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

«1»

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

Проверил:
Гапанюк Ю.Е.

Москва 2025

Задание:

Цель работы: Разработка программы на языке Python, демонстрирующей владение структурами данных и способами их обработки (запросы, сортировки, агрегация).

1. Предметная область (Вариант №24)

В соответствии с таблицей вариантов выбраны следующие классы:

Класс 1: Глава.

Класс 2: Книга.

2. Задачи и требования к реализации

Проектирование классов: Создать классы «Глава» и «Книга», связанные отношениями «один-ко-многим» и «многие-ко-многим».

Поля данных:

В класс «Глава» (сторона «много») ввести количественный признак — количество страниц.

В классы включить необходимые ID для реализации связей.

Тестовые данные: Заполнить списки объектов тестовыми данными (3–5 записей), обеспечив целостность связей по идентификаторам.

Технологии: Использовать возможности Python: list comprehensions и функции высших порядков.

3. Запросы (Вариант Г)

Запрос №1: «Книга» и «Глава» связаны соотношением один-ко-многим. Вывести список всех книг, у которых название начинается с буквы «А», и список входящих в них глав.

Запрос №2: «Книга» и «Глава» связаны соотношением один-ко-многим. Вывести список книг с максимальным количеством страниц в их главах, отсортированный по этому максимальному значению.

Запрос №3: «Книга» и «Глава» связаны соотношением многие-ко-многим. Вывести список всех связанных глав и книг, отсортированный по книгам (сортировка по главам произвольная).

Код:

```
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

def load_data(filename):
    try:
        with open(filename, 'r', encoding='utf-8') as f:
            data = json.load(f)
```

```
        books = [Book(b['id'], b['name']) for b in data['books']]
        chapters = [Chapter(c['id'], c['title'], c['pages'], c['book_id'])
                    for c in data['chapters']]
        chapters_books = [ChapterBook(cb['book_id'], cb['chapter_id'])
                          for cb in data['chapters_books']]

        return books, chapters, chapters_books
    except FileNotFoundError:
        print(f"Ошибка: Файл {filename} не найден.")
        return [], [], []

def main():
    # Загружаем данные
    books, chapters, chapters_books = load_data('data.json')

    if not books:
        return

    # 1. Соединение данных один-ко-многим [cite: 103]
    one_to_many = [(c.title, c.pages, b.name)
                   for b in books
                   for c in chapters
                   if c.book_id == b.id]

    # 2. Соединение данных многие-ко-многим [cite: 108, 109, 113, 114]
```

```
many_to_many_temp = [(b.name, cb.book_id, cb.chapter_id)
                     for b in books
                     for cb in chapters_books
                     if b.id == cb.book_id]

many_to_many = [(c.title, c.pages, book_name)
                for book_name, book_id, chapter_id in many_to_many_temp
                for c in chapters if c.id == chapter_id]

print('Задание Г1')
res_1 = [(b_name, c_title)
         for c_title, _, b_name in one_to_many
         if b_name.startswith('A')]
print(res_1)

print('\nЗадание Г2')
res_2_unsorted = []
for b in books:
    b_chapters = list(filter(lambda i: i[2] == b.name, one_to_many))
    if len(b_chapters) > 0:
        b_pages = [pages for _, pages, _ in b_chapters]
        res_2_unsorted.append((b.name, max(b_pages)))

res_2 = sorted(res_2_unsorted, key=itemgetter(1), reverse=True)
print(res_2)
```

```
    print('\nЗадание Г3')
    res_3 = sorted(many_to_many, key=itemgetter(2))
    print(res_3)

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

Тестировка:

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 }
 ]
}
```

```
Задание Г1
[('Алгоритмы', 'Введение'), ('Алгоритмы', 'Сортировка'), ('Архитектура ЭВМ', 'Процессоры'), ('Архитектура ЭВМ', 'Память'), ('Анализ данных', 'Регрессия')]

Задание Г2
[('Архитектура ЭВМ', 60), ('Анализ данных', 50), ('Алгоритмы', 45)]

Задание Г3
[('Введение', 15, 'Алгоритмы'), ('Сортировка', 45, 'Алгоритмы'), ('Регрессия', 50, 'Анализ данных'), ('Процессоры', 60, 'Архитектура ЭВМ'), ('Память', 35,

Process finished with exit code 0
```