МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по РК1

Выполнил:
Студент группы ИУ5-36Б
Шах А. М.
Преподаватель:
Гапанюк Ю. Е.

Москва 2025

Условия рубежного контроля №1 по курсу ПиК ЯП

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».

Результатом рубежного контроля является документ в формате PDF, который содержит текст программы и результаты ее выполнения.

Мой вариант запросов – Д

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений*).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.

Мои варианты предметной области – вариант 17 (Дирижер – оркестр)

Код программы

```
class Orchestra:
    #Класс Оркестр
    def __init__(self, id, name):
        self.id = id
        self.name = name

    def __str__(self):
        return f"Оркестр [ID: {self.id}, Название: {self.name}]"

class Conductor:
    #Класс Дирижер (для связи один-ко-многим)
    def __init__(self, id, last_name, salary, orchestra_id):
        self.id = id
        self.last_name = last_name
        self.salary = salary
        self.orchestra_id = orchestra_id

    def __str__(self):
        return f"Дирижер [ID: {self.id}, Фамилия: {self.last_name}, Зарплата: {self.salary}, ID оркестра: {self.orchestra_id}]"

class OrchestraConductor:
    #Класс для связи многие-ко-многим
    def __init__(self, conductor_id, orchestra_id):
        self.conductor_id = conductor_id
        self.orchestra_id = orchestra_id

    def __str__(self):
        return f"ОркестрДирижер [ID дирижера: {self.conductor_id}, ID оркестра: {self.orchestra_id}]"


def create_test_data():

    orchestras = [
        Orchestra(1, "Академический симфонический оркестр"),
        Orchestra(2, "Бостонский симфонический оркестр"),
        Orchestra(3, "Ансамбль старинной музыки"),
        Orchestra(4, "Камерный оркестр"),
        Orchestra(5, "Афинский государственный оркестр")
    ]


    conductors = [
```

```
        Conductor(1, "Иванов", 80000, 1),
        Conductor(2, "Петров", 75000, 1),
        Conductor(3, "Сидоров", 90000, 2),
        Conductor(4, "Кузнецов", 85000, 3),
        Conductor(5, "Николаев", 95000, 3),
        Conductor(6, "Федоров", 70000, 4),
        Conductor(7, "Алексеев", 100000, 5)
    ]


    orchestra_conductors = [
        OrchestraConductor(1, 1),
        OrchestraConductor(2, 1),
        OrchestraConductor(3, 2),
        OrchestraConductor(4, 3),
        OrchestraConductor(5, 3),
        OrchestraConductor(6, 4),
        OrchestraConductor(7, 5),

        OrchestraConductor(1, 3),
        OrchestraConductor(4, 1),
        OrchestraConductor(7, 1)
    ]

    return orchestras, conductors, orchestra_conductors


def query_1(orchestras, conductors):
    print("=== ЗАПРОС 1 ===")
    print("Дирижеры с фамилией на 'ов' и их оркестры:\n")


    result = [
        {
            'conductor': conductor,
            'orchestra': next((orch for orch in orchestras if orch.id == conductor.orchestra_id), None)
        }
        for conductor in conductors
        if conductor.last_name.endswith('ов')
    ]

    for item in result:
        if item['orchestra']:
            print(f"Дирижер: {item['conductor'].last_name}, Оркестр: {item['orchestra'].name}")
```

```
    return result


def query_2(orchestras, conductors):
    print("\n=== ЗАПРОС 2 ===")
    print("Оркестры со средней зарплатой дирижеров (отсортировано по средней зарплате):\n")


    orchestra_stats = {}

    for orchestra in orchestras:

        orch_conductors = [cond for cond in conductors if cond.orchestra_id == orchestra.id]

        if orch_conductors:

