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»

Отчёт по РК 2

Выполнил:

студент группы ИУ5-31Б

Шимолина Полина Кирилловна

Подпись:_____

Дата:_____

Проверил:

преподаватель каф. ИУ5

Гапанюк Юрий Евгеньевич

Подпись:_____

Дата:_____

Москва, 2021 г.

Задание:

Рубежный контроль представляет собой разработку тестов на языке Python.

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 Chapter:
    """Глава"""

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


class Book:
    """Книга"""

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


class ChapterBook:
    """
    'Глава книги' для реализации
    связи многие-ко-многим
    """

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


# Книги
books = [
    Book(1, 'Понедельник начинается в субботу'),
    Book(2, 'Горе от ума'),
    Book(3, 'Убить пересмешника (на языке оригинала)'),
    Book(11, 'Повелитель мух (на языке оригинала'),
    Book(22, '1984 (на языке оригинала)'),
    Book(33, 'Пикник на обочине'),
]
# Главы
chapters = [
    Chapter(1, 'Глава 1', 11, 1),
    Chapter(2, 'Глава 2', 35, 2),
    Chapter(3, 'Глава 3', 40, 3),
    Chapter(4, 'Глава 4', 30, 3),
    Chapter(5, 'Глава 5', 15, 3),
```

```
]
chapters_books = [
    ChapterBook(1, 1),
    ChapterBook(2, 2),
    ChapterBook(3, 3),
    ChapterBook(3, 4),
    ChapterBook(3, 5),
    ChapterBook(11, 1),
    ChapterBook(22, 2),
    ChapterBook(33, 3),
    ChapterBook(33, 4),
    ChapterBook(33, 5),
]


def task1(l):
    return sorted(l, key=itemgetter(2))


def task2(l):
    res_2_unsorted = []
    for b in books:
        b_chapters = list(filter(lambda i: i[2] == b.title, l))
        if len(b_chapters) > 0:
            b_pages = [pages for _, pages, _ in b_chapters]
            b_pages_sum = sum(b_pages)
            res_2_unsorted.append((b.title, b_pages_sum))

    return sorted(res_2_unsorted, key=itemgetter(1), reverse=True)


def task3(k):
    res = {}
    for b in books:
        if 'оригинала' in b.title:
            b_chapters = list(filter(lambda i: i[2] == b.title, k))
            b_chapters_titles = [x for x, _, _ in b_chapters]
            res[b.title] = b_chapters_titles

    return res


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

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(b.title, 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_title)
                    for book_title, book_id, chapter_id in many_to_many_temp
                    for c in chapters if c.id == chapter_id]

    print('Задание А1')
    print(task1(one_to_many))

    print('\nЗадание А2')
    print(task2(one_to_many))
```

```
    print('\nЗадание А3')
    print(task3(many_to_many))


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

test.py

```
from main import ChapterBook, Book, Chapter, task1, task2, task3
from unittest import TestCase


class Test(TestCase):
    def setUp(self) -> None:
        self.books = [
            Book(1, 'Понедельник начинается в субботу'),
            Book(2, 'Горе от ума'),
            Book(3, 'Убить пересмешника (на языке оригинала)'),
            Book(11, 'Повелитель мух (на языке оригинала'),
            Book(22, '1984 (на языке оригинала)'),
            Book(33, 'Пикник на обочине'),
        ]
        self.chapters = [
            Chapter(1, 'Глава 1', 11, 1),
            Chapter(2, 'Глава 2', 35, 2),
            Chapter(3, 'Глава 3', 40, 3),
            Chapter(4, 'Глава 4', 30, 3),
            Chapter(5, 'Глава 5', 15, 3),
        ]
        self.chapters_books = [
            ChapterBook(1, 1),
            ChapterBook(2, 2),
            ChapterBook(3, 3),
            ChapterBook(3, 4),
            ChapterBook(3, 5),
            ChapterBook(11, 1),
            ChapterBook(22, 2),
            ChapterBook(33, 3),
            ChapterBook(33, 4),
            ChapterBook(33, 5),
        ]
        self.one_to_many = [(c.title, c.pages, b.title)
                            for b in self.books
                            for c in self.chapters
                            if c.book_id == b.id]

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

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

    def test1(self):
        result = task1(self.one_to_many)
        desired = [('Глава 2', 35, 'Горе от ума'), ('Глава 1', 11,
'Понедельник начинается в субботу'),
                   ('Глава 3', 40, 'Убить пересмешника (на языке
оригинала)'),
                   ('Глава 4', 30, 'Убить пересмешника (на языке
```

```
оригинала)'),
                    ('Глава 5', 15, 'Убить пересмешника (на языке
оригинала)')]
        self.assertEqual(result, desired)

    def test2(self):
        result = task2(self.one_to_many)
        desired = [('Убить пересмешника (на языке оригинала)', 85), ('Горе от
ума', 35),
                    ('Понедельник начинается в субботу', 11)]
        self.assertEqual(result, desired)

    def test3(self):
        result = task3(self.many_to_many)
        desired = {'Убить пересмешника (на языке оригинала)': ['Глава 3',
'Глава 4', 'Глава 5'],
                    'Повелитель мух (на языке оригинала': ['Глава 1'], '1984
(на языке оригинала)': ['Глава 2']}
        self.assertEqual(result, desired)
```

Результаты работы программы:

```
Testing started at 17:58 ...
Launching unittests with arguments



Ran 3 tests in 0.005s

OK

Process finished with exit code 0
```