Оглавление

Тестовое задание «Бонусные карты»

Цели:

1. Проверить способность соискатель разбираться в спецификациях протоколов и реализовывать их поддержку;
2. Проверить, насколько соискатель знаком с ключевыми технологиями JavaEE (JPA/Hibernate, Servlet API, веб-фреймворки);

Критерии оценки

Для оценки будет использоваться следующий набор критериев:

1. Удобочитаемость и простота понимания кода;
2. Соответствие реализации спецификации;

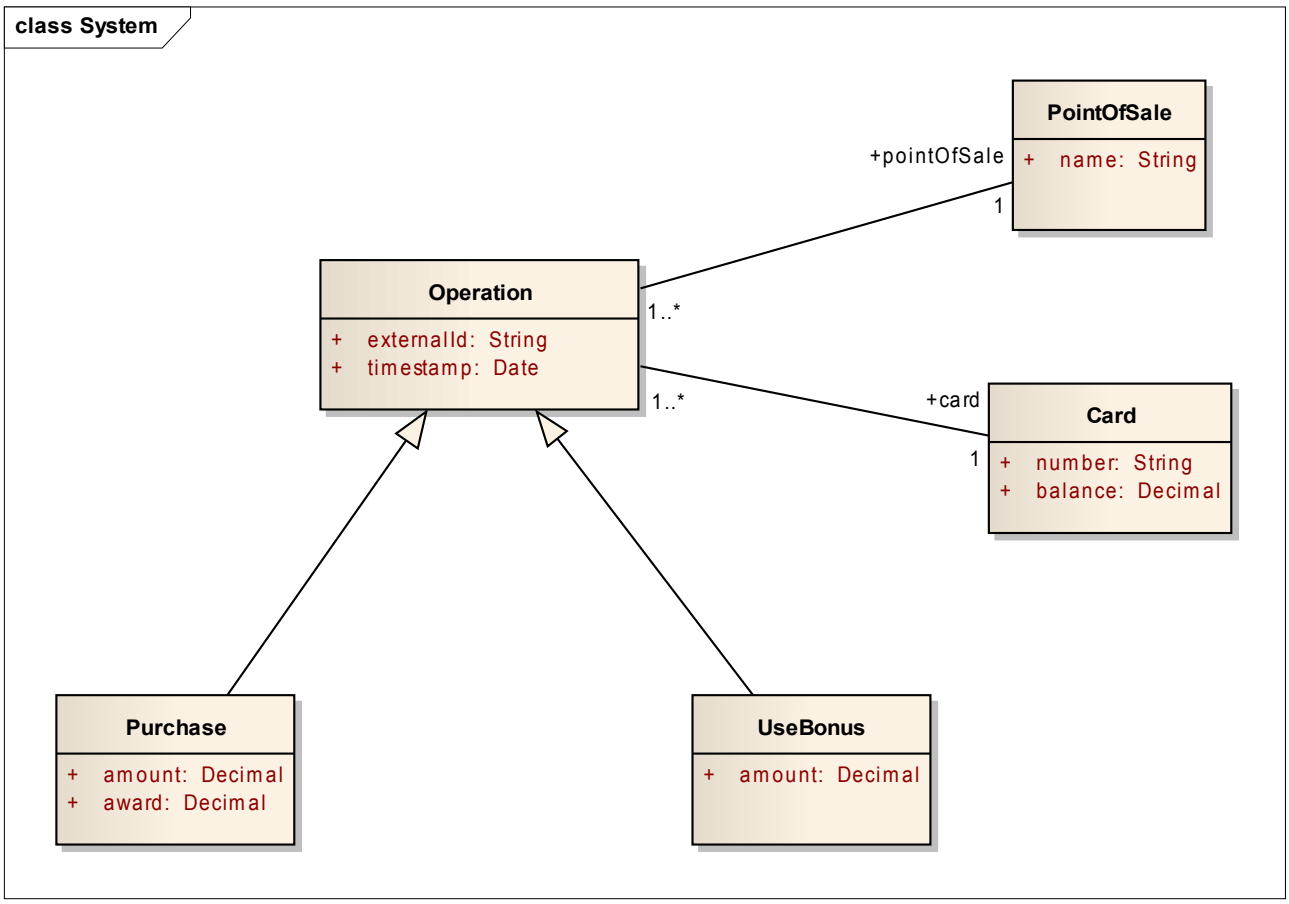
Описание бизнес-процесса

Процессинговая система «Бонусные карты» используется для поддержки программы лояльности частых покупателей в торговых организациях.

Программа лояльности функционирует следующим образом:

1. Покупатель получает карту при совершении любой покупки;
2. На карту начисляется число баллов, равное 10% от суммы совершённых покупок. Запрос на начисление баллов отправляется кассовым ПО в процессинговую систему после совершения покупки. Карта регистрируется в процессинговой системе при обработке первого такого запроса, а далее ведётся учёт баланса на карте;
3. Покупатель может частично или полностью расплачиваться за дальнейшие свои покупки накопленными баллами по курсу один к одному. Запрос на списание баллов отправляется кассовым ПО в процессинговую систему до совершения покупки. Запрос выполняется только если баланс на карте не меньше, чем запрошенная сумма на списание;
4. В любой момент покупатель может захотеть узнать текущий баланс на своей карте. Для этого ему нужно обратиться в любую кассу. Кассовое ПО отправляет в процессинговую систему запрос на получение текущего баланса, передавая при этом номер карты. Результат высвечивается на дисплее и сообщается покупателю кассиром.

