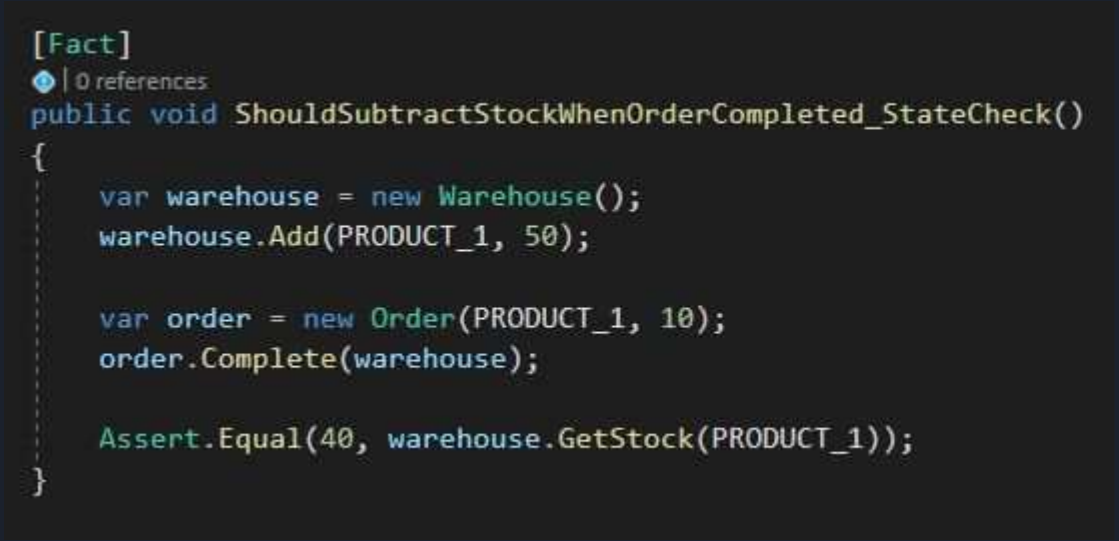
Types of double

- Dummy
    - Filler objects that are not really used in the tests
- Fake
    - Objects that implement a shortcut such as an in memory database
- Stub
    - Objects implemented with canned answers for the tests
- Mock
    - Objects implemented with expectations for the calls made to it
- Spy
    - Actually stubs but also records data about the calls made to it

Types of double (focus)

- Stub
  - Objects implemented with canned answers for the tests
  - Mostly when testing state
- Mock
  - Objects implemented with expectations for the calls to be made to it
  - Mostly when testing behavior

Asserting state

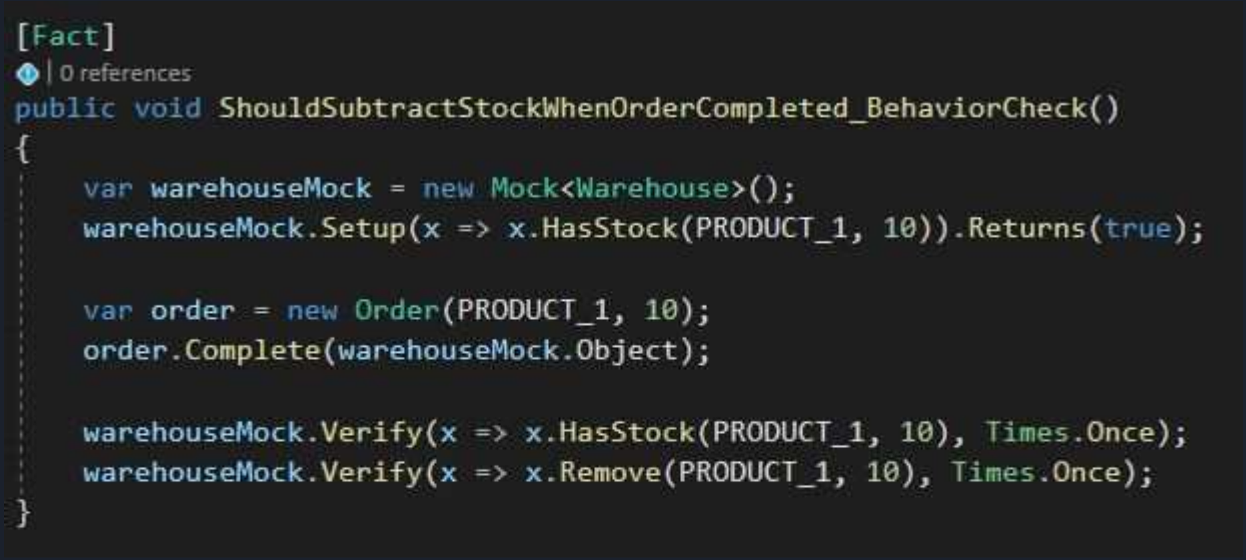


```
[Fact]
0 references
public void ShouldSubtractStockWhenOrderCompleted_StateCheck()
{
    var warehouse = new Warehouse();
    warehouse.Add(PRODUCT_1, 50);

    var order = new Order(PRODUCT_1, 10);
    order.Complete(warehouse);

    Assert.Equal(40, warehouse.GetStock(PRODUCT_1));
}
```

