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
Вариант №1

Выполнил:
студент группы РТ5-31Б
Агеев Алексей
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.

Вариант Е.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате. Средняя зарплата должна быть округлена до 2 знака после запятой (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений; для округления необходимо использовать функцию https://docs.python.org/3/library/functions.html#round*).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.

Предметная область:

№ варианта	Класс 1	Класс 2
1	Студент	Группа

Текст программы

```
# используется для сортировки
from operator import itemgetter

class Group:
    """
    Группа - id, название.
    """
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Student:
    """
    Студент - id, фамилия, id группы, кол-во статей.
    """
    def __init__(self, id, surname, group_id, article_num):
        self.id = id
        self.surname = surname
        self.group_id = group_id
        self.article_num = article_num

class StudentsOfGroup:
    """
    Студенты групы - id студента, id группы
    """
    def __init__(self, student_id, group_id):
        self.student_id = student_id
        self.group_id = group_id

#Создаем фамилии и названия групп
surname = ["Агеев", "Петров", "Иванов", "Гонт", "Гур", "Блур"]
group = ["Группа ИУ", "Группа РК", "РТ"]

#Создаем массивы объектов студент и групп
students = [Student(i, surname[i], (i + 1) % len(group), i % 4) for i in
range(len(surname))]
groups = [Group(i, group[i]) for i in range(len(group))]

#Создаём массив объектов типа студенты группы(многие ко многим)
Students_Of_Group = [StudentsOfGroup(students[i].id, groups[i%len(group)].id) for
i in range(len(surname))]

def main():

    one_to_many = [(stud.surname, stud.article_num, grp.name)
                   for grp in groups
                   for stud in students
                   if stud.group_id == grp.id]

    many_to_many = [(stud.surname, stud.article_num, grp.name)
                   for stofgr in Students_Of_Group
```

```
                    for stud in students
                    for grp in groups
                    if(stud.id == stofgr.student_id and grp.id ==
stofgr.group_id)]



    """
    1 задание
    """
    print("1 задание")
    gr = ""
    for name, num, grp in one_to_many:
        if (grp.find("Группа") != -1):
            if (gr != grp):
                gr = grp
                print(grp)
            if (gr == grp):
                print(name)

    """
    2 задание
    """
    print("2 задание")
    #массив среднего количества статей
    index = 0
    count = 0
    middle_num = 0
    answer2 = []
    gr = ""
    for name, num, grp in one_to_many:
        if (gr != grp):
            if (count != 0):
                answer2.append((gr, (round(middle_num/count, 2))))
            gr = grp
            middle_num = 0
            count = 0
        if (gr == grp):
            middle_num += num
            count += 1
    answer2.append((gr, (round(middle_num/count, 2)))) # так как последний append
не будет произведен

    answer2 = sorted(answer2, key=itemgetter(1))

    print(answer2)

    """
    3 задание
    """
    print("3 задание")
    answer3 = []
    for surname, num, grp in many_to_many:
        # если фамилия начинается с "А"
```

```
        if (surname[0] == "A"):
            answer3.append((surname, grp))
    print(answer3)


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

Пример работы программы