Московский государственный технический университет имени Н.Э.Баумана

Факультет РТ Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

Отчет по рубежному контролю № 1 по курсу Базовые компоненты интернет-технологий Вариант 20

Исполнитель

студент группы РТ5-31б _____ Титов А.Д.

“___” _____ 2021 г.

Проверил

Доцент кафедры ИУ5 _____ Гапанюк Ю.Е.

“___” _____ 2021 г.

Москва - 2021

Описание задания

1. Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.
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. **Вариант Е.**
    a. «Поставщик» и «Деталь» связаны соотношением один-ко-многим. Выведите список всех поставщиков, у которых фамилия состоит не менее, чем из 5 букв, и список поставляемых ими деталей.
    b. «Поставщик» и «Деталь» связаны соотношением один-ко-многим. Выведите список поставщиков со средней зарплатой, полученной от стоимости продажи деталей, отсортированный по средней зарплате поставщика
    c. «Поставщик» и «Деталь» связаны соотношением многие-ко-многим. Выведите список всех деталей, у которых название начинается с буквы «М», и фамилии их поставщиков.

Текст программы

```
from functools import *

class Detail:
    """Деталь"""

    def __init__(self, id, name, price, pr_id):
        """
        Args:
            id (int): id детали
            name (str): название детали
            price (int): стоимость детали
            pr_id (int): id поставщика (provider)
        """

        self.id = id
        self.name = name
        self.price = price
        self.pr_id = pr_id

class Provider:
    """Поставщик"""
```

```
        def __init__(self, id, name):
                """
                Args:
                        id (int): id поставщика
                        name (str): имя поставщика
                """
                self.id = id
                self.name = name

class DetPr:
    """
        Поставщики деталей (для реализации связи многие-ко-многим)
    """
    def __init__(self, pr_id, det_id):
        self.det_id = det_id
        self.pr_id = pr_id


# Поставщики
providers = [
    Provider(1, 'Титов'),
    Provider(2, 'Яковенко'),
    Provider(3, 'Бения'),

    Provider(4, 'Смирнов'),
    Provider(5, 'Васильев'),
    Provider(6, 'Агеев')
]

# Детали
details = [
    Detail(1, 'Процессор', 20000, 1),
    Detail(2, 'Материнская плата', 15000, 2),
    Detail(3, 'Кулер', 6000, 1),
    Detail(4, 'Жесткий диск', 9000, 3),
    Detail(5, 'Монитор', 12000, 1)
]

det_prov = [
    DetPr(1,1),
    DetPr(2,2),
    DetPr(3,3),
    DetPr(3,4),
    DetPr(3,5),
```

```
    DetPr(4,1),
    DetPr(5,2),
    DetPr(6,3),
    DetPr(6,4),
    DetPr(6,5)
]

def main():
        # Соединение данных один-ко-многим
        one_to_many = [(detail.name, detail.price, provider.name)
        for detail in details
        for provider in providers
        if detail.pr_id == provider.id]


        # Соединение данных многие-ко-многим
        many_to_many_temp = [(provider.name, dp.pr_id, dp.det_id)
                        for provider in providers
                        for dp in det_prov
                        if provider.id == dp.pr_id]

        many_to_many = [(detail.name, detail.price, provider_name)
                        for provider_name, provider_id, detail_id in
many_to_many_temp
                        for detail in details if detail.id == detail_id]

        # Задание E1
        """
                «Поставщик» и «Деталь» связаны соотношением один-ко-
многим.
                Выведите список всех поставщиков, у которых фамилия состоит
не менее, чем из 5 букв, и список поставляемых ими деталей.
        """

        print("Задание E1")

        res1 = {}

        for item in one_to_many:
                if (len(item[2]) <= 5):
                        if (item[2] in res1):
                                res1[item[2]].append(item[0])
                        else:
                                res1[item[2]] = [item[0]]
```

```
    res1 = list((key, res1[key]) for key in res1.keys())

    print(res1)

        # Задание E2
    """
        «Поставщик» и «Деталь» связаны соотношением один-ко-многим.
        Выведите список поставщиков со средней зарплатой, полученной от стоимости продажи деталей, отсортированный по средней зарплате поставщика.
    """
    print("Задание E2")

    res2 = []

    for provider in providers:
        # список деталей и их стоимость у каждого поставщика
        d_prov = list(filter(lambda i: i[2] == provider.name, one_to_many))

        if (len(d_prov) > 0):
            res2.append((provider.name, round(reduce(lambda x, y: x + y, [x[1] for x in d_prov], 0) / len(d_prov), 2)))

    res2 = sorted(res2, key = lambda x: x[1])

    print(res2)

    """
    «Поставщик» и «Деталь» связаны соотношением многие-ко-многим.
    Выведите список всех деталей, у которых название начинается с буквы «М», и фамилии их поставщиков.
    """
    print("Задание E3")

    res3 = list(filter(lambda x: x[0][0] == "M", many_to_many))

    d = {}

    for item in res3:
        if (item[0] in d):
            d[item[0]].append(item[2])
        else:
            d[item[0]] = [item[2]]
```

```
    res3 = [(key, d[key]) for key in d.keys()]
    print(res3)

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

Примеры выполнения программы

```
Задание E1
[('Титов', ['Процессор', 'Кулер', 'Монитор']), ('Бения', ['Жесткий диск'])]
Задание E2
[('Бения', 9000.0), ('Титов', 12666.67), ('Яковенко', 15000.0)]
Задание E3
[('Материнская плата', ['Яковенко', 'Васильев']), ('Монитор', ['Бения', 'Агеев'])]
```