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

е. Связь с адресом пользователя

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

- 4. Оплата
  - а. Уникальный идентификатор
  - b. Шаблон
  - с. Номер карты
  - d. Сумма
  - е. 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
  - а. ФИО
  - b. Е-мейл
  - с. телефон(выступает в качестве логина)
- 2. Отделение
  - а. Идентификатор
  - b. Описание
- 3. Отправка посылки
  - а. Идентификатор
  - b. Пользователь (отправитель)
  - с. Отделение получателя(из справочника)
  - d. Отделение получателя(из справочника)
  - е. Телефон получателя
  - f. ФИО получателя
  - g. Статус(Новый, Доставлена, Просрочена)
  - h. Дата+время создания

- і. Дата+время изменения статуса
- 4. Уведомление
  - а. Идентификатор отправки
  - b. Текст
  - с. 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:
  - Зарегистрировать пользователя
     Создать несколько отделений
  - 2) COSHATE RECKOTERO CIACTERIA
  - 3) Создать несколько отправок
  - 4) Дождаться перехода всех отправок в финальной статус и дождаться выполнения отправок

# Кинотеатр

### Сущности

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

- с. Место
- d. Пользователь
- е. Статус(Новый, Проведение оплаты, Оплачен, Забракован)
- f. Дата+время создания
- g. Дата+время изменения статуса
- h. Код платежа
- 6. Платеж
  - а. Код платежа
  - b. Сумма
  - с. 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