Задача 1: Система обработки заказов с инкапсуляцией данных заказчика

Представьте, что вы работаете над системой электронной коммерции, которая управляет заказами клиентов. Вам нужно разработать структуру классов, которая будет использовать инкапсуляцию для защиты данных заказчиков (например, адресов доставки, платежной информации и статусов заказа).

Требования:

1. Создайте класс Order, который будет содержать информацию о заказе:
    - Приватные атрибуты для идентификатора заказа, списка товаров, адреса доставки, статуса заказа и платежной информации.
    - Приватный атрибут для общей стоимости заказа, который будет обновляться при изменении товаров.
2. Реализуйте методы для добавления/удаления товаров в заказе, изменения статуса заказа (например, "Ожидается оплата", "Отправлен", "Доставлен"), при этом доступ к атрибутам должен быть строго ограничен.
3. Создайте отдельный класс Customer, который будет управлять заказами клиента. Этот класс должен предоставлять методы для создания нового заказа и изменения его состояния (например, изменить адрес доставки или статус). При этом, данные заказов должны быть скрыты от прямого доступа.
4. Реализуйте метод для обработки оплаты, который будет:
    - Проверять, оплачено ли уже;
    - В зависимости от статуса платежа, обновлять статус заказа и защищать от повторных попыток оплатить уже оплаченный заказ.
5. Добавьте механизм, который позволит клиенту изменить адрес доставки только до тех пор, пока заказ не отправлен (после отправки изменение адреса запрещено).

Задача 2: Генерация уникальных идентификаторов для распределённой системы

В распределённой системе, где несколько серверов работают одновременно, необходимо генерировать уникальные идентификаторы для объектов (например, записей в базе данных).

Требуется создать класс IDGenerator со следующими статическими методами:

1. Метод генерации идентификатора, который использует текущее время в формате UNIX (timestamp), уникальный номер сервера (например, серверы с номерами от 1 до 1000), и случайный элемент (например, случайное число).
    - Идентификатор должен быть строкой и включать в себя серверный номер и случайный элемент для обеспечения уникальности.
2. Метод для проверки валидности идентификатора, который принимает на вход идентификатор и проверяет, что он соответствует заданному формату

(например, начинается с номера сервера, имеет корректный timestamp, и длина идентификатора правильная).

3. Метод для генерации списка идентификаторов для партии объектов, где можно передать количество объектов и номер сервера, и метод вернёт список уникальных идентификаторов.

Задача 3: Управление доступом к полям объекта на основе ролей пользователя

Вам нужно реализовать систему управления доступом к атрибутам объектов на основе ролей пользователя. Для этого создайте дескриптор, который будет ограничивать доступ к определённым атрибутам класса в зависимости от роли текущего пользователя (например, admin, editor, viewer).

Условия:

1. У класса должен быть атрибут, который хранит роль текущего пользователя.
2. Дескриптор должен контролировать доступ к чтению и записи атрибутов в зависимости от роли:
   - admin может читать и изменять все поля.
   - editor может читать все поля, но не имеет права их изменять.
   - viewer может читать только определённые поля, но не имеет доступа к другим.
3. Если пользователь попытается получить доступ к запрещённому полю, выбрасывайте исключение PermissionError.