**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**

#Полное чтение

with open('example.txt', 'r', encoding='utf-8') as file:

content = file.read()

print(content)

#Построчное чтение

with open('example.txt', 'r', encoding='utf-8') as file:

for line in file:

print(line)

def reading(method):

if method == 'full':

#Полное чтение

with open('example.txt', 'r', encoding='utf-8') as file:

content = file.read()

print(content)

elif method == 'parts':

#Построчное чтение

with open('example.txt', 'r', encoding='utf-8') as file:

for line in file:

print(line)

else:

print('no such method')

n = input('reading using \'full\' or \'parts\'? \n')

reading(n)

def writing():

print ('\'stop\' чтобы прекратить \n')

n = 'start'

with open('user\_input.txt', 'a', encoding='utf-8') as file:

while n != 'stop':

n = input()

if n != 'stop':

file.write(n + '\n')

writing()

def spec\_reading(name, method):

try:

if method == 'full':

#Полное чтение

with open(str(name) + '.txt', 'r', encoding='utf-8') as file:

content = file.read()

print(content)

elif method == 'parts':

#Построчное чтение

with open(str(name) + '.txt', 'r', encoding='utf-8') as file:

for line in file:

print(line)

else:

print('no such method')

except Exception as FileNotFoundError:

print("Нет такого файла...")

n = input('write a name of file and you like reading using \'full\' or \'parts\'? \n')

a, b = n.split()

spec\_reading(a, b)

**Заключение**

Были освоены принципы определения и использования функций в языке программирования Python для открытия, чтения, редактирования файлов, поняты механизмы передачи аргументов в функции, связанные с этим, также было изучено - как применять функции для решения практических задач по записи текстовых данных в файлы.