### Задание

1. Создайте базовый класс User, представляющий пользователя.
2. Создайте производный класс Customer, наследующий от User.
3. Создайте производный класс Admin, наследующий от User.
4. Реализуйте регистрацию пользователей с проверкой уникальности имени пользователя и хешированием паролей.
5. Реализуйте аутентификацию пользователей с проверкой пароля.
6. Реализуйте управление сессиями, чтобы отслеживать текущего вошедшего пользователя.
7. Реализуйте функции для управления пользователями (просмотр списка пользователей, удаление пользователей) только для админа.

**Шаблон:**

import hashlib

import uuid

**class User:**

"""

Базовый класс, представляющий пользователя.

"""

users = [] # Список для хранения всех пользователей

def \_\_init\_\_(self, username, email, password):

@staticmethod

def hash\_password(password):

@staticmethod

def check\_password(stored\_password, provided\_password):

"""

Проверка пароля.

"""

def get\_details(self):

**class Customer(User):**

"""

Класс, представляющий клиента, наследующий класс User.

"""

def \_\_init\_\_(self, username, email, password, address):

def get\_details(self):

**class Admin(User):**

"""

Класс, представляющий администратора, наследующий класс User.

"""

def \_\_init\_\_(self, username, email, password, admin\_level):

def get\_details(self):

@staticmethod

def list\_users():

"""

Выводит список всех пользователей.

"""

@staticmethod

def delete\_user(username):

"""

Удаляет пользователя по имени пользователя.

"""

class AuthenticationService:

"""

Сервис для управления регистрацией и аутентификацией пользователей.

"""

def \_\_init\_\_(self):

def register(self, user\_class, username, email, password, \*args):

"""

Регистрация нового пользователя.

"""

def login(self, username, password):

"""

Аутентификация пользователя.

"""

def logout(self):

"""

Выход пользователя из системы.

"""

def get\_current\_user(self):

"""

Возвращает текущего вошедшего пользователя.

"""

# Пример использования

auth\_service = AuthenticationService()

### Пояснения к коду

1. **Классы** User**,** Customer **и** Admin:
   * Базовый класс User хранит общие свойства пользователей и включает методы для хеширования и проверки пароля.
   * Класс Customer наследует от User и добавляет свойство address.
   * Класс Admin наследует от User и добавляет свойство admin\_level. Также включает методы для управления пользователями, такие как list\_users и delete\_user.
2. **Класс** AuthenticationService:
   * Класс AuthenticationService управляет регистрацией, аутентификацией и управлением сессиями пользователей.
   * Метод register регистрирует нового пользователя с проверкой уникальности имени пользователя и хешированием пароля. Возвращает сообщение об успехе или ошибке.
   * Метод login аутентифицирует пользователя, проверяя его пароль, и генерирует токен сессии. Возвращает сообщение об успехе или ошибке.
   * Метод logout завершает сессию текущего пользователя. Возвращает сообщение об успехе или ошибке.
   * Метод get\_current\_user возвращает информацию о текущем вошедшем пользователе. Возвращает сообщение об успехе или ошибке.
3. **Пример использования**:
   * Демонстрирует регистрацию, аутентификацию, управление сессиями и административные функции, такие как просмотр и удаление пользователей.
   * Результаты операций выводятся на экран для наглядности.