Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»
Отчёт по рубежному контролю №2

Выполнил:
студент группы ИУ5-31Б
Лобанов Дмитрий
Сергеевич

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись:_____

Дата:_____

Москва, 2021 г.

Вариант предметной области – 13 (“Книга” и “Библиотека”)
Вариант запросов – А

Описание задания

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).

Текст программы

main.py

```
# используется для сортировки
from operator import itemgetter


class Book:
    """Книга"""

    def __init__(self, id, name, len, lib_id):
        self.id = id
        self.name = name
        self.len = len  # длина в страницах
        self.lib_id = lib_id


class Lib:
    """Библиотека"""

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


class BookLib:
    """
    'Книга в библиотеке' для реализации
    связи многие-ко-многим
    """

    def __init__(self, book_id, lib_id):
        self.book_id = book_id
        self.lib_id = lib_id


# Библиотеки
libs = [
    Lib(1, 'российская государственная библиотека'),
    Lib(2, 'научная библиотека МГТУ им.Баумана'),
    Lib(3, 'научная библиотека МГУ'),
```

```
    Lib(4, 'центральная молодежная библиотека')
]

# Книги
books = [
    Book(1, 'Война и мир', 1225, 1),
    Book(2, 'Горе от ума', 145, 1),
    Book(3, 'Прикладные информационные технологии', 334, 2),
    Book(4, 'Радио и связь', 543, 2),
    Book(5, 'Телекоммуникационные сети', 192, 2),
    Book(6, 'Первая научная история войны 1812 года', 896, 3),
    Book(7, 'Компьютерные сети', 907, 3),
    Book(8, 'Гарри Поттер и философский камень', 223, 4),
    Book(9, 'Бэтмен. Убийственная шутка', 72, 4),
]

books_libs = [
    BookLib(1, 1),
    BookLib(2, 1),
    BookLib(3, 2),
    BookLib(4, 2),
    BookLib(5, 2),
    BookLib(6, 3),
    BookLib(7, 3),
]


def sorting_by_name(table):
    return sorted(table, key=itemgetter(2))


def sorting_by_sum_of_books(table, libs):
    res_12_unsorted = []
    # Перебираем все библиотеки
    for l in libs:
        # Список книг библиотеки
        l_books = list(filter(lambda i: i[2] == l.name, table))
        # Если библиотека не пустая
        if len(l_books) > 0:
            # Длины всех книг
            l_lens = [len for _, len, _ in l_books]
            # Суммарная зарплата сотрудников отдела
            l_lens_sum = sum(l_lens)
            res_12_unsorted.append((l.name, l_lens_sum))

    # Сортировка по суммарной зарплате
    return sorted(res_12_unsorted, key=itemgetter(1), reverse=True)


def output_books_of_libs_with_NAUCHNAYA(table, libs):
    result = {}
    # Перебираем все библиотеки
    for l in libs:
        if 'научная' in l.name:
            # Список книг библиотеки
            l_books = list(filter(lambda i: i[2] == l.name, table))
            # Только ФИО сотрудников
            l_books_names = [x for x, _, _ in l_books]
            # Добавляем результат в словарь
            # ключ - отдел, значение - список фамилий
            result[l.name] = l_books_names
    return result
```

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

    # Соединение данных один-ко-многим
    one_to_many = [(b.name, b.len, l.name)
                   for l in libs
                   for b in books
                   if b.lib_id == l.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(l.name, bl.lib_id, bl.book_id)
                         for l in libs
                         for bl in books_libs
                         if l.id == bl.lib_id]

    many to many = [(b.name, b.len, lib name)
                    for lib_name, lib_id, book_id in many_to_many_temp
                    for b in books if b.id == book_id]

    print('Задание А1')
    print(sorting_by_name(one_to_many))

    print('\nЗадание А2')
    print(sorting_by_sum_of_books(one_to_many, libs))

    print('\nЗадание А3')
    print(output_books_of_libs_with_NAUCHNAYA(many_to_many, libs))


