Eric Henderson
CM 1158

1) The database is being mocked by a dynamic mock class that will return a certain value when it gets a certain input. It will return the value of anotherRoomOccupant when getRoomOccupant gets called with a value of 1025, and will return the value of roomOccupant when getRoomOccupant gets called with a value of 24. The activity is being recorded by the mock engine, and the database is not actually being contacted.

2) LastCall.Throw(new WhateverException("Whatever message"));

3) A stub is not needed if it does not return a value or do anything else that would require LastCall; yes

4) We set the mockDatabase.Rooms property to a list we created in the test case, and then that value is stored in the mockDatabase object. The IDatabase.Rooms property is called by Hotel when we request Hotel.AvailableRooms, and since we set the Hotel.Database property to mockDatabase, it returns the length of the list returned by the mockDatabase object. We then test to see if the value returned is the same as the length of the list we gave to the mockDatabase object.

5) A ServiceLocator object is created, then the cars are added to it, then we use reflection to set the static _instance field to refer to the object we created. After that, we continue with our regular testing techniques: creating the test User, booking the carToBook, and testing to see if the User.book method worked correctly.