Asserting behavior



```
[Fact]
| 0 references
public void ShouldSubtractStockWhenOrderCompleted_BehaviorCheck()
{
    var warehouseMock = new Mock<Warehouse>();
    warehouseMock.Setup(x => x.HasStock(PRODUCT_1, 10)).Returns(true);

    var order = new Order(PRODUCT_1, 10);
    order.Complete(warehouseMock.Object);

    warehouseMock.Verify(x => x.HasStock(PRODUCT_1, 10), Times.Once);
    warehouseMock.Verify(x => x.Remove(PRODUCT_1, 10), Times.Once);
}
```

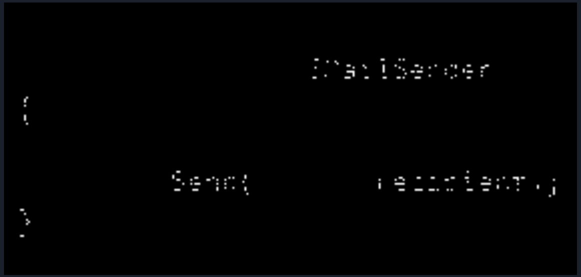
Classicist vs Mockist?

Let’s not go there but...

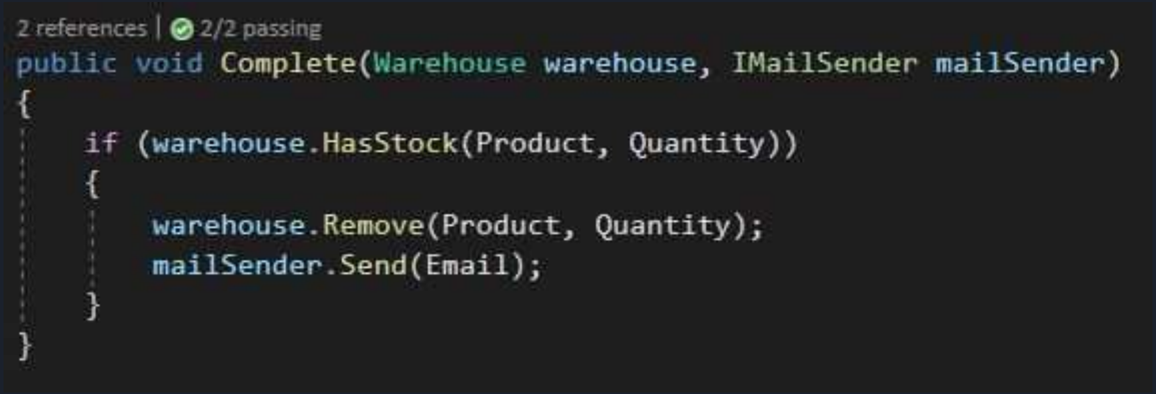
- Classicist
  - Will prefer to use the real objects when possible
  - new Warehouse()
- Mockist
  - Will prefer to mock all the dependencies out of the scope of the test
  - new Mock<Warehouse>()

Moving on

- Send an email after order is completed
  - How to test if it was called correctly when IMailSender doesn't hold state?

