

Факультет Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

**Отчет по рубежному контролю № 2 по курсу  
Парадигмы и конструкции языков программирования**

Исполнитель

Студент группы РТ5-31Б \_\_\_\_\_

Кочкин А.В.

«\_\_»\_\_\_\_\_ 2023 г.

Проверил

Доцент кафедры ИУ5 \_\_\_\_\_

Гапанюк Ю.Е.

«\_\_»\_\_\_\_\_ 2023 г.

## Задание РК2

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).

## Задание РК1

Предметная область E, вариант 14. Классы: CD-диск, Библиотека CD-дисков.

Задания:

1. «Библиотека CD-дисков» и «CD-диск» связаны соотношением один-ко-многим. Выведите список всех каталогов, у которых в названии присутствует буква «а», и список его файлов.
2. «Библиотека CD-дисков» и «CD-диск» связаны соотношением один-ко-многим. Выведите список библиотек со средним кол-вом дисков в них, отсортированный по среднему количеству дисков. Среднее кол-во дисков в библиотеке должно быть округлено до 2 знаков после запятой.
3. «Библиотека CD-дисков» и «CD-диск» связаны соотношением многие-ко-многим. Выведите список всех дисков, у которых название начинается с буквы «М», и названия их библиотек.

Листинг программы, в которой выполняются задания и для которой был проведён рефакторинг (PK1.py)

```

'''Вариант - E, вариант предметной области - 14
("диск - полка)'''

from operator import itemgetter

class disk:
    """Диск"""
    def __init__(self, id, name, size, ShelfId):
        self.id = id
        self.name = name
        self.size = size          #В секундах
        self.ShelfId = ShelfId

class Shelf:
```

```

"""Полка"""

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

class diskShelf:
    """
    'Диски Полки' для реализации
    связи многие-ко-многим
    """
    def __init__(self, ShelfId, diskId):
        self.ShelfId = ShelfId
        self.diskId = diskId

Shelfs = [Shelf(1, "Металл"),
          Shelf(2, "Рок-Н-Ролл"),
          Shelf(3, "Хардстайл")]

disks = [disk(1,"MetallicA",1134,1),
          disk(2,"Машина времени",2517,2),
          disk(3,"Bladee Mixtape",1488,3),
          disk(4,"Elvis Presley",1860,2),
          disk(5,"SOAD Mezmerize",3252,1)
          ]

disksShelfs = [diskShelf(1,1),
               diskShelf(2,2),
               diskShelf(1,3),
               diskShelf(3,3)]

ShelfsId = [c.id for c in Shelfs]

```

```
OneToMany = [(f.name, f.size, Shelves[ShelvesId.index(f.ShelfId)].name) for f in disks]
```

```
disksId = [f.id for f in disks]
```

```
ManyToMany = [(disks[disksId.index(fc.diskId)].name,
                disks[disksId.index(fc.diskId)].size,
                Shelves[ShelvesId.index(fc.ShelfId)].name)
               for fc in disksShelves]
```

```
def task1(OneToMany):
```

```
    word1 = "a"
```

```
    ShelvesE1 = [c.name for c in Shelves if word1 in c.name]
```

```
    disksE1 = [otm[0] for otm in OneToMany for c in ShelvesE1 if otm[2] == c]
```

```
    return ShelvesE1, disksE1
```

```
def task2(OneToMany):
```

```
    return sorted([(c.name, round(sum([otm[1] for otm in OneToMany if otm[2] ==
c.name]))/(lambda x: 1 if x==0 else x)(len([otm[1] for otm in OneToMany if otm[2] == c.name]])))
for c in Shelves], key=itemgetter(1), reverse=True)
```

```
def task3(ManyToMany):
```

```
    char3 = "M"
```

```
    return [[f.name, [mtm[2] for mtm in ManyToMany if mtm[0]==f.name]] for f in disks if
f.name[0] == char3]
```

```
if __name__ == '__main__':
```

```
    print(task1(OneToMany))
```

```
    print(task2(OneToMany))
```

```
    print(task3(ManyToMany))
```

Листинг программы, в которой проводятся тесты (PK2 14E.py)

```
import unittest
```

```
import PK1
```

```
class testPK1(unittest.TestCase):

    def setUp(self):

        self.test1 = (['Металл', 'Хардстайл'], ['MetallicA', 'Bladee Mixtape', 'SOAD Mezmerize'])

        self.test2 = [['Металл', 2193], ['Рок-Н-Ролл', 2188], ['Хардстайл', 1488]]

        self.test3 = [['MetallicA', ['Металл']], ['Машина времени', ['Рок-Н-Ролл']]]

    def test1_rk(self):

        self.assertEqual(PK1.task1(PK1.OneToMany), self.test1)

    def test2_rk(self):

        self.assertEqual(PK1.task2(PK1.OneToMany), self.test2)

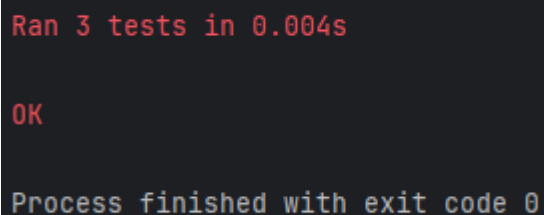
    def test3_rk(self):

        self.assertEqual(PK1.task3(PK1.ManyToMany), self.test3)

if __name__ == '__main__':

    unittest.main()
```

## Результаты работы программы



```
Ran 3 tests in 0.004s

OK

Process finished with exit code 0
```