Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Базовые компоненты интернет-технологий»
Отчет по рубежному контролю №2

Выполнил:
студент группы ИУ5-31Б
Кондрахин Сергей
Сергеевич

Проверил:
преподаватель каф.ИУ5
Гапанюк Юрий
Евгеньевич

Москва, 2021 г.

Задание

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

```
# используется для сортировки
from operator import itemgetter

class Prog:
    """Программа"""
    def __init__(self, id, name, price, comp_id):
        self.id = id
        self.name = name
        self.price = price
        self.comp_id = comp_id

class Comp:
    """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, 'компьютер Macbook'),
    Comp(2, 'компьютер Xiaomi'),
    Comp(3, 'компьютер Honor'),
    Comp(4, 'MSI'),

    Comp(11, 'компьютер (другой) Macbook'),
```

```
    Comp(22, '(другой) Xiaomi'),
    Comp(33, '(другой) Honor'),
    Comp(44, 'компьютер (другой) MSI'),
]

# Программы
progs = [
    Prog(1, 'Pycharm', 15000, 1),
    Prog(2, 'Visual Studio', 40000, 2),
    Prog(3, 'Inventor', 60000, 3),
    Prog(4, 'Access', 10000, 4),
    Prog(5, 'Workbench', 35000, 4),
]

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

    ProgComp(11, 1),
    ProgComp(22, 2),
    ProgComp(33, 3),
    ProgComp(44, 4),
    ProgComp(44, 5),
]
def sort_by_name(dat):
    return sorted(dat, key=itemgetter(2))

def sort_by_price_progs(dat, comps):
    result_unsorted = []
    # Перебираем все компьютеры
    for c in comps:
        # Список программ компьютера
        com_progs = list(filter(lambda i: i[2] == c.name, dat))
        # Если компьютер не пустой
        if len(com_progs) > 0:
            # Стоимость программы на компьютере
            com_prices = [price for _, price, _ in com_progs]
            # Суммарная стоимость программ на компьютере
            com_price_sum = sum(com_prices)
            result_unsorted.append((c.name, com_price_sum))
    # Сортировка по суммарной стоимости
    return sorted(result_unsorted, key=itemgetter(1), reverse=True)

def out_put_progs_of_comp(dat, comps):
    result = {}
    # Перебираем все компьютеры
    for c in comps:
```

```
        if 'компьютер' in c.name:
            # Список программ на компьютере
            com_progs = list(filter(lambda i: i[2] == c.name, dat))
            # Только название программы
            com_progs_name = [x for x, _, _ in com_progs]
            # Добавляем результат в словарь
            # ключ - компьютер, значение - список программ
            result[c.name] = com_progs_name
    return result

def main():
    """Основная функция"""
    # Соединение данных один-ко-многим
    one_to_many = [(p.name, p.price, 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.price, comp_name)
                    for comp_name, comp_id, prog_id in many_to_many_temp
                    for p in progs if p.id == prog_id]

    print('Задание 1')
    res_1 = sort_by_name(one_to_many)
    print(res_1)

    print('\nЗадание 2')
    res_2 = sort_by_price_progs(one_to_many, comps)
    print(res_2)

    print('\nЗадание 3')
    res_3 = out_put_progs_of_comp(many_to_many, comps)
    print(res_3)

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

Tests.py

```
from main import Prog, Comp, ProgComp, sort_by_name, sort_by_price_progs,
out_put_progs_of_comp
import unittest

class Tests(unittest.TestCase):
    def setUp(self) -> None:
        # Компьютеры
```

```
        self.comps = [
            Comp(1, 'компьютер Macbook'),
            Comp(2, 'компьютер Xiaomi'),
            Comp(3, 'компьютер Honor'),
            Comp(4, 'MSI'),

            Comp(11, 'компьютер (другой) Macbook'),
            Comp(22, '(другой) Xiaomi'),
            Comp(33, '(другой) Honor'),
            Comp(44, 'компьютер (другой) MSI'),
        ]

        # Программы
        self.progs = [
            Prog(1, 'Pycharm', 15000, 1),
            Prog(2, 'Visual Studio', 40000, 2),
            Prog(3, 'Inventor', 60000, 3),
            Prog(4, 'Access', 10000, 4),
            Prog(5, 'Workbench', 35000, 4),
        ]

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

            ProgComp(11, 1),
            ProgComp(22, 2),
            ProgComp(33, 3),
            ProgComp(44, 4),
            ProgComp(44, 5),
        ]
        # Соединение данных один-ко-многим
        self.one_to_many = [(p.name, p.price, c.name)
                            for c in self.comps
                            for p in self.progs
                            if p.comp_id == c.id]
        # Соединение данных многие-ко-многим
        self.many_to_many_temp = [(c.name, pc.comp_id, pc.prog_id)
                                  for c in self.comps
                                  for pc in self.progs_comps
                                  if c.id == pc.comp_id]
        self.many_to_many = [(p.name, p.price, comp_name)
                             for comp_name, comp_id, prog_id in self.many_to_many_temp
                             for p in self.progs if p.id == prog_id]

    def test_sort_by_name(self):
        result = sort_by_name(self.one_to_many)
```

```
        desired_result = [('Access', 10000, 'MSI'), ('Workbench', 35000, 'MSI'), ('Inventor', 60000,
'компьютер Honor'), ('Pycharm', 15000, 'компьютер Macbook'), ('Visual Studio', 40000, 'компьютер
Xiaomi')]
        self.assertEqual(desired_result, result)

    def test_sort_by_price_progs(self):
        result = sort_by_price_progs(self.one_to_many, self.comps)
        desired_result = [('компьютер Honor', 60000), ('MSI', 45000), ('компьютер Xiaomi', 40000),
('компьютер Macbook', 15000)]
        self.assertEqual(desired_result, result)

    def test_out_put_progs_of_comp(self):
        result = out_put_progs_of_comp(self.many_to_many, self.comps)
        desired_result = {'компьютер Macbook': ['Pycharm'], 'компьютер Xiaomi': ['Visual Studio'],
'компьютер Honor': ['Inventor'], 'компьютер (другой) Macbook': ['Pycharm'], 'компьютер (другой)
MSI': ['Access', 'Workbench']}
        self.assertEqual(desired_result, result)
```

Пример выполнения программы

```
Testing started at 15:05 ...


Ran 3 tests in 0.003s
Launching unittests with arguments python -m unittest

OK
```