

Lab 4 – Tyler Hannan and Mark Vitale

1. We made a stub database and using the Record method within mocks we assigned specific behaviors to calling `getRoomOccupant` with the inputs 24 and 1025. We set the targets database to our mock database. We then called `getRoomOccupant` with those two values and checked that the proper result was received.
2. If you use `LastCall.Throw(Exception exception)` the database would have thrown an exception on being called.
3. If you have no need of returning anything or throwing an error, there is no need for a stub. You could replace it with a `DynamicMock` in this case.
4. We create the stub database again and defined a list of rooms. We then assigned that list of rooms to our `mockDatabase.Rooms` (as an expectation), and when hotel calls on its `database.Rooms`, the `Rooms` list assigned in the test is passed as the database's `Rooms` field.
5. 2 Cars and a service locator are made, the cars are added to the service locator. It is then set to be a global instance of the `ServiceLocator`. We book one of the cars, and the user class calls on the global instance of the `ServiceLocator`, modifying it. Then in the test case, as the global instance has been modified, the changes are checked and found to be true.