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 Рубежному контролю №2

Выполнил:
студент группы ИУ5-33Б
Власов Александр

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2023 г.

Вариант запросов В. Предметная область 7.

1. «Микросхема» и «Компьютер» связаны соотношением один-комногим. Выведите список всех компьютеров, у которых название производителя начинается с буквы «А», и микросхемы, которые в них установлены.

2. «Микросхема» и «Компьютер» связаны соотношением один-комногим. Выведите список компьютеров с наименьшем количеством ядер, отсортированный по минимальному числу ядер.

3. «Микросхема» и «Компьютер» связаны соотношением многие-комногим. Выведите список всех связанных компьютеров и микросхем, отсортированный по компьютерам, сортировка по микросхемам произвольная.

Условия рубежного контроля №2 по курсу ПиКЯП:

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).

Код программы
Main.py

```
class MicroScheme:
    def __init__(self, id: int, company: str,  m_id: int, core: int):
        self._id = id
        self._company = company
        self._m_id = m_id
        self._core = core

    @property
    def id(self) -> int:
        return self._id

    @property
    def m_id(self) -> int:
        return self._m_id



    @property
    def core(self) -> int:
        return self._core


class Computer:
    def __init__(self, id: int, name: str):
        self._id = id
        self._name = name

    @property
```

```
        def id(self) -> int:
            return self._id

        @property
        def name(self) -> str:
            return self._name


    class MicroSchemeComputer:
        def __init__(self, MicroScheme_id: int, m_id: int):
            self._MicroScheme_id = MicroScheme_id
            self._m_id = m_id

        @property
        def m_id(self) -> int:
            return self._m_id

        @property
        def MicroScheme_id(self) -> int:
            return self._MicroScheme_id


    def task1(Computers: list[Computer], MicroSchemes: list[MicroScheme]):
        print("Запрос 1")
        data = [(a, b) for a in MicroSchemes for b in Computers if a.m_id ==
b.id and a._company.startswith("A")]
        for (a, b) in data:
            print(a._company, b.name)
        print()


    def task2(Computers: list[Computer], MicroSchemes: list[MicroScheme]):
        print("Запрос 2")
        data = {}
        for Computer in Computers:
            Computer_core = [a.core for a in MicroSchemes for b in Computers
if a.m_id == b.id and b.id == Computer.id]
            data[Computer.name] = min(Computer_core)

        data_items = list(data.items())
        data_items.sort(key=lambda x: x[1])
        for (Computer, min_core) in data_items:
            print(Computer, min_core)
        print()


    def task3(Computers: list[Computer], MicroSchemes: list[MicroScheme],
MicroSchemes_Computers: list[MicroSchemeComputer]):
        print("Запрос 3")
        data = [(a, b) for ab in MicroSchemes_Computers for a in
MicroSchemes for b in Computers if ab.MicroScheme_id == a.id and ab.m_id ==
b.id]

        data.sort(key=lambda x: x[0]._company)

        for (MicroScheme, Computer) in data:
            print(MicroScheme._company, Computer.name)
        print()


    def main():
        Computers = [
            Computer(1, "Enigma"),
            Computer(2, "Altair-8800"),
```

```
        Computer(3, "Agat"),
        Computer(4, "Macintosh"),
        Computer(5, "Datapoint-2200")
    ]

    MicroSchemes = [
        MicroScheme(1, "BAIKAL", 1, 24),
        MicroScheme(2, "BAIKAL", 1, 20),
        MicroScheme(3, "AMD", 2, 20),
        MicroScheme(4, "BAIKAL", 2, 16),
        MicroScheme(5, "BAIKAL", 3, 8),

        MicroScheme(6, "BAIKAL", 3, 24),
        MicroScheme(7, "AMD", 4, 24),
        MicroScheme(8, "BAIKAL", 4, 8),
        MicroScheme(9, "AMD", 5, 16)
    ]

    MicroSchemes_Computers = [
        MicroSchemeComputer(1, 1),
        MicroSchemeComputer(1, 2),
        MicroSchemeComputer(1, 4),
        MicroSchemeComputer(2, 1),
        MicroSchemeComputer(3, 2),
        MicroSchemeComputer(4, 4),
        MicroSchemeComputer(5, 5),
        MicroSchemeComputer(9, 3)
    ]

    task1(Computers, MicroSchemes)
    task2(Computers, MicroSchemes)
    task3(Computers, MicroSchemes, MicroSchemes_Computers)


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

Test.py

```
import unittest

class TestMicroSchemeFunctions(unittest.TestCase):

    def setUp(self):
        self.computers = [
            Computer(1, "Enigma"),
            Computer(2, "Altair-8800"),
            Computer(3, "Agat"),
            Computer(4, "Macintosh"),
            Computer(5, "Datapoint-2200")
        ]

        self.microschemes = [
            MicroScheme(1, "BAIKAL", 1, 24),
            MicroScheme(2, "BAIKAL", 1, 20),
            MicroScheme(3, "AMD", 2, 20),
            MicroScheme(4, "BAIKAL", 2, 16),
            MicroScheme(5, "BAIKAL", 3, 8),
            MicroScheme(6, "BAIKAL", 3, 24),
            MicroScheme(7, "AMD", 4, 24),
            MicroScheme(8, "BAIKAL", 4, 8),
            MicroScheme(9, "AMD", 5, 16)
```

```
        ]

        self.microscheme_computers = [
            MicroSchemeComputer(1, 1),
            MicroSchemeComputer(1, 2),
            MicroSchemeComputer(1, 4),
            MicroSchemeComputer(2, 1),
            MicroSchemeComputer(3, 2),
            MicroSchemeComputer(4, 4),
            MicroSchemeComputer(5, 5),
            MicroSchemeComputer(9, 3)
        ]
```

```
          def test_task1(self):
              expected_output = [('BAIKAL', 'Enigma'), ('BAIKAL',
'Altair-8800')]
              result = task1(self.computers, self.microschemes)
              # Add your assertion here to compare the result with the
expected output

          def test_task2(self):
              expected_output = [('Agat', 8), ('Altair-8800', 16),
('Datapoint-2200', 8), ('Macintosh', 24), ('Enigma', 8)]
              result = task2(self.computers, self.microschemes)
              # Add your assertion here to compare the result with the
expected output

          def test_task3(self):
              expected_output = [('AMD', 'Agat'), ('AMD', 'Enigma'),
('AMD', 'Macintosh'), ('BAIKAL', 'Altair-8800'), ('BAIKAL', 'Datapoint-
2200')]
              result = task3(self.computers, self.microschemes,
self.microscheme_computers)
              # Add your assertion here to compare the result with the
expected output

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

Анализ результатов