Chris Hoorn

CM 594

Lab 4 Questions

1. A stub of the Database is created using the mocks .Stub method with the IDatabase interface. The fake database is given two possible inputs and two outputs mapped to those inputs. If 24 is passed to the mockDatabase.getRoomOccupant method, then it will return “Whale Rider”. If 1025 is passed, it returns “Raptor Wrangler”. Next we create a Hotel object and assign the fake database to the Hotel.Database. We then call the Hotel.getRoomOccupant method with 1025 and 24 and assert that the outputs of both are as expected.
2. To throw an exception with LastCall, we call the LastCall.Throw(<Some Exception>) method.
3. If the mocked object did not return a value, we can use a DynamicMock instead of a stub.
4. We first create a list of rooms. We then assign the list of rooms to a mocked Database. Then we assign the mocked database to a new Hotel object. When the AvailableRooms method is called from the Hotel class, it will use the mocked database to check the number of rooms, which is determined by the list we created. If the returned value from the method call equals the size of the list we originally created, then the property functions as expected.
5. We create a serviceLocator object and add 2 cars to it, then makes an instance of it. We then create a new User and have it book a car. This invokes the User.book() method given a car parameter. This should call ServiceLocator.Instance.RemoveCar(), which remove CarToBook from the Instance of the ServiceLocator. We then use Assert to test that there is only one Car in the Instance and that the car is remainingCar.