МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра ИУ5 “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по рубежному контролю №2
“Модульное тестирование”

Выполнил:
Студент группы ИУ5-33Б
Просвирякова Д. С.
Преподаватель:
Гапанюк Ю. Е.

Москва 2025

Листинг кода

```
from operator import itemgetter

class Computer:
    def __init__(self, id, model, ram_gb, classroom_id):
        self.id = id
        self.model = model
        self.ram_gb = ram_gb
        self.classroom_id = classroom_id

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

class ComputerClassroom:
    def __init__(self, computer_id, classroom_id):
        self.computer_id = computer_id
        self.classroom_id = classroom_id

class DataManager:
    def __init__(self):
        self.classrooms = [
            Classroom(1, "Основной компьютерный класс"),
            Classroom(2, "Графическая лаборатория"),
            Classroom(3, "Программирование"),
            Classroom(4, "Мультимедийный центр"),
        ]

        self.computers = [
            Computer(1, "Dell Optiplex", 16, 1),
            Computer(2, "HP EliteDesk", 32, 1),
            Computer(3, "Lenovo ThinkCentre", 8, 2),
            Computer(4, "Apple iMac", 16, 2),
            Computer(5, "Asus ProArt", 64, 3),
            Computer(6, "Acer Aspire", 8, 3),
            Computer(7, "MSI Creator", 32, 4),
            Computer(8, "Acer Predator", 16, 2),
        ]

        self.computer_classrooms = [
            ComputerClassroom(1, 1),
            ComputerClassroom(2, 1),
            ComputerClassroom(3, 2),
            ComputerClassroom(4, 2),
            ComputerClassroom(5, 3),
            ComputerClassroom(6, 3),
            ComputerClassroom(7, 4),
            ComputerClassroom(8, 2),
            ComputerClassroom(1, 2),
            ComputerClassroom(5, 4),
            ComputerClassroom(8, 1),
        ]
```

```
    def get_one_to_many(self):
        return [
            (c.model, c.ram_gb, r.name)
            for r in self.classrooms
            for c in self.computers
            if c.classroom_id == r.id
        ]

    def get_many_to_many(self):
        many_to_many_temp = [
            (r.name, cr.classroom_id, cr.computer_id)
            for r in self.classrooms
            for cr in self.computer_classrooms
            if r.id == cr.classroom_id
        ]

        return [
            (c.model, c.ram_gb, classroom_name)
            for classroom_name, classroom_id, computer_id in many_to_many_temp
            for c in self.computers if c.id == computer_id
        ]

    def task1_computers_starting_with_A(self):
        one_to_many = self.get_one_to_many()

        result = [
            (model, ram, classroom_name)
            for model, ram, classroom_name in one_to_many
            if model.startswith('A')
        ]

        return sorted(result, key=itemgetter(0))

    def task2_classrooms_min_ram(self):
        one_to_many = self.get_one_to_many()

        min_ram_by_classroom = {}

        for model, ram, classroom_name in one_to_many:
            if classroom_name not in min_ram_by_classroom:
                min_ram_by_classroom[classroom_name] = ram
            else:
                min_ram_by_classroom[classroom_name] =
min(min_ram_by_classroom[classroom_name], ram)

        result = sorted(
            [(classroom_name, min_ram) for classroom_name, min_ram in
min_ram_by_classroom.items()],
            key=itemgetter(1)
        )

        return result
```

```
    def task3_all_relationships_sorted(self):
        many_to_many = self.get_many_to_many()

        result = sorted(many_to_many, key=itemgetter(0))

        return result

def main():
    manager = DataManager()

    print("Предметная область: Компьютер – Дисплейный класс")
    print("Вариант B")

    print("\n1. Компьютеры, модель которых начинается с буквы 'A':")
    result1 = manager.task1_computers_starting_with_A()
    if result1:
        for model, ram, classroom_name in result1:
            print(f"Компьютер: {model}, RAM: {ram}GB, Класс: {classroom_name}")
    else:
        print("Компьютеры с моделями на 'A' не найдены")

    print("\n2. Дисплейные классы с минимальным RAM компьютеров:")
    result2 = manager.task2_classrooms_min_ram()
    for classroom_name, min_ram in result2:
        print(f"Класс: {classroom_name}, Мин. RAM: {min_ram}GB")

    print("\n3. Все связи компьютеров и дисплейных классов (отсортировано по
компьютерам):")
    result3 = manager.task3_all_relationships_sorted()
    for model, ram, classroom_name in result3:
        print(f"Компьютер: {model} ({ram}GB) -> Класс: {classroom_name}")

if __name__ == '__main__':
    main()
import unittest
from rk2 import DataManager

class TestComputerClassroomSystem(unittest.TestCase):

    def setUp(self):
        self.manager = DataManager()

    def test_task1_computers_starting_with_A(self):
        result = self.manager.task1_computers_starting_with_A()

        self.assertIsNotNone(result)
        self.assertGreater(len(result), 0)

        for model, ram, classroom_name in result:
            self.assertTrue(model.startswith('A'))
```

```
        expected_models = ['Acer Aspire', 'Acer Predator', 'Apple iMac', 'Asus
ProArt']
        actual_models = [item[0] for item in result]

        for expected_model in expected_models:
            self.assertIn(expected_model, actual_models)

        sorted_models = sorted(actual_models)
        self.assertEqual(actual_models, sorted_models)

    def test_task2_classrooms_min_ram(self):
        result = self.manager.task2_classrooms_min_ram()

        self.assertIsNotNone(result)
        self.assertGreater(len(result), 0)

        for classroom_name, min_ram in result:
            self.assertIsInstance(classroom_name, str)
            self.assertIsInstance(min_ram, int)
            self.assertGreater(min_ram, 0)

        ram_values = [item[1] for item in result]
        sorted_ram_values = sorted(ram_values)
        self.assertEqual(ram_values, sorted_ram_values)

        expected_results = [
            ("Графическая лаборатория", 8),
            ("Программирование", 8),
            ("Основной компьютерный класс", 16),
            ("Мультимедийный центр", 32)
        ]

        for classroom_name, expected_min_ram in expected_results:
            found = False
            for actual_classroom, actual_min_ram in result:
                if actual_classroom == classroom_name:
                    self.assertEqual(actual_min_ram, expected_min_ram)
                    found = True
                    break
            self.assertTrue(found)

    def test_task3_all_relationships_sorted(self):
        result = self.manager.task3_all_relationships_sorted()

        self.assertIsNotNone(result)
        self.assertGreater(len(result), 0)

        for model, ram, classroom_name in result:
            self.assertIsInstance(model, str)
            self.assertIsInstance(ram, int)
            self.assertIsInstance(classroom_name, str)
            self.assertGreater(ram, 0)
```

```
        models = [item[0] for item in result]
        sorted_models = sorted(models)
        self.assertEqual(models, sorted_models)

        expected_relationships = [
            ("Acer Aspire", 8, "Программирование"),
            ("Acer Predator", 16, "Графическая лаборатория"),
            ("Dell Optiplex", 16, "Основной компьютерный класс"),
        ]

        for expected_model, expected_ram, expected_classroom in
expected_relationships:
            found = False
            for actual_model, actual_ram, actual_classroom in result:
                if (actual_model == expected_model and
                    actual_ram == expected_ram and
                    actual_classroom == expected_classroom):
                    found = True
                    break
            self.assertTrue(found)

if __name__ == '__main__':
    unittest.main(verbosity=2)
```

