МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Факультет «Информационные технологии»

Кафедра «Искусственный интеллект и машинное обучение»

Лабораторная работа №7

Работа с классами часть 3

Автор:

Голиков Михаил Вячеславович

Группа:

БВТ2402

Москва, 2024

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'
}
```

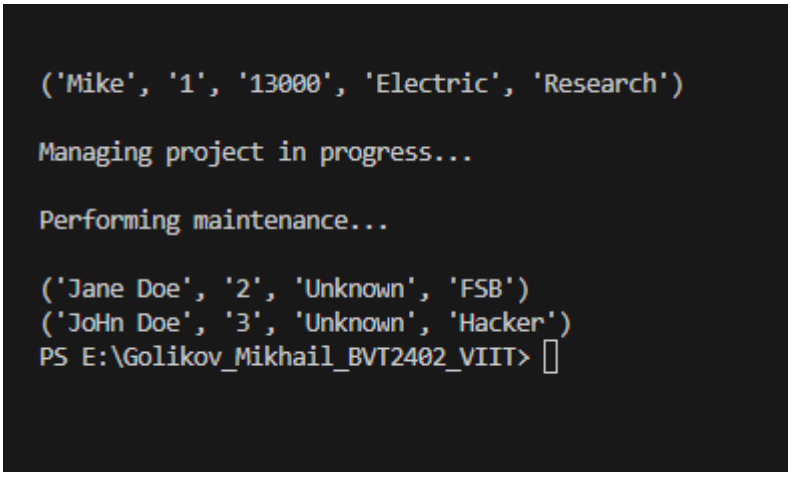
```
dat2 = {
    'name':'JoHn Doe',
    'id' : '3',
    'salary':'Unknown',
    'specialization':'Hacker'
}
Worker_1 = Manager(dat1)

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.