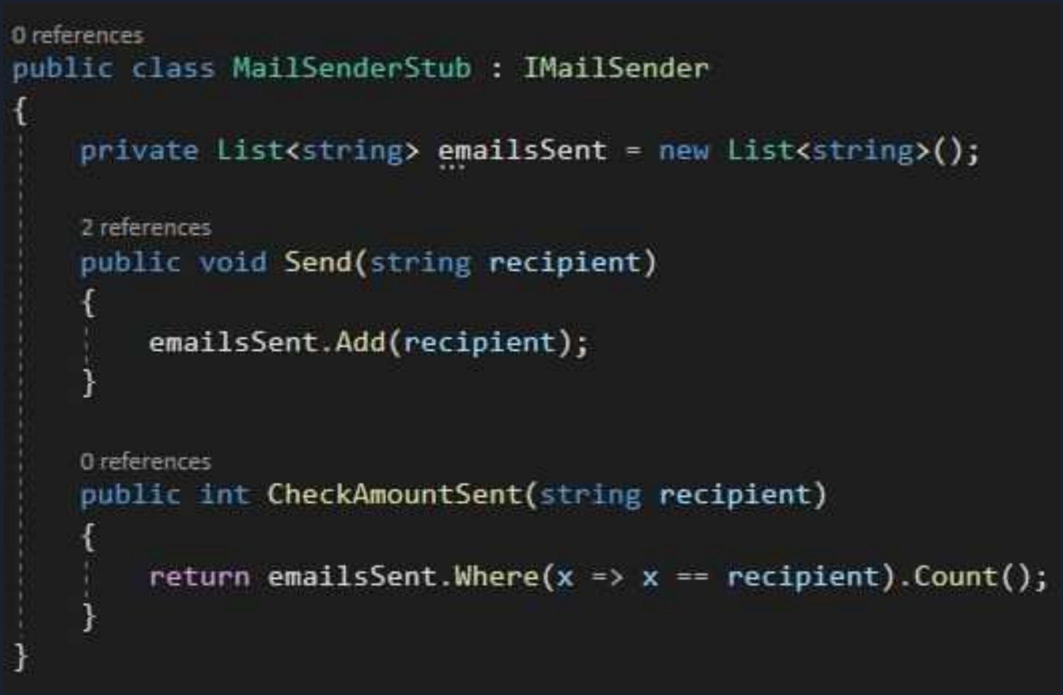


```
[illegible] IMailSender
{
    [illegible] Send( [illegible] recipient);
}
```



```
2 references | 2/2 passing
public void Complete(Warehouse warehouse, IMailSender mailSender)
{
    if (warehouse.HasStock(Product, Quantity))
    {
        warehouse.Remove(Product, Quantity);
        mailSender.Send(Email);
    }
}
```

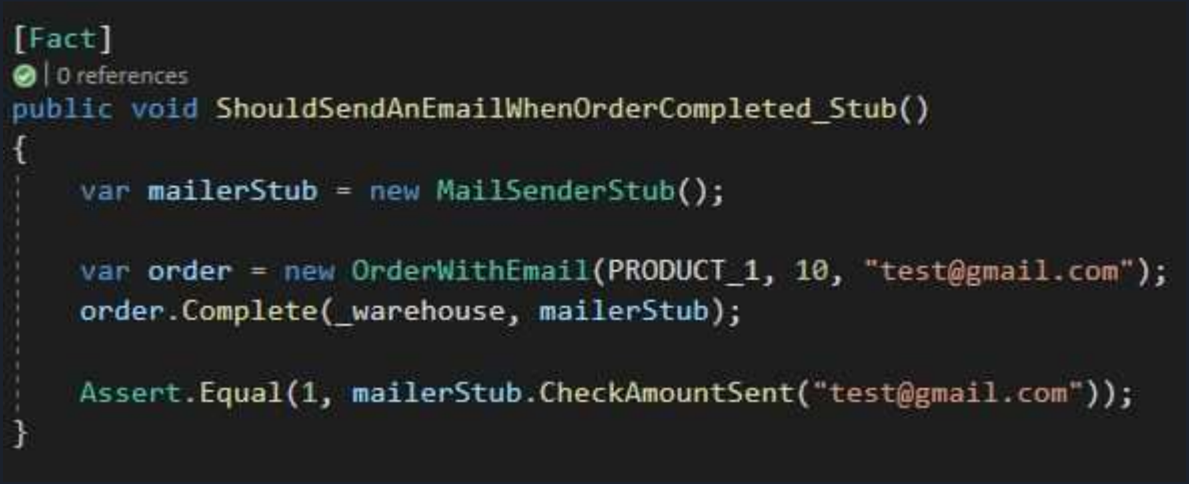
Stub it!


```
0 references
public class MailSenderStub : IMailSender
{
    private List<string> emailsSent = new List<string>();

    2 references
    public void Send(string recipient)
    {
        emailsSent.Add(recipient);
    }

    0 references
    public int CheckAmountSent(string recipient)
    {
        return emailsSent.Where(x => x == recipient).Count();
    }
}
```

