Прокат автомобилей.

Система **Прокат автомобилей**. **Клиент** выбирает **Автомобиль** из списка доступных. Заполняет форму **Заказа**, указывая данные, срок аренды. **Администратор** может отклонить **Заявку**. При подтверждении **Заявки**, **Клиент** оплачивает **Заказ**. В случае повреждения **Автомобиля Клиентом**, **Администратор** вносит соот- ветствующие пометки.

Требования:

1) Необходимо разобраться в предметной области и соответствующими данными, определиться с родом таблиц и связей между ними, провести соответствующую нормализацию данных (минимум все таблицы должны соответствовать Третьей Нормальной Форме, 3НФ).
2) Необходимо спроектировать и разработать ER-диаграмму базы данных на основании выбранной предметной области и физически её реализовать с использованием СУБД MySQL.
3) Необходимо спроектировать и разработать иерархию классов-сущностей доменной области (бизнес объектов) и объектов DTO (*Data Transfer Object* – объекты, которые нужны для передачи данных от одного слоя другому), а также специальных вспомогательных (утилитных) интерфейсов и классов DAO (*Data Access Object* – объекты для доступа к базе данных и управления транзакциями), которые необходимы для реализации общего слоя будущего Web-приложения для доступа к данным (*Data Access Layer, DAL*).
4) При проектировании системы необходимо использовать принципы ООП-методологии, шаблон проектирования MVC, а также фундаментальные принципы ***GRASP*** и ***SOLID***. Основные классы системы должны быть самодостаточными, т.е. не зависеть, к примеру, от консоли! Любые типы отношений между классами (агрегация, композиция, ассоциация, наследование и т.д.) должны применяться лишь тогда, когда это имеет смысл.
5) Каждый класс, который моделирует отдельную сущность предметной области, должен иметь адекватное осмысленное имя и информативный состав (***get***- и ***set***-методы для доступа к состоянию объекта, корректно переопределённые методы базового класса *Object*: ***toString()***, ***equals()***, ***hashCode()*** и др.), default-конструктор, конструктор с параметрами и т.д.
6) Создаваемые классы (или другие пользовательские типы данных) необходимо грамотно разложить по соответствующим пакетам, которые должны иметь «адекватные» названия и быть вложены в указанные пакеты: ***by.kursy.surnameOfStudent*** *(к примеру,* классы-сущности в одном пакете, классы бизнес логики – в другом, утилитные/вспомогательные классы – в третьем и т.д.*)*.

7) Все конфигурационные параметры программы для подключения к базе данных и выполнения соответствующих транзакций необходимо разместить в отдельном файле-свойств.
8) SQL-команды должны быть также размещены в соответствующем файле-свойств.
9) Каждая программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом.
10) При разработке программ придерживайтесь соглашений по написанию кода на JAVA (Java Code-Convention).