            total_salary = sum(cond.salary for cond in orch_conductors)
            conductor_count = len(orch_conductors)
            average_salary = total_salary / conductor_count

            orchestra_stats[orchestra] = {
                'total_salary': total_salary,
                'conductor_count': conductor_count,
                'average_salary': average_salary
            }


    sorted_orchestras = sorted(orchestra_stats.items(),
                               key=lambda x: x[1]['average_salary'])


    for orchestra, stats in sorted_orchestras:
        print(f"Оркестр: {orchestra.name}, "
              f"Средняя зарплата: {stats['average_salary']:.2f}, "
              f"Кол-во дирижеров: {stats['conductor_count']}")

    return sorted_orchestras


def query_3(orchestras, conductors, orchestra_conductors):
    print("\n=== ЗАПРОС 3 ===")
    print("Оркестры с названием на 'A' и их дирижеры:\n")

    a_orchestras = [orch for orch in orchestras if orch.name.startswith('A')]
```

```
    result = []

    for orchestra in a_orchestras:

        conductor_ids = [
            oc.conductor_id for oc in orchestra_conductors
            if oc.orchestra_id == orchestra.id
        ]

        orchestra_conductors_list = [
            cond for cond in conductors
            if cond.id in conductor_ids
        ]

        result.append({
            'orchestra': orchestra,
            'conductors': orchestra_conductors_list
        })


    for item in result:
        print(f"\nОркестр: {item['orchestra'].name}")
        if item['conductors']:
            for conductor in item['conductors']:
                print(f"  - Дирижер: {conductor.last_name}")
        else:
            print("  - Дирижеров нет")

    return result


def main():
    print("РУБЕЖНЫЙ КОНТРОЛЬ - ВАРИАНТ 17")
    print("Предметная область: Дирижер - Оркестр\n")


    orchestras, conductors, orchestra_conductors = create_test_data()

    print("ТЕСТОВЫЕ ДАННЫЕ:")
    print("\nОркестры:")
    for orchestra in orchestras:
        print(f"  {orchestra}")

    print("\nДирижеры:")
```

```
    for conductor in conductors:
        print(f"  {conductor}")

    print("\nСвязи многие-ко-многим:")
    for oc in orchestra_conductors:
        print(f"  {oc}")

    print("\n" + "="*50)


    result1 = query_1(orchestras, conductors)
    result2 = query_2(orchestras, conductors)
    result3 = query_3(orchestras, conductors, orchestra_conductors)

    print("\n" + "="*50)
    print("СТАТИСТИКА ВЫПОЛНЕНИЯ:")
    print(f"Запрос 1: найдено {len(result1)} дирижеров с фамилией на 'ов'")
    print(f"Запрос 2: обработано {len(result2)} оркестров")
    print(f"Запрос 3: найдено {len(result3)} оркестров с названием на 'А'")


def demonstrate_higher_order_functions(orchestras, conductors):
    print("\n" + "="*50)
    print("ДЕМОНСТРАЦИЯ ФУНКЦИЙ ВЫСШЕГО ПОРЯДКА:")

    conductor_names = list(map(lambda c: c.last_name, conductors))
    print(f"\nСписок фамилий дирижеров (map): {conductor_names}")


    high_salary_conductors = list(filter(lambda c: c.salary > 85000, conductors))
    print(f"\nДирижеры с зарплатой > 85000 (filter):")
    for cond in high_salary_conductors:
        print(f"  {cond.last_name}: {cond.salary}")


    from functools import reduce
    total_salary = reduce(lambda x, y: x + y.salary, conductors, 0)
    print(f"\nОбщая зарплата всех дирижеров (reduce): {total_salary}")

if __name__ == "__main__":
    main()


    orchestras, conductors, orchestra_conductors = create_test_data()
    demonstrate_higher_order_functions(orchestras, conductors)
```

