МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра “Математическая кибернетика и информационные технологии”

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Введение в информационные технологии»

Тема: «Работа с классами ч.2»

Выполнил: студент группы БВТ2505

Хардиков ВладиславДмитриевич

Проверил: Павликов. А.Е.

Москва, 2025

Цель работы

Получить практический опыт работы с ООП в Python. использование инкапсуляции, наследования

Задание 1:

1. Создать класс UserAccount, который представляет аккаунт пользователя с атрибутами: имя пользователя (username), электронная почта (email) и приватный атрибут пароль (password).

2. Использовать конструктор __init__ для инициализации этих атрибутов.

3. Реализовать метод set_password(new_password), который позволяет безопасно изменить пароль аккаунта.

4. Реализовать метод check_password(password), который проверяет, соответствует ли введённый пароль текущему паролю аккаунта и возвращает True или False.

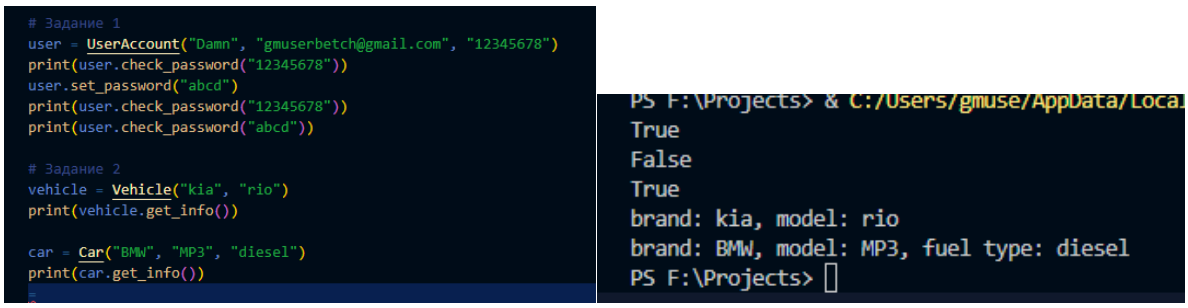
5. Создать объект класса UserAccount, попробовать изменить пароль и проверить его с помощью методов set_password и check_password.

Задание 2:

1. Определить базовый класс Vehicle с атрибутами: make (марка) и model (модель), а также методом get_info(), который возвращает информацию о транспортном средстве.

2. Создайть класс Car, наследующий от Vehicle, и добавьте в него атрибут fuel_type (тип топлива). Переопределить метод get_info() таким образом, чтобы он включал информацию о типе топлива.

Скриншоты выполнения:



Результаты выполнения программ

Исходный код программы:

```
class UserAccount:
    def __init__(self, username: str, email: str, password: str):
        self.username = username
        self.email = email
        self.__password = password


    def set_password(self, new_password: str) -> None:
```

```
        self.__password = new_password

    def check_password(self, password: str) -> bool:
        return self.__password == password

class Vehicle:
    def __init__(self, make: str, model: str):
        self.make = make
        self.model = model

    def get_info(self) -> str:
        return f"brand: {self.make}, model: {self.model}"

class Car(Vehicle):
    def __init__(self, make: str, model: str, fuel_type: str):
        super().__init__(make, model)
        self.fuel_type = fuel_type

    def get_info(self) -> str:
        return f"brand: {self.make}, model: {self.model}, fuel type: {self.fuel_type}"

# Задание 1
user = UserAccount("Damn", "gmuserbetch@gmail.com", "12345678")
print(user.check_password("12345678"))
user.set_password("abcd")
print(user.check_password("12345678"))
print(user.check_password("abcd"))

# Задание 2
vehicle = Vehicle("kia", "rio")
print(vehicle.get_info())
```

```
car = Car("BMW", "MP3", "diesel")

print(car.get_info())
```

Заключение

В ходе лабораторной работы были изучены инкапсуляции, наследования и полиморфизма в ООП на Python. На примере класса UserAccount показано, как создавать приватные атрибуты и реализовывать методы для безопасной работы с конфиденциальными данными пользователя, включая изменение и проверку пароля.

Были рассмотрены принципы наследования на примере базового класса Vehicle и производного класса Car. Была реализована переопределённая версия метода get_info(), что позволило продемонстрировать использование полиморфизма.