Скриншот работы приложения

```
daraprosvirakova@MacBook-Air-Dara-2 Prosviryakova-PCPL % python3 rk2.py
Предметная область: Компьютер - Дисплейный класс
Вариант В

1. Компьютеры, модель которых начинается с буквы 'A':
Компьютер: Acer Aspire, RAM: 8GB, Класс: Программирование
Компьютер: Acer Predator, RAM: 16GB, Класс: Графическая лаборатория
Компьютер: Apple iMac, RAM: 16GB, Класс: Графическая лаборатория
Компьютер: Asus ProArt, RAM: 64GB, Класс: Программирование

2. Дисплейные классы с минимальным RAM компьютеров:
Класс: Графическая лаборатория, Мин. RAM: 8GB
Класс: Программирование, Мин. RAM: 8GB
Класс: Основной компьютерный класс, Мин. RAM: 16GB
Класс: Мультимедийный центр, Мин. RAM: 32GB

3. Все связи компьютеров и дисплейных классов (отсортировано по компьютерам):
Компьютер: Acer Aspire (8GB) -> Класс: Программирование
Компьютер: Acer Predator (16GB) -> Класс: Основной компьютерный класс
Компьютер: Acer Predator (16GB) -> Класс: Графическая лаборатория
Компьютер: Apple iMac (16GB) -> Класс: Графическая лаборатория
Компьютер: Asus ProArt (64GB) -> Класс: Программирование
Компьютер: Asus ProArt (64GB) -> Класс: Мультимедийный центр
Компьютер: Dell Optiplex (16GB) -> Класс: Основной компьютерный класс
Компьютер: Dell Optiplex (16GB) -> Класс: Графическая лаборатория
Компьютер: HP EliteDesk (32GB) -> Класс: Основной компьютерный класс
Компьютер: Lenovo ThinkCentre (8GB) -> Класс: Графическая лаборатория
Компьютер: MSI Creator (32GB) -> Класс: Мультимедийный центр
daraprosvirakova@MacBook-Air-Dara-2 Prosviryakova-PCPL % python3 tests.py
test_task1_computers_starting_with_A (__main__.TestComputerClassroomSystem.test_task1_computers_starting_with_A) ... ok
test_task2_classrooms_min_ram (__main__.TestComputerClassroomSystem.test_task2_classrooms_min_ram) ... ok
test_task3_all_relationships_sorted (__main__.TestComputerClassroomSystem.test_task3_all_relationships_sorted) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```