**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 Программная инженерия

Отчет по лабораторной работе №6

“**Работа с классами ч.2**”

по дисциплине «Введение в информационные технологии»

Выполнил: студент группы

БПИ2501

Карпеко Никита Александрович

Проверила: Мосева Марина Сергеевна

Москва, 2025

**Цель работы**

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

**Задания**

**1. Защита данных пользователя**

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

1.2. Использовать конструктор \_\_init\_\_ для инициализации этих атрибутов.

1.3. Реализовать метод set\_password(new\_password), который позволяет безопасно изменить пароль аккаунта.

1.4. Реализовать метод check\_password(password), который проверяет, соответствует ли введённый пароль текущему паролю аккаунта и возвращает True или False.

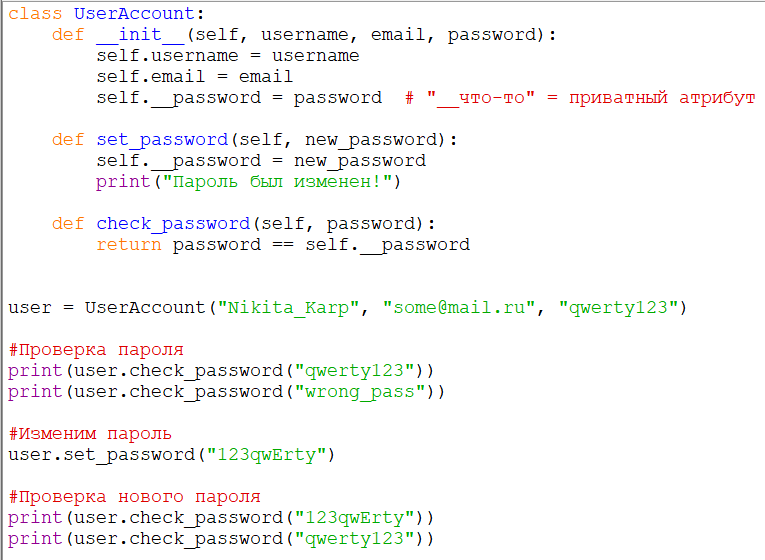
1.5. Создать объект класса UserAccount, попробуйте изменить пароль и проверить его с помощью методов set\_password и check\_password.

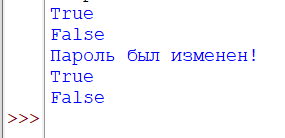
**2. Полиморфизм и наследование**

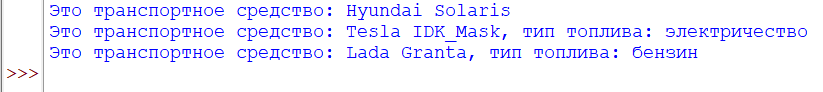
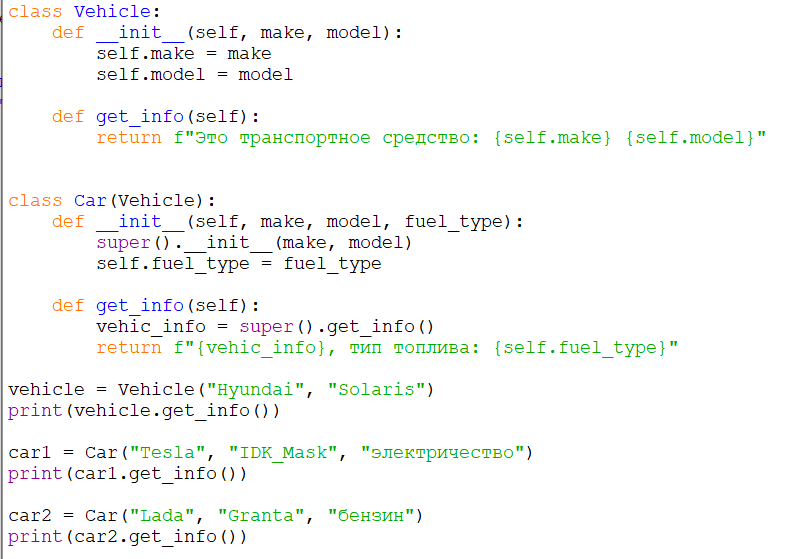
2.1. Определить базовый класс Vehicle с атрибутами: make (марка) и model (модель), а также методом get\_info(), который возвращает информацию о транспортном средстве.

2.2. Создать класс Car, наследующий от Vehicle, и добавьте в него атрибут fuel\_type (тип топлива). Переопределите метод get\_info() таким образом, чтобы он включал информацию о типе топлива.

**Скриншоты выполнения (группы – код и вывод выполнения)**







**Исходный код**

class UserAccount:

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

self.username = username

self.email = email

self.\_\_password = password # "\_\_что-то" = приватный атрибут

def set\_password(self, new\_password):

self.\_\_password = new\_password

print("Пароль был изменен!")

def check\_password(self, password):

return password == self.\_\_password

user = UserAccount("Nikita\_Karp", "some@mail.ru", "qwerty123")

#Проверка пароля

print(user.check\_password("qwerty123"))

print(user.check\_password("wrong\_pass"))

#Изменим пароль

user.set\_password("123qwErty")

#Проверка нового пароля

print(user.check\_password("123qwErty"))

print(user.check\_password("qwerty123"))

class Vehicle:

def \_\_init\_\_(self, make, model):

self.make = make

self.model = model

def get\_info(self):

return f"Это транспортное средство: {self.make} {self.model}"

class Car(Vehicle):

def \_\_init\_\_(self, make, model, fuel\_type):

super().\_\_init\_\_(make, model)

self.fuel\_type = fuel\_type

def get\_info(self):

vehic\_info = super().get\_info()

return f"{vehic\_info}, тип топлива: {self.fuel\_type}"

vehicle = Vehicle("Hyundai", "Solaris")

print(vehicle.get\_info())

car1 = Car("Tesla", "IDK\_Mask", "электричество")

print(car1.get\_info())

car2 = Car("Lada", "Granta", "бензин")

print(car2.get\_info())

**Заключение**

Была проведена работа с ООП в Python, использованием инкапсуляции, наследования.