Табахов Евгений
ИУ5-32Б

Рубежный контроль №2

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

```
from operator import itemgetter

class Library:
    """Библиотека"""
    def __init__(self, id, title, rows_count, pr_id):
        self.id = id
        self.title = title
        self.rows_count = rows_count
        self.pr_id = pr_id

class ProgrammingLanguage:
    """Язык программирования"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class LibProg:
    """
    'Библиотеки языка программирования' для реализации
    связи многие-ко-многим
    """
    def __init__(self, pr_id, lib_id):
        self.pr_id = pr_id
        self.lib_id = lib_id

progs = [
    ProgrammingLanguage(1, 'Python'),
    ProgrammingLanguage(2, 'C++'),
    ProgrammingLanguage(3, 'C#'),
    ProgrammingLanguage(4, 'Java'),
]

libs = [
    Library(1, 'CV2', 4500, 1),
    Library(2, 'Numpy', 2000, 1),
    Library(3, 'Math', 1500, 2),
    Library(4, 'Libpq', 6000, 2),
    Library(5, 'NUnit', 4000, 3),
    Library(6, 'Moq', 3000, 3),
    Library(7, 'JHipster', 7000, 4),
    Library(8, 'Maven', 4500, 4),
]

libs_progs = [
    LibProg(1, 1),
    LibProg(1, 2),
    LibProg(2, 3),
    LibProg(2, 4),
    LibProg(3, 5),
    LibProg(3, 6),
    LibProg(4, 7),
    LibProg(4, 8),
]

def join_one_to_many(progs, libs):
    return [(l.title, l.rows_count, p.name) for p in progs for l in libs if l.pr_id == p.id]

def join_many_to_many(progs, libs_progs, libs):
    many_to_many_temp = [(p.name, lp.pr_id, lp.lib_id) for p in progs for lp in libs_progs if p.id
== lp.pr_id]
    return [(l.title, l.rows_count, prog_name) for prog_name, pr_id, lib_id in many_to_many_temp
for l in libs if l.id == lib_id]

def sort_one_to_many(one_to_many_list):
    return sorted(one_to_many_list, key=itemgetter(0))

def sum_rows_by_prog(progs, one_to_many_list):
    res_unsorted = []
    for p in progs:
        p_libs = list(filter(lambda i: i[2] == p.name, one_to_many_list))
        if len(p_libs) > 0:
            p_rows = [rows_count for _, rows_count, _ in p_libs]
            p_rows_sum = sum(p_rows)
            res_unsorted.append((p.name, p_rows_sum))
    return sorted(res_unsorted, key=itemgetter(1), reverse=True)

def find_libs_by_title(many_to_many_list):
    res = {}
    for l in libs:
        if ('m' in l.title) or ('M' in l.title):
            l_progs = list(filter(lambda i: i[0] == l.title, many_to_many_list))
            l_progs_titles = [x for _, _, x in l_progs]
            res[l.title] = l_progs_titles
    return res

def main():
    one_to_many = join_one_to_many(progs, libs)
    many_to_many = join_many_to_many(progs, libs_progs, libs)

    print('Задание Б1')
    res_11 = sort_one_to_many(one_to_many)
    print(res_11)

    print('\nЗадание Б2')
    res_12 = sum_rows_by_prog(progs, one_to_many)
    print(res_12)
    print('\nЗадание Б3')
    res_13 = find_libs_by_title(many_to_many)
    print(res_13)

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

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

```
import unittest
from RK2 import join_one_to_many, join_many_to_many, sort_one_to_many,
sum_rows_by_prog, Library, ProgrammingLanguage

class TestLibrary(unittest.TestCase):

    def test_join_one_to_many(self):
        progs = [
            ProgrammingLanguage(1, 'Python'),
            ProgrammingLanguage(2, 'C++'),
            ProgrammingLanguage(3, 'C#'),
            ProgrammingLanguage(4, 'Java'),
        ]

        libs = [
            Library(1, 'CV2', 4500, 1),
            Library(2, 'Numpy', 2000, 1),
            Library(3, 'Math', 1500, 2),
            Library(4, 'Libpq', 6000, 2),
            Library(5, 'NUnit', 4000, 3),
            Library(6, 'Moq', 3000, 3),
            Library(7, 'JHipster', 7000, 4),
            Library(8, 'Maven', 4500, 4),
        ]

        expected_result = [
            ('CV2', 4500,'Python'),
            ('Numpy',2000,'Python'),
            ('Math',1500,'C++'),
            ('Libpq',6000,'C++'),
            ('NUnit',4000,'C#'),
            ('Moq',3000,'C#'),
            ('JHipster',7000,'Java'),
            ('Maven',4500,'Java')
        ]

        self.assertEqual(join_one_to_many(progs,libs), expected_result)

    def test_sort_one_to_many(self):
        input_list = [
            ('CV2', 4500,'Python'),
            ('Numpy',2000,'Python'),
            ('Math',1500,'C++'),
            ('Libpq',6000,'C++'),
            ('NUnit',4000,'C#'),
            ('Moq',3000,'C#'),
            ('JHipster',7000,'Java'),
            ('Maven',4500,'Java')
        ]

        expected_result = [
            ('CV2', 4500,'Python'),
            ('JHipster',7000,'Java'),
            ('Libpq',6000,'C++'),
            ('Math',1500,'C++'),
            ('Maven',4500,'Java'),
            ('Moq',3000,'C#'),
            ('NUnit',4000,'C#'),
            ('Numpy',2000,'Python')
        ]

        self.assertEqual(sort_one_to_many(input_list), expected_result)

    def test_sum_rows_by_prog(self):
        progs = [
            ProgrammingLanguage(1, 'Python'),
            ProgrammingLanguage(2, 'C++'),
            ProgrammingLanguage(3, 'C#'),
            ProgrammingLanguage(4, 'Java'),
        ]

        one_to_many_list = [
            ('CV2', 4500,'Python'),
            ('Numpy',2000,'Python'),
            ('Math',1500,'C++'),
            ('Libpq',6000,'C++'),
            ('NUnit',4000,'C#'),
            ('Moq',3000,'C#'),
            ('JHipster',7000,'Java'),
            ('Maven',4500,'Java')
        ]

        expected_result = [
            ('Java',11500),
            ('Python',6500),
            ('C#',7000),
            ('C++',7500)
        ]

        self.assertEqual(sum_rows_by_prog(progs, one_to_many_list), expected_result)

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