Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования» РК №1

Выполнил:

Студент ИУ5-34Б

Шимко Д. Д.

Подпись и дата:

Проверил:

Преподаватель ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2023 г.

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.

Код:

```
from operator import itemgetter

class Operator:
    """Оператор"""
    def __init__(self, id, name, speed, pl_id):
        self.id = id
        self.name = name
        self.speed = speed
        self.pl_id = pl_id
class Progrlang:
    """Язык программирования"""
    def __init__(self, id, name):
        self.id = id
        self.name = name
class OperatorInProgrlang:
    """
    'Операторы' для реализации
    связи многие-ко-многим
    """
    def __init__(self, pl_id, emp_id):
        self.pl_id = pl_id
        self.operator_id = emp_id

# Языки Программирования
progrlangs = [Progrlang(1, 'Java'), Progrlang(2, 'JS'), Progrlang(3, 'Golang'),
              Progrlang(11, 'Php'), Progrlang(22, 'C++'), Progrlang(33, 'C#'),]
# Операторы
operators = [Operator(1, '+' , 0.1818, 1), Operator(2, 'x++',  0.17839, 2),
           Operator (3, 'SDK',  0.32021, 3), Operator (4, '/=',  0.25001, 3),
Operator(5, '=',  0.0911, 3),]

goslings = [OperatorInProgrlang(1, 1), OperatorInProgrlang(2, 2),
OperatorInProgrlang(3, 3), OperatorInProgrlang(3, 4), OperatorInProgrlang(3, 5),
              OperatorInProgrlang(11, 1), OperatorInProgrlang(22, 2),
OperatorInProgrlang(33, 3), OperatorInProgrlang(33, 4), OperatorInProgrlang(33,
5),]

def main():
    # Соединение данных один-ко-многим
    one_to_many_fq = [(pl.name, operator.name, operator.speed) for pl in
progrlangs for operator in operators if pl.id == operator.pl_id]
    # Соединение данных один-ко-многим
    one_to_many_curr = [(pl.name, dia.pl_id, dia.operator_id) for pl in progrlangs
for dia in goslings if pl.id == dia.pl_id]
    # Соединение данных многие-ко-многим
    many_to_many_ans = [(pl_name, d.name) for pl_name, pl_id, operator_id in
one_to_many_curr for d in operators if d.id == operator_id]

    print("First Question")
    sorted(one_to_many_fq, key=itemgetter(0))
    i = 0
    j = 0
    """Sliding windows"""
    while i < len(one_to_many_fq) and one_to_many_fq[i][0].startswith('J'):
        if i == j:
            print(one_to_many_fq[j][0])
        while j < len(one_to_many_fq) and one_to_many_fq[j][0] ==
one_to_many_fq[i][0]:
            print(one_to_many_fq[j][1] + ' ' + str(one_to_many_fq[j][2]))
            j += 1
        i = j
```

```
    print("Second Question")
    sorted(one_to_many_fq, key=itemgetter(0,2))
    i = 0
    j = 0
    parks_maximus = []
    """Sliding windows"""
    while i < len(one_to_many_fq):
        curr = 0
        while j < len(one_to_many_fq) and one_to_many_fq[j][0] ==
one_to_many_fq[i][0]:
            if one_to_many_fq[j][2] > curr:
                curr = one_to_many_fq[j][2]
            j += 1
        parks_maximus.append((one_to_many_fq[i][0], curr))
        i = j
    for e in parks_maximus:
        print(e)
    print("Third Question")
    sorted(many_to_many_ans, key=itemgetter(0, 1))
    i = 0
    j = 0
    while i < len(many_to_many_ans) and j < len(many_to_many_ans):
        print(many_to_many_ans[i][0])
        while j < len(many_to_many_ans) and many_to_many_ans[j][0] ==
many_to_many_ans[i][0]:
            print('\t' + str(many_to_many_ans[j][1]))
            j += 1
        i = j
if __name__ == '__main__':
    main()
```