Stub it!

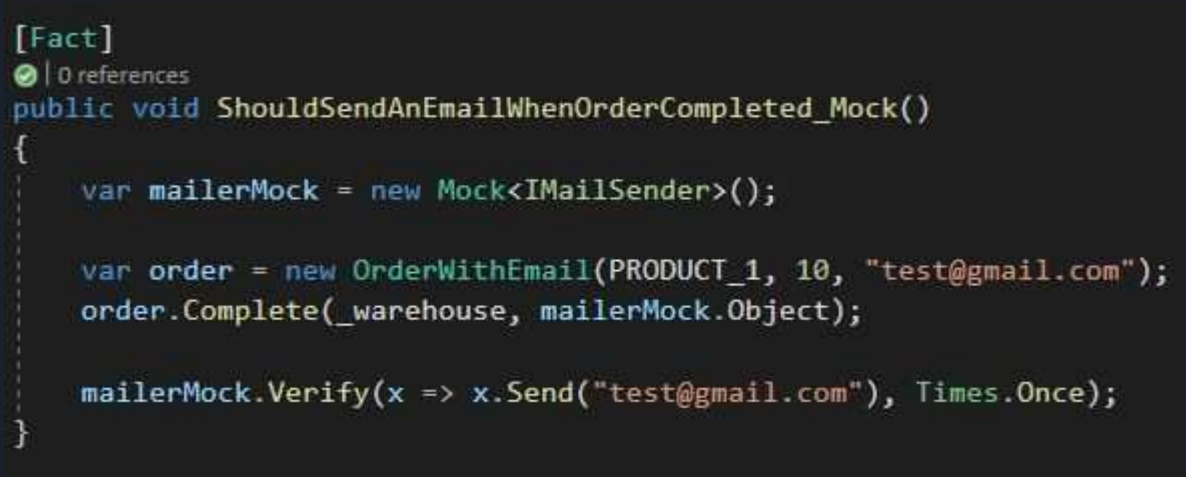


```
[Fact]
0 references
public void ShouldSendAnEmailWhenOrderCompleted_Stub()
{
    var mailerStub = new MailSenderStub();

    var order = new OrderWithEmail(PRODUCT_1, 10, "test@gmail.com");
    order.Complete(_warehouse, mailerStub);

    Assert.Equal(1, mailerStub.CheckAmountSent("test@gmail.com"));
}
```

Or mock it!



```
[Fact]
0 references
public void ShouldSendAnEmailWhenOrderCompleted_Mock()
{
    var mailerMock = new Mock<IMailSender>();

    var order = new OrderWithEmail(PRODUCT_1, 10, "test@gmail.com");
    order.Complete(_warehouse, mailerMock.Object);

    mailerMock.Verify(x => x.Send("test@gmail.com"), Times.Once);
}
```

Conclusions

- Both will have advantages
- There is no right or wrong
- It depends on what is being tested
  - State
  - Behavior
- It depends also on your approach
  - Classicist
  - Mockist

Thank you

Robson da Silva (sc.robson@gmail.com)

Source: https://martinfowler.com/articles/mocksArentStubs.html