

**Московский государственный технический
университет им. Н.Э. Баумана**

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

Курс «Парадигмы и конструкции языков программирования»

Отчет по РК №2

Выполнил:

студент группы ИУ5-32Б

Кудрявцев Андрей

Подпись и дата:

Проверила:

преподаватель каф. ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2023 г.

Постановка задачи

- 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

Текст программы PK1_refactor.py

вариант 11Б (Программа Компьютер)

используется для сортировки
`from operator import itemgetter`

`class Prog: # program`
"""Программа"""

`def __init__(self, id, name, vers, mem, comp_id):`
 `self.id = id`
 `self.name = name`
 `self.vers = vers # version`
 `self.mem = mem # memory`
 `self.comp_id = comp_id`

`class Comp: # computer`
"""Компьютер"""

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

`class ProgComp:`
"""
'Программы компьютера' для реализации
связи многие-ко-многим
"""

`def __init__(self, comp_id, prog_id):`
 `self.comp_id = comp_id`
 `self.prog_id = prog_id`

Компьютеры
`comps = [`
 `Comp(1, 'Имя моего компьютера'),`
 `Comp(2, 'Мой ноутбук'),`
 `Comp(3, 'Рабочий компьютер'),`

```

Comp(11, 'имя (другого) моего компьютеров'),
Comp(22, 'мой (другой) ноутбук'),
Comp(33, '(другой) рабочий компьютер'),
]

```

Программы

```

progs = [
    Prog(1, 'Chrome', '10.02.8', 262444000, 1), # ~250 MB
    Prog(1, 'Firefox', '11.22.8', 255444000, 1), # ~250 MB
    Prog(2, 'PyCharm', '12.02.8', 2600354000, 2), # ~2,5 GB
    Prog(3, 'Adobe', '6.02.8', 8100437000, 3), # ~7,6 GB
    Prog(4, 'Visual Studio', '1.02.8', 4200967000, 3), # ~4 GB
    Prog(5, 'LibreOffice', '7.02.8', 419444000, 3), # ~400 MB
]

```

```

progs_comps = [
    ProgComp(1, 1),
    ProgComp(1, 2),
    ProgComp(2, 3),
    ProgComp(3, 4),
    ProgComp(3, 5),
    ProgComp(3, 6),

    ProgComp(11, 1),
    ProgComp(11, 2),
    ProgComp(22, 3),
    ProgComp(33, 4),
    ProgComp(33, 5),
    ProgComp(33, 6),
]

```

```

def g1_solution(one_to_many):
    res_11 = sorted(one_to_many, key=itemgetter(0))
    return res_11

```

```

def g2_solution(one_to_many):
    res_12_unsorted = []
    # Перебираем все компьютеры
    for c in comps:
        # Список программ компьютера
        c_progs = list(filter(lambda i: i[2] == c.name, one_to_many))
        # Если список не пустой
        if len(c_progs) > 0:
            c_prog_amount = len(c_progs)
            res_12_unsorted.append((c.name, c_prog_amount))

    # Сортировка по количеству программ
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12

```

```

def g3_solution(many_to_many):
    res_13 = {}
    # Перебираем все компьютеры
    for c in comps:
        # Список программ компьютера
        c_progs = list(filter(lambda i: i[2] == c.name, many_to_many))
        # Только программы с названием, заканчивающимся на "е"
        c_progs_names = [x for x, _, _ in c_progs if x[-1] == 'е']
        # Добавляем результат в словарь

```

```

# ключ - компьютер, значение - список программ
if len(c_progs_names) > 0:
    res_13[c.name] = c_progs_names
return res_13

def main():
    """Основная функция"""

    # -----
    # Соединение данных один-ко-многим
    one_to_many = [(p.name, p.mem, c.name)
                    for c in comps
                    for p in progs
                    if p.comp_id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.name, pc.comp_id, pc.prog_id)
                          for c in comps
                          for pc in progs_comps
                          if c.id == pc.comp_id]

    many_to_many = [(p.name, p.mem, comp_name)
                     for comp_name, com_id, prog_id in many_to_many_temp
                     for p in progs if p.id == prog_id]

    print("Задание A1")
    # «Компьютер» и «Программа» связаны соотношением один-ко-многим. Выведите список всех связанных программ и компьютеров, отсортированный по программам, сортировка по компьютерам произвольная.
    print(g1_solution(one_to_many))

    print("\nЗадание A2")
    # «Компьютер» и «Программа» связаны соотношением один-ко-многим. Выведите список компьютеров с количеством программ на каждом компьютере, отсортированный по количеству программ.
    print(g2_solution(one_to_many))

    print("\nЗадание A3")
    # «Компьютер» и «Программа» связаны соотношением многие-ко-многим. Выведите список всех программ, у которых название заканчивается на «е», и названия их компьютеров.
    print(g3_solution(many_to_many))

if __name__ == '__main__':
    main()

