Московский государственный технический

университет им. Н.Э. Баумана

Отчёт по рубежному контролю №2
по курсу «Парадигмы и конструкции языков программирования»
Вариант Е11.

Проверил:
Преподаватель каф. ИУ5
Гапанюк Ю. Е.

Выполнил:
Студент группы РТ5-31Б
Иванов А. А.

Москва, 2024 г.

Рубежный контроль представляет собой разработку тестов на языке Python.

Цели:

- Провести рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
- Для текста программы рубежного контроля №1 необходимо создать модульные тесты с применением TDD - фреймворка (3 теста).

Текст программы:

```
from operator import itemgetter

class Program:
    """Программа"""

    def __init__(self, id, name, price, computer_id=None):
        self.id = id
        self.name = name
        self.price = price
        self.computer_id = computer_id

class Computer:
    """Компьютер"""

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

class ComputerProgram:
    """
    'Компьютеры-программы' для реализации связи многие-ко-многим
    """

    def __init__(self, computer_id, program_id):
        self.computer_id = computer_id
        self.program_id = program_id


def res1(computers, programs, computers_programs):
    """Список компьютеров с названием 'Компьютер' и программами"""

    sort_computers = [c for c in computers if 'Компьютер' in c.name]
    comp_progs = {}

    for comp in sort_computers:
        progs_otm = [p.name for p in programs if p.computer_id == comp.id]
        progs_mtm = [
            p.name for cp in computers_programs
            if cp.computer_id == comp.id
```

```
            for p in programs if p.id == cp.program_id
        ]
        all_progs = sorted(set(progs_otm + progs_mtm))#
        comp_progs[comp.name] = all_progs

    return comp_progs


def res2(computers, programs, computers_programs):
    """Список компьютеров со средней ценой программ, отсортированный по средней
цене"""

    computer_avg_prices = []

    for comp in computers:
        prog_otm = [p.price for p in programs if p.computer_id == comp.id]
        prog_mtm = [
            p.price for cp in computers_programs
            if cp.computer_id == comp.id
            for p in programs if p.id == cp.program_id
        ]
        all_prices = prog_otm + prog_mtm

        if all_prices:
            avg_price = round(sum(all_prices) / len(all_prices), 2)
            computer_avg_prices.append((comp.name, avg_price))
    comp_avg_prices_sort = sorted(computer_avg_prices, key=itemgetter(1))

    return comp_avg_prices_sort


def res3(computers, programs, computers_programs):
    """Список программ, начинающихся с 'A', и компьютеров"""

    sort_programs = [p for p in programs if p.name.startswith('A')]
    comp_dict = {c.id: c.name for c in computers}
    prog_comps = {}

    for program in sort_programs:
        comp_list = []
        if program.computer_id:
            comp_list.append(comp_dict.get(program.computer_id))

        for cp in computers_programs:
            if cp.program_id == program.id:
                comp_list.append(comp_dict.get(cp.computer_id))

        prog_comps[program.name] = sorted(list(filter(None, set(comp_list))))#

    return prog_comps
```

```python
# Модульные тесты

import unittest

class TestProgramFunctions(unittest.TestCase):

    def setUp(self):
        self.computers = [
            Computer(1, 'Компьютер Alpha'),
            Computer(2, 'Компьютер Beta'),
            Computer(3, 'Компьютер Gamma'),
            Computer(4, 'Персональный Компьютер Delta')
        ]
        self.programs = [
            Program(1, 'Антивирус Kaspersky', 500, 1),
            Program(2, 'Браузер Yandex', 150, 2),
            Program(3, 'Графический редактор Adobe', 1200, 3),
            Program(4, 'Текстовый редактор Word', 300, 1),
            Program(5, 'Античит Faceit', 800, 4)
        ]
        self.computers_programs = [
            ComputerProgram(1, 1),
            ComputerProgram(1, 4),
            ComputerProgram(2, 2),
            ComputerProgram(3, 3),
            ComputerProgram(4, 5),
            ComputerProgram(2, 5),
            ComputerProgram(3, 1),
            ComputerProgram(4, 3)
        ]

    def test_res1(self):
        """Тест для функции res1"""
        result = res1(self.computers, self.programs, self.computers_programs)
        expected = {
            'Компьютер Alpha': ['Антивирус Kaspersky', 'Текстовый редактор
Word'],
            'Компьютер Beta': ['Античит Faceit', 'Браузер Yandex'],
            'Компьютер Gamma': ['Антивирус Kaspersky', 'Графический редактор
Adobe'],
            'Персональный Компьютер Delta': ['Античит Faceit', 'Графический
редактор Adobe']
        }
        self.assertEqual(result, expected)

    def test_res2(self):
        """Тест для функции res2"""
        result = res2(self.computers, self.programs, self.computers_programs)
        expected = [
            ('Компьютер Beta', 366.67),
            ('Компьютер Alpha', 400.0),
```

```
            ('Персональный Компьютер Delta', 933.33),
            ('Компьютер Gamma', 966.67)
        ]
        self.assertEqual(result, expected)

    def test_res3(self):
        """Тест для функции res3"""
        result = res3(self.computers, self.programs, self.computers_programs)
        expected = {
            'Антивирус Kaspersky': ['Компьютер Alpha', 'Компьютер Gamma'],
            'Античит Faceit': ['Компьютер Beta', 'Персональный Компьютер Delta']
        }
        self.assertEqual(result, expected)

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

Вывод программы:

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```