ИУ5-34Б Матвеев Илья

Отчет РК1 по дисциплине
«Парадигмы и конструкции языков программирования»

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).

Текст программы

main.py

```
from operator import itemgetter


class Computer:
    """Компьютер"""

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


class OperatingSystem:
    """Операционная система"""

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


class ComputersOS:
    """
    'Компьютеры с ОС' для реализации
    связи многие-ко-многим
    """

    def __init__(self, os_id, comp_id):
        self.os_id = os_id
        self.comp_id = comp_id


def get_one_to_many(oses, computers):
    """Соединение данных один-ко-многим"""
    return [(c.name, c.price, os.name)
            for os in oses
            for c in computers
            if c.os_id == os.id]
```

```python
def get_many_to_many(oses, computers, computers_os):
    """Соединение данных многие-ко-многим"""
    many_to_many_temp = [(os.name, co.os_id, co.comp_id)
                         for os in oses
                         for co in computers_os
                         if os.id == co.os_id]

    return [(c.name, c.price, os_name)
            for os_name, os_id, comp_id in many_to_many_temp
            for c in computers if c.id == comp_id]


def task_1(oses, computers):
    """Задание 1"""
    selected_os = [os for os in oses if os.name.startswith('W')]
    return {os.name: [comp.name for comp in computers if comp.os_id == os.id]
for os in selected_os}


def task_2(oses, one_to_many):
    """Задание 2"""
    res_2_unsorted = []
    for os in oses:
        os_comps = list(filter(lambda i: i[2] == os.name, one_to_many))
        if os_comps:
            os_prices = [price for _, price, _ in os_comps]
            os_max_price = max(os_prices)
            res_2_unsorted.append((os.name, os_max_price))

    return sorted(res_2_unsorted, key=itemgetter(1), reverse=True)


def task_3(oses, computers, computers_os):
    """Задание 3"""
    result_3 = {os.name: [comp.name for relation in computers_os for comp in
computers if
                          relation.os_id == os.id and relation.comp_id ==
comp.id]
                for os in oses}

    return {os: result_3[os] for os in sorted(result_3.keys())}


def main():
    oses = [
        OperatingSystem(1, 'Windows'),
        OperatingSystem(2, 'Linux'),
        OperatingSystem(3, 'macOS'),
        OperatingSystem(11, 'Windows Server'),
        OperatingSystem(22, 'Ubuntu'),
    ]

    computers = [
        Computer(1, 'PC1', 1000, 1),
        Computer(2, 'PC2', 1200, 1),
        Computer(3, 'PC3', 900, 11),
        Computer(4, 'PC4', 1500, 3),
        Computer(5, 'PC5', 1100, 2),
    ]

    computers_os = [
        ComputersOS(1, 1),
```

```
        ComputersOS(1, 2),
        ComputersOS(2, 3),
        ComputersOS(2, 5),
        ComputersOS(3, 4),
        ComputersOS(11, 1),
        ComputersOS(22, 3),
    ]

    one_to_many = get_one_to_many(oses, computers)
    many_to_many = get_many_to_many(oses, computers, computers_os)

    print("\nTask 1")
    print(task_1(oses, computers))

    print("\nTask 2")
    print(task_2(oses, one_to_many))

    print("\nTask 3")
    print(task_3(oses, computers, computers_os))


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

test_computers_os.py

```
import unittest
from main import Computer, OperatingSystem, ComputersOS, get_one_to_many,
get_many_to_many, task_1, task_2, task_3


class TestComputers(unittest.TestCase):

    def setUp(self):
        self.oses = [
            OperatingSystem(1, 'Windows'),
            OperatingSystem(2, 'Linux'),
            OperatingSystem(3, 'macOS'),
            OperatingSystem(11, 'Windows Server'),
            OperatingSystem(22, 'Ubuntu'),
        ]

        self.computers = [
            Computer(1, 'PC1', 1000, 1),
            Computer(2, 'PC2', 1200, 1),
            Computer(3, 'PC3', 900, 11),
            Computer(4, 'PC4', 1500, 3),
            Computer(5, 'PC5', 1100, 2),
        ]

        self.computers_os = [
            ComputersOS(1, 1),
            ComputersOS(1, 2),
            ComputersOS(2, 3),
            ComputersOS(2, 5),
            ComputersOS(3, 4),
            ComputersOS(11, 1),
            ComputersOS(22, 3),
        ]

    def test_task_1(self):
        result = task_1(self.oses, self.computers)
        expected = {
```

```
            'Windows': ['PC1', 'PC2'],
            'Windows Server': ['PC3']
        }
        self.assertEqual(result, expected)

    def test_task_2(self):
        one_to_many = get_one_to_many(self.oses, self.computers)
        result = task_2(self.oses, one_to_many)
        expected = [('macOS', 1500), ('Windows', 1200), ('Linux', 1100),
('Windows Server', 900)]
        self.assertEqual(result, expected)

    def test_task_3(self):
        result = task_3(self.oses, self.computers, self.computers_os)
        expected = {
            'Linux': ['PC3', 'PC5'],
            'Ubuntu': ['PC3'],
            'Windows': ['PC1', 'PC2'],
            'Windows Server': ['PC1'],
            'macOS': ['PC4']
        }
        self.assertEqual(result, expected)


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

Анализ результатов

```
ilamatveev@MacBook-Pro-Ila rk1-PCPL % cd RK2
ilamatveev@MacBook-Pro-Ila RK2 % python3 -m unittest test_computers_os.py

...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
ilamatveev@MacBook-Pro-Ila RK2 %
```