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 Б9

Выполнил:
студент группы ИУ5-32Б:
Неустроева А.В.
Подпись и дата:
21.12.2025 г.

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

Москва, 2025г.

Задание:

Условия рубежного контроля №2 по курсу ПиК ЯП

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).

Файл test_data_manager.py:

```
import unittest

from data_manager import DataManager, OperatingSystem, Computer, ComputerOS


class TestDataManager(unittest.TestCase):


    def setUp(self):
        """Настройка тестовых данных"""
        self.test_os = [
            OperatingSystem(1, "Windows", "11"),
            OperatingSystem(2, "Linux", "Ubuntu 22.04"),
            OperatingSystem(3, "macOS", "Ventura"),
        ]


        self.test_computers = [
            Computer(1, "Dell", 16, 1),
            Computer(2, "HP", 8, 1),
            Computer(3, "Lenovo", 32, 2),
        ]


        self.test_relations = [
```

```
            ComputerOS(1, 1),
            ComputerOS(1, 2),
            ComputerOS(2, 1),
            ComputerOS(3, 2),
        ]

        self.manager = DataManager(
            operating_systems=self.test_os,
            computers=self.test_computers,
            computer_os_relations=self.test_relations
        )

    def test_get_one_to_many_relations(self):
        """Тест 1: Проверка получения связей один-ко-многим"""
        result = self.manager.get_one_to_many_relations()

        self.assertEqual(len(result), 3)
        computer, os_obj = result[0]
        self.assertEqual(computer.id, 1)
        self.assertEqual(computer.brand, "Dell")
        self.assertEqual(os_obj.id, 1)
        self.assertEqual(os_obj.name, "Windows")

        for computer, os_obj in result:
            self.assertEqual(computer.os_id, os_obj.id)

    def test_get_os_computer_count(self):
        """Тест 2: Проверка подсчета компьютеров по ОС"""
        result = self.manager.get_os_computer_count()
```

```
        self.assertEqual(len(result), 3)
        self.assertEqual(result["Windows"], 2)
        self.assertEqual(result["Linux"], 1)
        self.assertEqual(result["macOS"], 0)
        self.assertIn("Windows", result)
        self.assertIn("Linux", result)
        self.assertIn("macOS", result)

    def test_get_computers_with_ov_brand_and_os(self):
        """Тест 3: Проверка поиска компьютеров с брендом на 'ov'"""
        result = self.manager.get_computers_with_ov_brand_and_os()

        self.assertEqual(len(result), 1)

        computer, os_list = result[0]

        self.assertEqual(computer.id, 3)
        self.assertEqual(computer.brand, "Lenovo")
        self.assertEqual(computer.ram_gb, 32)

        self.assertEqual(len(os_list), 1)
        self.assertEqual(os_list[0].id, 2)
        self.assertEqual(os_list[0].name, "Linux")

    def test_get_computers_with_ov_brand_no_results(self):
        """Тест 4: Проверка случая, когда нет компьютеров с брендом на 'ov'"""
        computers_without_ov = [
            Computer(1, "Dell", 16, 1),
```

```
            Computer(2, "HP", 8, 1),
            Computer(3, "Apple", 32, 3),
        ]

        manager = DataManager(
            operating_systems=self.test_os,
            computers=computers_without_ov,
            computer_os_relations=self.test_relations
        )

        result = manager.get_computers_with_ov_brand_and_os()

        self.assertEqual(len(result), 0)

    def test_sorting_one_to_many(self):
        """Тест 5: Проверка сортировки связей один-ко-многим"""
        result = self.manager.get_one_to_many_relations()

        sorted_result = sorted(result, key=lambda x: x[0].brand)

        brands = [computer.brand for computer, _ in sorted_result]
        self.assertEqual(brands, ["Dell", "HP", "Lenovo"])

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

Файл data_manager.py:

