Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

Курс «Парадигмы и конструкции языков программирования»

Отчёт по РК №2

Подготовила:
Студентка группы ИУ5Ц-54Б
Тихонова Д.Д. 20.11.2023
(подпись, дата)

Проверил:
Гапанюк Ю.Е.
(подпись, дата)

МГТУ им. Н.Э. Баумана

Москва, 2023 г.

Вариант запросов РК№1- Г.

Предметная область – «Глава-Книга».

1. «Книга» и «Глава» связаны соотношением один-ко-многим. Выведите список всех книг, у которых название начинается с буквы «П», и список их глав.

2. «Книга» и «Глава» связаны соотношением один-ко-многим. Выведите список книг с максимальным количеством знаков в каждой книге, отсортированный по максимальному количеству знаков.

3. «Книга» и «Глава» связаны соотношением многие-ко-многим. Выведите список всех связанных глав и книг, отсортированный по книгам, сортировка по главам произвольная.

Задание РК№2

1) Провести рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создать модульные тесты с применением TDD - фреймворка (3 теста).

Листинг программы.

1) **main.py:**

```
# main.py


class Chapter:
    """Глава"""


    def __init__(self, id, title, book_id, num_of_chars):
        self.id = id
        self.title = title
        self.book_id = book_id
        self.num_of_chars = num_of_chars



class Book:
    """Книга"""
```

```
    def __init__(self, id, title, author):
        self.id = id
        self.title = title
        self.author = author


class BookChapter:
    """Книги и главы"""

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


def get_one_to_many_relationship(books, chapters):
    return [(c.title, b.title) for b in books for c in chapters if c.book_id ==
b.id]


def get_many_to_many_relationship(book_chapters, books, chapters):
    return [(b.title, c.title) for bc in book_chapters for b in books for c in
chapters
            if bc.book_id == b.id and bc.chapter_id == c.id]


def task_G1(books, one_to_many):
    result = []
    for book in books:
        if book.title.startswith('П'):
            chapters_list = [c for c, b in one_to_many if b == book.title]
            result.append((book.title, chapters_list))
    return result


def task_G2(books, chapters):
    book_max_chars = [(b.title, max(c.num_of_chars for c in chapters if c.book_id ==
b.id)) for b in books]
    sorted_books = sorted(book_max_chars, key=lambda x: x[1], reverse=True)
```

```
    result = [(book_title, max_chars) for book_title, max_chars in sorted_books]
    return result


def task_G3(many_to_many):
    return many_to_many


def main():
    books = [
        Book(1, 'Война и мир', 'Лев Толстой'),
        Book(2, 'Преступление и наказание', 'Федор Достоевский'),
        Book(3, 'Мастер и Маргарита', 'Михаил Булгаков'),
    ]

    chapters = [
        Chapter(1, 'Часть первая. Война', 1, 100),
        Chapter(2, 'Часть вторая. Мир', 1, 120),
        Chapter(3, 'Часть первая. Преступление', 2, 80),
        # Добавьте другие главы по необходимости
    ]

    book_chapters = [
        BookChapter(1, 1),
        BookChapter(1, 2),
        BookChapter(2, 3),
        # Добавьте другие связи между книгами и главами по необходимости
    ]

    # Соединение данных один-ко-многим
    one_to_many = get_one_to_many_relationship(books, chapters)

    # Соединение данных многие-ко-многим
    many_to_many = get_many_to_many_relationship(book_chapters, books, chapters)

    # Задание Г1
    task_G1_result = task_G1(books, one_to_many)
```

```
    print('Задание Г1 - выводит книги, у которых название начинается с буквы "П" и список их глав.')
    for result in task_G1_result:
        print(f'{result[0]}: {result[1]}')


    # Задание Г2
    task_G2_result = task_G2(books, chapters)
    print('\nЗадание Г2 - выводит книги с максимальным количеством знаков в каждой книге, отсортированные по максимальному количеству знаков.')
    for result in task_G2_result:
        print(f'{result[0]}: {result[1]}')


    # Задание Г3
    task_G3_result = task_G3(many_to_many)
    print('\nЗадание Г3 - выводит список связанных глав и книг, отсортированный по книгам.')
    for result in task_G3_result:
        print(result)


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

2) tddtests.py:

```
import unittest
from main import get_one_to_many_relationship, get_many_to_many_relationship, task_G1, task_G2, task_G3, Book, Chapter, BookChapter


class TestBookChapterFunctions(unittest.TestCase):


    def setUp(self):
        # Тестовые данные
        self.books = [
            Book(1, 'Война и мир', 'Лев Толстой'),
            Book(2, 'Преступление и наказание', 'Федор Достоевский'),
            Book(3, 'Мастер и Маргарита', 'Михаил Булгаков'),
```

```
        ]

        self.chapters = [
            Chapter(1, 'Часть первая. Война', 1, 100),
            Chapter(2, 'Часть вторая. Мир', 1, 120),
            Chapter(3, 'Часть первая. Преступление', 2, 80),
            # Добавьте другие главы по необходимости
        ]

        self.book_chapters = [
            BookChapter(1, 1),
            BookChapter(1, 2),
            BookChapter(2, 3),]

    def test_get_one_to_many_relationship(self):
        one_to_many = get_one_to_many_relationship(self.books, self.chapters)
        self.assertEqual(len(one_to_many), 2)

    def test_get_many_to_many_relationship(self):
        many_to_many = get_many_to_many_relationship(self.book_chapters,
self.books, self.chapters)
        self.assertEqual(len(many_to_many), 2)

    def test_task_G1(self):
        one_to_many = get_one_to_many_relationship(self.books, self.chapters)
        task_G1_result = task_G1(self.books, one_to_many)
        self.assertEqual(task_G1_result, 'Ожидаемый результат')  # Заменить
на фактический ожидаемый результат

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