Trevor Krenz
Lab 4
3/19/12

1. First, the mock repository is created in the setup. Then, using the mock repository, a stub for a mockDatabase is created to act like a IDatabase object. Using the Record method of the mock repository, a list of expected calls and return values is made. Then, the mock object is tested by the mock repository checking if the correct calls and return values to the database is made. If the correct call and return values are done, the mock passes, otherwise it fails.
2. Call LastCall.throw(Exception exception).
3. If no return value is needed, the mocked object need not be a stub. A DynamicMock object can be used instead.
4. First, a stub is created of the type of the database. A list of rooms are created and fed to the mock database. A Hotel object is created, and has the mock database as its database. Then, the available rooms are calculated through the hotel object. If the number of rooms is equal to the rooms in the mock database, the test passes, otherwise it fails.
5. A ServiceLocator object is created. Then, two cars are added to the new ServiceLocator object. Then, the private, static variable "_instance" is set to the new ServiceLocator object. A car is then booked using a user. Then, the ServiceLocator object is tested to see if it has only 1 remaining car left, and the remaining car points to the 2nd car given. If these hold, then the test passes.