Модель бизнес-объектов процессинговой системы показана на диаграмме:



Пояснения к модели:

- Класс Operation – операция по карте:

    - атрибут externalId – уникальный идентификатор операции, назначаемый извне;
    - атрибут timestamp – время совершения операции;
    - ссылка card – карта, по которой совершается операция;
    - ссылка pointOfSale – место совершения операции.
- Класс PointOfSale – справочник мест совершения операций;
- Класс Card – учётная запись карты в системе:
    - атрибут number – уникальный номер карты в формате DDDD-DDDD-DDDD;
    - атрибут balance – текущий баланс по карте;
- Класс Purchase – тип операции, соответствующий совершённой покупке:
    - атрибут amount – сумма покупки;
    - атрибут award – сумма начисленных бонусов;
- Класс UseBonus – тип операции, соответствующий оплате бонусами;
    - атрибут amount – сумма списанных с карты бонусов.

Функциональные требования

Требуется разработать процессинговую систему, предназначенную для обеспечения автоматического функционирования описанного выше бизнес-процесса программы лояльности.

Требования:

1 Реализовать протокол онлайн-взаимодействия в соответствии со спецификацией:
  1.1 запрос на регистрацию покупки и начисление баллов;
  1.2 запрос на получение текущего баланса;
  1.3 запрос на списание баллов с карты.
2 Сохранять информацию о совершённых операциях в базе данных.

Нефункциональные требования

1. Использование встраиваемой СУБД HyperSQL 2.0 (см. http://hsqldb.org/);
2. Использование Hibernate или JPA для работы с СУБД;
3. Сборка проекта с помощью Apache Maven или Apache Ant;
4. Выполняемые процессинговой системой действия должны логироваться в файл;
5. Для управления жизненным циклом компонентов желательно использование Spring Framework.

Что не нужно делать:

1. Механизм аутентификации и авторизации;
2. Веб-интерфейс;

Приложение А: Описание онлайн-протокола

Протокол построен на базе HTTP. Операция осуществляется при получении POST запроса на некий URL (определяется реализацией).

Запрос содержит параметры в query string (после знака ? в URL). Параметры специфичны для типа операции (см. ниже).

Операция регистрации покупки и начисления баллов

Запрос принимает следующие обязательные параметры:

- operation — тип операции, всегда **award**;
- card — номер карты, строка;
- pointOfSale — место совершения операции, строка;
- amount — общая сумма покупки;
- id – уникальный идентификатор операции.

Алгоритм обработки запроса:

1. Если запрос является некорректным, то:
   - возвращается HTTP ошибка 400 Bad Request.
2. Если операция с указанным уникальным идентификатором уже существует, то:
   - возвращается HTTP результат 304 Not Modified.
3. Если карта с заданным номером не существует, то:
   - создаётся новая карта с указанным номером и нулевым балансом;
4. Если всё корректно, то:
   - создаётся новая операция типа Purchase, в неё сохраняется:
   а) списываемая сумма;
   б) указываемое место совершения операции;
   в) карта, с которой происходит списание;
   г) переданный уникальный идентификатор операции;
   - баланс карты увеличивается в соответствии с бизнес-процессом (см. выше);
   - возвращается HTTP результат 200 OK;
   - тело HTTP ответа содержит обновлённый баланс карты.

Пример корректной строки запроса:

```
POST /bonus?operation=award&card=1234-5678-
9090&pointOfSale=terminal321&amount=120.50&id=334455 HTTP/1.1
```

Операция получения текущего баланса

Запрос принимает следующие обязательные параметры:

- operation — тип операции, всегда **check**;
- card — номер карты, строка.

Алгоритм обработки запроса:

1. Если запрос является некорректным, то
   - возвращается HTTP ошибка 400 Bad Request.
2. Если карта с заданным номером не существует, то:
   - возвращается HTTP ошибка 404 Not Found.
3. Если всё корректно, то:
   - возвращается HTTP результат 200 OK;
   - тело HTTP ответа содержит текущий баланс карты.

Пример корректной строки запроса:

```
POST /bonus?operation=check&card=1234-5678-9090 HTTP/1.1
```

Операция списания баллов

Запрос принимает следующие обязательные параметры:

- operation — тип операции, всегда **pay**;
- card — номер карты, строка;
- pointOfSale — место совершения операции, строка;
- amount — сумма списываемых с карты баллов;
- id – уникальный идентификатор операции.

Алгоритм обработки запроса:

1. Если запрос является некорректным, то
   - возвращается HTTP ошибка 400 Bad Request.
2. Если карта с заданным номером не существует, то:
   - возвращается HTTP ошибка 404 Not Found.
3. Если операция с указанным уникальным идентификатором уже существует, то:
   - возвращается HTTP результат 304 Not Modified.
4. Если карта с заданным номером существует, но на её балансе не достаточно средств для списания, то:
   - возвращается HTTP ошибка 409 Conflict;
   - тело HTTP ответа содержит текущий баланс карты.
5. Если карта с заданным номером существует и на балансе достаточно средств для списания, то:
   - создаётся новая операция типа UseBonus, в ней сохраняется
   а) списываемая сумма;
   б) указываемое место совершения операции;
   в) карта, с которой происходит списание;
   г) переданный уникальный идентификатор операции;
   - сокращается баланс на карте на запрошенную сумму;
   - возвращается HTTP результат 200 OK;
   - тело HTTP ответа содержит текущий баланс карты.

Пример корректной строки запроса:

```
POST /bonus?operation=pay&card=1234-5678-
9090&pointOfSale=terminal123&amount=17.35&id=667788 HTTP/1.1
```