```

Текст программы RK2.py

```
import unittest
from RK1_refactor import *

class Test_Program(unittest.TestCase):
    # Компьютеры
    comps = [
        Comp(1, 'Имя моего компьютера'),
        Comp(2, 'Мой ноутбук'),
        Comp(3, 'Рабочий компьютер'),

        Comp(11, 'имя (другого) моего компьютеров'),
        Comp(22, 'мой (другой) ноутбук'),
        Comp(33, '(другой) рабочий компьютер'),
    ]

    # Программы
    progs = [
        Prog(1, 'Chrome', '10.02.8', 262444000, 1), # ~250 MB
        Prog(1, 'Firefox', '11.22.8', 255444000, 1), # ~250 MB
        Prog(2, 'PyCharm', '12.02.8', 2600354000, 2), # ~2,5 GB
        Prog(3, 'Adobe', '6.02.8', 8100437000, 3), # ~7,6 GB
        Prog(4, 'Visual Studio', '1.02.8', 4200967000, 3), # ~4 GB
        Prog(5, 'LibreOffice', '7.02.8', 419444000, 3), # ~400 MB
    ]

    progs_comps = [
        ProgComp(1, 1),
        ProgComp(1, 2),
        ProgComp(2, 3),
        ProgComp(3, 4),
        ProgComp(3, 5),
        ProgComp(3, 6),

        ProgComp(11, 1),
        ProgComp(11, 2),
        ProgComp(22, 3),
        ProgComp(33, 4),
        ProgComp(33, 5),
        ProgComp(33, 6),
    ]

    def test_g1(self):
        # Соединение данных один-ко-многим
        one_to_many = [(p.name, p.mem, c.name)
                        for c in comps
                        for p in progs
                        if p.comp_id == c.id]
        self.assertEqual(g1_solution(one_to_many),
                         [('Adobe', 8100437000, 'Рабочий компьютер'), ('Chrome', 262444000, 'Имя моего компьютера'),
                          ('Firefox', 255444000, 'Имя моего компьютера'),
                          ('LibreOffice', 419444000, 'Рабочий компьютер'), ('PyCharm', 2600354000, 'Мой ноутбук'),
                          ('Visual Studio', 4200967000, 'Рабочий компьютер')])

    def test_g2(self):
        one_to_many = [(p.name, p.mem, c.name)
                        for c in comps
                        for p in progs
                        if p.comp_id == c.id]
        self.assertEqual(g2_solution(one_to_many),
```

```
[('Рабочий компьютер', 3), ('Имя моего компьютера', 2), ('Мой ноутбук', 1)])
```

```
def test_g3(self):
```

```
    # Соединение данных многие-ко-многим
```

```
    many_to_many_temp = [(c.name, pc.comp_id, pc.prog_id)
```

```
                          for c in comps
```

```
                          for pc in progs_comps
```

```
                          if c.id == pc.comp_id]
```

```
    many_to_many = [(p.name, p.mem, comp_name)
```

```
                    for comp_name, com_id, prog_id in many_to_many_temp
```

```
                    for p in progs if p.id == prog_id]
```

```
    self.assertEqual(g3_solution(many_to_many),
```

```
                     {'Имя моего компьютера': ['Chrome'], 'Мой ноутбук': ['Adobe'],
```

```
                     'Рабочий компьютер': ['LibreOffice'], 'имя (другого) моего компьютеров': ['Chrome'],
```

```
                     'мой (другой) ноутбук': ['Adobe'], '(другой) рабочий компьютер': ['LibreOffice']})
```

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

```
    unittest.main()
```

Результат выполнения программы

```
Ran 3 tests in 0.005s
```

```
OK
```

```
Process finished with exit code 0
```

Если специально допустить ошибку:

✖ Tests failed: 1, passed: 2 of 3 tests – 21 ms

```
Ran 3 tests in 0.023s
```

```
FAILED (failures=1)
```

```
[('AAAAAdobe', 8100437000, 'Рабочий компьютер'),
 ('Chrome', 262444000, 'Имя моего компьютера'),
 ('Firefox', 255444000, 'Имя моего компьютера'),
 ('LibreOffice', 419444000, 'Рабочий компьютер'),
 ('PyCharm', 2600354000, 'Мой ноутбук'),
 ('Visual Studio', 4200967000, 'Рабочий компьютер')] != [('Adobe', 8100437000, 'Рабочий компьютер'),
 ('Chrome', 262444000, 'Имя моего компьютера'),
 ('Firefox', 255444000, 'Имя моего компьютера'),
 ('LibreOffice', 419444000, 'Рабочий компьютер'),
 ('PyCharm', 2600354000, 'Мой ноутбук'),
 ('Visual Studio', 4200967000, 'Рабочий компьютер')]
```

[<Click to see difference>](#)

Traceback (most recent call last):

File "[C:\Users\User\PycharmProjects\pythonProject\RK2\RK2.py](#)", line 49, in test_g1

self.assertEqual(g1_solution(one_to_many),

AssertionError: Lists differ: [('Adobe', 8100437000, 'Рабочий компьютер'), [230 chars]ep')] != [('AAAA/

First differing element 0:

('Adobe', 8100437000, 'Рабочий компьютер')

('AAAAAdobe', 8100437000, 'Рабочий компьютер')