Stubs and Mocks

Stub:

→ Used to create a class that returns data to a test

→ Data should be easily created and will always stay the same.

→ They don't show how a class interacts with the system.

→ Just provides data.

→ MOQ frameworks can create Stubs.

Stubs and Mocks

Mock:

→ When your class interacts with the system.

→ Created using MOQ frameworks.

→ Can provide the same functionality as the Stub
but Stubs are easier to create.

→ They could create the same functionality as the class under test.

Creating mocks and stubs with C#

Using a hand-crafted Stub – Using an interface

```
public interface IFileReader
{
    string ReadFirstLine();
}
```

```
public class StubFileReader : IFileReader {
    public string ReadFirstLine()  {
        return "10 Pen 60p";
    }
}
```

```
public class Order {
    public string ProcessOrder(IFileReader fileReader)  {
        string line= fileReader.ReadFirstLine();
        // process information
        // if(process OK) {
                email();
                return true;
           }
           return false;
    }
    public virtual void email() {
        // code to email a message
    }
}
```

```
[TestMethod]
public void TestOrder() {
    Order ord = new Order();
    Assert.IsTrue(ord.ProcessOrder(new StubFileReader());
}
```

Using Moq to mock objects

```
public interface IFileReader
{
    string ReadLine(int ln);
}
```

```
[TestMethod]
public void TestMethod1()
{
    var moqFileReader = new Mock<IFileReader>();

    moqFileReader.Setup(f => f.ReadLine(1)).Returns("10 Pen 60p");
    moqFileReader.Setup(f => f.ReadLine(2)).Returns("15 Ruler 120p");
    var oReader = new Order();
    bool res = oReader.ProcessOrder(moqFileReader.Object);
    Assert.IsTrue(res);
}
```

Using TestInitialize (Moq with MsTest)

```
[TestClass]
public class Temp {

    Order oReader;
    IFileReader fileReader;

    [TestInitialize]
    public void Initialize()
    {
        var moqFileReader = new Mock<IFileReader>();
        moqFileReader.Setup(f => f.ReadFirstLine()).Returns("10 Pen 60p");
        fileReader = moqFileReader.Object;
        oReader = new Order();
    }

    [TestMethod]
    public void TestMethod1()
    {
        bool res = oReader.ProcessOrder(fileReader);
        Assert.IsTrue(res);
    }
}
```

```
public interface IFileReader
{
    string ReadFirstLine();
}
```

Creating mocks and stubs with Java

Using Mockito with Java

- Mockito is a very useful app for java developers to mock external dependencies.
- To use **Mockito**, you need to set it up for you application.
- See here for a Maven project's POM file.

```
<dependencies>
        <dependency>
                <groupId>org.junit.jupiter</groupId>
                <artifactId>junit-Jupiter-api</artifactId>
                <version>5.9.1</version>
                <scope>test</scope>
        </dependency>
        <dependency>
                <groupId>org.mockito</groupId>
                <artifactId>mockito-core</artifactId>
                <version>4.6.1</version>
                <scope>test</scope>
        </dependency>
```

```
        <dependency>
                <groupId>org.mockito</groupId>
                <artifactId>mockito-junit jupiter</artifactId>
                <version>4.6.1</version>
                <scope>test</scope>
        </dependency>
</dependencies>
```

Using Mockito - setting expectations

- Create an interface for the actual system that your application depends on.
- Then ask Mockito to create a mock and setup the methods' expected values.
- You can create a Stub, but you will have to code it and maintain it in your project.

```
public class StubDatabase implements QADatabase {
  public String getUsernameByID(int id) {
    String[] names = {"Bob","Anna","Mike","David","Lily", "Fred", "Kimberly"};

    if (id < names.length)
       return names[id];
    else
       return null;
    }
}
```

So, how do we get Mockito to do all this for us? Let’s see…

Mocking an object using Mockito

```
public interface QADatabase {
    public String getUsernameByID(int id);
}
```

Object to mock

Class requiring the mock

```
public class QAController {
  private QADatabase qaDb;

  public QAController(QADatabase db) {
      this.qaDb = db;
  }
}
```

```
@ExtendWith(MockitoExtension.class)
public class FirstTest {

   @Mock
   QADatabase db;

   @InjectMocks
   QAController controller;

   @Before
   public void setUp() {
      Mockito.when(db.getUsernameByID(1)).thenReturn("Bob");
    }
}
```

SUMMARY

- In this chapter you learn how to create a Stub and how to create a mock

C# EXTRA MATERIAL

- Let’s explore using mock objects in more detail

MOQ –Another example

```
public interface ILogin {
          bool isValidUser(string uname, string pass, int userType);
}
```

```
public class Company {
    public bool registerEmployee(string uname, string password, ILogin login, int userType) {
        // ... some code
        if (!isValidUser(uname, password, login, userType))
            return false;
        //... Register the user
        return true;
    }

    public bool isValidUser(string uname, string password, int userType) {
        return login.isValidUser(uname, password, userType);
    }
}
```

```
[TestMethod]
public void StockInformationTestTDD_GetStockInfoIsCalled() {
    Mock<ILogin> mockLogin = new Mock<ILogin>();
    mockLogin.Setup(log => log.isValidUser("mike", "password123", 1)).Returns(true);
    mockLogin.Setup(log => log.isValidUser("Bob", "pass1", 1)).Returns(false);
    Company qa = new Company();

    Assert.IsTrue(qa.registerEmployee("mike", "password123", mockLogin.Object, 1));
}
```

