Для срезовой контрольной работы

- Для взаимодействия с базой данных используем чистый jdbc.
- Для обеспечения многопоточности используем Thread или классы из java.util.concurrent
- Программу выполнять в виде консольного приложения, ввод и вывод осуществлять с помощью файлов
- Демонстрация программы должна выглядеть следующим образом
    - Программа должна считать входящий файл, который включает в себя набор команд. Например, для регистрации пользователя, строка с командой может выглядеть следующим образом REGISTRATION|ФИО|email|phone
    - После выполнения программы в файле вывода должны сохраниться все операции с используемым набором данных и временем выполнения. В самом конце сделать вывод из всех таблиц

К демонстрации необходимо подготовить рабочий исходный код и скрипты для создания таблиц. В качестве базы данных рекомендуется использовать постгрес, поднятый в докере:

Команда создания и запуска
```
docker run -d \
   --name some-postgres \
   -e POSTGRES_PASSWORD=mysecretpassword \
   -p 5432:5432 \
   postgres
```
Команда запуска созданного контейнера
```
docker start some-postgres
```

Варианты:

Коммунальные платежи
Сущности

1. Пользователь
    a. ФИО
    b. E-мейл (выступает в качестве логина)
    c. телефон
2. Платежный адрес пользователя
    a. Уникальный идентификатор
    b. Строка с адресом
    c. Связь с пользователем
3. Шаблон
    a. Уникальный идентификатор
    b. Наименование шаблона
    c. Iban
    d. Назначение платежа

    e. Связь с адресом пользователя

У одного пользователя может быть несколько платежных адресов. Под адресом может быть несколько шаблонов.

4. Оплата
    a. Уникальный идентификатор
    b. Шаблон
    c. Номер карты
    d. Сумма
    e. Статус(Новый, Оплачен, Провален)
    f. Дата+время создания
    g. Дата+время изменения статуса

Все сущности должны иметь отражение в базе данных

Процесс оплаты

1) Создаем оплату в статусе новый
2) В отдельном потоке 1 раз в секунду вычитываем оплаты в статусе новый
3) Запрашиваем статус у компонента, который первые 2 секунды для платежа всегда выдает статус новый, а после 2-й секунды с одинаковой вероятностью выдает все 3 статуса

Сценарий выполнения демонстрационной программы:

1) Зарегистрировать пользователя
2) Завести ему 2 адреса
3) Под каждым адресом создать по 1-2 шаблона
4) Создать несколько платежей
5) Дождаться, когда платежи перейдут в финальный статус

Почтовая служба

Сущности

1. Пользователь
    a. ФИО
    b. E-мейл
    c. телефон(выступает в качестве логина)
2. Отделение
    a. Идентификатор
    b. Описание
3. Отправка посылки
    a. Идентификатор
    b. Пользователь (отправитель)
    c. Отделение получателя(из справочника)
    d. Отделение получателя(из справочника)
    e. Телефон получателя
    f. ФИО получателя
    g. Статус(Новый, Доставлена, Просрочена)
    h. Дата+время создания

        i. Дата+время изменения статуса
    4. Уведомление
        a. Идентификатор отправки
        b. Текст
        c. Статус(новый, отправлено)

Процесс доставки

1) Пользователь создает посылку
2) Получатель забирает или не забирает посылку (это организовать с помощью отдельно расписания, которое будет раз в секунду принимать по каждой посылке решение забрали посылку или нет. Причем решение, что посылку не забрали в 5 раз вероятнее. Через 5 секунд, если посылку не забрали, то она считается не доставленной.
3) В случае перехода в финальное состояние в таблицу уведомлений добавляется сообщение для отправителя об успешной или неуспешной доставке
4) Отдельный поток периодически вычитывает неотправленные сообщения и осуществляет отправку(вывод в файл), затем проставляет статус об отправке

Сценарий выполнения демонстрационной программы:

1) Зарегистрировать пользователя
2) Создать несколько отделений
3) Создать несколько отправок
4) Дождаться перехода всех отправок в финальной статус и дождаться выполнения отправок

Кинотеатр

Сущности

1. Пользователь
    a. ФИО
    b. Е-мейл
    c. Телефон(логин)
2. Кинотеатр
    a. Идентификатор
    b. Название
    c. Количество рядов
    d. Количество мест в ряду
3. Фильм
    a. Идентификатор
    b. Наименование
4. Сеанс
    a. Идентификатор
    b. Фильм
    c. Кинотеатр
    d. Время
    e. Стоимость
5. Билет
    a. Сеанс
    b. Ряд

    c. Место
    d. Пользователь
    e. Статус(Новый, Проведение оплаты, Оплачен, Забракован)
    f. Дата+время создания
    g. Дата+время изменения статуса
    h. Код платежа
6. Платеж
    a. Код платежа
    b. Сумма
    c. Карта
    d. Статус(новый/прошел/не прошел)

Процесс покупки билета

1) Первым методом создаем билет в статусе новый (без ид платежа). Обязательно контролируем, что место не занято(можно контроль осуществлять на уровне базы)
2) Вторым методом создаем платеж для билета и сохраняем идентификатор платежа в билет и переводим билет состояние “Проведение оплаты”
3) Отдельный поток вычитывает платежи в состояние новый и случайный образом присваивает платежу 1 из трех статусов.
4) Еще один поток вычитывает билеты в состоянии “Проведение оплаты” и для них по коду платежа получает состояние оплаты, и если оплата имеет финальный статус, то переводит билет в соответствующий статус

Сценарий выполнения демонстрационной программы:

1) Зарегистрировать пользователя
2) Создать несколько кинотеатров
3) Создать несколько фильмов
4) Создать несколько сеансов
5) Создать несколько билетов
6) Дождаться получения финальных статусов