```
import unittest
```

```
from data_manager import DataManager, OperatingSystem, Computer,
ComputerOS


class TestDataManager(unittest.TestCase):


    def setUp(self):
        """Настройка тестовых данных"""
        self.test_os = [
            OperatingSystem(1, "Windows", "11"),
            OperatingSystem(2, "Linux", "Ubuntu 22.04"),
            OperatingSystem(3, "macOS", "Ventura"),
        ]


        self.test_computers = [
            Computer(1, "Dell", 16, 1),
            Computer(2, "HP", 8, 1),
            Computer(3, "Lenovo", 32, 2),
        ]


        self.test_relations = [
            ComputerOS(1, 1),
            ComputerOS(1, 2),
            ComputerOS(2, 1),
            ComputerOS(3, 2),
        ]


        self.manager = DataManager(
            operating_systems=self.test_os,
            computers=self.test_computers,
```

```
        computer_os_relations=self.test_relations
    )

def test_get_one_to_many_relations(self):
    """Тест 1: Проверка получения связей один-ко-многим"""
    result = self.manager.get_one_to_many_relations()

    self.assertEqual(len(result), 3)
    computer, os_obj = result[0]
    self.assertEqual(computer.id, 1)
    self.assertEqual(computer.brand, "Dell")
    self.assertEqual(os_obj.id, 1)
    self.assertEqual(os_obj.name, "Windows")

    for computer, os_obj in result:
        self.assertEqual(computer.os_id, os_obj.id)

def test_get_os_computer_count(self):
    """Тест 2: Проверка подсчета компьютеров по ОС"""
    result = self.manager.get_os_computer_count()

    self.assertEqual(len(result), 3)
    self.assertEqual(result["Windows"], 2)
    self.assertEqual(result["Linux"], 1)
    self.assertEqual(result["macOS"], 0)
    self.assertIn("Windows", result)
    self.assertIn("Linux", result)
    self.assertIn("macOS", result)
```

```
def test_get_computers_with_ov_brand_and_os(self):
    """Тест 3: Проверка поиска компьютеров с брендом на 'ov'"""
    result = self.manager.get_computers_with_ov_brand_and_os()

    self.assertEqual(len(result), 1)

    computer, os_list = result[0]

    self.assertEqual(computer.id, 3)
    self.assertEqual(computer.brand, "Lenovo")
    self.assertEqual(computer.ram_gb, 32)

    self.assertEqual(len(os_list), 1)
    self.assertEqual(os_list[0].id, 2)
    self.assertEqual(os_list[0].name, "Linux")

def test_get_computers_with_ov_brand_no_results(self):
    """Тест 4: Проверка случая, когда нет компьютеров с брендом на 'ov'"""
    computers_without_ov = [
        Computer(1, "Dell", 16, 1),
        Computer(2, "HP", 8, 1),
        Computer(3, "Apple", 32, 3),
    ]

    manager = DataManager(
        operating_systems=self.test_os,
        computers=computers_without_ov,
        computer_os_relations=self.test_relations
    )
```

```
        result = manager.get_computers_with_ov_brand_and_os()

        self.assertEqual(len(result), 0)

    def test_sorting_one_to_many(self):
        """Тест 5: Проверка сортировки связей один-ко-многим"""
        result = self.manager.get_one_to_many_relations()

        sorted_result = sorted(result, key=lambda x: x[0].brand)

        brands = [computer.brand for computer, _ in sorted_result]
        self.assertEqual(brands, ["Dell", "HP", "Lenovo"])

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

Файл run_test.py:

```
import unittest
import sys

if __name__ == "__main__":
    test_loader = unittest.TestLoader()
    test_suite = test_loader.discover('.', pattern='test_*.py')

    test_runner = unittest.TextTestRunner(verbosity=2)
    result = test_runner.run(test_suite)

    sys.exit(0 if result.wasSuccessful() else 1)
```

Файл main.py:

```
from data_manager import DataManager


def main():
    manager = DataManager()


    print("=== ЗАПРОС 1 ===")
    print("Список всех связанных компьютеров и ОС (сортировка по компьютерам):")
    print("-" * 50)
    manager.display_one_to_many_relations()
    print()


    print("=== ЗАПРОС 2 ===")
    print("Список ОС с количеством компьютеров (сортировка по количеству):")
    print("-" * 50)
    manager.display_os_computer_count()
    print()


    print("=== ЗАПРОС 3 ===")
    print("Список компьютеров с брендом, заканчивающимся на 'ov', и их ОС:")
    print("-" * 50)
    manager.display_computers_with_ov_brand()


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