Московский государственный технический
Университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»
Отчет рубежному контролю №2

Выполнил:
студент группы ИУ5-34Б
Нигматуллин А. Р.

Проверил:
Гапанюк Е.Ю.

Москва, 2022

Условия рубежного контроля №2 по курсу БКИТ

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).

Измененный код РК1:

```
# используется для сортировки
from operator import itemgetter


class Disk:
    """CD-диск"""

    def __init__(self, id, maker, memory, lib_id):
        self.id = id
        self.maker = maker
        self.memory = memory
        self.lib_id = lib_id


class Lib:
    """Библиотека CD-дисков"""

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


class DiskLib:
    """
    'CD-диски библиотек' для реализации
    связи многие-ко-многим
    """

    def __init__(self, lib_id, disk_id):
        self.lib_id = lib_id
        self.disk_id = disk_id


def task1(libs, disks):
    res = {}
    one_to_many = [(d.maker, d.memory, l.name)
                   for l in libs
                   for d in disks
                   if d.lib_id == l.id]
    for l in libs:
        if l.name[0] == 'a':
            l_disks = list(filter(lambda i: i[2] == l.name, one_to_many))
            l_disks_names = [x for x, _, _ in l_disks]
            res[l.name] = l_disks_names

    return res
```

```python
def task2(libs, disks):
    res_unsorted = []
    one_to_many = [(d.maker, d.memory, l.name)
                   for l in libs
                   for d in disks
                   if d.lib_id == l.id]
    for l in libs:
        l_disks = list(filter(lambda i: i[2] == l.name, one_to_many))
        if len(l_disks) > 0:
            l_members = [mem for _, mem, _ in l_disks]
            l_members_max = max(l_members)
            res_unsorted.append((l.name, l_members_max))

    res = sorted(res_unsorted, key=itemgetter(1), reverse=True)
    return res


def task3(libs, disks_libs, disks):
    many_to_many_temp = [(l.name, ld.lib_id, ld.disk_id)
                         for l in libs
                         for ld in disks_libs
                         if l.id == ld.lib_id]

    many_to_many = [(d.maker, d.memory, lib_name)
                    for lib_name, lib_id, disk_id in many_to_many_temp
                    for d in disks if d.id == disk_id]
    res = sorted(many_to_many, key=itemgetter(2))
    return res


if __name__ == '__main__':
    # Библиотеки CD-дисков
    Libs = [
        Lib(1, 'архивы'),
        Lib(2, 'фильмы'),
        Lib(3, 'игры'),
        Lib(4, 'музыка'),
        Lib(5, 'утилиты'),
        Lib(6, 'прочее'),
    ]

    # CD-диски
    Disks = [
        Disk(1, 'Verbatim', 700, 1),
        Disk(2, 'Ritek', 650, 2),
        Disk(3, 'Sonnen', 500, 3),
        Disk(4, 'Mirex', 250, 1),
        Disk(5, 'Vs', 350, 3),
    ]

    Disks_libs = [
        DiskLib(1, 1),
        DiskLib(2, 2),
        DiskLib(3, 3),
        DiskLib(3, 4),
        DiskLib(3, 5),
        DiskLib(4, 1),
        DiskLib(5, 2),
        DiskLib(6, 3),
        DiskLib(4, 4),
        DiskLib(3, 5),
    ]
    """Основная функция"""
    print('Задание Г1')
```

```
    res_11 = task1(Libs, Disks)
    print(res_11)

    print('\nЗадание Г2')
    res_12 = task2(Libs, Disks)
    print(res_12)

    print('\nЗадание Г3')
    res_13 = task3(Libs, Disks_libs, Disks)
    for i in res_13:
        print(i)
```

Тестирование:

```
import unittest
from main import task1, task2, task3, Disk, Lib, DiskLib


class FieldTest(unittest.TestCase):
    def test1(self):
        Libs = [
            Lib(1, 'архивы'),
            Lib(2, 'фильмы'),
            Lib(3, 'игры'),
            Lib(4, 'музыка'),
            Lib(5, 'утилиты'),
            Lib(6, 'прочее'),
        ]

        # CD-диски
        Disks = [
            Disk(1, 'Verbatim', 700, 1),
            Disk(2, 'Ritek', 650, 2),
            Disk(3, 'Sonnen', 500, 3),
            Disk(4, 'Mirex', 250, 1),
            Disk(5, 'Vs', 350, 3),
        ]
        self.assertEqual(task1(Libs, Disks), {'архивы': ['Verbatim',
'Mirex']})

    def test2(self):
        Libs = [
            Lib(1, 'архивы'),
            Lib(2, 'фильмы'),
            Lib(3, 'игры'),
            Lib(4, 'музыка'),
            Lib(5, 'утилиты'),
            Lib(6, 'прочее'),
        ]

        # CD-диски
        Disks = [
            Disk(1, 'Verbatim', 700, 1),
            Disk(2, 'Ritek', 650, 2),
            Disk(3, 'Sonnen', 500, 3),
            Disk(4, 'Mirex', 250, 1),
            Disk(5, 'Vs', 350, 3),
        ]
        self.assertEqual(task2(Libs, Disks), [('архивы', 700), ('фильмы',
650), ('игры', 500)])

    def test3(self):
        Libs = [
```

```
            Lib(1, 'архивы'),
            Lib(2, 'фильмы'),
            Lib(3, 'игры'),
            Lib(4, 'музыка'),
            Lib(5, 'утилиты'),
            Lib(6, 'прочее'),
        ]

        # CD-диски
        Disks = [
            Disk(1, 'Verbatim', 700, 1),
            Disk(2, 'Ritek', 650, 2),
            Disk(3, 'Sonnen', 500, 3),
            Disk(4, 'Mirex', 250, 1),
            Disk(5, 'Vs', 350, 3),
        ]

        Disks_libs = [
            DiskLib(1, 1),
            DiskLib(2, 2),
            DiskLib(3, 3),
            DiskLib(3, 4),
            DiskLib(3, 5),
            DiskLib(4, 1),
            DiskLib(5, 2),
            DiskLib(6, 3),
            DiskLib(4, 4),
            DiskLib(3, 5),
        ]
        self.assertEqual(task3(Libs, Disks_libs, Disks), [('Verbatim', 700,
'архивы'),
                          ('Sonnen', 500, 'игры'),
                          ('Mirex', 250, 'игры'),
                          ('Vs', 350, 'игры'),
                          ('Vs', 350, 'игры'),
                          ('Verbatim', 700, 'музыка'),
                          ('Mirex', 250, 'музыка'),
                          ('Sonnen', 500, 'прочее'),
                          ('Ritek', 650, 'утилиты'),
                          ('Ritek', 650, 'фильмы')])
```

Результат тестирования: