Ethan Veatch
CM 1411
March 16, 2012

Lab 4 – Advanced Unit Testing

1. In order to test the getRoomOccupant function, a mock was created based on the interface that a real database will follow. From there, values were set and expectations were established. The test then calls the getRoomOccupant function on the mock database and, assuming everything works as planned, the mock returns the values that were previously set.
2. LastCall can throw an exception using the Throw method. As an example, LastCall.Throw(new Exception(myException)) would throw myException when it is called.
3. If the mock object does not return a value there is no need to use a stub. A DynamicMock would suffice in this situation.
4. In order to test the AvailableRooms function, a mock database is established by creating a stub based on the interface a real database would follow. A list of rooms was then created and the Rooms value for the stub was set to always be equal to this list. From there a new Hotel was created and linked to the database stub. It was then asserted that number of rooms returned by AvailableRooms was equal to the number of rooms in our created list.
5. When a User creates a Booking, there is a check to see if it is a Car. If this check returns true, the User class tells the current instance of the ServiceLocator to remove the Car.