1) Make a mock database and then set the value of getRoomOccupant in the mock database to something expected, this is the record part. The next step is to make an actual call to getRoomOccupant and then compare that value with the expected one in the mock database created earlier, “replaying” this preset result.
2) By making a call to LastCall.throw if a certain value that shouldn’t be allowed is encountered.
3) A stub is only used when a valued is returned and that value is needed for something, so yes, a dynamic mock would be a good idea here.
4) Make a mock database, create a list of 100 rooms and make this the number of rooms for the mock database, then create a new Hotel object (not a mock one) and check to see that its AvailableRooms equal what we had in the mock database.
5) By using reflection with the ServiceLocator class (associating its current instance with the serivceLocator we created) and checking whether the size of its AvailableCars list is correct and that the first element in the list is the expected remaining car.