**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена Трудов ого Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра Программная инженерия

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Введение в информационные технологии»

Тема: «Работа с классами ч.3»

Выполнил: студент группы БВТ2501

Лебедев Даниил Владимирович

Москва, 2025

**Цель работы**

Разработать систему управления сотрудниками, демонстрирующую множественное наследование, инкапсуляцию и полиморфизм в Python. Система должна уметь обрабатывать различные типы сотрудников, включая менеджеров и технических специалистов, а также предоставлять возможность для расширения и добавления новых ролей.

**Индивидуальное задание**

1. Создать класс Employee с общими атрибутами, такими как name (имя), id (идентификационный номер) и методами, например, get\_info(), который возвращает базовую информацию о сотруднике.  
2. Создать класс Manager с дополнительными атрибутами, такими как department (отдел) и методами, например, manage\_project(), символизирующим управление проектами.  
3. Создать класс Technician с уникальными атрибутами, такими как specialization (специализация), и методами, например, perform\_maintenance(), означающим выполнение технического обслуживания.  
4. Создать класс TechManager, который наследует как Manager, так и Technician. Этот класс должен комбинировать управленческие способности и технические навыки, например, иметь методы для управления проектами и выполнения технического обслуживания.  
5. Добавить метод add\_employee(), который позволяет TechManager добавлять сотрудников в список подчинённых.  
6. Реализовать метод get\_team\_info(), который выводит информацию о всех подчинённых сотрудниках.  
7. Создать объекты каждого класса и демонстрируйте их функциональность.

Ход работы:

1. Импортируем sleep из библиотеки time, она мне понадобится дальше в программе.
2. Напишем класс Employee (рис 1)

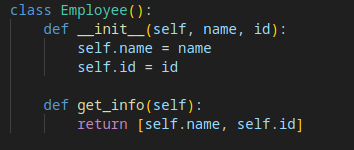


Рис 1

1. Напишем класс Manager, который будем наследовать от Employee (рис 2), тут мы будем использовать вместо super(), прямую ссылку на класс, чтобы программа, не запуталась в них

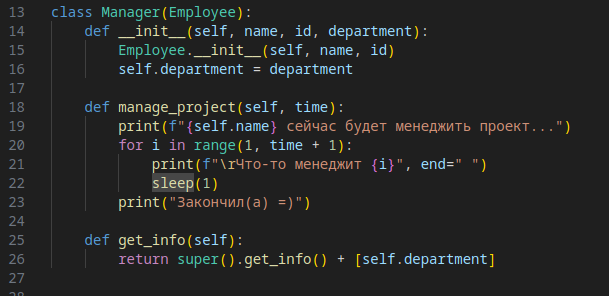


Рис 2

1. Напишем класс Technician, с которым ситуация будет аналогична с классом Manager (рис 3)

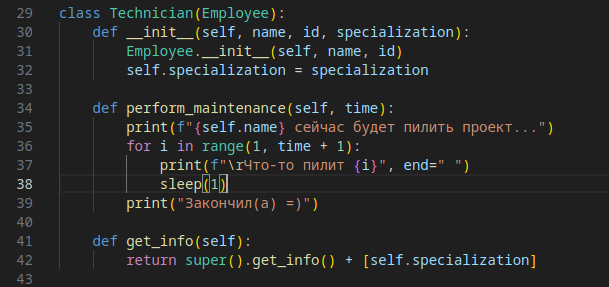
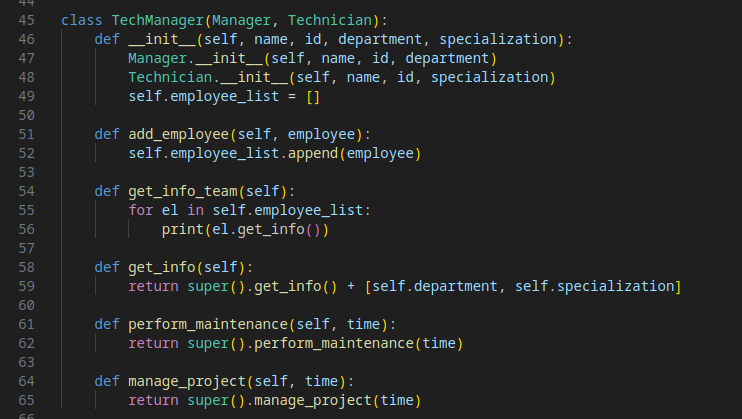


Рис 3

1. Создадим класс Techmanager, который будет наследоваться от трёх предыдущих классов, тут мы тоже будем использовать прямые ссылки к классам и конструкцию super

Рис 4

1. Перейдём к главной части, создадим объекты от этих классов и вызовем их методы (рис 5)

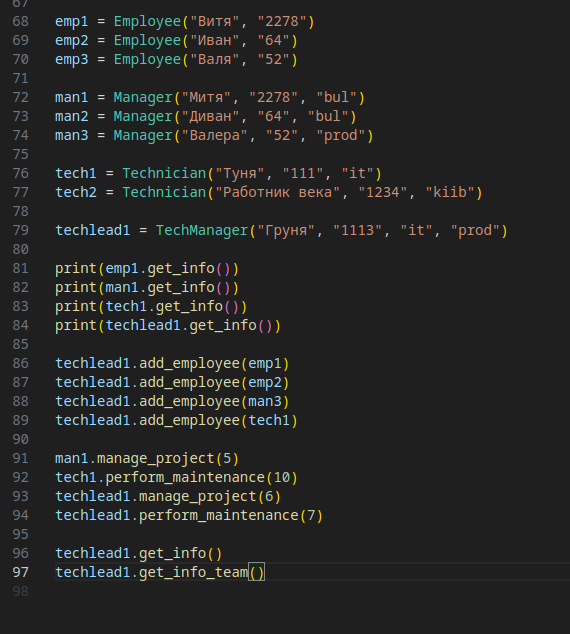


Рис 5

То что всё вывело правильно, убедимся на рисунке 6.

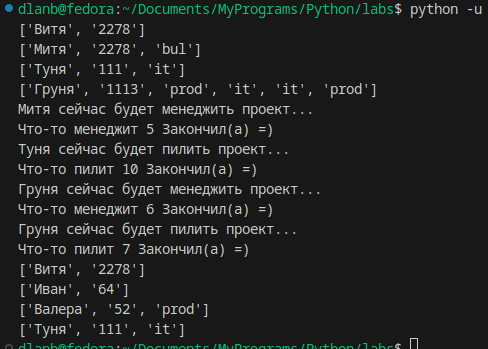


Рис 6

Заключениe:

Мы разработали систему управления сотрудниками, демонстрирующую множественное наследование, инкапсуляцию и полиморфизм в Python. Система должна умеет обрабатывать различные типы сотрудников, включая менеджеров и технических специалистов, а также предоставлять возможность для расширения и добавления новых ролей.

Приложение:

**github с лабами -** [**https://github.com/Bulbanator123/labsmtucivvit**](https://github.com/Bulbanator123/labsmtucivvit)