Here is the overview of my design.

I have used abstraction, encapsulation, inheritance and polymorphism. Also, applied factory pattern and SOLID principles.

I have considered car types like Compact, Standard, Intermediate and Full Size which I derived from an abstract class Vehicle. Since price calculation is different for each car types I have created an abstract method on the base class and implemented the method on each child classes. Reservation attribute in Vehicle class will hold multiple non-overlapping time frames. Also added few common attributes of a car like model, passenger capacity and number of doors. As all cars have a unique identification, I have added an ID attribute.

I have created a repository project to mock data. The DBContext class will return all Vehicle objects and also takes care of modifying the objects.

ServiceController class will take care of the search and reservation of cars based on the given type, date and time and number of days. Factory class will take care of all instantiation of different objects. In addition to that I have extended ServiceController to a WCF web service RentalService.svc.

For unit testing I have used Microsoft Unit Test Framework which is available in Visual Studio. To test the core reservation workflow I have created an ordered test file ReservationWorkflow.orderedtest.

Following scenarios are covered in the unit test

Search Compact from 1/30 to 2 days - Positive test case

Search Compact from 1/31 to 1 days - Positive test case

Search Compact from 1/30 to 3 days - Positive test case

Search Compact from 1/28 to 1 days - Negative test case

Search Compact from 2/3 to 3 days - Outside boundary test case

Search Standard from 2/1 to 4 days - Positive test case

Search Standard from 2/2 to 1 days - Positive test case

Search Standard from 2/3 to 1 days - Positive test case

Search Standard from 2/2 to 1 days - Inside boundary test case

Please scroll down to see unit test results.

