1. We setup a mock database using the IDatabase interface and the mocks.Stub call, because the object we need to mock returns a value. We made our own occupants and used the “using” method to do a record. We told it to return the occupants based on the getRoomOccupant function where we passed in an integer of the room the person occupies. We created a new Hotel object and made the Database of it to be our mockDatabase. Then we got the results of getRoomOccupant() with the room numbers as the parameter and asserted if the result and the name of the occupant were equal.
2. We can use the LastCall.Throw(Exception exception) method to throw an exception.
3. No. Yes.
4. We set up the mock database as a stub and used a list of numbers to represent the rooms in the hotel. Then we make a new hotel object and use its AvailableRooms getter to get the room count. Finally we check this number against the number of items in the list.
5. We created two cars and added them to a ServiceLocator object. Then we used reflection to set the value of the global ServiceLocator.Instance. Next we created a new User and booked one of the cars for the user. Finally we checked to see if there was indeed one less available car remaining and that the remaining car matches what it should be.