Лабораторная работа №7

Работа с классами часть 3

Автор:

Голиков Михаил Вячеславович

Группа:

БВТ2402

**Цель лабораторной 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.

**Задачи**

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 выполнения лабораторной работы**

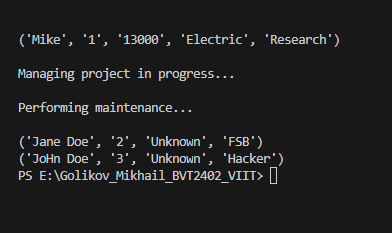
Для передачи данных между родительскими и дочерними классами будем использовать условный json файл со всей необходимой информацией.

|  |
| --- |
| class Employee:     def \_\_init\_\_(self, name, id, salary, \*\*kwargs):         self.name = name         self.id = id         self.salary = salary  ​     def get\_info(self):         return (self.name, self.id, self.salary)    ​  class Manager(Employee):     def \_\_init\_\_(self, department, \*\*kwargs):         super().\_\_init\_\_(\*\*kwargs)         self.department = department       def get\_info(self):         return (\*super().get\_info(), self.department)       def get\_department(self):         return self.department       def manage\_project(self):         return "Managing project in progress..."    ​  class Technician(Employee):     def \_\_init\_\_(self, specialization, \*\*kwargs):         super().\_\_init\_\_(\*\*kwargs)         self.specialization = specialization  ​     def get\_info(self):         return (\*super().get\_info(), self.specialization)       def get\_specialization(self):         return self.specialization       def perform\_maintenance(self):         return "Performing maintenance..."    ​  class TechManager(Manager, Technician):     def \_\_init\_\_(self, \*\*kwargs):         super().\_\_init\_\_(\*\*kwargs)         self.team\_list = []  ​     def get\_info(self):         info = super().get\_info()         return info       def add\_employee(self, employee):         self.team\_list.append(employee)  ​     def get\_team\_info(self):         for employee in self.team\_list:             print(employee.get\_info())           return True  ​  ​  # Example Usage  data = {     'name': 'Mike',     'id': '1',     'salary': '13000',     'department': 'Research',     'specialization': 'Electric'  }  ​  My\_TechManager = TechManager(\*\*data)  ​  ​  print(My\_TechManager.get\_info(), end='\n\n')  ​  print(My\_TechManager.manage\_project(), end='\n\n')  ​  print(My\_TechManager.perform\_maintenance(), end='\n\n')  ​  dat1 = {     'name': 'Jane Doe',     'id': '2',     'salary': 'Unknown',     'department': 'FSB'  }  ​  dat2 = {     'name': 'John Doe',     'id': '3',     'salary': 'Many',     'specialization': 'Hacker'  }  ​  Worker\_1 = Manager(\*\*dat1)  Worker\_2 = Technician(\*\*dat2)  ​  My\_TechManager.add\_employee(Worker\_1)  My\_TechManager.add\_employee(Worker\_2)  ​  My\_TechManager.get\_team\_info()  ​ |

Элемент 1 ― Код программы

|  |
| --- |
| class Employee:     def \_\_init\_\_(self, data):         self.name = data['name']         self.id = data['id']         self.salary = data['salary']         #super().\_\_init\_\_(\*\*kwargs)  ​     def get\_info(self):         return (self.name, self.id, self.salary)    ​  class Manager(Employee):     def \_\_init\_\_(self, data):         super().\_\_init\_\_(data)         self.department = data['department']       def get\_info(self):         return (\*super().get\_info(), self.department)         def get\_department(self):         return self.department       def manage\_project(self):         return "Managing project in progress..."    ​  class Technician(Employee):     def \_\_init\_\_(self, data):         super().\_\_init\_\_(data)         self.specialization = data['specialization']  ​     def get\_info(self):         return (\*super().get\_info(), self.specialization)         def get\_specialization(self):         return self.specialization       def perform\_maintenance(self):         return "Performing maintenance..."    ​  class TechManager(Manager, Technician):  ​     def \_\_init\_\_(self, data):           super().\_\_init\_\_(data)         self.team\_list = []  ​     def get\_info(self):  ​         info = super().get\_info()         return info       def add\_employee(self, employee):         self.team\_list.append(employee)  ​     def get\_team\_info(self):         for employee in self.team\_list:             print(employee.get\_info())           return True  ​  ​  ​  ​  ​  ​  ​  ​  data = {     'name' : 'Mike',     'id' : '1',     'salary' : '13000',     'department':'Research',     'specialization':'Electric'  ​  }  ​  My\_TechManager = TechManager(data)  ​  print('\n')  print(My\_TechManager.get\_info(), end ='\n\n')  ​  print(My\_TechManager.manage\_project(), end ='\n\n')  ​  print(My\_TechManager.perform\_maintenance(), end ='\n\n')  ​  ​  dat1 = {     'name':'Jane Doe',     'id' : '2',     'salary':'Unknown',     'department':'FSB'  }  ​  dat2 = {     'name':'JoHn Doe',     'id' : '3',     'salary':'Unknown',     'specialization':'Hacker'  }  Worker\_1 = Manager(dat1)  ​  Worker\_2 = Technician(dat2)  ​  ​  My\_TechManager.add\_employee(Worker\_1)  My\_TechManager.add\_employee(Worker\_2)  ​  ​  My\_TechManager.get\_team\_info() |

Элемент 2 ― Код программы через “json” файл



Элемент 3 ― Проверка работоспособности

**Заключение**

Была изучена работа с классами и наследование.