Московский государственный технический университет им. Н.Э. Баумана

Факультет информатика и системы управления
Кафедра системы обработки информации и управления

Курс «Парадигмы и конструкции языков программирования»
Отчет по рубежному контролю №2
Вариант Б7

Выполнил:
Студент группы ИУ5-32Б:
Купрюшин Егор А.
Подпись и дата:

Проверил:
Преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2025 г.

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

rk2.py

```python
class Microprocessor:
    def __init__(self, id, model, clock_speed_ghz, computer_id):
        self.id = id
        self.model = model
        self.clock_speed_ghz = clock_speed_ghz
        self.computer_id = computer_id

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

class MicroprocessorComputer:
    def __init__(self, computer_id, microprocessor_id):
        self.computer_id = computer_id
        self.microprocessor_id = microprocessor_id

computers = [
    Computer(1, "Игровой ПК 'Аврора'"),
    Computer(2, "Игровой ПК 'Счастье'"),
    Computer(3, "Ноутбук 'HP'"),
    Computer(4, "Ноутбук 'Титан'"),
]

microprocessors = [
    Microprocessor(1, "Intel Core i9-13900X", 5.8, 1),
    Microprocessor(2, "AMD Ryzen 9 7950X", 5.7, 1),
    Microprocessor(3, "Intel Core i7-13700", 5.2, 2),
    Microprocessor(4, "AMD Ryzen 7 7800X3D", 5.0, 3),
    Microprocessor(5, "Intel Xeon W-3400", 4.8, 4),
    Microprocessor(6, "AMD EPYC 9654", 3.7, 4),
]

microprocessors_computers = [
    MicroprocessorComputer(1, 1),
    MicroprocessorComputer(1, 2),
    MicroprocessorComputer(2, 3),
    MicroprocessorComputer(3, 4),
    MicroprocessorComputer(4, 5),
    MicroprocessorComputer(4, 6),
    MicroprocessorComputer(1, 3),
    MicroprocessorComputer(2, 1),
]

def get_one_to_many(procs, comps):
    """Соединение данных один-ко-многим"""
    return [(proc.model, proc.clock_speed_ghz, comp.name)
            for proc in procs
            for comp in comps
            if proc.computer_id == comp.id]

def task_b1(procs, comps):
    """Запрос Б1: Список всех связанных микропроцессоров и компьютеров, сортировка по модели"""
```

```
    one_to_many = get_one_to_many(procs, comps)
    return sorted(one_to_many, key=lambda x: x[0])

def task_b2(procs, comps):
    """Запрос Б2: Список компьютеров с количеством процессоров, сортировка по количеству"""
    one_to_many = get_one_to_many(procs, comps)
    result = []
    for comp in comps:
        procs_in_comp = list(filter(lambda x: x[2] == comp.name, one_to_many))
        if len(procs_in_comp) > 0:
            result.append((comp.name, len(procs_in_comp)))

    return sorted(result, key=lambda x: x[1])

def task_b3(procs, comps, links):
    """Запрос Б3: Список процессоров, заканчивающихся на 'X' и их компьютеров (M:M)"""
    many_to_many_first = [[comp.name, mc.computer_id, mc.microprocessor_id]
                          for comp in comps
                          for mc in links
                          if comp.id == mc.computer_id]

    many_to_many = [[proc.model, comp_name]
                    for comp_name, comp_id, proc_id in many_to_many_first
                    for proc in procs
                    if proc.id == proc_id]

    result = []
    for model, comp_name in many_to_many:
        if model.endswith("X"):
            result.append((model, comp_name))

    return sorted(result, key=lambda x: x[0])


def main():
    print("--- Запрос Б1 ---")
    result_b1 = task_b1(microprocessors, computers)
    for i in result_b1:
        print(f"  Процессор: {i[0]}, Частота: {i[1]} ГГц, Компьютер: {i[2]}")

    print("\n--- Запрос Б2 ---")
    result_b2 = task_b2(microprocessors, computers)
    for i in result_b2:
        print(f"  Компьютер: {i[0]}, Количество процессоров: {i[1]}")

    print("\n--- Запрос Б3 ---")
    result_b3 = task_b3(microprocessors, computers, microprocessors_computers)
    for i in result_b3:
        print(f"  Процессор: {i[0]}, Компьютер: {i[1]}")

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

rk2_test.py

```
import unittest
from rk2 import Microprocessor, Computer, MicroprocessorComputer, task_b1, task_b2, task_b3

class TestRK2(unittest.TestCase):
    def setUp(self):
        self.computers = [
            Computer(1, "Comp A"),
            Computer(2, "Comp B"),
            Computer(3, "Comp C (Empty)")
        ]

        self.microprocessors = [
            Microprocessor(1, "Proc-100X", 3.5, 1),
            Microprocessor(2, "Proc-200", 4.0, 1),
            Microprocessor(3, "Proc-300X", 2.5, 2),
        ]
```

```
        self.links = [
            MicroprocessorComputer(1, 1),
            MicroprocessorComputer(1, 2),
            MicroprocessorComputer(2, 3),
        ]

    def test_task_b1(self):
        expected = [
            ("Proc-100X", 3.5, "Comp A"),
            ("Proc-200", 4.0, "Comp A"),
            ("Proc-300X", 2.5, "Comp B")
        ]
        result = task_b1(self.microprocessors, self.computers)
        self.assertEqual(result, expected)

    def test_task_b2(self):
        expected = [
            ("Comp B", 1),
            ("Comp A", 2)
        ]
        result = task_b2(self.microprocessors, self.computers)
        self.assertEqual(result, expected)

    def test_task_b3(self):
        expected = [
            ("Proc-100X", "Comp A"),
            ("Proc-300X", "Comp B")
        ]
        result = task_b3(self.microprocessors, self.computers, self.links)
        self.assertEqual(result, expected)

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

Скриншот работы приложения

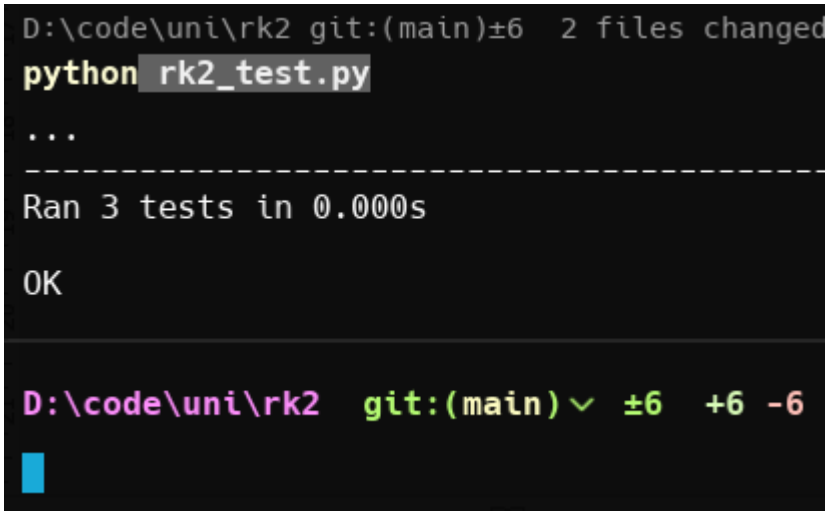

Рисунок 1. Вывод результатов программы