Марянян Артур ИУ5-34Б

Отчет РК-2 по дисциплине

Парадигмы и конструкторы языков программирования

Задание

1. Проведите рефакторинг текста программы таким образом, чтобы он был пригоден для тестирования

2. Для текста программы создайте модульное тестирование с применением TDD-фреймворка from operator import itemgetter

Текст программы main.py

```
from operator import itemgetter

class Microprocessor:
    def __init__(self, id, count, price, comp_id):
        self.id = id
        self.count = count
        self.price = price
        self.comp_id = comp_id

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

class MicrComp:
    def __init__(self, micr_id, comp_id):
        self.micr_id = micr_id
        self.comp_id = comp_id
```

```
def get_one_to_many(microprocessors, computers):
    return [
        (m.count, m.price, c.name)
        for m in microprocessors
        for c in computers
        if m.comp_id == c.id
    ]

def get_many_to_many(microprocessors, computers,
micr_comp):
    many_to_many_temp = [
        (c.name, mc.micr_id, mc.comp_id)
        for c in computers
        for mc in micr_comp
        if c.id == mc.comp_id
    ]

    return [
        (m.count, m.price, c.name)
        for mc in micr_comp
        for m in microprocessors if m.id ==
mc.micr_id
        for c in computers if c.id == mc.comp_id
    ]

def task_g1(computers, microprocessors):
    result = {}
    for c in computers:
        if int(c.name[-1]) < 4:
            m_c = [
                (m.count, m.price) for m in
microprocessors if m.comp_id == c.id
```

```
            ]
            result[c.name] = m_c
    return result

def task_g2(computers, one_to_many):
    result = []
    for c in computers:
        c_micrs = list(filter(lambda i: i[2] ==
c.name, one_to_many))
        if c_micrs:
            s_price = [price for _, price, _ in
c_micrs]
            s_max = max(s_price)
            result.append((c.name, s_max))
    return sorted(result, key=itemgetter(1),
reverse=True)

def task_g3(many_to_many):
    return sorted(many_to_many, key=itemgetter(2))

# Пример использования
def main():
    computers = [
        Computer(1, 'Компьютер 1'),
        Computer(2, 'Компьютер 2'),
        Computer(3, 'Компьютер 3'),
        Computer(4, 'Компьютер 4'),
        Computer(5, 'Компьютер 5'),
        Computer(6, 'Компьютер 6'),
    ]

    microprocessors = [
```

```
        Microprocessor(1, 1001, 12000, 2),
        Microprocessor(2, 270011, 12442, 3),
        Microprocessor(3, 323312, 147977, 1),
        Microprocessor(4, 664623, 2356, 3),
        Microprocessor(5, 374223, 2467, 4),
        Microprocessor(6, 12654, 2357, 5),
    ]

    micr_comp = [
        MicrComp(1, 1),
        MicrComp(2, 2),
        MicrComp(3, 3),
        MicrComp(3, 4),
        MicrComp(4, 4),
        MicrComp(5, 6),
        MicrComp(5, 2),
        MicrComp(1, 2),
    ]

    one_to_many = get_one_to_many(microprocessors,
computers)
    many_to_many =
get_many_to_many(microprocessors, computers,
micr_comp)

    print('Задание Г1:', task_g1(computers,
microprocessors))
    print('Задание Г2:', task_g2(computers,
one_to_many))
    print('Задание Г3:', task_g3(many_to_many))

if __name__ == '__main__':
```

```
    main()
```

Текст программы tests.py

```
import unittest
from main import Computer, Microprocessor,
MicrComp, get_one_to_many, get_many_to_many,
task_g1, task_g2, task_g3

class
TestMicroprocessorFunctions(unittest.TestCase):
    def setUp(self):
        self.computers = [
            Computer(1, 'Компьютер 1'),
            Computer(2, 'Компьютер 2'),
            Computer(3, 'Компьютер 3'),
            Computer(4, 'Компьютер 4'),
            Computer(5, 'Компьютер 5'),
            Computer(6, 'Компьютер 6'),
        ]

        self.microprocessors = [
            Microprocessor(1, 1001, 12000, 2),
            Microprocessor(2, 270011, 12442, 3),
            Microprocessor(3, 323312, 147977, 1),
            Microprocessor(4, 664623, 2356, 3),
            Microprocessor(5, 374223, 2467, 4),
            Microprocessor(6, 12654, 2357, 5),
        ]

        self.micr_comp = [
            MicrComp(1, 1),
            MicrComp(2, 2),
```

```
            MicrComp(3, 3),
            MicrComp(3, 4),
            MicrComp(4, 4),
            MicrComp(5, 6),
            MicrComp(5, 2),
            MicrComp(1, 2),
        ]

    def test_get_one_to_many(self):
        result =
get_one_to_many(self.microprocessors,
self.computers)
        self.assertTrue(len(result) > 0)
        self.assertIn((1001, 12000, 'Компьютер 2'),
result)

    def test_get_many_to_many(self):
        result =
get_many_to_many(self.microprocessors,
self.computers, self.micr_comp)
        self.assertTrue(len(result) > 0)
        self.assertIn((1001, 12000, 'Компьютер 1'),
result)

    def test_task_g1(self):
        result = task_g1(self.computers,
self.microprocessors)
        self.assertIn('Компьютер 1', result)

    def test_task_g2(self):
```

```
        one_to_many =
get_one_to_many(self.microprocessors,
self.computers)
        result = task_g2(self.computers,
one_to_many)
        self.assertTrue(result[0][1] > result[-
1][1])

    def test_task_g3(self):
        many_to_many =
get_many_to_many(self.microprocessors,
self.computers, self.micr_comp)
        result = task_g3(many_to_many)
        self.assertTrue(len(result) > 0)

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

Вывод программы

Задание Г1: {'Компьютер 1': [(323312, 147977)], 'Компьютер 2': [(1001, 12000)], 'Компьютер 3': [(270011, 12442), (664623, 2356)]}

Задание Г2: [('Компьютер 1', 147977), ('Компьютер 3', 12442), ('Компьютер 2', 12000), ('Компьютер 4', 2467), ('Компьютер 5', 2357)]

Задание Г3: [(1001, 12000, 'Компьютер 1'), (270011, 12442, 'Компьютер 2'), (374223, 2467, 'Компьютер 2'), (1001, 12000, 'Компьютер 2'), (323312, 147977, 'Компьютер 3'), (323312, 147977, 'Компьютер 4'), (664623, 2356, 'Компьютер 4'), (374223, 2467, 'Компьютер 6')]

....

Ran 5 tests in 0.000s

OK