Тестовое задание Sigur

Предоставить программное решение, в котором:

I. Существуют следующие сущности:

1. Базовая сущность Person. Сущность имеет следующие поля:
- ID – уникальный целочисленный идентификатор. Уникален в том числе для всех дочерних типов. Другими словами, не может быть сущности Employee и Guest с одним и тем же идентификатором.
- CARD – бинарные данные, не превышающие длину в 16 байт. Также должны быть уникальными.
- TYPE — тип сущности: EMPLOYEE или GUEST.

1.1. Сущность Employee, наследник Person. Сущность имеет те же базовые поля как наследник Person, в дополнение к ним имеет следующие поля:
- HIRE_TIME — дата и время принятия сотрудника на работу.
- FIRED_TIME – дата и время "увольнения" сотрудника.
- DEPARTMENT_ID – идентификатор отдела, в котором он состоит.

1.2. Сущность Guest, также наследник Person. Сущность имеет те же базовые поля как наследник Person, в дополнение к ним имеет следующие поля:
- VISIT_DATE – дата планируемого посещения.
- EMP_ID — идентификатор сотрудника, к которому пришёл гость.

2. Department – отдел, в котором работает сотрудник (Employee). Сущность имеет следующие поля:
– ID — уникальный числовой идентификатор отдела.
– NAME — имя, может содержать кириллицу. Не превышает 32 символов в длину.

По умолчанию должно существовать некоторое количество отделов, скажем, 10 штук. Их число должно оставаться постоянным даже при нескольких запусках приложения.

II. Первый компонент — EmployeesMgr.

EmployeesMgr генерирует сотрудников, т. е. "нанимает на работу" новых сотрудников с некоторой частотой, например каждую секунду. Каждая новая итерация (каждая секунда) будет означать новый день, т. е. каждая итерация это +1 день к предыдущему. Будем называть данное время виртуальным. Нулевая итерация означает 1 января 2022 года. Последняя итерация это 31 декабря 2022 года.

- Время найма (HIRE_TIME) сотрудника на работу должно быть случайным и находиться в пределах "от текущего виртуального дня" и до конца виртуального периода (31 декабря 2022 года).
- CARD должен быть случайным, но уникальным среди всех сущностей Person.
- Идентификатор отдела DEPARTMENT_ID должен быть случайным и равен одному из заранее созданных сущностей Department.
- Каждый 5-ый найм "увольняет" случайное число сотрудников (от 1 до 3). Таким образом, проставляется дата и время FIRED_TIME. Также является виртуальным временем.

- Информация о произведённых наймах и увольнениях должна быть доступна даже в случае "падения" компонента EmployeesMgr.
- Все наймы и увольнения сотрудников логгируются в консоли в виде:
  - {Дата текущего дня, отсчитываемого компонентом (виртуальное время)}. Сотрудник {идентификатор сотрудника} нанят {дата и время найма (виртуальное время)}. Отдел: {имя отдела сотрудника}.
  - {Дата текущего дня, отсчитываемого компонентом (виртуальное время)}. Сотрудник {идентификатор сотрудника} уволен {дата и время увольнения (виртуальное время)}. Отдел: {имя отдела сотрудника}. Проработал: {количество дней, проведённых в штате (кол-во дней между наймом и увольнением виртуального времени)}.

III. Второй компонент GuestsMgr отвечает за планирование визитов гостей. Гость всегда приходит к какому-то конкретному сотруднику Employee. Компонент следит за тем, отменяется ли встреча, если сотрудник был уволен раньше, чем назначена встреча. Компонент оперирует тем же виртуальным временем.

- На каждого "нанятого" сотрудника с вероятностью 1/2 создаётся сущность Guest и проставляется дата визита этим гостем данного сотрудника. Дата также может быть случайной, но должна находится в пределах 6 месяцев после найма сотрудника.
- Информация о запланированном посещении или его отмене должна быть доступна даже после "падения" компонента GuestsMgr.
- Все изменения встреч логгируются в консоли в виде:
  - "Гостю ${Идентификатор гостя} назначена встреча сотруднику ${Идентификатор сотрудника. Отдел: {имя отдела сотрудника}. Дата: {дата встречи (виртуальное время)}. До встречи осталось: ${число дней (виртуальное время)}"
  - "Встреча гостя ${идентификатор гостя} с сотрудником ${идентификатор сотрудника} отменена. Отдел: {имя отдела сотрудника}. Дата встречи: ${дата встречи (виртуальное время)}, дата увольнения сотрудника: ${дата увольнения *виртуальное время)}".

IV. Третий компонент PassEmulator эмулирует проходы сотрудников и гостей. Он работает по тому же виртуальному времени.

- 10 раз "в виртуальный" день компонент либо генерирует случайный код карты длиной до 16 байт (с вероятностью 1/5), либо читает одну из карт, поле CARD сущностей Person (вероятность 4/5).
- Далее он должен понять, может ли "пройти" персона по этой карте:
  - Если карта не известна системе, то пишет в консоль: "Поднесена неизвестная карта: {HEX-представление карты}".
  - Если карта известна, нужно понять кому она принадлежит.
    - Если это сотрудник, и он нанят, но ещё не уволен, то пишем: "{Текущая виртуальная дата и время} Предоставлен доступ сотруднику {Идентификатор сотрудника}. Отдел: {имя отдела сотрудника}. Карта: {HEX-представление карты}".
    - Если он уже уволен, то пишет: "{Текущая виртуальная дата и время} Доступ запрещен сотруднику {Идентификатор сотрудника}. Отдел: {имя отдела сотрудника}. Карта: {HEX-представление карты}".
    - Если это гость, то проверяем назначена ли встреча. Пишем те же сообщения, но для гостя:
      - "{Текущая виртуальная дата и время} Предоставлен доступ гостю {Идентификатор гостя}. Пришёл к {Идентификатор гостя} из отдела: {имя отдела сотрудника}. Карта: {HEX-представление карты}".

- "{Текущая виртуальная дата и время} Доступ запрещён гостю {Идентификатор гостя}. Карта: {HEX-представление карты}".

Основной технический стек:
- Java 8
- Spring/Spring Boot
- Relational DB: H2 (embedded database) / MySQL / MariaDB / PostgreSQL

При желании/необходимости возможно использование любых дополнительных библиотек/программных компонентов.