**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

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

Кафедра «Математическая Кибернетика и Информационные технологии»

Лабораторная работа №7.

Работа с классами ч.3.

Выполнил: Студент группы

БВТ2402

Ларин Максим

Москва

2024

**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.

**Задание:**

1. Создайте класс Employee с общими атрибутами, такими как name (имя), id (идентификационный номер) и методами, например, get\_info(), который возвращает базовую информацию о сотруднике.

2. Создайте класс Manager с дополнительными атрибутами, такими как department (отдел) и методами, например, manage\_project(), символизирующим управление проектами.

3. Создайте класс Technician с уникальными атрибутами, такими как specialization (специализация), и методами, например, perform\_maintenance(), означающим выполнение технического обслуживания.

4. Создайте 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. Добавьте метод add\_employee(), который позволяет TechManager добавлять сотрудников в список подчинённых.

6. Реализуйте метод get\_team\_info(), который выводит информацию о всех подчинённых сотрудниках.

7. Создайте объекты каждого класса и демонстрируйте их функциональность.

**Ход работы:**

class Employee():

    def \_\_init\_\_(self, name, id):

        self.name = name

        self.id = id

    def get\_info(self):

        return ("Имя: {} ID: {}.".format(self.name,self.id))

class Manager(Employee):

    def \_\_init\_\_(self, name="", id="", department=""):

        super().\_\_init\_\_(name, id)

        self.department = department

    def manage\_project(self, pr\_name):

        return ("Менеджер {} работает над проектом {}.".format(self.name, pr\_name))

class Technician(Employee):

    def \_\_init\_\_(self, name, id, specialization=""):

        self.specialization = specialization

        super().\_\_init\_\_(name, id)

    def perform\_maintenance(self):

        return("Техник {} выполняет техническое обслуживание.".format(self.name))

class TechManager(Manager, Technician):

    team\_info=[]

    def \_\_init\_\_(self, name, emp\_id, department, specialization):

        super().\_\_init\_\_(name, emp\_id, department)

        super().\_\_init\_\_(name, emp\_id, specialization)

    @classmethod

    def add\_employee(cls, member):

        cls.member = member

        TechManager.team\_info.append(cls.member)

    @classmethod

    def get\_team\_info(cls):

        print("Наша команда:")

        for member in cls.team\_info:

            print("Имя: {} ID: {}".format(member.name,member.id))

emp1 = Employee("Алиса", 1)

emp2 = Employee("Андрей", 2)

manager = Manager("Максим", 3, "Продажи")

technician = Technician("Ваня", 4, "Электрик")

tech\_manager = TechManager("Егор", 5, "IT-разработка", "Специалист по машинному обучению")

print(emp1.get\_info())

print(emp2.get\_info())

print(manager.get\_info())

print(technician.get\_info())

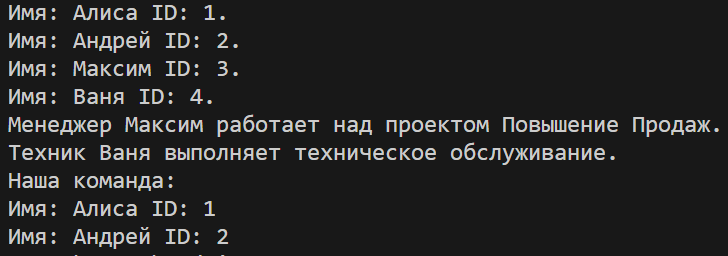
print(manager.manage\_project("Повышение Продаж"))

print(technician.perform\_maintenance())

tech\_manager.add\_employee(emp1)

tech\_manager.add\_employee(emp2)

tech\_manager.get\_team\_info()

****

**Вывод:** я разработал систему управления сотрудниками, демонстрирующую множественное наследование, инкапсуляцию и полиморфизм в Python. Система умеет обрабатывать различные типы сотрудников, включая менеджеров и технических специалистов, а также предоставляет возможность для расширения и добавления новых ролей.