CSSE376 – Lab4 – Caio Balthazar – CM1478

1. [Question 1 (4 pts)] Explain, in your own words, the process used to mock the database and test the getRoomOccupant function.

The first step was to create the stub object using the Rhino framework. Then, the expected results were set in the respective string objects. After that, we instructed (through recording) the stub with the proper behavior in order to emulate the database communication. Then we created the Hotel target and set the stub as the database property. Lastly, we made the calls and asserted the results.

2. [Question 2 (4 pts)] Specify how you can use the LastCall class to throw an exception.

Through the method LastCall.Throw(Exception x) after the troublesome call ;

3. [Question 3 (4 pts)] In the above example, we used a \stub" since the mocked object returned a value. Do I need to use a stub if the mocked object did not return a value? Can I replace the stub with a DynamicMock?

You don’t need to use a stub if there are no returning values. You could use a Strict or Dynamic mock. In order to replace the stub with a Dynamic Mock, you would need to instantiate one and would need to tell the mock to expect some behavior. Then use the verify method to force the mock into checking if the expected behavior really occurred.

1. [Question 4 (4 pts)] Explain, in your own words, the process used to mock the database and test the AvailableRooms property.

After the instantiation of the stub, we created an object with the expected values for that property (An int32 list). Then we assign that value for the property in the mock trough straight usage of the assignment symbol. Lastly, we create the target object and run the desired tests.

1. [Question 5 (4 pts)] Explain, in your own words, the process used to ensure that the service locator removes a car from its available cars when a car is booked.

The first step was to set up the service locator. We created a ServiceLocator instance and added some cars to it. Then, we set the global instance inside the object class using reflection. After this, the service locator must have couple of cars available for centralized booking. Continuing, we created a user and tried to book a car. Finally, we checked if the list of cars available had the expected lower size and that the remaining car was the correct one.