МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Рубежный контроль

«2»

Выполнила:
Студент группы ИУ5-36Б
Топорец Г.А.

Проверил:
Гапанюк Ю.Е.

Москва 2025

Задание:

Рефакторинг: необходимо переработать текст программы рубежного контроля №1 так, чтобы он стал пригоден для модульного тестирования.

Разработка тестов: для программы рубежного контроля №1 нужно создать модульные тесты с использованием TDD-фреймворка в количестве 3 штук.

Код:

main.py:

```
import json
from operator import itemgetter

class Chapter:
    def __init__(self, id, title, pages, book_id):
        self.id = id
        self.title = title
        self.pages = pages
        self.book_id = book_id

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

class ChapterBook:
    def __init__(self, book_id, chapter_id):
        self.book_id = book_id
        self.chapter_id = chapter_id

# Логика запросов, вынесенная для тестирования
def query_g1(books, chapters):
    """Список книг на 'А' и их глав (один-ко-многим)"""
    one_to_many = [(c.title, b.name) for b in books for c in chapters if c.book_id == b.id]
    return [(b_name, c_title) for c_title, b_name in one_to_many if b_name.startswith('A')]

def query_g2(books, chapters):
```

```
    """Книги и макс. страницы в их главах (один-ко-многим)"""
    res = []
    for b in books:
        b_chapters_pages = [c.pages for c in chapters if c.book_id == b.id]
        if b_chapters_pages:
            res.append((b.name, max(b_chapters_pages)))
    return sorted(res, key=itemgetter(1), reverse=True)


def query_g3(books, chapters, chapters_books):
    """Связи многие-ко-многим, сортировка по книгам"""
    many_to_many_temp = [(b.name, cb.chapter_id) for b in books for cb in chapters_books if
b.id == cb.book_id]
    res = [(c.title, book_name) for book_name, c_id in many_to_many_temp for c in chapters
if c.id == c_id]
    return sorted(res, key=itemgetter(1))


def main():
    pass


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

test_main.py:

```
import unittest
from main import Book, Chapter, ChapterBook, query_g1, query_g2, query_g3


class TestLibraryQueries(unittest.TestCase):
    def setUp(self):
        # Инициализация тестовых данных
        self.books = [
            Book(1, 'Алгоритмы'),
            Book(2, 'Базы данных')
        ]
        self.chapters = [
            Chapter(1, 'Введение', 10, 1),
            Chapter(2, 'Сложность', 50, 1),
            Chapter(3, 'SQL', 30, 2)
        ]
        self.chapters_books = [
            ChapterBook(1, 1),
            ChapterBook(2, 3)
        ]

    def test_query_g1(self):
        """Тест запроса №1: Книги на букву 'А'"""
        result = query_g1(self.books, self.chapters)
        self.assertEqual(len(result), 2)
        self.assertEqual(result[0][0], 'Алгоритмы')

    def test_query_g2(self):
        """Тест запроса №2: Максимальные страницы"""
        result = query_g2(self.books, self.chapters)
        # У 'Алгоритмов' макс страницы 50, у 'Баз данных' 30
```

```
        self.assertEqual(result[0], ('Алгоритмы', 50))
        self.assertEqual(result[1], ('Базы данных', 30))

    def test_query_g3(self):
        """Тест запроса №3: Многие-ко-многим"""
        result = query_g3(self.books, self.chapters, self.chapters_books)
        self.assertEqual(result[0][1], 'Алгоритмы')
        self.assertEqual(result[1][1], 'Базы данных')

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

Data.json:


```
{
 "books": [
  { "id": 1, "name": "Алгоритмы" },
  { "id": 2, "name": "Архитектура ЭВМ" },
  { "id": 3, "name": "Анализ данных" },
  { "id": 11, "name": "Атлас систем (доп)" },
  { "id": 22, "name": "Базы данных" },
  { "id": 33, "name": "Веб-разработка" }
 ],
 "chapters": [
  { "id": 1, "title": "Введение", "pages": 15, "book_id": 1 },
  { "id": 2, "title": "Сортировка", "pages": 45, "book_id": 1 },
  { "id": 3, "title": "Процессоры", "pages": 60, "book_id": 2 },
  { "id": 4, "title": "Память", "pages": 35, "book_id": 2 },
  { "id": 5, "title": "Регрессия", "pages": 50, "book_id": 3 }
 ],
 "chapters_books": [
  { "book_id": 1, "chapter_id": 1 },
  { "book_id": 1, "chapter_id": 2 },
  { "book_id": 2, "chapter_id": 3 },
  { "book_id": 2, "chapter_id": 4 },
  { "book_id": 3, "chapter_id": 5 },
  { "book_id": 11, "chapter_id": 1 },
  { "book_id": 22, "chapter_id": 2 },
  { "book_id": 33, "chapter_id": 5 }
 ]
}
```

Случай если, что то не так в тестировании:

```
('Алгоритмы', 15) != ('Алгоритмы', 50)

Expected :('Алгоритмы', 50)
Actual   :('Алгоритмы', 15)
<Click to see difference>

Traceback (most recent call last):
  File "C:\Users\grigo\IdeaProjects\untitled\test_main.py", line 31, in test_query_g2
    self.assertEqual(result[0], ('Алгоритмы', 15))
    ~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: Tuples differ: ('Алгоритмы', 50) != ('Алгоритмы', 15)

First differing element 1:
50
15

- ('Алгоритмы', 50)
?                -

+ ('Алгоритмы', 15)
?               +



Ran 3 tests in 0.006s

FAILED (failures=1)
```

Если все хорошо

```
Ran 3 tests in 0.002s

OK

Process finished with exit code 0
```