МГТУ им. Н.Э. Баумана

Отчёт по рубежному контролю №2
по курсу «Базовые компоненты и интернет-технологии»
Вариант 13.

Руководитель
Гапанюк Ю.Е.
15.12.2022

Студент группы ИУ5-34Б
Мкртчян Д.А.
15.12.2022

2022 г.

Полученное 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).

Предметная область: класс_1 – Книга, класс_2 – Библиотека, вариант запросов: Г.

Запросы:

1. «Библиотека» и «Книга» связаны соотношением один-ко-многим. Выведите список всех библиотек, у которых название начинается с буквы «А», и список содержащихся в них книг.

2. «Библиотека» и «Книга» связаны соотношением один-ко-многим. Выведите список библиотек с максимальным размером книг в каждой библиотеке, отсортированный по максимальному количеству страниц.

3. «Библиотека» и «Книга» связаны соотношением многие-ко-многим. Выведите список всех связанных книг и библиотек, отсортированный по библиотекам, сортировка по книгам произвольная.

Текст программы:

1) Файл main.py:

```
# используется для сортировки from operator import itemgetter
from operator import itemgetter


class Book:
    """Книга"""

    def __init__(self, id, name, pg, lib_id):
        self.id = id
        self.name = name
        self.pg = pg
        self.lib_id = lib_id


class Lib:
    """Библиотека"""

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


class BookLib:
    """
    'Книги библиотек' для реализации связи многие-ко-многим
    """

    def __init__(self, lib_id, book_id):
        self.lib_id = lib_id
```

```
        self.book_id = book_id


# Библиотеки
libs = [
    Lib(1, 'аниме'),
    Lib(2, 'драма'),
    Lib(3, 'комедии'),
    Lib(4, 'биография'),
    Lib(5, 'романтика'),
    Lib(6, 'прочее'),
]

# Книги
books = [
    Book(1, 'Дракула', 500, 1),
    Book(2, 'Террор', 350, 2),
    Book(3, 'Манюня', 600, 3),
    Book(4, 'Дюна', 250, 1),
    Book(5, 'Компромисс', 750, 3),
]

books_libs = [BookLib(1, 1),
              BookLib(2, 2),
              BookLib(3, 3),
              BookLib(3, 4),
              BookLib(3, 5),
              BookLib(4, 1),
              BookLib(5, 2),
              BookLib(6, 3),
              BookLib(4, 4),
              BookLib(3, 5),
              ]

# Соединение данных один-ко-многим
one_to_many = [(b.name, b.pg, l.name)
               for l in libs for b in books
               if b.lib_id == l.id]

# Соединение данных многие-ко-многим
many_to_many_temp = [(l.name, lb.lib_id, lb.book_id)
                     for l in libs
                     for lb in books_libs if l.id == lb.lib_id]

many_to_many = [(b.name, b.pg, lib_name)
                for lib_name, lib_id, book_id in many_to_many_temp
                for b in books if b.id == book_id]


def func1(one_to_many):
    res_11 = {}
    for l in libs:
        if l.name[0] == 'а':
            l_books = list(filter(lambda i: i[2] == l.name, one_to_many))
            l_books_names = [x for x, _, _ in l_books]
            res_11[l.name] = l_books_names
    return res_11


def func2(one_to_many):
    res_12_unsorted = []
    for l in libs:
        l_books = list(filter(lambda i: i[2] == l.name, one_to_many))
        if len(l_books) > 0:
            l_members = [mem for _, mem, _ in l_books]
            l_members_max = max(l_members)
            res_12_unsorted.append((l.name, l_members_max))
```

```
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12


def func3(many_to_many):
    res_13 = sorted(many_to_many, key=itemgetter(2))
    return res_13
```

2) Файл test.py:

```
import unittest

from main import *


class Test(unittest.TestCase):
    def test_1(self):
        res = func1(one_to_many)
        exp = {'аниме': ['Дракула', 'Дюна']}
        self.assertEqual(res, exp)

    def test_2(self):
        res = func2(one_to_many)
        exp = [('комедии', 750), ('аниме', 500), ('драма', 350)]
        self.assertEqual(res, exp)

    def test_3(self):
        res = func3(many_to_many)
        exp = [('Дракула', 500, 'аниме'),
               ('Дракула', 500, 'биография'),
               ('Дюна', 250, 'биография'),
               ('Террор', 350, 'драма'),
               ('Манюня', 600, 'комедии'),
               ('Дюна', 250, 'комедии'),
               ('Компромисс', 750, 'комедии'),
               ('Компромисс', 750, 'комедии'),
               ('Манюня', 600, 'прочее'),
               ('Террор', 350, 'романтика')]


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

Результаты выполнения:

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK

Process finished with exit code 0
```