Данное задание является стартовым для курса и включает цель отработки навыка решения типовых объектно-ориентированных задач с использованием языка Java. В ходе выполнения работы необходимо разработать группу классов, которые реализуют прописанные в задании требования.

Задание состоит из набора требований, каждое из которых необходимо решать в указанной очередности.

**Часть 1. Разработка базового класса**

Необходимо разработать класс **Account**, который будет представлять собой абстракцию банковского счета. Он будет хранить имя владельца и количество всех имеющихся валют.

Реализовать класс необходимо придерживаясь следующих требований:

1. **Account** имеет поле для *имени владельца*: **String**

2. **Account** имеет поле, в котором хранятся пары валюта-количество.

Количество валюты хранится в виде целочисленного значения. Валюта должна быть представлена таким образом, чтобы указать можно было только значение из некоторого фиксированного списка значений (конкретный перечень допустимо указать произвольно в коде).

3. Создание объекта **Account** возможно только с указанием имени владельца счета.

4. Для имени необходимо сделать геттеры и сеттеры

5. Для пар валюта-количество необходимо сделать только геттер.

6. Необходим метод, который принимает Валюту и её количество и заменяет текущее количество данной Валюты на указанное. Если такой валюты ранее не было – она добавляется в список.

Также необходимо реализовать следующие ограничения:

**—** Имя не может быть null или пустым

**—** Количество валюты не может быть отрицательным

В случае нарушения ограничений необходимо бросать подходящее исключение.

Необходимо реализовать и приложить модульные тесты, проверяющие выполнение обозначенных требований.

**Часть 2. Отмена**

Необходимо реализовать в классе **Account** метод **undo**, который будет отменять одно последнее изменение объекта класса **Account**. Метод должен поддерживать следующие требования:

**—** Вызывать метод можно несколько раз подряд. Каждый вызов откатывает еще одно изменение. Изменениями считаются смена имени владельца и смена значений для валют. Например:

* после создания объекта, мы сначала добавили сто рублей,
* потом сменили имя на “Василий Иванов”
* потом установили количество рублей на 300.
* Первый вызов отмены установит число рублей на 100, второй вызов вернет начальное имя, третий вызов уберет рубли из списка вообще.

**—** Откатывать изменения можно до тех пор, пока объект не вернется к состоянию, в котором он был на момент создания. Необходимо предоставить метод проверки возможности отмены.

**—** Попытка отменить изменения, если их не было — это ошибка.

**—** Реализация отмены должна быть выполнена таким образом, чтобы, когда в класс будут добавлены новые поля, их можно было учитывать в отмене, однако ранее реализованный код не требовал бы изменения.

Например: к уже реализованному коду Account необходимо добавить тип: обычный или премиальный счет.

Реализация отмены смены типа счета не должна требовать изменений в код метода undo или методов работы с именем и валютами.

**—** Метод undo не имеет параметров

Реализуйте модульные тесты для проверки работоспособности кода.

**Часть 3. Сохранение**

Необходимо реализовать возможность получать Сохранения у объектов класса Account. Требования включают:

**—** Метод сохранения возвращает объект, который хранит состояние Account на момент запроса сохранения.

**—** Объект сохранения не должен изменяться после его создания (другими словами – иммутабелен) и не нарушает ранее реализованные требования инкапсуляции.

**—** Изменения оригинального объекта Account также не оказывают влияния на Сохранение.

**—** Объектов Сохранений может быть сколько угодно для каждого из Account.

**—** Любое Сохранение может быть использовано для приведения соответствующего ему объекта Account в состояние соответствующее моменту создания сохранения.

**—** Всю информацию для отмены действий можно не сохранять.

Реализуйте модульные тесты для проверки работоспособности кода.