1 лабораторная

Есть несколько Банков, которые предоставляют финансовые услуги по операциям с деньгами.

В банке есть Счета и Клиенты. У клиента есть имя, фамилия, адрес и номер паспорта (имя и фамилия обязательны, остальное – опционально).

Счета и проценты
Счета бывают трёх видов: Дебетовый счет, Депозит и Кредитный счет. Каждый счет принадлежит какому-то клиенту.

Дебетовый счет – обычный счет с фиксированным процентом на остаток. Деньги можно снимать в любой момент, в минус уходить нельзя. Комиссий нет.

Депозитный счет – счет, с которого нельзя снимать и переводить деньги до тех пор, пока не закончится его срок (пополнять можно). Процент на остаток зависит от изначальной суммы, например, если открываем депозит до 50 000 р. - 3%, если от 50 000 р. до 100 000 р. - 3.5%, больше 100 000 р. - 4%. Комиссий нет. Проценты должны задаваться для каждого банка свои.

Кредитный счет – имеет кредитный лимит, в рамках которого можно уходить в минус (в плюс тоже можно). Процента на остаток нет. Есть фиксированная комиссия за использование, если клиент в минусе.

Комиссии
Периодически банки проводят операции по выплате процентов и вычету комиссии. Это значит, что нужен механизм проматывания времени, чтобы посмотреть, что будет через день/месяц/год и т.п.

Процент на остаток начисляется ежедневно от текущей суммы в этот день, но выплачивается раз в месяц (и для дебетовой карты и для депозита). Например, 3.65% годовых. Значит в день: 3.65% / 365 дней = 0.01%. У клиента сегодня 100 000 р. на счету - запомнили, что у него уже 10 р. Завтра ему пришла ЗП и стало 200 000 р. За этот день ему добавили ещё 20 р. На следующий день он купил себе новый ПК и у него осталось 50 000 р. - добавили 5 р. Таким образом, к концу месяца складываем все, что запоминали. Допустим, вышло 300 р. - эта сумма добаляется к счету или депозиту в текущем месяце.

Разные банки предлагают разные условия. В каждом банке известны величины процентов и комиссий.

Центральный банк
Регистрацией всех банков, а также взаимодействием между банками занимается центральный банк. Он должен управлять банками (предоставлять возможность создать банк) и предоставлять необходимый функционал, чтобы банки могли взаимодействовать с другими банками (например, можно реализовать переводы между банками через него). Он также занимается уведомлением других банков о том, что нужно начислять остаток или комиссию - для этого механизма не требуется создавать таймеры и завязываться на реальное время.

Операции и транзакции
Каждый счет должен предоставлять механизм снятия, пополнения и перевода денег (то есть счетам нужны некоторые идентификаторы).

Еще обязательный механизм, который должны иметь банки - отмена транзакций. Если вдруг выяснится, что транзакция была совершена злоумышленником, то такая транзакция должна быть отменена. Отмена транзакции подразумевает возвращение банком суммы обратно. Транзакция не может быть повторно отменена.

Создание клиента и счета

Клиент должен создаваться по шагам. Сначала он указывает имя и фамилию (обязательно), затем адрес (можно пропустить и не указывать), затем паспортные данные (можно пропустить и не указывать).

Если при создании счета у клиента не указаны адрес или номер паспорта, мы объявляем такой счет (любого типа) сомнительным, и запрещаем операции снятия и перевода выше определенной суммы (у каждого банка своё значение). Если в дальнейшем клиент указывает всю необходимую информацию о себе - счет перестает быть сомнительным и может использоваться без ограничений.

Обновление условий счетов
Для банков требуется реализовать методы изменений процентов и лимитов не перевод. Также требуется реализовать возможность пользователям подписываться на информацию о таких изменениях - банк должен предоставлять возможность клиенту подписаться на уведомления. Стоит продумать расширяемую систему, в которой могут появится разные способы получения нотификаций клиентом (да, да, это референс на тот самый сайт). Например, когда происходит изменение лимита для кредитных карт – все пользователи, которые подписались и имеют кредитные карты, должны получить уведомление.

Консольный интерфейс работы
Для взаимодействия с банком требуется реализовать консольный интерфейс, который будет взаимодействовать с логикой приложения, отправлять и получать данные, отображать нужную информацию и предоставлять интерфейс для ввода информации пользователем.

Дополнения

На усмотрение студента можно ввести свои дополнительные идентификаторы для пользователей, банков etc.

На усмотрение студента можно пользователю добавить номер телефона или другие характеристики, если есть понимание зачем это нужно.

