Lab 4

Dan Schepers

1) First, create a mock database. Then use mocks.Record to declare the behavior of that database. Then assign the database to the database to the target object. Because you told the database what message to send back when it is called with certain parameters, you can test this expected value against what value the method that uses the database returns.
2) Use LastCall.throw(exception)
3) No, you do not need to use a stub if the mocked value doesn't return a value. Instead, you can use a DynamicMock, which will create the stubs for you.
4) Create mock database. Create a list of rooms, then assign that list to the mocked database. Create a target hotel, then assign the mocked database to that hotel. Now whenever the AvailableRooms property of the hotel is called, the mock database accesses the list of rooms.
5) Create two cars and a service locator. Add the cars to the service locator, then use reflection to set the value of Instance in ServiceLocator to your service locator. After that, book one of the cars you created. Check to see if there is only one remaining car to be booked, and make sure that the remaining car is the same object as the one left in ServiceLocator.Instance.AvailableCars