МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ И МАССОВЫХ КОММУНИКАЦИЙ

Ордена Трудового Красного Знамени
Федеральное государственное бюджетное общеобразовательное учреждение высшего образования

“Московский технический университет связи и информатики”
(МТУСИ)
Кафедра “Программная инженерия”

Лабораторная работа №6

по дисциплине “Введение в информационные технологии”

Выполнил: Студент группы
БПИ2503
Яричевский Даниил

Москва
2025

Цель работы

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

Задачи

1. Защита данных пользователя
2. Полиморфизм и наследование

Ход работы

1.1 Создаем класс UserAccount.

```
1   class UserAccount:
```

Рис. (1.1)

1.2 Используя конструктор __init__ добавляем три атрибута: имя пользователя (username), электронная почта (email) и приватный атрибут пароль (password).

```
1   class UserAccount:
2       def __init__(self, username, email, password):
3           self.username = username
4           self.email = email
5           self.__password = password
```

Рис. (1.2)

1.3 Определяем метод **set_password**(**new_password**), который позволяет безопасно изменить пароль аккаунта.

```
7       def set_password(self, old_password, new_password):
8           if self.__password == old_password:
9               self.__password = new_password
10          return False
```

Рис. (1.3)

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

```
12      def check_password(self, password):
13          if self.__password == password:
14              return True
15          return False
```

Рис. (1.4)

1.5 Создаем объект класса UserAccount, попробуйте изменить пароль и проверить его с помощью методов set_password и check_password.

```
18  Me = UserAccount("DYF", "me@dyf.com", "1234")
19  print(Me.set_password("1234", "0987"))
20  print(Me.check_password("1234"))
```

Рис. (1.5)

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

```
1   class Vechicle:
2       def __init__(self, make, model):
3           self.make = make
4           self.model = model
5
6       def get_info(self):
7           return f"Марка: {self.model}, Модель: {self.model}"
```

Рис. (2.1)

2.2 Создаем 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.

```
9   class Car(Vechicle):
10      def __init__(self, make, model, fuel_type):
11          self.fuel = fuel_type
12          super().__init__(make, model)
13      def get_info(self):
14          car_info = super().get_info()
15          return f"{car_info}, Тип топлива: {self.fuel}"
```

Рис. (2.2)

Вывод

Я получил практический опыт работы с ООП в Python. Использовал инкапсуляцию, наследование.