

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчёт по рубежному контролю №1

Выполнил:

студент группы РТ5-31Б
Андреев Виктор

Подпись и дата:

Проверил:

преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2021 г

Описание задания

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.

Текст программы

```
# используется для сортировки
from operator import itemgetter
```

```
class Stu:
    """Школьник"""

    def __init__(self, id, fio, avg, cls_id):
        self.id = id
        self.fio = fio
        self.avg = avg # Средний балл
        self.cls_id = cls_id
```

```
class Cls:
    """Класс"""

    def __init__(self, id, name):
        self.id = id
        self.name = name
```

```
class ClsStu:
    """
    'Школьники класса' для реализации
    связи многие-ко-многим
    """

    def __init__(self, cls_id, stu_id):
        self.cls_id = cls_id
        self.stu_id = stu_id
```

```
# Классы
classes = [
    Cls(1, '11А'),
    Cls(2, '11Б (углубленный)'),
    Cls(3, '11В'),
    Cls(4, '11Г (углубленный)'),
    Cls(5, '11Д'),

    Cls(6, '11Е'),
    Cls(7, '11Ж (углубленный)'),
    Cls(8, '11З'),
]
```

```
# Сотрудники
students = [
    Stu(1, 'Алексеев', 4.9, 1),
    Stu(2, 'Андреев', 4.7, 1),
    Stu(3, 'Иванов', 3.0, 1),
    Stu(4, 'Петров', 4.8, 2),
    Stu(5, 'Семенов', 5.0, 2),
    Stu(6, 'Николаев', 4.2, 2),
    Stu(7, 'Фёдоров', 3.2, 3),
    Stu(8, 'Аксаков', 2.8, 3),
    Stu(9, 'Яковлев', 3.9, 4),
    Stu(10, 'Менделеев', 5.0, 4),
    Stu(11, 'Ломоносов', 4.8, 5),
    Stu(12, 'Бауман', 4.9, 5),
```

```

        Stu(13, 'Туполев', 5.0, 5),
        Stu(14, 'Александров', 4.5, 5),
        Stu(15, 'Гагарин', 4.8, 5),
    ]

# Классы и студенты, для связи многие-ко-многим
classes_students = [
    ClsStu(1, 1),
    ClsStu(1, 2),
    ClsStu(1, 3),
    ClsStu(2, 4),
    ClsStu(2, 5),
    ClsStu(2, 6),
    ClsStu(3, 7),
    ClsStu(3, 8),
    ClsStu(4, 9),
    ClsStu(4, 10),
    ClsStu(5, 11),
    ClsStu(5, 12),
    ClsStu(5, 13),
    ClsStu(5, 14),
    ClsStu(5, 15),

    ClsStu(6, 1),
    ClsStu(6, 2),
    ClsStu(6, 3),
    ClsStu(7, 4),
    ClsStu(7, 5),
    ClsStu(7, 6),
    ClsStu(8, 7),
    ClsStu(8, 8),
    ClsStu(8, 9),
]

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

    # Соединение данных один-ко-многим
    one_to_many = [(s.fio, s.avg, c.name)
                    for c in classes
                    for s in students
                    if s.cls_id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.name, cs.cls_id, cs.stu_id)
                           for c in classes
                           for cs in classes_students
                           if c.id == cs.cls_id]

    many_to_many = [(s.fio, s.avg, cls_name)
                     for cls_name, cls_id, stu_id in many_to_many_temp
                     for s in students if s.id == stu_id]

    print('Задание E1')
    task1 = []
    # перебираем всех учеников во всех классах
    for fio, avg, name in one_to_many:
        # если в строке найдена строка "углубленный"
        if name.find("углубленный") != -1:
            task1.append((name, fio))
    print(task1)

```

```

print('\nЗадание E2')
task2_uns = []
# перебираем все классы
for c in classes:
    # все школьники класса
    s_cls = list(filter(lambda i: i[2] == c.name, one_to_many))
    # если в классе больше 0 школьников
    if len(s_cls) > 0:
        # массив средних баллов
        c_avg = [avg for _, avg, _ in s_cls]
        # средний средний балл
        c_avgAvg = sum(c_avg)/len(c_avg)
        task2_uns.append((c.name, round(c_avgAvg, 2)))
task2 = sorted(task2_uns, key=itemgetter(1))
print(task2)

print('\nЗадание E3')
task3 = []
# перебираем всех учеников во всех классах
for fio, avg, name in many_to_many:
    # если строка начинается с "А"
    if fio[0] == "А":
        task3.append((fio, name))
print(task3)

if __name__ == '__main__':
    main()

```

Пример работы программы

```
# используется для сортировки
from operator import itemgetter
```

```
class Stu:
    """Школьник"""
```

```
    def __init__(self, id, fio, avg, cls_id):
        self.id = id
        self.fio = fio
        self.avg = avg # Средний балл
        self.cls_id = cls_id
```

```
class Cls:
    """Класс"""
```

```
    def __init__(self, id, name):
        self.id = id
        self.name = name
```

```
class ClsStu:
```

```
    """
    'Школьники класса' для реализации
    связи многие-ко-многим
    """
```

```
    def __init__(self, cls_id, stu_id):
        self.cls_id = cls_id
        self.stu_id = stu_id
```

```
# Классы
```

```
classes = [
    Cls(1, '11А'),
    Cls(2, '11Б (углубленный)'),
    Cls(3, '11В'),
    Cls(4, '11Г (углубленный)'),
    Cls(5, '11Д'),
    Cls(6, '11Е'),
```

```
Python 3.9.7 (v3.9.7:1016ef3790, Aug 30 2021, 16:39:15)
```

```
[Clang 6.0 (clang-600.0.57)] on darwin
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>>
```

```
= RESTART: /Users/viktorandreev/Desktop/МГТУ им. Н. Э. Баумана/БКИТ/RK1/main.py
```

```
Задание Е1
```

```
[('11Б (углубленный)', 'Петров'), ('11Б (углубленный)', 'Семенов'), ('11Б (углубленный)', 'Николаев'), ('11Г (углубленный)', 'Яковлев'), ('11Г (углубленный)', 'Менделеев')]
```

```
Задание Е2
```

```
[('11В', 3.0), ('11А', 4.2), ('11Г (углубленный)', 4.45), ('11Б (углубленный)', 4.67), ('11Д', 4.8)]
```

```
Задание Е3
```

```
[('Алексеев', '11А'), ('Андреев', '11А'), ('Аксаков', '11В'), ('Александров', '11Д'), ('Алексеев', '11Е'), ('Андреев', '11Е'), ('Аксаков', '11З')]
```

```
>>>
```

Ln: 134 Col: 0

Ln: 14 Col: 4