**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 рубежному контролю № 2
Вариант Г-13

Выполнил:
студент группы ИУ5-34Б
Багин М. В.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2024 г

## Постановка задачи:

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).

Рефакторинг кода заключался в перенос логики задания в функции для последующего тестирования.

## Текст программы

main.py

```
from operator import itemgetter
 class
Cath:
    def __init__(self, id, fio,
sal, dep_id):         self.id =
id         self.fio = fio
self.sal = sal
self.dep_id = dep_id
  class
Facul:
    def __init__(self, id, name):
        self.id = id
self.name = name
  class
CathFacul:
    def __init__(self, dep_id,
emp_id):          self.dep_id =
dep_id         self.emp_id =
emp_id
  faculs = [
Facul(1, 'РК'),
    Facul(2, 'РТ'),
    Facul(3, 'ИУ'),
    Facul(11, 'СМ'),
    Facul(22, 'Эн'),
    Facul(33, 'МТ'),
]  caths
= [
    Cath(1, 'ИУ5', 25000, 3),
    Cath(2, 'ИУ6', 35000, 3),
    Cath(3, 'ИУ7', 45000, 3),
    Cath(4, 'РК6', 35000, 1),
    Cath(5, 'РТ5', 25000, 2),
]

CathFaculs = [
    CathFacul(1, 1),
    CathFacul(2, 2),
    CathFacul(3, 3),
    CathFacul(3, 4),
    CathFacul(3, 5),
    CathFacul(11, 1),
    CathFacul(22, 2),
```

```
    CathFacul(33, 3),
    CathFacul(33, 4),
    CathFacul(33, 5),
]

 def get_one_to_many(caths,
faculs):     return [(e.fio, e.sal,
d.name)              for d in
faculs             for e in caths
if e.dep_id == d.id]
  def get_many_to_many(caths,
faculs,
CathFaculs):
    many_to_many_temp = [(d.name,
ed.dep_id, ed.emp_id)
for d in faculs
                        for ed in
CathFaculs
if d.id == ed.dep_id]
return [(e.fio, e.sal, dep_name)
for dep_name, dep_id, emp_id in
many_to_many_temp             for e
in caths if e.id == emp_id]
  def
count_caths_by_fio(one_to_many):
    one_to_many =
sorted(one_to_many,
key=itemgetter(2))     temp =
''     cur = 0     res2 =
[[name, 0] for _, _, name in
one_to_many]          for i in
range(len(res2)):         if
res2[i][0] != temp:
temp = res2[i][0]
cur = i             res2[i][1]
+= 1         else:
            res2[cur][1] += 1
     return list(filter(lambda
y:
y[1] != 0, res2))
  def
filter_caths_by_fio(many_to_many):
return [[fio, name] for fio, _, name
in many_to_many if fio[0] ==
'И']
  def
main():
one_to_many
=
get_one_to_many(caths, faculs)
    res = sorted(one_to_many,
key=itemgetter(0))
    print(res)

    res2 = count_caths_by_fio(res)
print(res2)
     many_to_many
=
```

```
    get_many_to_many(caths, faculs,
    CathFaculs)     res3 =
    filter_caths_by_fio(many_to_many)
        print(res3)
      if __name__ ==
    '__main__':
        main()
```

unit_test.py

```
import unittest

class TestCathFunctions(unittest.TestCase):

                                    def setUp(self):
                                self.faculs = [
            Facul(1, 'РК'),
            Facul(2, 'РТ'),
            Facul(3, 'ИУ'),
                                        ]
                                 self.caths = [
                                      Cath(1, 'ИУ5', 25000, 3),
                                      Cath(2, 'ИУ6', 35000, 3),
                                      Cath(3, 'ИУ7', 45000, 3),
                                      Cath(4, 'РК6', 35000, 1),
                                      Cath(5, 'РТ5', 25000, 2),
                                       ]
                             self.CathFaculs = [
            CathFacul(1, 1),
            CathFacul(2, 2),
            CathFacul(3, 3),
            CathFacul(3, 4),
            CathFacul(3, 5),
        ]

    def test_get_one_to_many(self):
                result = get_one_to_many(self.caths, self.faculs)
         expected = [
            ('ИУ5', 25000, 'ИУ'),
            ('ИУ6', 35000, 'ИУ'),
            ('ИУ7', 45000, 'ИУ'),
            ('РК6', 35000, 'РК'),
            ('РТ5', 25000, 'РТ'),
        ]
        self.assertEqual(sorted(result), sorted(expected))

    def test_get_many_to_many(self):
        result = get_many_to_many(self.caths, self.faculs, self.CathFaculs)
expected = [
            ('ИУ5', 25000, 'ИУ'),
                                    ('ИУ6', 35000, 'ИУ'),
                                    ('ИУ7', 45000, 'ИУ'),
                                    ('РК6', 35000, 'РК'),
                                    ('РТ5', 25000, 'РТ'),
                                            ]
                    self.assertEqual(sorted(result), sorted(expected))

                         def test_count_caths_by_fio(self):
                            one_to_many = [
                                    ('ИУ5', 25000, 'ИУ'),
                                    ('ИУ6', 35000, 'ИУ'),
```

```
                ('ИУ7', 45000, 'ИУ'),
                ('РК6', 35000, 'РК'),
                ('РТ5', 25000, 'РТ'),
                    ]
        result = count_caths_by_fio(one_to_many)
                    expected = [
                        ['ИУ', 3],
                        ['РК', 1],
                        ['РТ', 1],
                        ]
    self.assertEqual(sorted(result), sorted(expected))

        def test_filter_caths_by_fio(self):
            many_to_many = [
                ('ИУ5', 25000, 'ИУ'),
                ('ИУ6', 35000, 'ИУ'),
                ('ИУ7', 45000, 'ИУ'),
                ('РК6', 35000, 'РК'),
                ('РТ5', 25000, 'РТ'),
                    ]
        result = filter_caths_by_fio(many_to_many)
            expected = [
                    ['ИУ5', 'ИУ'],
                    ['ИУ6', 'ИУ'],
                    ['ИУ7', 'ИУ'],
                    ]
    self.assertEqual(sorted(result), sorted(expected))


            if __name__ == '__main__':

    unittest.main()
```

## Результат выполнения программы:

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

OK
```