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 РК№2
Вариант 11

Выполнил:

студент группы ИУ5-31Б

Карпова К.П.

Подпись и дата:

Проверил:

преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2023г.

Постановка задачи

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

```
from operator import itemgetter
class Program:
    def __init__(self, id, name, comp_id, size_in_gb):
        self.id = id
        self.name = name
        self.comp_id = comp_id
        self.size_in_gb = size_in_gb

class Computer:
    def __init__(self, id, model, RAM, owner):
        self.id = id
        self.model = model
        self.RAM = RAM
        self.owner = owner

class ProgramComputerLink:
    def __init__(self, pr_id, comp_id):
        self.pr_id = pr_id
        self.comp_id = comp_id

def generate_one_to_many(programs, computers):
    return [(p.name, p.size_in_gb, c.owner, c.model) for p in programs for c in computers if p.comp_id == c.id]

def generate_many_to_many(programs, program_computer_links, computers):
    many_to_many_temp = [(p.name, p_c.pr_id, p_c.comp_id) for p_c in program_computer_links for p in programs if
p.id == p_c.pr_id]
    return [(pr_name, c.model, c.owner) for pr_name, _, p_c_id in many_to_many_temp for c in computers if c.id ==
p_c_id]

def filter_programs_by_name(programs, name_substring):
    return [p for p in programs if name_substring in p.name]

def main():
    programs = [
        Program(1, 'Microsoft Word', 1, 2.0),
        Program(2, 'Microsoft Excel', 1, 1.5),
        Program(3, 'Google Docs', 3, 1.0),
        Program(4, 'LibreOffice Writer', 1, 1.8),
        Program(5, 'OpenOffice Calc', 4, 1.2)
```

```
    ]

    computers = [
        Computer(1, 'Model A-2000X', '8 GB DDR4', 'Иванов Иван Иванович'),
        Computer(2, 'UltraBook Pro 15S', '16 GB DDR4', 'Петрова Анна Сергеевна'),
        Computer(3, 'GamingBeast X9000', '32 GB DDR4', 'Смирнов Сергей Владимирович'),
        Computer(4, 'OfficeMaster 500', '64 GB DDR4', 'Козлова Екатерина Павловна'),
        Computer(5, 'PerformanceElite 3000', '128 GB DDR4', 'Михайлов Алексей Дмитриевич')
    ]

    pr_comp = [ProgramComputerLink(1, 1), ProgramComputerLink(1, 2), ProgramComputerLink(1, 4),
               ProgramComputerLink(2, 3), ProgramComputerLink(2, 1),
               ProgramComputerLink(3, 4), ProgramComputerLink(4, 5),
               ProgramComputerLink(3, 5)]

    one_to_many_sorted = sorted(generate_one_to_many(programs, computers), key=itemgetter(0))
    many_to_many = generate_many_to_many(programs, pr_comp, computers)

    one_to_many = [(p.name, p.size_in_gb, c.owner, c.model)
                   for p in programs
                   for c in computers
                   if p.comp_id == c.id]

    many_to_many_temp = [(p.name, p_c.pr_id, p_c.comp_id)
                         for p_c in pr_comp
                         for p in programs
                         if p.id == p_c.pr_id]

    return one_to_many_sorted, many_to_many, many_to_many_temp


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

Программа тестов

```
import unittest
from main import generate_one_to_many, generate_many_to_many, filter_programs_by_name, Program, Computer,
ProgramComputerLink

class TestProgramManagement(unittest.TestCase):
    def setUp(self):
        self.programs = [
            Program(1, 'Microsoft Word', 1, 2.0),
            Program(2, 'Microsoft Excel', 1, 1.5),
            Program(3, 'Google Docs', 3, 1.0),
            Program(4, 'LibreOffice Writer', 1, 1.8),
            Program(5, 'OpenOffice Calc', 4, 1.2)
        ]

        self.computers = [
            Computer(1, 'Model A-2000X', '8 GB DDR4', 'Иванов Иван Иванович'),
            Computer(2, 'UltraBook Pro 15S', '16 GB DDR4', 'Петрова Анна Сергеевна'),
            Computer(3, 'GamingBeast X9000', '32 GB DDR4', 'Смирнов Сергей Владимирович'),
            Computer(4, 'OfficeMaster 500', '64 GB DDR4', 'Козлова Екатерина Павловна'),
```

```
            Computer(5, 'PerformanceElite 3000', '128 GB DDR4', 'Михайлов Алексей Дмитриевич')
        ]

        self.program_computer_links = [
            ProgramComputerLink(1, 1),
            ProgramComputerLink(1, 2),
            ProgramComputerLink(2, 3),
            ProgramComputerLink(2, 1),
            ProgramComputerLink(3, 4),
            ProgramComputerLink(4, 5),
            ProgramComputerLink(3, 5)
        ]

    def test_generate_one_to_many(self):
        result = generate_one_to_many(self.programs, self.computers)
        # Add assertions based on the expected output

    def test_generate_many_to_many(self):
        result = generate_many_to_many(self.programs, self.program_computer_links, self.computers)
        # Add assertions based on the expected output

    def test_filter_programs_by_name(self):
        result = filter_programs_by_name(self.programs, 'Microsoft')
        # Add assertions based on the expected output

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

Описание тестов

1. `test_generate_one_to_many`: Этот тест проверяет корректность работы функции `generate_one_to_many`, которая создает отношение "один ко многим" между программами и компьютерами.
2. `test_generate_many_to_many`: Этот тест проверяет корректность работы функции `generate_many_to_many`, которая создает отношение "многие ко многим" между программами, программно-компьютерными связями и компьютерами.
3. `test_filter_programs_by_name`: Этот тест проверяет корректность работы функции `filter_programs_by_name`, которая фильтрует программы по подстроке в их названии.

Результаты Тестов

```
Ran 3 tests in 0.002s

OK
```

Вывод: Все тесты успешно выполнены