QnA
Q: Нужно ли предоставлять механизм отписки от информации об изменениях в условии счетов
A: Не обговорено, значит на ваше усмотрение (это вообще не критичный момент судя по условию лабы)

Q: Транзакциями считаются все действия со счётом, или только переводы между счетами. Если 1, то как-то странно поддерживать отмену операции снятия, а то после отмены деньги удвоятся: они будут и у злоумышлениика на руках и на счету. Или просто на это забить
A: Все операции со счетами - транзакции.

Q: Фиксированная комиссия за использование кредитного счёта, когда тот в минусе измеряется в % или рублях, и когда её начислять: после выполнения транзакции, или до. И нужно ли при отмене транзакции убирать и начисленную за неё комиссию.
A: Фиксированная комиссия означает, что это фиксированная сумма, а не процент. Да, при отмене транзакции стоит учитывать то, что могла быть также комиссия.

Q: Если транзакция подразумевает возвращение суммы обратно - но при этом эта же сумма была переведена на несколько счетов (пример перевод денег со счета 1 на счёт 2, со счёта 2 на счёт 3) Что происходит если клиент 1 отменяет транзакцию?
Подразумевается ли что деньги по цепочке снимаются со счёта 3? (на счету 2 их уже физически нет)
Либо у нас банк мошеннический и деньги "отмываются" и возмещаются клиенту 1 с уводом счёта 2 в минус
A: Банк не мошеннический, просто упрощённая система. Транзакции не связываются между собой. Так что да, можно считать, что может уйти в минус.

Иными словами: Переписать лабораторную 4 из курса по ООП на Java

2 лабораторная

Нужно написать сервис по учету котиков и их владельцев.

Существующая информация о котиках:

- Имя
- Дата рождения
- Порода
- Цвет (один из заранее заданных вариантов)
- Хозяин
- Список котиков, с которыми дружит этот котик (из представленных в базе)

Существующая информация о хозяевах:
- Имя
- Дата рождения
- Список котиков

Сервис должен реализовывать архитектуру controller-service-dao.

Вся информация хранится в БД PostgreSQL. Для связи с БД должен использоваться Hibernate.

Проект должен собираться с помощью Maven или Gradle (на выбор студента). Слой доступа к данным и сервисный слой должны являться двумя разными модулями Maven/Gradle. При этом проект должен полностью собираться одной командой.

При тестировании рекомендуется использовать Mockito, чтобы избежать подключения к реальным базам данных. Фреймворк для тестирования рекомендуется Junit 5.

В данной лабораторной нельзя использовать Spring или подобные ему фреймворки.

3 лабораторная

К созданному в прошлой лабораторной сервису добавляется Spring.

Сервис должен предоставлять http интерфейс (REST API) для получения информации о конкретных котиках и владельцах и для получения фильтрованной информации (например, получить всех рыжих котиков)

Внимание: недопустимо отдавать через HTTP интерфейс сущности JPA. Рекомендуется создать отдельные оберточные классы.

Сервисы и dao должны превратиться в Spring Bean’ы с использованием Dependency Injection (Autowired). Dao при этом наследуют JpaRepository и имеет шаблонные Spring Data JPA методы: https://www.baeldung.com/spring-data-repositories#repositories

При сдаче лабораторной нужно будет показать работоспособность endpoint’ов через http запросы (рекомендуется Postman).

4 лабораторная

Владельцы недовольны, что инфрмацию о котиках может получить кто угодно. В этой лабораторной мы добавим авторизацию к сервису.

Добавляется роль администратора. Он имеет доступ ко всем методам и может создавать новых пользователей. Пользователь связан с владельцем в соотношении 1:1.

Методы по получению информации и котиках и владельцах должны быть защищены Spring Security. Доступ к соотвутсвующим endpoint'ам имеют только владельцы котиков и администраторы. Доступ к методам для фильтрации имеют все авторизованные пользователи, но на выходе получают только данные о своих котиках.

Внимание: эндпоинты, созданные на предыдущем этапе, не должны быть удалены.

5 лабораторная

Бизнес прочитал статью о том, что микросервисы это круто и попросил нас разбить программу на микросервисы.

Из созданного приложения выделяются три микросервиса:
- Микросервис доступа к котикам
- Микросервис доступа к владельцам
- Микросервис с внешними интерфейсами.

Все они являются **разными приложениями**.

Все созданные ранее эндпоинты и авторизация переезжает на третий микросервис.

Общение между микросервисами происходит посредством RabbitMQ/Kafka (на выбор студента).

Сервисы доступа к котикам и доступа к владельцам могут либо быть подключены к одной БД, либо иметь разные БД. Во втором случае недопустимо делать один запрос на получение данных из двух БД, запроса должно быть два (по одному в каждую).

Внимание: недопустимо передавать через RabbitMQ/Kafka JPA сущности. Рекомендуется создать отдельные оберточные классы.

Рекомендуется выделить модуль с JPA сущностями в отдельный подключаемый модуль.