Защищено:
Нардид А.Н.

"__"_____2024 г.

Демонстрация:
Нардид А.Н.

"__"_____2024 г.

Отчет по рубежному контролю № 1 по курсу Парадигмы и конструкции языков программирования ГУИМЦ

Тема работы: "Рубежный контроль представляет собой разработку программы на языке Python."

3
(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-53Б _____
(подпись)

Пронин В.К.

"__"_____2024 г.

Москва, МГТУ - 2024 г.

1. Тема и задание для выполнения рубежного контроля.

Тема работы: Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

1) Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Пример классов данных для предметной области Сотрудник-Отдел:

1. Класс «Сотрудник», содержащий поля:
   - ID записи о сотруднике;
   - Фамилия сотрудника;
   - Зарплата (количественный признак);
   - ID записи об отделе. (для реализации связи один-ко-многим)
2. Класс «Отдел», содержащий поля:
   - ID записи об отделе;
   - Наименование отдела.
3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:
   - ID записи о сотруднике;
   - ID записи об отделе.

2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак, например, «зарплата сотрудника».

Вариант В.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

Варианты предметной области

№ варианта: 30
Класс 1: Факультет
Класс 2: Университет

2. Листинг программы

```
from operator import itemgetter

class Faculty:
    """Факультет"""
    def __init__(self, id, name, salary, uni_id):
        self.id = id
        self.name = name
        self.salary = salary
        self.uni_id = uni_id

class University:
    """Университет"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class FacultyUniversity:
    """Факультеты университетов для связи многие-ко-многим"""
    def __init__(self, faculty_id, university_id):
        self.faculty_id = faculty_id
        self.university_id = university_id

# Университеты
universities = [
    University(1, "МГТУ"),
    University(2, "МГУ"),
    University(3, "НИУ ВШЭ")
]

# Факультеты
faculties = [
    Faculty(1, "ИУ5", 50000, 1),
    Faculty(2, "МТ4", 45000, 1),
    Faculty(3, "ГУИМЦ", 30000, 2),
    Faculty(4, "ИУ8", 60000, 3),
    Faculty(5, "РК9", 35000, 3)
]

# Связь многие-ко-многим
fac_uni = [
    FacultyUniversity(1, 1),
    FacultyUniversity(2, 1),
    FacultyUniversity(3, 2),
    FacultyUniversity(4, 3),
    FacultyUniversity(5, 3)
]

def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(f.name, f.salary, u.name)
                   for u in universities
                   for f in faculties
                   if f.uni_id == u.id]
```

```
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(u.name, fu.university_id, fu.faculty_id)
                         for u in universities
                         for fu in fac_uni
                         if u.id == fu.university_id]

    many_to_many = [(f.name, f.salary, uni_name)
                    for uni_name, uni_id, fac_id in many_to_many_temp
                    for f in faculties if f.id == fac_id]

    # Задание B1: Все факультеты, где название начинается с "И", и их университеты
    print("\nЗадание B1\n")
    res_1 = [(f_name, u_name) for f_name, _, u_name in one_to_many if f_name.startswith("И")]
    print(res_1)

    # Задание B2: Университеты с минимальной зарплатой на каждом факультете, отсортировано по
зарплате
    print("\nЗадание B2\n")
    res_2_unsorted = []
    for u in universities:
        u_faculties = list(filter(lambda x: x[2] == u.name, one_to_many))
        if u_faculties:
            min_salary = min(sal for _, sal, _ in u_faculties)
            res_2_unsorted.append((u.name, min_salary))
    res_2 = sorted(res_2_unsorted, key=itemgetter(1))
    print(res_2)

    # Задание B3: Список всех факультетов и университетов, отсортированный по факультетам
    print("\nЗадание B3\n")
    res_3 = sorted(many_to_many, key=itemgetter(0))
    print(res_3)

if __name__ == "__main__":
    main()
```

3. Результаты работы программы

```
C:\Users\Вячеслав\python\П_и_К_ЯП\РК_1>python RK_1.py

Задание B1

[('ИУ5', 'МГТУ'), ('ИУ8', 'НИУ ВШЭ')]

Задание B2

[('МГУ', 30000), ('НИУ ВШЭ', 35000), ('МГТУ', 45000)]

Задание B3

[('ГУИМЦ', 30000, 'МГУ'), ('ИУ5', 50000, 'МГТУ'), ('ИУ8', 60000, 'НИУ ВШЭ'), ('МТ4', 45000,
'МГТУ'), ('РК9', 35000, 'НИУ ВШЭ')]
```