Project 2

Software Verification and Validation

Pedro Carrega – nº 49480

Exercise 4 – Mockito

Using the implemented SUT is possible to mock some of the business layer's modules, for example, it is possible to mock the CustomerFinder module to test the CustomerService class. The CustomerFinder module is suitable for mocking since it is not a final or anonymous class, being the principal limitations for mocking a class not being Final or Anonymous. The methods can also be stubbed since the single method it contains it public and non-static.

One possible implementation of testing the mocked module would be the following:

```
25    @Test
26    void mockCustomerFinder() throws PersistenceException, ApplicationException {
27
28        //Mocking the CustomerFinder
29        CustomerFinder mockedFinder = mock(CustomerFinder.class);
30
31        //Stubbing Methods
32        when(mockedFinder.getCustomerByVATNumber(197672337)).thenReturn(new CustomerRowDataGateway(1, 197672337, "Mock Customer", 123456789));
33        when(mockedFinder.getCustomerByVATNumber(anyInt())).thenThrow(new ApplicationException("Customer not found"));
34
35        //Test return values
36        assertEquals(CustomerService.INSTANCE.getCustomerByVat(197672337), new CustomerDTO(1, 197672337, "Mock Customer", 123456789));
37        assertThrows(ApplicationException.class, () -> {CustomerService.INSTANCE.getCustomerByVat(168027852);});
38
39        //Verify called times
40        verify(mockedFinder, times(1)).getCustomerByVATNumber(197672337);
41        verify(mockedFinder, times(1)).getCustomerByVATNumber(168027852);
42
43    }
```

Initializing the mocked class, stubbing the methods of the class so we can control the return values, testing the expected return values of CustomerService and finally verify that the stubbed methods were called the excepted number of times.

Found Bugs and Modifications

1. One of the bugs found was the non-removal of entries in the sales table when removing a customer from the application. To fix this it was implemented a method in the SaleRowDataGateway which given a client vat it removes every entry from the sales table associated to said vat number. This method is called during the removeClient method provided by the CustomerService class.

2. The program also didn't delete the sale deliveries when removing a client. The solution to this bug is exactly the same as the previous one but applied to the saleDelivery table.
3. The application also allowed to insert the same address twice to the same customer. To fix this bug it was created a verification if the given address was already associated with the given client. This verification was implemented in the AddressRowDataGateway and is executed when adding an address.
4. When inserting a new sale delivery no verification was made to see if the sale was already closed. During the execution of the service it's acquired the sale entry associated to the given sale_id and then checked the status of the sale.
5. The text present on the submit button on addCustomer.html to be more intuitive.
6. When adding a new sale the only verification made was if the vat number was valid, the service was changed so that is also verified if the vat number is associated to a customer present in the database.
7. When removing a customer the addresses associated to said customer were not removed. Like on point 2 and 3, it was implemented a method so that, given a vat number, the addresses associated to that vat number were removed.
8. The constructors of RowDataGateway whose parameter were a ResultSet failed to fill all attributes, that bug was fixed in the fillAttributes method present in each RowDataGateway class.
9. Classes that featured an empty constructor had said constructor removed and some public methods were given the "static" attribute. This was implemented due to the fact that the original implementation did not follow good programing practices.