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»
Отчет по рубежному контролю №2

Выполнил:
студент группы ИУ5-31Б
Прошкин Георгий Павлович

Проверил:
преподаватель каф.ИУ5
Гапанюк Юрий Евгеньевич

Москва, 2021 г.

Задание

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

```
from operator import itemgetter


class Book:
    """Книга"""

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


class Shop:
    """Магазин"""

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


class BookShop:
    """
    'Книга в магазине' для реализации
    связи многие-ко-многим
    """

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


# Магазины
shops = [
    Shop(1, 'центральный магазин'),
    Shop(2, 'магазин знаний'),
    Shop(3, 'дом книги'),
]

# Книги
books = [
    Book(1, 'Преступление и наказание', 500, 1),
    Book(2, 'На дне', 700, 2),
    Book(3, 'Мцыри', 450, 3),
    Book(4, 'Преступление и наказание', 480, 2),
```

```
    Book(5, 'На дне', 710, 3),
    Book(6, 'Мцыри', 465, 1),
    Book(7, 'Преступление и наказание', 480, 3),
    Book(8, 'На дне', 690, 1),
    Book(9, 'Мцыри', 445, 2),
]

books_shops = [
    BookShop(1, 1),
    BookShop(1, 6),
    BookShop(1, 8),
    BookShop(2, 2),
    BookShop(2, 4),
    BookShop(2, 9),
    BookShop(3, 3),
    BookShop(3, 5),
    BookShop(3, 7),
]


def sortbyname(info):
    return sorted(info, key=itemgetter(2))


def sortbyprice(info, shops):
    res_12_unsorted = []
    # Перебираем все магазины
    for s in shops:
        # Список книг в магазине
        s_books = list(filter(lambda i: i[2] == s.name, info))
        # Если магазин не пустой
        if len(s_books) > 0:
            # Цены книг в магазине
            s_price = [price for _, price, _ in s_books]
            # Средняя цена книги в магазине
            s_price_sum_sr = sum(s_price) // len(s_books)
            res_12_unsorted.append((s.name, s_price_sum_sr))

    # Сортировка по средней цене
    return sorted(res_12_unsorted, key=itemgetter(1))


def output(info, shops):
    res_13 = {}
    # Перебираем все магазины
    for s in shops:
        if 'магазин' in s.name:
            # Список книг магазина
            s_books = list(filter(lambda i: i[2] == s.name, info))
            # 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
    return res_13


def main():
    one_to_many = [(b.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
```

```
    many_to_many_temp = [(s.name, b.shop_id, b.book_id)
                         for s in shops
                         for b in books_shops
                         if s.id == b.shop_id]

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

    print('Задание A1')
    res_11 = sortbyname(one_to_many)
    print(res_11)

    print('\nЗадание A2')
    res_12 = sortbyprice(one_to_many, shops)
    print(res_12)

    print('\nЗадание A3')
    res_13 = output(many_to_many, shops)
    print(res_13)


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

tests.py

```
from main import Book, Shop, BookShop, sortbyname, sortbyprice, output
import unittest


class Tests(unittest.TestCase):
    def setUp(self) -> None:
        # Магазины
        self.shops = [
            Shop(1, 'центральный магазин'),
            Shop(2, 'магазин знаний'),
            Shop(3, 'дом книги'),
        ]
        # Книги
        self.books = [
            Book(1, 'Преступление и наказание', 500, 1),
            Book(2, 'На дне', 700, 2),
            Book(3, 'Мцыри', 450, 3),
            Book(4, 'Преступление и наказание', 480, 2),
            Book(5, 'На дне', 710, 3),
            Book(6, 'Мцыри', 465, 1),
            Book(7, 'Преступление и наказание', 480, 3),
            Book(8, 'На дне', 690, 1),
            Book(9, 'Мцыри', 445, 2),
        ]

        self.books_shops = [
            BookShop(1, 1),
            BookShop(1, 6),
            BookShop(1, 8),
            BookShop(2, 2),
            BookShop(2, 4),
            BookShop(2, 9),
            BookShop(3, 3),
            BookShop(3, 5),
            BookShop(3, 7),
```

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

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

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


    def testsortbyname(self):
        result = sortbyname(self.one_to_many)
        desiredresult = [('Мцыри', 450, 'дом книги'), ('На дне', 710, 'дом
книги'), ('Преступление и наказание', 480, 'дом книги'), ('На дне', 700,
'магазин знаний'), ('Преступление и наказание', 480, 'магазин знаний'),
('Мцыри', 445, 'магазин знаний'), ('Преступление и наказание', 500,
'центральный магазин'), ('Мцыри', 465, 'центральный магазин'), ('На дне',
690, 'центральный магазин')]
        self.assertEqual(desiredresult, result)


    def testsortbyprice(self):
        result = sortbyprice(self.one_to_many, self.shops)
        desiredresult = [('магазин знаний', 541), ('дом книги', 546),
('центральный магазин', 551)]
        self.assertEqual(desiredresult, result)


    def testoutput(self):
        result = output(self.many_to_many, self.shops)
        desiredresult = {'центральный магазин': ['Преступление и наказание',
'Мцыри', 'На дне'], 'магазин знаний': ['На дне', 'Преступление и наказание',
'Мцыри']}
        self.assertEqual(desiredresult, result)
```

Пример выполнения программы

```
Testing started at 13:54 ...
Launching unittests with arguments python -m unittest tests.Tests in C:\Users\Admin\Desktop\БКИТ\RK_2



Ran 3 tests in 0.005s

OK

Process finished with exit code 0
```