ИУ5-32Б
Плюшко Дмитрий

Рубежный контроль №2

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

RK2.py

```
from operator import itemgetter

class BookStore:
    def __init__(self, ID, NAME):
        self.id = ID
        self.name = NAME


class Book:
    def __init__(self, ID, NAME, IDSTORE):
        self.id = ID
        self.name = NAME
        self.idStore = IDSTORE


class MBookStore:
    def __init__(self, IDBOOK, IDSTORE):
        self.idBook = IDBOOK
        self.idStore = IDSTORE


books = [
    Book(1, "Вечный кошмар", 1),
    Book(2, "Забудь меня", 1),
    Book(3, "Перекрёсток судеб", 1),
    Book(4, "Любовь не вечна", 2),
    Book(5, "Два желания", 3),
    Book(6, "Мир лжецов", 3),
    Book(7, "Северные ветра", 1),
    Book(8, "Последняя остановка", 3),
    Book(9, "Множество лесов", 2)
]
```

```
stores = [
    BookStore(1, "Читай-Город"),
    BookStore(2, "Лабиринт"),
    BookStore(3, "Букбридж"),
]

mbs = [
    MBookStore(1, 1),
    MBookStore(2, 1),
    MBookStore(3, 1),
    MBookStore(4, 2),
    MBookStore(5, 3),
    MBookStore(6, 3),
    MBookStore(7, 1),
    MBookStore(8, 3),
    MBookStore(9, 2),
]


def get_one_to_many(books, stores):
    return [(a.name, b.name)
            for a in books
            for b in stores
            if a.idStore == b.id]


def get_many_to_many(books, mbs):
    many_to_many_temp = [(a.name, b.idBook, b.idStore)
                         for a in books
                         for b in mbs
                         if a.id == b.idBook]
    return [(a, d.name)
            for a, b, c in many_to_many_temp
            for d in stores if d.id == c]


def task1(one_to_many):
    print(one_to_many)
    return sorted(one_to_many, key=itemgetter(0))


def task2(one_to_many):
    res = []
    temp = dict()
    for a in one_to_many:
```

```
        if a[1] not in temp:
            temp[a[1]] = 1
        else:
            temp[a[1]] += 1
    for i in temp.keys():
        res.append((i, temp[i]))
    res.sort(key=itemgetter(1), reverse=True)
    return res


def task3(many_to_many):
    return [many_to_many[i] for i in range(len(many_to_many)) if
many_to_many[i][0][-2:] == 'ов']


def main():
    one_to_many = get_one_to_many(books, stores)
    many_to_many = get_many_to_many(books, mbs)

    res1 = task1(one_to_many)
    print(f'Задание Б1\n{res1}')

    res2 = task2(one_to_many)
    print(f'Задание Б2\n{res2}')

    res3 = task3(many_to_many)
    print(f'Задание Б3\n{res3}')

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

RK2_unittest.py

```
import RK2
from operator import itemgetter
import unittest


class MyTestCase(unittest.TestCase):
    def test_task1(self):
        result = RK2.task1(RK2.get_one_to_many(RK2.books,
RK2.stores))
        reference = [('Вечный кошмар', 'Читай-Город'), ('Два
желания', 'Букбридж'), ('Забудь меня', 'Читай-Город'),
                     ('Любовь не вечна', 'Лабиринт'), ('Мир
лжецов', 'Букбридж'), ('Множество лесов', 'Лабиринт'),
```

```
                    ('Перекрёсток судеб', 'Читай-Город'),
('Последняя остановка', 'Букбридж'), ('Северные ветра',
'Читай-Город')]
        self.assertEqual(result, reference)

    def test_task2(self):
        result = RK2.task2(RK2.get_one_to_many(RK2.books,
RK2.stores))
        reference = [('Читай-Город', 4), ('Букбридж', 3),
('Лабиринт', 2)]
        self.assertEqual(result, reference)

    def test_task3(self):
        result = RK2.task3(RK2.get_many_to_many(RK2.books,
RK2.mbs))
        reference = [('Мир лжецов', 'Букбридж'), ('Множество
лесов', 'Лабиринт')]
        self.assertEqual(result, reference)

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