ИУ5-32Б,

Яковлев Сергей

Рубежный контроль 2

Условия:

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

```
from operator import itemgetter



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



class Chapter:
    def __init__(self, id, name, length, book_chap_id):
        self.id = id
        self.name = name
        self.length = length
        self.book_chap_id = book_chap_id



class BookChap:
    def __init__(self, book_chap_id, ch_id):
        self.book_chap_id = book_chap_id
        self.ch_id = ch_id
```

```
books = [
    Book(1, "All Quiet on the Western Front"),
    Book(2, "Luftwaffe aces"),
    Book(3, "Achtung Panzer"),
]

chapters = [
    Chapter(1, "Chapter 11", 43, 1),
    Chapter(2, "Chapter 24", 45, 2),
    Chapter(3, "Chapter 32", 61, 3),
    Chapter(4, "Chapter 47", 38, 2),
    Chapter(5, "Chapter 52", 47, 2),
    Chapter(6, "Chapter 27", 42, 1)
]

book_chap = [
    BookChap(1, 1),
    BookChap(2, 2),
    BookChap(3, 3),
    BookChap(3, 4),
    BookChap(1, 5),
]



def task1(ch_list):
    result = sorted(ch_list, key=itemgetter(0))
    return result



def task2(ch_list):
    result = []
    temp = dict()
    for i in ch_list:
        if i[2] in temp:
            temp[i[2]] += 1
        else:
            temp[i[2]] = 1
    for i in temp.keys():
        result.append((i, temp[i]))

    result.sort(key=itemgetter(1), reverse=True)
    return result



def task3(op_list, end_ch):
    result = [(i[0], i[1]) for i in op_list if i[0].endswith(end_ch)]
    return result

one_to_many = [(ch.name, ch.length, bk.name)
                for bk in books
                for ch in chapters
                if ch.book_chap_id == bk.id]

many_to_many_temp = [(bk.name, bc.book_chap_id, bc.ch_id)
            for bk in books
            for bc in book_chap
            if bc.book_chap_id == bk.id]

many_to_many = [(ch.name, ch.length, bk_name)
            for bk_name, bk_id, ch_id in many_to_many_temp
            for ch in chapters if ch.id == ch_id]
def main():
    print("Задание Б1\n")
    print(task1(one_to_many))

    print("Задание Б2\n")
    print(task2(one_to_many))

    print("Задание Б3")
    print(many_to_many)
    print(task3(many_to_many, '7'))


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

Тесты

```
import unittest
from operator import itemgetter
from main import Book, Chapter, BookChap, task1, task2, task3

class TestBookChapterFunctions(unittest.TestCase):
    def setUp(self):
        self.books = [
            Book(1, "All Quiet on the Western Front"),
            Book(2, "Luftwaffe aces"),
            Book(3, "Achtung Panzer"),
        ]

        self.chapters = [
            Chapter(1, "Chapter 11", 43, 1),
            Chapter(2, "Chapter 24", 45, 2),
            Chapter(3, "Chapter 32", 61, 3),
            Chapter(4, "Chapter 47", 38, 2),
            Chapter(5, "Chapter 52", 47, 2),
            Chapter(6, "Chapter 27", 42, 1)
        ]

        self.book_chap = [
            BookChap(1, 1),
            BookChap(2, 2),
            BookChap(3, 3),
            BookChap(3, 4),
            BookChap(1, 5),
        ]

        self.one_to_many = [(ch.name, ch.length, bk.name)
                            for bk in self.books
                            for ch in self.chapters
                            if ch.book_chap_id == bk.id]

    def test_task1(self):
        expected = sorted(self.one_to_many, key=itemgetter(0))
        result = task1(self.one_to_many)
        self.assertEqual(result, expected)

    def test_task2(self):
        expected = [("Luftwaffe aces", 3),
                    ("All Quiet on the Western Front", 2),
                    ("Achtung Panzer", 1)]
        result = task2(self.one_to_many)
        self.assertEqual(result, expected)

    def test_task3(self):
        many_to_many_temp = [(bk.name, bc.book_chap_id, bc.ch_id)
                             for bk in self.books
```

```
                        for bc in self.book_chap
                        if bc.book_chap_id == bk.id]

        many_to_many = [(ch.name, ch.length, bk_name)
                    for bk_name, bk_id, ch_id in many_to_many_temp
                    for ch in self.chapters if ch.id == ch_id]

        expected = [("Chapter 47", 38)]  # Указывает на главы, которые действительно есть
        result = task3(many_to_many, '7')
        self.assertEqual(result, expected)

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