МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное учреждение высшего образования

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

Лабораторная работа №7

**Работа с классами ч.3**

**по дисциплине**

**«Введение в информационные технологии»**

Выполнил: студент гр. БВТ2403

Сесицкий И.Д.

Проверил: Кузнецов А В

Москва, 2024 г.

**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.

**Оборудование:** компьютер с установленным языком программирования Python.

**Практика:**

Задание:

class Employee():

    def \_\_init\_\_(self, name, id):

        self.name = name

        self.\_id = id

    def get\_info(self):

*return* f"Name: {self.name}, id: {self.\_id}"

class Manager(Employee):

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

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

        self.department = department

    def manage\_project(self):

*return* f"Менеджер {self.name} управляет проектом в отделе ({self.department})"

class Technician(Employee):

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

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

        self.specialization = specialization

    def perform\_maintenance(self):

*return* f"Техник {self.name} выполняет техническое обслуживание в области ({self.specialization})"

class TechManager(Manager, Technician):

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

        self.name= name

        self.\_id = id

        self.department = department

        self.specialization = specialization

        self.subordinates = []

    def add\_employee(self, \*employees):

        self.subordinates.extend(employees)

    def get\_team\_info(self):

        subordinates\_info = "Список подчинённых:\n"

*for* emp *in* self.subordinates:

            subordinates\_info += emp.get\_info() + '\n'

*return* subordinates\_info

employee = Employee("Иван", 62931)

manager = Manager("Брайан", 72912, "Разработка искусственного интеллекта")

technician = Technician("Артём", 12532, "Кибербезопасность")

techmanager = TechManager("Уильям", 54552, "Логистика", "Веб-разработка")

print(employee.get\_info())

print(manager.get\_info())

print(manager.manage\_project())

print(technician.get\_info())

print(technician.perform\_maintenance())

print(techmanager.get\_info())

print(techmanager.manage\_project())

print(techmanager.perform\_maintenance())

techmanager.add\_employee(employee, manager, technician)

print(techmanager.get\_team\_info())

Результат задания:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

**Вывод:** на данной лабораторной работе мы разработали систему управления сотрудниками, демонстрирующую множественное наследование, инкапсуляцию и полиморфизм в Python.