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

tests.py

```
from main import Book, Lib, BookLib, sorting_by_name, sorting_by_sum_of_books,
output_books_of_libs_with_NAUCHNAYA
import unittest


class Tests(unittest.TestCase):
    def setUp(self):
        # Библиотеки
        self.libs = [
            Lib(1, 'российская государственная библиотека'),
            Lib(2, 'научная библиотека МГТУ им.Баумана'),
            Lib(3, 'научная библиотека МГУ'),
            Lib(4, 'центральная молодежная библиотека')
        ]

        # Книги
        self.books = [
            Book(1, 'Война и мир', 1225, 1),
            Book(2, 'Горе от ума', 145, 1),
            Book(3, 'Прикладные информационные технологии', 334, 2),
            Book(4, 'Радио и связь', 543, 2),
            Book(5, 'Телекоммуникационные сети', 192, 2),
            Book(6, 'Первая научная история войны 1812 года', 896, 3),
            Book(7, 'Компьютерные сети', 907, 3),
            Book(8, 'Гарри Поттер и философский камень', 223, 4),
            Book(9, 'Бэтмен. Убийственная шутка', 72, 4),
        ]

        self.books_libs = [
            BookLib(1, 1),
            BookLib(2, 1),
```

```
            BookLib(3, 2),
            BookLib(4, 2),
            BookLib(5, 2),
            BookLib(6, 3),
            BookLib(7, 3),
        ]
        # Соединение данных один-ко-многим
        self.one_to_many = [(b.name, b.len, l.name)
                            for l in self.libs
                            for b in self.books
                            if b.lib_id == l.id]

        # Соединение данных многие-ко-многим
        self.many_to_many_temp = [(l.name, bl.lib_id, bl.book_id)
                                  for l in self.libs
                                  for bl in self.books libs
                                  if l.id == bl.lib_id]

        self.many_to_many = [(b.name, b.len, lib_name)
                             for lib_name, lib_id, book_id in self.many_to_many_temp
                             for b in self.books if b.id == book_id]

    def test_sorting_by_name(self):
        result = sorting_by_name(self.one_to_many)
        desired_result = [('Прикладные информационные технологии', 334, 'научная
библиотека МГТУ им.Баумана'),
                          ('Радио и связь', 543, 'научная библиотека МГТУ
им.Баумана'),
                          ('Телекоммуникационные сети', 192, 'научная библиотека МГТУ
им.Баумана'),
                          ('Первая научная история войны 1812 года', 896, 'научная
библиотека МГУ'),
                          ('Компьютерные сети', 907, 'научная библиотека МГУ'),
                          ('Война и мир', 1225, 'российская государственная
библиотека'),
                          ('Горе от ума', 145, 'российская государственная
библиотека'),
                          ('Гарри Поттер и философский камень', 223, 'центральная
молодежная библиотека'),
                          ('Бэтмен. Убийственная шутка', 72, 'центральная молодежная
библиотека')]
        self.assertEqual(result, desired_result)

    def test_sorting_by_sum(self):
        result = sorting_by_sum_of_books(self.one_to_many, self.libs)
        desired_result = [('научная библиотека МГУ', 1803), ('российская
государственная библиотека', 1370),
                          ('научная библиотека МГТУ им.Баумана', 1069), ('центральная
молодежная библиотека', 295)]
        self.assertEqual(result, desired_result)

    def test_output_NAUCHNAYA(self):
        result = output_books_of_libs_with_NAUCHNAYA(self.many_to_many, self.libs)
        desired_result = {
            'научная библиотека МГТУ им.Баумана': ['Прикладные информационные
технологии', 'Радио и связь',
                                                   'Телекоммуникационные сети'],
            'научная библиотека МГУ': ['Первая научная история войны 1812 года',
'Компьютерные сети']}
        self.assertEqual(result, desired_result)
```

Результат выполнения программы