**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

Деминова Виктория

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:**

Код программы 1:

class Employee:  
  
 def \_\_init\_\_(self, name, id):  
 self.name = name  
 self.id = id  
  
 def get\_info(self):  
 return f'Имя: {self.name}, ID: {self.id}'

Код программы 2:

from employee import Employee  
  
  
class Manager(Employee):  
 def \_\_init\_\_(self, name, id, department=''):  
 super().\_\_init\_\_(name, id)  
 self.department = department  
  
 def get\_info(self):  
 return f'{super().get\_info()}, менеджер отдела {self.department}'  
  
 def manage\_project(self):  
 return f'Управляет проектами в отделе {self.department}'

Код программы 3:

from employee import Employee  
  
  
class Technician(Employee):  
 def \_\_init\_\_(self, name, id, specialization):  
 super().\_\_init\_\_(name, id)  
 self.specialization = specialization  
  
 def get\_info(self):  
 return f'{super().get\_info()}, специализация: {self.specialization}'  
  
 def perform\_maintenance(self):  
 return f'Выполняет техническое обслуживание по специализации {self.specialization}'

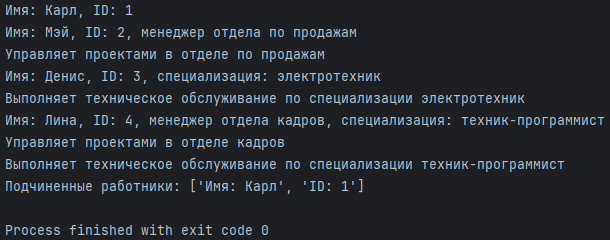
Код программ 4-6:

from manager import Manager  
from technician import Technician  
  
  
class TechManager(Technician, Manager):  
 def \_\_init\_\_(self, name, id, specialization, department):  
 super().\_\_init\_\_(name, id, specialization)  
 self.department = department  
 self.subordinates = []  
  
 def add\_employee(self, employee):  
 self.subordinates.append(employee.get\_info().split(', '))  
  
 def get\_team\_info(self):  
 print('Подчиненные работники:', \*self.subordinates)

Код программы 7:

from employee import Employee  
from manager import Manager  
from technician import Technician  
from techmanager import TechManager  
  
  
ordinary\_employee = Employee('Карл', 1)  
print(ordinary\_employee.get\_info())  
manager\_employee = Manager('Мэй', 2, 'по продажам')  
print(manager\_employee.get\_info())  
print(manager\_employee.manage\_project())  
technician\_employee = Technician('Денис', 3, 'электротехник')  
print(technician\_employee.get\_info())  
print(technician\_employee.perform\_maintenance())  
techmanager\_employee = TechManager('Лина', 4, 'техник-программист', 'кадров')  
print(techmanager\_employee.get\_info())  
print(techmanager\_employee.manage\_project())  
print(techmanager\_employee.perform\_maintenance())  
techmanager\_employee.add\_employee(ordinary\_employee)  
techmanager\_employee.get\_team\_info()

Консоль:



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