MOQ – Parameter type of Any

```
public interface Ilogin {
    bool VerifyUser(string userName, string password, int userType);
}
```

```
Mock<ILogin> mockLogin;

[TestInitialize]
public void testInitialize() {
    mockLogin = new Mock<ILogin>();
    mockLogin.Setup(x => x.VerifyUser("Bob", "Password123", It.IsAny<int>())).Returns(true);
    mockLogin.Setup(x => x.VerifyUser("Steve", "sa", 2)).Returns(true);

    login.Setup(x => x.VerifyUser(It.IsAny<string>(), It.IsAny<string>(),
                                                      It.IsNotIn(1, 2))).
                                                      Throws<ArgumentException>();
}

[TestMethod]
public void TestRegisterEmployeeWithValidUserType() {
    var qa = new Company();
    Assert.IsTrue( qa.registerEmployee("mike", "password123", mockLogin.Object, 1) );
}
```

MOQ – Mock Exceptions

```
public interface Ilogin {
    bool Verify(string userName, string password, int userType);
}
```

```
[TestInitialize]
public void testInitialize() {
    mockLogin = new Mock<ILogin>();
    mockLogin.Setup(x => x.VerifyUser("Bob", "Password123", It.IsAny<int>())).Returns(true);
    mockLogin.Setup(x => x.VerifyUser("Steve", "sa", 2)).Returns(true);

    login.Setup(x => x.VerifyUser(It.IsAny<string>(), It.IsAny<string>(),
                                                      It.IsNotIn(1, 2))).
                                                      Throws<ArgumentException>();
}

[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void TestNullUserName()
{
    var qa = new Company();
    qa.registerEmployee("mike", "password123", mockLogin.Object, 99) );
}
```

MOQ – verify a method was called

```
public interface ILogWriter  {
    void Write(string message);
}
public class LogWriter : ILogWriter
{
    public void Write(string message)  {
        Console.WriteLine(message);
    }
}
```

```
public class MyProcessor
{
    public void Start(ILogWriter _writer)
    {
        _writer.Write("my message");
    }
}
```

```
[TestClass]
public class MyProcessorTest {
    private Mock<ILogWriter> _writer;

    [TestInitialize]
    public void SetUp() {
        _writer = new Mock<ILogWriter>();
        _writer.Setup(x => x.Write(It.IsAny<string>()));
    }
    [TestMethod]
    public void Succesfully_writeLog()
    {
        new MyProcessor().Start(_writer.Object);
        _writer.Verify(x => x.Write(It.IsAny<string>()), Times.Once());
    }
}
```

Real-life example of a Stub

The following code example shows a practical example of using a Stub.

There are cases when the data modules are not available or it is undesirable to use.

Repository example – The Interface

First define an interface

All data operation which the business layer wish to perform

```
public interface INorthwindRepository
{
    IEnumerable<Customer> GetCustomers();
    IEnumerable<Customer> GetCustomersByCity(string city);
    Customer GetCustomerByID(string id);
    void DeleteCustomer(Customer cus);
    void DeleteCustomerByID(string id);
    void AddCustomer(Customer cus);
}
```

You may then add real business methods in another layer which uses this interface

Repository Classes – Implement the interface

```
public class SQLNorthwindRepository : INorthwindRepository  {
    Northwind context = new Northwind();       // better in a constructor
    public void AddCustomer(Customer cus) {
        context.Customers.Add(cus);
        context.SaveChanges();
    }
    public void DeleteCustomer(Customer cus) {
        context.Customers.Remove(cus);
        context.SaveChanges();
    }
    public void DeleteCustomerByID(string id) {
        context.Customers.Remove(GetCustomerByID(id));
        context.SaveChanges();
    }
    public IEnumerable<Customer> GetCustomers() {
        return context.Customers;
    }
    public IEnumerable<Customer> GetCustomersByCity(string city) {
        return context.Customers.Where(c => c.City == city);
    }
    public Customer GetCustomerByID(string id) {
        return context.Customers.Single(c => c.CustomerID == id);
    }
}
```

Refactor

When many Controller's Actions need to use a repository consider:

→ Creating class level variable (based on the repository interface)

→ Instantiate the object in constructor chain

```
public class HomeController : Controller
{
    INorthwindRepository northwindRepository;
    public HomeController() : this(new SQLNorthwindRepository())  {
    }
    public HomeController(INorthwindRepository repository) {
        this.northwindRepository = repository;
    }
    public ActionResult Index()  {
        return View(northwindRepository.GetCustomers().ToList());
    }
}
```

Testing Your Controller – Building a Stub

By basing the repository on an interface, it's easy to implement a "stub" for testing:

```
public class StubNorthwindRepository : INorthwindRepository
{
        IEnumerable<Customer> customers;
        public StubNorthwindRepository ()  {
            customers = new List<Customer>() {
new Customer(){CompanyName="QA", ContactName="Mike", City="London", CustomerID="AAAAA" },
new Customer(){CompanyName="BA", ContactName="Dean", City="London", CustomerID="BBBBB" },
new Customer(){CompanyName="QA", ContactName="Steve", City="Leeds", CustomerID="CCCCC" },
new Customer(){CompanyName="QA", ContactName="Victor", City="London",CustomerID="DDDDD"}
        };
        }
        public IEnumerable<Customer> GetCustomers() {
            return customers;
        }
        public IEnumerable<Customer> GetCustomersByCity(string city)   {
            return customers.Where(c => c.City == city);
        }
   // other methods
}
```

Testing Your Controller – Using the Stub

Use the overloaded constructors to pass in your test repository and test the returned object

```
[TestClass]
public class HomeControllerTest     {
    [TestMethod]
    public void Index() {
        // Arrange
        HomeController controller = new HomeController( new StubNorthwindRepository());

        // Act
        ViewResult result = controller.Index() as ViewResult;
        IEnumerable<Customer> model = (IEnumerable<Customer>)result.Model;

        // Assert
        Assert.AreEqual(96, model.Count());
    }
}
```

Use the Stub

Cast

Check the Model