Лабораторная работа №2
«Наследование в Java»

Задание:

С использованием механизма наследования в Java, реализовать слой доступа к данным.

1. На основе сущностей предметной области создать классы их описывающие.
2. Информацию о предметной области хранить в БД (или любом другом хранилище: XML-файлы, текстовые, бинарные файлы, веб-сервисы и т.д. Тип хранилища выбирается студентом самостоятельно). Для доступа в случае с БД использовать API JDBC с использованием пула соединений, стандартного или разработанного самостоятельно. В качестве СУБД рекомендуется MySQL или MS Sql server.
3. Доступ к данным реализовать с использованием шаблонов Factory Method или Abstract Factory.

Требования к оформлению кода:

1. Оформление кода должно соответствовать Java Code Convention.
2. Код должен содержать комментарии.
3. Классы и методы должны иметь отражающую их функциональность названия и должны быть грамотно структурированы по пакетам.

Варианты предметных областей

1. Система **Факультатив**. Существует перечень **Курсов,** за каждым из которых закреплен один **Преподаватель. Студент** записывается на один или несколько **Курсов**. По окончании обучения **Преподаватель** выставляет **Студенту** и добавляет отзыв.
2. Система **Платежи. Клиент** имеет одну или несколько **Кредитных Карт,** каждая из которых соответствует некоторому **Счету** в системе платежей. **Клиент** может при помощи **Счета** сделать **Платеж**, заблокировать **Счет** и пополнить **Счет. Администратор** снимает блокировку.
3. Система **Прокат автомобилей. Клиент** выбирает **Автомобиль** из списка доступных. Заполняет форму **Заказа**, указывая паспортные данные, срок аренды. **Клиент** оплачивает **Заказ. Администратор** регистрирует возврат автомобиля. В случае повреждения **Автомобиля, Администратор** вносит информацию и выставляет счет за ремонт. **Администратор** может отклонить **Заявку**, указав причины отказа.

4. Система **Библиотека**. **Читатель** имеет возможность осуществлять поиск и заказ **Книг** в **Каталоге**. **Библиотекарь** выдает **Читателю Книгу** на абонемент или в читальный зал. **Книга** может присутствовать в **Библиотеке** в одном или нескольких экземплярах.
5. Система **Больница**. **Врач** определяет диагноз, делает назначение **Пациенту** (процедуры, лекарства, операции). Назначение может выполнить **Медсестра (**процедуры, лекарства**)** или **Врач** (любое назначение). **Пациент** может быть выписан из **Больницы**, при этом фиксируется окончательный диагноз.
6. Система **Турагентство**. **Заказчик** выбирает и оплачивает **Тур (отдых, экскурсия, шоппинг)**. **Турагент** определяет тур как «горящий», размеры скидок постоянным клиентам.
7. Система **Приемная комиссия**. **Абитуриент** регистрируется на один из **Факультетов** с фиксированным планом набора, вводит баллы по соответствующим **Предметам и аттестату**. Результаты **Администратором** регистрируются в **Ведомости**. Система подсчитывает сумму баллов и определяет **Абитуриентов**, зачисленных в учебное заведение.
8. Система **Интернет-магазин**. **Администратор** осуществляет ведение каталога **Товаров**. **Клиент** делает и оплачивает **Заказ** на **Товары**. **Администратор** может занести неплательщиков в “черный список”.
9. Система **Авиакомпания**. **Авиакомпания** имеет список рейсов. **Диспетчер** формирует летную **Бригаду** (пилоты, штурман, радист, стюардессы) на **Рейс**. **Администратор** управляет списком рейсов.
10. Система **Тестирование**. **Тьютор** создает **Тест** из нескольких **Вопросов** закрытого типа (выбор одного или более вариантов из N предложенных) по определенному **Предмету**. **Студент** просматривает список доступных **Тестов,** отвечает на **Вопросы**.
11. Система **Периодические издания**. **Администратор** осуществляет ведение каталога периодических **Изданий**. **Читатель** может оформить **Подписку**, предварительно выбрав периодические **Издания** из списка. Система подсчитывает сумму для оплаты и регистрирует **Платеж.**
12. Система **Заказ гостиницы**. **Клиент** заполняет **Заявку**, указывая количество мест в номере, класс апартаментов и время пребывания. **Администратор** просматривает поступившую **Заявку**, выделяет наиболее подходящий из доступных **Номеров**, после чего система выставляет **Счет Клиенту**.
13. Система **Жилищно-коммунальные услуги**. **Квартиросъемщик** отправляет **Заявку**, в которой указывает род работ, масштаб, и желаемое время выполнения. **Диспетчер** формирует соответствующую **Бригаду** и регистрирует её в **Плане работ**.
14. Система **Парк**. **Владелец** парка дает указания **Леснику** о высадке (лечении, художественной обработке, уничтожении) **Растений**. **Лесник** отчитывается о выполнении. **Владелец** просматривает результаты и подтверждает исполнение.

