Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Рубежный контроль №2

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:

студент группы ИУ5-35Б

Титов Е.А.

2021 г.

Описание задания.

(Вариант предметной области - 16, вариант запросов - Д)

1. «Книги» и «Книжный магазин» связаны соотношением один-ко-многим. Выведите список всех книг, у которых название заканчивается на «а», и названия их магазинов.
2. «Книги» и «Книжный магазин» связаны соотношением один-ко-многим. Выведите список магазинов со средней стоимостью книг в каждом магазине, отсортированный по средней стоимости (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений*).
3. «Книги» и «Книжный магазин» связаны соотношением многие-ко-многим. Выведите список всех магазинов, у которых в названии есть слово «книги», и список продаваемых в них книг.
4. Нужно провести рефакторинг текста программы таким образом, чтобы он был пригоден для модульного тестирования.
5. Для текста программы нужно создать модульные тесты с применением TDD - фреймворка (3 теста).

Листинг программы.

Файл «ds»:

```
class Book:
    """Книга"""

    def __init__(self, id, name, price, shop_id):
        self.id = id
        self.name = name
        self.price = price
        self.shop_id = shop_id


class Shop:
    """Книжный магазин"""

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


class BookShop:
    """
    'Книги книжного магазина' для реализации
    связи многие-ко-многим
    """

    def __init__(self, book_id, shop_id):
        self.book_id = book_id
```

```
        self.shop_id = shop_id


# Магазины (ID, название)
shops = [
    Shop(1, 'Азбука'),
    Shop(2, 'Букварики'),
    Shop(3, 'Дом книги'),
    Shop(4, 'Книжный бум'),
    Shop(5, 'Литера'),
    Shop(6, 'Мир книги'),
]

# Книги (ID, название, стоимость, ID магазина)
books = [
    Book(1, 'Властелин колец', 760, 1),
    Book(2, 'Гарри Поттер', 670, 2),
    Book(3, 'Война и мир', 550, 3),
    Book(4, 'Унесённые ветром', 550, 3),
    Book(5, 'Остров сокровищ', 500, 5),
    Book(6, 'Дюна', 560, 6),
    Book(7, 'Граф Монте-Кристо', 610, 4),
    Book(8, 'Анна Каренина', 450, 4),
    Book(9, 'Повелитель мух', 500, 6),
    Book(10, 'Дракула', 600, 1),
]

# одна книга в нескольких магазинах, связь многие-ко-многим
books_shops = [
    BookShop(1, 1),
    BookShop(1, 2),
    BookShop(2, 2),
    BookShop(2, 4),
    BookShop(3, 3),
    BookShop(3, 5),
    BookShop(4, 3),
    BookShop(4, 1),
    BookShop(5, 5),
    BookShop(5, 6),
    BookShop(6, 6),
    BookShop(6, 2),
    BookShop(7, 4),
    BookShop(7, 5),
    BookShop(8, 4),
    BookShop(8, 1),
    BookShop(9, 6),
    BookShop(10, 1),
    BookShop(10, 4),
    BookShop(10, 5),
]
```

