Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Рубежный контроль 2

Вариант Б 16

Выполнил:

студент группы

ИУ5-32Б

Насруллаев Арсен

Подпись и дата:

_____

_____

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Подпись и дата:

_____

_____

Москва, 2023 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).

Код программы

Программа после рефакторинга:

```
class Book:
    def init(self, book_id, title, author):
        self.book_id = book_id
        self.title = title
        self.author = author

class Bookstore:
    def init(self, store_id, name, books):
        self.store_id = store_id
        self.name = name
        self.books = books

def get_bookstore_books(bookstores):
    bookstore_books = [(bookstore.name, book.title) for bookstore in bookstores for book in bookstore.books]
    return sorted(bookstore_books, key=lambda x: x[1])

def get_bookstore_books_count(bookstores):
    bookstore_books_count = [(bookstore.name, len(bookstore.books)) for bookstore in bookstores]
    return sorted(bookstore_books_count, key=lambda x: x[1], reverse=True)

def get_books_with_i(bookstores):
    books_with_i = [book for book in books if book.title.endswith("и")]
    bookstores_with_i_books = [(bookstore.name, book.title) for bookstore in bookstores for book in bookstore.books if book in books_with_i]
    return bookstores_with_i_books

# Создание списков объектов классов с тестовыми данными
books = [
    Book(1, "Война и мир", "Лев Толстой"),
    Book(2, "Преступление и наказание", "Федор Достоевский"),
    Book(3, "Евгений Онегин", "Александр Пушкин"),
    Book(4, "Мастер и Маргарита", "Михаил Булгаков"),
    Book(5, "Идиот", "Федор Достоевский"),
    Book(6, "Отцы и дети", "Иван Тургенев"),
    Book(7, "Герой нашего времени", "Михаил Лермонтов")
]

bookstores = [
    Bookstore(1, "Магазин книг 'Читай-город'", [books[0], books[1], books[2], books[5]]),
    Bookstore(2, "Книжный мир", [books[3], books[4], books[5]]),
    Bookstore(3, "Буквоед", [books[0], books[1], books[4], books[6]])
]

# Запросы
print("Задание Б1")
bookstore_books_sorted = get_bookstore_books(bookstores)
print("Список всех связанных книг и книжных магазинов, отсортированный по книгам:")
for item in bookstore_books_sorted:
    print(f"Книга 'item[1]' в магазине 'item[0]'")

print("Задание Б2")
bookstore_books_count_sorted = get_bookstore_books_count(bookstores)
print("Список книжных магазинов с количеством книг в каждом магазине:")
for item in bookstore_books_count_sorted:
    print(f"Магазин 'item[0]': item[1] книг")

print("Задание Б3")
bookstores_with_i_books = get_books_with_i(bookstores)
print("Задание Б3")
bookstores_with_i_books = get_books_with_i(bookstores)
print("Список всех книг, у которых название заканчивается на 'и', и книжные магазины, в которых они находятся:")
for item in bookstores_with_i_books:
    print(f"Книга 'item[1]' в магазине 'item[0]'")
```

ТЕСТЫ

```
import unittest

class TestBookstore(unittest.TestCase):
    def setUp(self):
        self.books = [
            Book(1, "Война и мир", "Лев Толстой"),
            Book(2, "Преступление и наказание", "Федор Достоевский"),
            Book(3, "Евгений Онегин", "Александр Пушкин"),
            Book(4, "Мастер и Маргарита", "Михаил Булгаков"),
            Book(5, "Идиот", "Федор Достоевский"),
            Book(6, "Отцы и дети", "Иван Тургенев"),
            Book(7, "Герой нашего времени", "Михаил Лермонтов")
        ]

        self.bookstores = [
            Bookstore(1, "Магазин книг 'Читай-город'", [self.books[0], self.books[1], self.books[2], self.books[5]]),
            Bookstore(2, "Книжный мир", [self.books[3], self.books[4], self.books[5]]),
            Bookstore(3, "Буквоед", [self.books[0], self.books[1], self.books[4], self.books[6]])
        ]

    def test_get_bookstore_books(self):
        expected_output = [
            ("Буквоед", "Война и мир"),
            ("Магазин книг 'Читай-город'", "Война и мир"),
            ("Буквоед", "Герой нашего времени"),
            ("Книжный мир", "Идиот"),
            ("Магазин книг 'Читай-город'", "Отцы и дети"),
            ("Буквоед", "Отцы и дети"),
            ("Магазин книг 'Читай-город'", "Евгений Онегин"),
            ("Магазин книг 'Читай-город'", "Преступление и наказание"),
            ("Буквоед", "Преступление и наказание"),
            ("Книжный мир", "Мастер и Маргарита"),
            ("Книжный мир", "Идиот")
        ]
        self.assertEqual(get_bookstore_books(self.bookstores), expected_output)

    def test_get_bookstore_books_count(self):
        expected_output = [
            ("Буквоед", 4),
            ("Магазин книг 'Читай-город'", 4),
            ("Книжный мир", 3)
        ]
        self.assertEqual(get_bookstore_books_count(self.bookstores), expected_output)

    def test_get_books_with_i(self):
        expected_output = [
            ("Магазин книг 'Читай-город'", "Евгений Онегин"),
            ("Буквоед", "Война и мир"),
            ("Буквоед", "Герой нашего времени"),
            ("Буквоед", "Преступление и наказание")
        ]
        self.assertEqual(get_books_with_i(self.bookstores), expected_output)

if name == 'main':
    unittest.main()
```