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

Сыса П.В.

2021 г.

Задание рубежного контроля №2:

Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Задание рубежного контроля № 1:

Вариант Д.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.

2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате (отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений).

3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.

Вариант предметной области 13.

Книга-Библиотека

В соответствии с предметной областью, задание было немного изменено:

3. «Библиотека» и «Книга» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «б», и список работающих в них сотрудников.

Программа:

Файл main.py:

```
# используется для сортировки
from operator import itemgetter

class Book:
    """Книга"""

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

class Lib:
    """Библиотека"""

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

class LibBook:
    """
    'Книги библиотеки' для реализации
    связи многие-ко-многим
    """

    def __init__(self, libID, bookID):
        self.libID = libID
        self.bookID = bookID

# Библиотеки
libs = [
    Lib(1, 'Москвоская библиотека'),
    Lib(2, 'Петербургская библиотека'),
    Lib(3, 'библиотека имени Пушкина'),

    # для связи многие-ко-многим:
    Lib(11, 'Новгородская библиотека'),
    Lib(22, 'библиотека имени Лермонтова'),
    Lib(33, 'библиотека имени Толстого'),
]

# Книги
books = [
    Book(1, 'Война и мир', 1000, 1),
    Book(2, 'Капитанская дочка', 200, 2),
    Book(3, 'Богач, бедняк', 600, 2),
    Book(4, 'Преступление и наказание', 500, 3),
    Book(5, 'Гордость и предубеждение', 400,3),

]

libBooks = [
    LibBook(1, 1),
    LibBook(2, 2),
    LibBook(2, 3),
    LibBook(3, 4),
    LibBook(3, 5),

    LibBook(11, 1),
```

```
    LibBook(22, 2),
    LibBook(22, 3),
    LibBook(33, 4),
    LibBook(33, 5),
]

def one_to_many_relation():
    # Соединение данных один-ко-многим
    one_to_many = [(h.name, h.cost, l.name)
                   for l in libs
                   for h in books
                   if h.LibID == l.id]
    return one_to_many

def many_to_many_relation():
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(l.name, lb.libID, lb.bookID)
                         for l in libs
                         for lb in libBooks
                         if l.id == lb.libID]

    many_to_many = [(b.name, b.cost, bookName)
                    for bookName, streetID, bookID in many_to_many_temp
                    for b in books if b.id == bookID]
    return many_to_many

def func1(one_to_many):
    res1 = list(filter(lambda x: x[0].endswith("ие"), one_to_many))
    return res1

def func2(one_to_many):
    res2unsorted = []
    # Перебираем все библиотеки
    for l in libs:
        # Список книг в библиотеке
        bookss = list(filter(lambda i: i[2] == l.name, one_to_many))
        # Если на улице есть дома
        if len(bookss) > 0:
            # Все цены книг в библиотеке
            allCosts = [sal for _, sal, _ in bookss]
            # Средняя цена книги в библиотеке
            averageCosts = round(sum(allCosts) / len(allCosts), 2)
            res2unsorted.append((l.name, averageCosts))

    # Сортировка по средней стоимости
    res2 = sorted(res2unsorted, key=itemgetter(1), reverse=True)
    return res2

def func3(many_to_many):
    res3 = {}
    for l in libs:
        if l.name.startswith("б"):
            # Список книг в библиотеке
            bookss = list(filter(lambda i: i[2] == l.name, many_to_many))
            # Только имя книг
            booksNames = [x for x, _, _ in bookss]
            # Добавляем результат в словарь
            # ключ - бибилиотека, значение - список названий книг
            res3[l.name] = booksNames
    return res3


if __name__ == '__main__':
    one_to_many = one_to_many_relation()
```

```
    many_to_many = many_to_many_relation()
    print('Задание D1\n{}'.format(func1(one_to_many)))
    print('Задание D2\n{}'.format(func2(one_to_many)))
    print('Задание D3\n{}'.format(func3(many_to_many)))
```

Файл tdd.py:

```
import unittest
import main

class TestBooksAndLibraries(unittest.TestCase):

    def test_task1(self):
        relations = main.one_to_many_relation()
        expected_result = [('Преступление и наказание', 500, 'библиотека имени
Пушкина'),
                           ('Гордость и предубеждение', 400, 'библиотека имени
Пушкина')]
        self.assertEqual(main.func1(relations), expected_result)
    def test_task2(self):
        relations = main.one_to_many_relation()
        expected_result = [('Москвоская библиотека', 1000.0), ('библиотека
имени Пушкина', 450.0), ('Петербургская библиотека', 400.0)]
        self.assertEqual(main.func2(relations), expected_result)
    def test_task3(self):
        relations = main.many_to_many_relation()
        expected_result = {'библиотека имени Пушкина': ['Преступление и
наказание', 'Гордость и предубеждение'],
                           'библиотека имени Лермонтова': ['Капитанская
дочка', 'Богач, бедняк'],
                           'библиотека имени Толстого': ['Преступление и
наказание', 'Гордость и предубеждение']}

        self.assertEqual(main.func3(relations), expected_result)

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

Результаты выполнения программы:

Файл main.py:

```
Задание D1
[('Преступление и наказание', 500, 'библиотека имени Пушкина'), ('Гордость и предубеждение', 400, 'библиотека имени Пушкина')]
Задание D2
[('Москвоская библиотека', 1000.0), ('библиотека имени Пушкина', 450.0), ('Петербургская библиотека', 400.0)]
Задание D3
{'библиотека имени Пушкина': ['Преступление и наказание', 'Гордость и предубеждение'], 'библиотека имени Лермонтова': ['Капитанская дочка', 'Богач, бедняк'],

Process finished with exit code 0
```

```
'библиотека имени Толстого': ['Преступление и наказание', 'Гордость и предубеждение']}
```

Файл tdd.py:

```
Testing started at 22:21 ...

Launching unittests with arguments python -m unittest


Ran 3 tests in 0.002s

OK

Process finished with exit code 0
```