МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э.Баумана

Отчет по рубежному контролю №1
по курсу «Разработка интернет-приложений»

Функциональные возможности языка Python

Выполнила:
Светашева Ю.В
ИУ5-54Б

Москва, 2021

Вариант запросов: Г

№ варианта	Класс 1	Класс 2
19	Деталь	Производитель

Вариант Г.

1) «Производитель» и «Деталь» связаны соотношением один-ко-многим. Выведите список всех Производителей, у которых имя начинается с буквы «А», и список производимых ими Деталей.
2) «Производитель» и «Деталь» связаны соотношением один-ко-многим. Выведите список Производителей с максимальной стоимостью Деталей у каждого Производителя, отсортированный по максимальной стоимости.
3) «Производитель» и «Деталь» связаны соотношением многие-ко-многим. Выведите список всех связанных Деталей и Производителей, отсортированный по Производителям, сортировка по Деталям произвольная.

Код программы:

```
# используется для сортировки
from operator import itemgetter


class Manufacturer:
    """Производитель"""

    def __init__(self, id_man, fio):
        self.id_man = id_man
        self.fio = fio



class Detal:
    """Деталь"""

    def __init__(self, id_det, name, cost, id_man):
        self.id_man = id_man
        self.id_det = id_det
        self.name = name
        self.cost = cost


class EmpDep:
    """
    'Детали производителя' для реализации
    связи многие-ко-многим
    """

    def __init__(self, id_det, id_man):
        self.id_det = id_det
        self.id_man = id_man


# Детали
detals = [
    Detal(1, 'шуруп', 50, 1),
```

```
    Detal(2, 'гайка', 70, 2),
    Detal(3, 'круглая деталь', 100, 3),

    Detal(11, 'квадратная деталь', 150, 4),
    Detal(22, 'прямоугольная деталь', 120, 4),
    Detal(33, 'шестеренка', 200, 1),
]

# Производители
manufs = [
    Manufacturer(1, 'Артамонов'),
    Manufacturer(2, 'Петров'),
    Manufacturer(3, 'Иваненко'),
    Manufacturer(4, 'Добролюбов'),
    Manufacturer(5, 'Селеванов'),
]

detals_manuf = [
    EmpDep(1, 1),
    EmpDep(2, 2),
    EmpDep(3, 3),
    EmpDep(3, 4),
    EmpDep(3, 5),

    EmpDep(11, 1),
    EmpDep(22, 2),
    EmpDep(33, 3),
    EmpDep(33, 4),
    EmpDep(33, 5),
]


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

    # Соединение данных один-ко-многим
    one_to_many = [(e.name, e.cost, d.fio)
                   for d in manufs #Производители
                   for e in detals # Детали
                   if e.id_man == d.id_man]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.fio, ed.id_man, ed.id_det)
                         for d in manufs #Производители
                         for ed in detals_manuf #многие ко многим
                         if d.id_man == ed.id_man]

    many_to_many = [(e.name, e.cost, manuf_name)
                    for manuf_name, id_man, id_det in many_to_many_temp
                    for e in detals if e.id_det == id_det]

   #получили список из кортежеей
    print('Задание А1')
    res1={}
    #Перебираем всем производителей
    for d in manufs:
        if (d.fio[0] == "А"):
            # список деталей производителя
            d_emps = list(filter(lambda i: i[2] == d.fio, one_to_many)) #
список деталей производителя
            # Только название детали
            d_emps_name = [x for x, _, _ in d_emps]
            #добавляем результат в словарь
            # ключ - имя производителя, значение - список деталей
```

```
            res1[d.fio]=d_emps_name

    print(res1)

    print('\nЗадание А2')
    res_12_unsorted = []
    # Перебираем все отделы
    for d in manufs:
        # Список деталей производителя
        d_emps = list(filter(lambda i: i[2] == d.fio, many_to_many))
        # Если производитель имеет детали
        if len(d_emps) > 0:
            #  стоимости деталей
            d_sals = [sal for _, sal, _ in d_emps]
            # Максимальняа стоимость детали
            d_sals_max = max(d_sals)
            res_12_unsorted.append((d.fio, d_sals_max))

    # Сортировка по максиальной стоимости
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(res_12)

    print('Задание А3')
    res_11 = sorted(many_to_many, key=itemgetter(2))
    print(res_11)


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

Результат выполнения программы: