David McGinnis
CM 2351

CSSE 376 Lab #4

1. We create a mock Database class stub called mockDatabase, which then inherits all of the functions of the IDatabase interface which we are mocking. This stub then is told if it receives a 24 as an argument to getRoomOccupant, it should return “Whale Rider”, while if it receives 1025, it should return “Raptor Wrangler”. We then test that this is working by calling the code on those two functions, and asserting the result is what it should be.
2. You would call LastCall.Throw, with the single argument being the exception which should be thrown for the call that is made.
3. If the mocked object did not return a value, you could just use a DynamicMock, since it would need to assert inside of the method, instead of returning some value.
4. We create a mock Database class stub called mockDatabase, which then inherits all of the functions of the IDatabase interface which we are mocking. This stub then is told that its Rooms property is a list of integers, which we have defined. This mockDatabase instance is added to a new Hotel instance. We then test that this is working by calling the availableRooms property, to get the number of rooms which are in the database, which in this case is our mockDatabase instance. We then make sure that this number agrees with the number of elements in the list which we attached to the mockDatabase instance.
5. We are only inserting two cars into the queue for serviceLocator, so if we remove one to be rented, the size of the list of available cars should be 1 (the first assertion), and the one in the list should be the one not rented by the user (the second assertion).