Результат

РУБЕЖНЫЙ КОНТРОЛЬ - ВАРИАНТ 17
Предметная область: Дирижер - Оркестр

ТЕСТОВЫЕ ДАННЫЕ:

Оркестры:
  Оркестр [ID: 1, Название: Академический симфонический оркестр]
  Оркестр [ID: 2, Название: Бостонский симфонический оркестр]
  Оркестр [ID: 3, Название: Ансамбль старинной музыки]
  Оркестр [ID: 4, Название: Камерный оркестр]
  Оркестр [ID: 5, Название: Афинский государственный оркестр]

Дирижеры:
  Дирижер [ID: 1, Фамилия: Иванов, Зарплата: 80000, ID оркестра: 1]
  Дирижер [ID: 2, Фамилия: Петров, Зарплата: 75000, ID оркестра: 1]
  Дирижер [ID: 3, Фамилия: Сидоров, Зарплата: 90000, ID оркестра: 2]
  Дирижер [ID: 4, Фамилия: Кузнецов, Зарплата: 85000, ID оркестра: 3]
  Дирижер [ID: 5, Фамилия: Николаев, Зарплата: 95000, ID оркестра: 3]
  Дирижер [ID: 6, Фамилия: Федоров, Зарплата: 70000, ID оркестра: 4]
  Дирижер [ID: 7, Фамилия: Алексеев, Зарплата: 100000, ID оркестра: 5]

Связи многие-ко-многим:
  ОркестрДирижер [ID дирижера: 1, ID оркестра: 1]
  ОркестрДирижер [ID дирижера: 2, ID оркестра: 1]
  ОркестрДирижер [ID дирижера: 3, ID оркестра: 2]
  ОркестрДирижер [ID дирижера: 4, ID оркестра: 3]
  ОркестрДирижер [ID дирижера: 5, ID оркестра: 3]
  ОркестрДирижер [ID дирижера: 6, ID оркестра: 4]
  ОркестрДирижер [ID дирижера: 7, ID оркестра: 5]
  ОркестрДирижер [ID дирижера: 1, ID оркестра: 3]
  ОркестрДирижер [ID дирижера: 4, ID оркестра: 1]
  ОркестрДирижер [ID дирижера: 7, ID оркестра: 1]

==
=== ЗАПРОС 1 ===
Дирижеры с фамилией на 'ов' и их оркестры:

Дирижер: Иванов, Оркестр: Академический симфонический оркестр
Дирижер: Петров, Оркестр: Академический симфонический оркестр
Дирижер: Сидоров, Оркестр: Бостонский симфонический оркестр
Дирижер: Кузнецов, Оркестр: Ансамбль старинной музыки
Дирижер: Федоров, Оркестр: Камерный оркестр

=== ЗАПРОС 2 ===
Оркестры со средней зарплатой дирижеров (отсортировано по средней зарплате):

Оркестр: Камерный оркестр, Средняя зарплата: 70000.00, Кол-во дирижеров: 1
Оркестр: Академический симфонический оркестр, Средняя зарплата: 77500.00, Кол-во дирижеров: 2
Оркестр: Бостонский симфонический оркестр, Средняя зарплата: 90000.00, Кол-во дирижеров: 1
Оркестр: Ансамбль старинной музыки, Средняя зарплата: 90000.00, Кол-во дирижеров: 2
Оркестр: Афинский государственный оркестр, Средняя зарплата: 100000.00, Кол-во дирижеров: 1

=== ЗАПРОС 3 ===
Оркестры с названием на 'А' и их дирижеры:

Оркестр: Академический симфонический оркестр
  - Дирижер: Иванов
  - Дирижер: Петров
  - Дирижер: Кузнецов
  - Дирижер: Алексеев

Оркестр: Ансамбль старинной музыки
  - Дирижер: Иванов
  - Дирижер: Кузнецов
  - Дирижер: Николаев

Оркестр: Афинский государственный оркестр
  - Дирижер: Алексеев

==
СТАТИСТИКА ВЫПОЛНЕНИЯ:
Запрос 1: найдено 5 дирижеров с фамилией на 'ов'
Запрос 2: обработано 5 оркестров
Запрос 3: найдено 3 оркестров с названием на 'А'

==
ДЕМОНСТРАЦИЯ ФУНКЦИЙ ВЫСШЕГО ПОРЯДКА:

Список фамилий дирижеров (map): ['Иванов', 'Петров', 'Сидоров', 'Кузнецов', 'Николаев', 'Федоров', 'Алексеев']

Дирижеры с зарплатой > 85000 (filter):
  Сидоров: 90000
  Николаев: 95000
  Алексеев: 100000

Общая зарплата всех дирижеров (reduce): 595000