Kyle Spiegel

1. The database is mocked by creating an imaginary database for the test to communicate with which stores the assigned parameters. It then simulates a call to a database by creating the return value and confirming the room association with Hotel.cs.
2. Put a throw exception statement into the LastCall call.
3. If there was no return value a stub would not have been necessary and could have been replaced with a DynamicMock. However, since there was a returned value, the stub was necessary.
4. An imaginary database is created again this time "populating" imaginary rooms. It then sets the number of rooms available and confirms the room count with Hotel.cs.
5. It creates two entries on the database one for the car and a tracker for available cars. When the car is created and booked it is removed from the available cars entry