15. Система **Ресторан**. **Клиент** осуществляет заказ из **Меню**. **Администратор** подтверждает **Заказ** и отправляет его на кухню для исполнения. **Администратор** выставляет **Счет**. **Клиент** производит его оплату.
16. Система **Автобаза**. **Диспетчер** распределяет **Заявки** на **Рейсы** между **Водителями,** за каждым из которых закреплен свой **Автомобиль**. На **Рейс** может быть назначен **Автомобиль**, находящийся в исправном состоянии и характеристики которого соответствуют **Заявке. Водитель** делает отметку о выполнении **Рейса** и состоянии **Автомобиля**.
17. Система **Скачки. Клиент** делает **Ставки** разных видов на **Забеги**. **Букмекер** устанавливает уровень выигрыша. **Администратор** фиксирует результаты **Забегов**.
18. Система **Телефонная станция**. **Администратор** осуществляет подключение **Абонентов**. **Абонент** может выбрать одну или несколько из предоставляемых **Услуг**. **Абонент** оплачивает **Счет** за разговоры и **Услуги**. **Администратор** может просмотреть список неоплаченных **Счетов** и заблокировать **Абонента**.
19. Система **LowCost-Авиакомпания**. **Клиент** заказывает и оплачивает **Билет** на **Рейс** с учетом наличия\отсутствия багажа и права первоочередной регистрации и посадки (Цена **Билета** может быть ниже стоимости провоза багажа). С приближением даты **Рейса** или наполнением самолета, цена на **Билет** может повышаться.
20. Система **Кофе-машина**. **Пользователь** обладает **Счетом**. **Кофе-машина** содержит набор **Напитков**, с заданным числом порций и дополнительных **Ингредиентов**. **Пользователь** может купить один или несколько **Напитков**. **Администратор Кофе-машины** осуществляет ее наполнение.

Вопросы к защите

1. Что такое Heap и Stack память в Java? Какая разница между ними?
2. Каков порядок вызова конструкторов и блоков инициализации с учётом иерархии классов?
3. Может ли статический метод быть переопределён или перегружен?
4. Могут ли нестатические методы перегрузить статические?
5. Можно ли сузить уровень доступа/тип возвращаемого значения при переопределении метода?
6. Возможно ли при переопределении метода изменить: модификатор доступа; возвращаемый тип; тип аргумента или их количество; имена аргументов или их порядок, убирать, добавлять, изменять порядок следования элементов секции throws?
7. Что такое autoboxing («автоупаковка») в Java и каковы правила упаковки примитивных типов в классы-обертки?
8. Что такое класс Object? Какие в нем есть методы?

9. Расскажите про клонирование объектов.
10. В чем отличие между поверхностным и глубоким клонированием?
11. Какой способ клонирования предпочтительней?
12. Почему метод clone() объявлен в классе Object, а не в интерфейсе Cloneable?
13. Дайте определение понятию «конструктор».
14. Что такое «конструктор по умолчанию»?
15. Чем отличаются конструкторы по-умолчанию, копирования и конструктор с параметрами?
16. Где и как вы можете использовать закрытый конструктор?
17. Расскажите про классы-загрузчики и про динамическую загрузку классов.
18. Что такое Reflection?
19. Зачем нужен equals(). Чем он отличается от операции ==?
20. equals() порождает отношение эквивалентности. Какими свойствами обладает такое отношение?
21. Если вы хотите переопределить equals(), какие условия должны удовлетворяться для переопределенного метода?
22. Правила переопределения метода Object.equals().
23. Какая связь между hashCode() и equals()?
24. Если equals() переопределен, есть ли какие-либо другие методы, которые следует переопределить?
25. Что будет, если переопределить equals() не переопределяя hashCode()? Какие могут возникнуть проблемы?
26. Каким образом реализованы методы hashCode() и equals() в классе Object?
27. Для чего нужен метод hashCode()?
28. Правила переопределения метода Object.hashCode().
29. Есть ли какие-либо рекомендации о том, какие поля следует использовать при подсчете hashCode()?
30. Могут ли у разных объектов быть одинаковые hashCode()?
31. Если у класса Point{int x, y;} реализовать метод equals(Object that) {(return this.x == that.x && this.y == that.y)}, но сделать хэш-код в виде int hashCode() {return x;}, то будут ли корректно такие точки помещаться и извлекаться из HashSet?
32. Могут ли у разных объектов (ref0 != ref1) быть ref0.equals(ref1) == true?
33. Могут ли у разных ссылок на один объект (ref0 == ref1) быть ref0.equals(ref1) == false?
34. Можно ли так реализовать метод equals(Object that) {return this.hashCode() == that.hashCode()}?
35. В equals() требуется проверять, что аргумент equals(Object that) такого же типа что и сам объект. В чем разница между this.getClass() == that.getClass() и that instanceof MyClass?
36. Можно ли реализовать метод equals() класса MyClass вот так: class MyClass {public boolean equals(MyClass that) {return this == that;}}?

37. Есть класс Point{int x, y;}. Почему хэш-код в виде 31 * x + y предпочтительнее чем x + y?