Отчет по рубежному контролю №2 по дисциплине “Парадигмы и конструкции языков программирования”

1. Исходный код

   main.py

```python
from operator import itemgetter
from statistics import mean


class Book:
    """Класс Книга"""
    def __init__(self, id, title, price, bookstore_id):
        self.id = id
        self.title = title
        self.price = price
        self.bookstore_id = bookstore_id


class Bookstore:
    """Класс Книжный магазин"""
    def __init__(self, id, name):
        self.id = id
        self.name = name


class BookBookstore:
    """Связь Книги и Книжного магазина для реализации многие-ко-многим"""
    def __init__(self, bookstore_id, book_id):
        self.bookstore_id = bookstore_id
```

```python
        self.book_id = book_id


def get_one_to_many(books, bookstores):
    """Связь один-ко-многим между книгами и магазинами."""
    return [(b.title, b.price, s.name)
            for s in bookstores
            for b in books
            if b.bookstore_id == s.id]


def get_avg_price_per_store(books, bookstores):
    """Средняя цена книг по каждому магазину."""
    one_to_many = get_one_to_many(books, bookstores)
    res_2_unsorted = []
    for s in bookstores:
        s_books = list(filter(lambda i: i[2] == s.name, one_to_many))
        if len(s_books) > 0:
            s_prices = [price for _, price, _ in s_books]
            avg_price = mean(s_prices)
            res_2_unsorted.append((s.name, avg_price))
    return sorted(res_2_unsorted, key=itemgetter(1))


def get_books_in_a_stores(books, bookstores):
    """Список магазинов, название которых начинается с 'А', и их книги."""
    one_to_many = get_one_to_many(books, bookstores)
    res_3 = {}
```

```
    for s in bookstores:
        if s.name.startswith("A"):
            s_books = list(filter(lambda i: i[2] == s.name, one_to_many))
            s_books_titles = [x for x, _, _ in s_books]
            res_3[s.name] = s_books_titles
    return res_3


# Тестовые данные
bookstores = [
    Bookstore(1, "Академическая литература"),
    Bookstore(2, "Книжный мир"),
    Bookstore(3, "Азбука знаний")
]


books = [
    Book(1, "Философия науки", 500, 1),
    Book(2, "Математика для всех", 400, 1),
    Book(3, "История России", 700, 2),
    Book(4, "Алгебра и анализ", 450, 2),
    Book(5, "Биология", 550, 3)
]


# Основная функция
if __name__ == "__main__":
    print("Запрос 1: Список книг и магазинов (связь один-ко-многим):")
```

```
    print(get_one_to_many(books, bookstores))



    print("\nЗапрос 2: Средняя цена книг в каждом магазине,
отсортированная по средней цене:")
    print(get_avg_price_per_store(books, bookstores))



    print("\nЗапрос 3: Магазины, название которых начинается с 'А', и
список книг в них:")
    print(get_books_in_a_stores(books, bookstores))
```

tests.py

```
import unittest

from refactor_tests import Book, Bookstore, get_one_to_many,
get_avg_price_per_store, get_books_in_a_stores



class TestRefactoredCode(unittest.TestCase):

    def setUp(self):

        self.bookstores = [

            Bookstore(1, "Академическая литература"),

            Bookstore(2, "Книжный мир"),

            Bookstore(3, "Азбука знаний"),

            Bookstore(4, "Библиотека мира"),

        ]
```

```
        self.books = [
            Book(1, "Философия науки", 500, 1),
            Book(2, "Математика для всех", 400, 1),
            Book(3, "История России", 700, 2),
            Book(4, "Алгебра и анализ", 450, 2),
            Book(5, "Биология", 550, 3),
        ]


    def test_one_to_many_relationship(self):
        expected = [
            ("Философия науки", 500, "Академическая литература"),
            ("Математика для всех", 400, "Академическая литература"),
            ("История России", 700, "Книжный мир"),
            ("Алгебра и анализ", 450, "Книжный мир"),
            ("Биология", 550, "Азбука знаний"),
        ]
        result = get_one_to_many(self.books, self.bookstores)
        self.assertEqual(result, expected)


    def test_avg_price_per_store(self):
        expected = [
            ("Академическая литература", 450.0),
            ("Азбука знаний", 550.0),
            ("Книжный мир", 575.0),
        ]
```

```
        result = get_avg_price_per_store(self.books, self.bookstores)
        self.assertEqual(result, expected)


    def test_books_in_a_stores(self):
        expected = {
            "Академическая литература": ["Философия науки", "Математика
для всех"],
            "Азбука знаний": ["Биология"],
        }
        result = get_books_in_a_stores(self.books, self.bookstores)
        self.assertEqual(result, expected)


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

2. Вывод

main.py:
Запрос 1: Список книг и магазинов (связь один-ко-многим):

[('Философия науки', 500, 'Академическая литература'), ('Математика для всех', 400, 'Академическая литература'), ('История России', 700, 'Книжный мир'), ('Алгебра и анализ', 450, 'Книжный мир'), ('Биология', 550, 'Азбука знаний')]

Запрос 2: Средняя цена книг в каждом магазине, отсортированная по средней цене:

[('Академическая литература', 450), ('Азбука знаний', 550), ('Книжный мир', 575)]

Запрос 3: Магазины, название которых начинается с 'А', и список книг в них:

{'Академическая литература': ['Философия науки', 'Математика для всех'], 'Азбука знаний': ['Биология']}

tests.py

...

Ran 3 tests in 0.000s

OK