INFO 6067

Testing for Development Lab Week 8

## Description – Mocks, Stubs, Drivers

Instructions

You have been assigned to test object interactions in isolation. Mainly testing the receptionist class. You will be provided the interfaces for Objects that will be mocked.

The process is as follows.

1. Download the Lab files from FoL
   1. Import them into eclipse
2. Create a test class and test the Receptionist class
   1. The first test should consist of valid interaction from the receptionist -> mechanic and back to the receptionist.

Ans : As our main concern is to test the receptionist class, So we are going to use real object of the receptionist class and we will mock the mechanic class.

As we already know we need provide the mockery all the expectations that we are going to use on the mocked object. So all the communication will happen in the mocked mechanic object.

So the test is very simple actually. We are going to use the mocked object as it’s a real mechanic classes object and do the communication through it’s publicly provided methods. And for each method call on the mechanic class there will be an expectaion for that.

The source code has a lot of comment to explain itself how it works.

* 1. The second test should consist of valid interaction from the customer->receptionist->mechanic and back to the customer.

Ans : Same as before ans we will use the real classes object for the receptionist class, and the other two classes i.e. customer, mechanic classes will be mocked using it’s interface.

To do the communication we will use as same as previously that mechanic and customer are object of an real class and we wil communicate between them via their provided public methods and fill the expectations to provide support for all the method calls.

The communication will start from the customer. The customer will receive an issue and the return it to send to the receptionist. We will hold that in an String object and send it to the receptionist class. The receptionist class will accept the String and return another String to send to the mechanic. We will hold that in another String object and send it to the mechanic class so that the mechanic can start working on it.

Now for communicating back to the customer we will first ask the mechanic for what he has been finished working. Mechanic will return an String with the issue that fixed in it. We will hold that in as String object and send it back to the receptionist. The receptionist will process it and return back an String object holding information to send back to the customer. We will hold that in another String object and send it back to the customer.

Now to send and receive informatin i.e. String object to the mechanic and customer class we need set the proper expectations for the mocked object of the mechanic interface and the customer interface.

The source code has proper explanaitation on it with comment on almost every line.

* 1. The third test should consist of an invalid communication from the -receptionist->mechanic and back to the customer with an exception handled.

Ans : To do the miscommunication we need to mock the receptionist because the receptionist’s real object won’t make that happen so in the mock object’s expectation we will make the miscommunication. We will use the real object of the mechanic. So that it can detect he miscommunication and throw the exception when it happended.

We need to use some try catch for this test to handle the exception that will be thrown and handle it properly.

Mostly all the communication is trivial as same as the previous test just some modified expectations and some try catch to handle the exception for the miscommunication.

* 1. You need to perform an assertion test. Find it and assert the message.

Ans : To perform the assertion test we are going to take help assertThrows method that will help us accept the exception and if no exception thrown then it will immediately fail the test.

After the exception accepted by the excetion variable we can use assert to vaify the smae exception happend that we expedted

* 1. You need to mock an object to throw an exception. Find the object, mock the exception then assert True the exception message, then validate the communication

Ans: In this test we are going to use an mocked object of the customer object from the customer interface and throw an excetion through the expectation of the mocked object.

As same as the previous test we will use the assertThrows method to handle this exception that will be thrown fromthe customer mocked object.

After the excetion found we can verify the exception message using the assertTrue method with the the same exception message that we thrown from the mocked expectation.

1. All the arguments that you pass between object arguments should be of type string
2. You will need to submit this file and the TestTemplate.java file that’s provided and do not change any of the method names!!!!!!!
3. Take a screenshot of all your test passing/failing below this

Some additional file are included to give you an idea on how to do the exception portion

|  |  |  |
| --- | --- | --- |
| **Marks Available** | **What Are the Marks Awarded For?** | **Marks Awarded** |
| **If your code does not compile you will get ZERO (0).** | | |
| **1** | **2.a** |  |
| **3** | **2.b** |  |
| **5** | **2.c** |  |
| **3** | **2.d** |  |
| **4** | **2.e** |  |
| **2** | **Readable Code (Proper structure)** |  |
| **1** | **Screenshot** |  |
| **\_\_\_ 20** | **TOTAL MARKS** |  |