# **Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»**.

# Лабораторная работа №7:

**‘’Работа с классами ч.3’’**

Выполнил: Безматерных Иван Алексеевич БВТ2402

Москва 2024

**Цель**: 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.

**Код программы:**

class Employee:  
 def \_\_init\_\_(self, name, employee\_id):  
 self.name = name  
 self.employee\_id = employee\_id  
 def get\_info(self):  
 return f'Name: {self.name}, ID: {self.employee\_id}'  
class Manager(Employee):  
 def \_\_init\_\_(self, name='', employee\_id='', department=''):  
 super().\_\_init\_\_(name, employee\_id)  
 self.department = department  
 def manage\_project(self):  
 return f'{self.name} управляет проектом в отделе {self.department}'  
class Technician(Employee):  
 def \_\_init\_\_(self, name='', employee\_id='', specialization=''):  
 super().\_\_init\_\_(name, employee\_id)  
 self.specialization = specialization  
 def perform\_maintenance(self):  
 return f'{self.name} выполняет техническое обслуживание по специальности {self.specialization}'  
class TechManager(Manager, Technician):  
 def \_\_init\_\_(self, name, employee\_id, department, specialization):  
 super().\_\_init\_\_(name, employee\_id, department)  
 super().\_\_init\_\_(name, employee\_id, specialization)  
 self.team = []  
 def add\_employee(self, employee):  
 self.team.append(employee)  
 def get\_team\_info(self):  
 team\_info = "Command:\n"  
 for member in self.team:  
 team\_info += member.get\_info() + '\n'  
 return team\_info  
if \_\_name\_\_ == "\_\_main\_\_":  
 tech\_manager = TechManager("Иван", 1, "ИТ", "Системный администратор")  
 manager = Manager("Катя", 2, "Экономики")  
 technician = Technician("Вова", 3, "Связист")  
  
 tech\_manager.add\_employee(manager)  
 tech\_manager.add\_employee(technician)  
  
 print(tech\_manager.get\_info(), tech\_manager.manage\_project(),\  
 tech\_manager.perform\_maintenance(), tech\_manager.get\_team\_info(), sep="\n")  
  
 print(manager.get\_info(), manager.manage\_project(), sep="\n")  
  
 print(technician.get\_info(), technician.perform\_maintenance(), sep="\n")

**Объяснение кода**:

1. Класс Employee инкапсулирует общие атрибуты и методы для всех сотрудников.

2. Класс Manager добавляет атрибуты и методы для управленцев.

3. Класс Technician добавляет специфические атрибуты и методы для технических специалистов.

4. Класс TechManager наследует как от Manager, так и от Technician, объединяя их функциональности.

5. Методы add\_employee() и get\_team\_info() позволяют управлять подчиненными и получать их сведения.

6. В разделе if \_\_name\_\_ == "\_\_main\_\_": создаются объекты класса и демонстрируется их работа.

**Вывод:** я разработал систему управления сотрудниками, демонстрирующую множественное наследование, инкапсуляцию и полиморфизм в Python.