Файл «RK1»:

```
# используется для сортировки
from operator import itemgetter
from ds import shops, books, books_shops

class RK1:
    def __init__(self, shops, books, books_shops):
        self.shops = shops
        self.books = books
        self.books_shops = books_shops

        # Соединение данных один-ко-многим
        self.one_to_many_1 = [(b.name, b.price, s.name)
                        for s in shops
                        for b in books
                        if b.shop_id == s.id]

        # Соединение данных многие-ко-многим
        self.many_to_many_temp = [(s.name, ed.shop_id, ed.book_id)
                            for s in shops
                            for ed in books_shops
                            if s.id == ed.shop_id]

        self.many_to_many = [(b.name, b.price, shop_name)
                        for shop_name, shop_id, book_id in self.many_to_many_temp
                        for b in books if b.id == book_id]


    def N1(self):
        print('Задание Д1')
        res_11 = []
        # Перебираем все магазины
        for s in shops:
            # Список книг магазина
            s_books = list(filter(lambda i: i[2] == s.name, self.one_to_many_1))
            # Если магазин не пустой
            if len(s_books) > 0:
                for i in range(len(s_books)):
                    if s_books[i][0][-1] == 'a':
                        res_11.append((s_books[i][0], s.name))
        print(res_11)
        return res_11


    def N2(self):
        #нужно вывести список магазинов со средней стоимостью книг
        print('\nЗадание Д2')
        res_12_unsorted = []
        # Перебираем все магазины
        for s in shops:
```

```
            # Список книг в магазине
            s_books = list(filter(lambda i: i[2]==s.name, self.one_to_many_1))
            # Если магазин не пустой
            if len(s_books) > 0:
                # Стоимость книг в магазине
                s_prices = [price for _,price,_ in s_books]
                # Суммарная стоимость книг в магазине
                s_prices_sum = sum(s_prices)
                #Средняя стоимость книг в магазине
                av_prices = s_prices_sum/len(s_prices)
                res_12_unsorted.append((s.name, av_prices))

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


    def N3(self):
        #Нужно вывести все магазины, в названии которых есть слово "книги", и
список продаваемых в них книг
        print('\nЗадание Д3')
        res_13 = {}
        # Перебираем все магазины
        for s in shops:
            if ('книги' in s.name):
                # Список книг магазина
                s_books = list(filter(lambda i: i[2]==s.name, self.many_to_many))
                # Только название книг
                s_books_names = [x for x,_,_ in s_books]
                # Добавляем результат в словарь
                # ключ - магазин, значение - список названий книг
                res_13[s.name] = s_books_names

        print(res_13)
        return res_13


if __name__ == '__main__':
    rk1 = RK1(shops, books, books_shops)
    rk1.N1()
    rk1.N2()
    rk1.N3()
```

Файл «testRK1»:

```
import unittest
from RK1 import RK1
from ds import books, shops, books_shops
```

```python
res_1 = [
    ('Дракула', 'Азбука'),
    ('Анна Каренина', 'Книжный бум'),
    ('Дюна', 'Мир книги')
]


res_2 = [
    ('Азбука', 680.0),
    ('Букварики', 670.0),
    ('Дом книги', 550.0),
    ('Книжный бум', 530.0),
    ('Мир книги', 530.0),
    ('Литера', 500.0)
]

res_3 = {
    'Дом книги':
        [
            'Война и мир',
            'Унесённые ветром'
        ],
    'Мир книги':
        [
            'Остров сокровищ',
            'Дюна',
            'Повелитель мух'
        ]
}


class MyTestCase(unittest.TestCase):
    def test_N1(self):
        rk = RK1(shops, books, books_shops)
        self.assertEqual(res_1, rk.N1())

    def test_N2(self):
        rk = RK1(shops, books, books_shops)
        self.assertEqual(res_2, rk.N2())

    def test_N3(self):
        rk = RK1(shops, books, books_shops)
        self.assertEqual(res_3, rk.N3())

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

Результаты выполнения.

```
PS C:\Users\egork\OneDrive\Рабочий стол\Учеба\Питон\РК2> & C:/Users/egork/AppData/Local/Programs/Python/Python310/python.exe
"c:/Users/egork/OneDrive/Рабочий стол/Учеба/Питон/РК2/testRK1.py"
Задание Д1
[('Дракула', 'Азбука'), ('Анна Каренина', 'Книжный бум'), ('Дюна', 'Мир книги')]
.
Задание Д2
[('Азбука', 680.0), ('Букварики', 670.0), ('Дом книги', 550.0), ('Книжный бум', 530.0), ('Мир книги', 530.0), ('Литера', 500.
0)]
.
Задание Д3
{'Дом книги': ['Война и мир', 'Унесённые ветром'], 'Мир книги': ['Остров сокровищ', 'Дюна', 'Повелитель мух']}
.
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
PS C:\Users\egork\OneDrive\Рабочий стол\Учеба\Питон\РК2>
```