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 ИУ5-36Б
Ордиянц Эрик

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2024 г.

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

```python
from models import Part, Supplier, PartSupply
from services import PartService

def create_test_data():
    """Создание тестовых данных"""
    suppliers = [
        Supplier(1, 'Алексей'),
        Supplier(2, 'Николай'),
        Supplier(3, 'Пётр'),
    ]

    parts = [
        Part(1, "Резистор", 1),
        Part(2, "Конденсатор", 2),
        Part(3, "Транзистор", 1),
        Part(4, "Диод", 3),
    ]

    part_supplies = [
        PartSupply(1, 1, 100),
        PartSupply(2, 1, 200),
        PartSupply(3, 2, 150),
        PartSupply(4, 3, 50),
        PartSupply(1, 3, 300),
    ]

    return parts, suppliers, part_supplies

def main():
    """Основная функция"""
    parts, suppliers, part_supplies = create_test_data()
    service = PartService(parts, suppliers, part_supplies)

    print('Задание Д1')
    res_1 = service.find_parts_ending_with_or()
    print(res_1)

    print('\nЗадание Д2')
```

```
    res_2 = service.get_suppliers_avg_supply()
    print(res_2)

    print('\nЗадание ДЗ')
    res_3 = service.get_suppliers_starting_with_p()
    print(res_3)

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

test_services.py

```
import unittest

import sys
import os

sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))

from models import Part, Supplier, PartSupply
from services import PartService

class TestPartService(unittest.TestCase):
    def setUp(self):
        self.suppliers = [
            Supplier(1, 'Алексей'),
            Supplier(2, 'Николай'),
            Supplier(3, 'Пётр'),
        ]

        self.parts = [
            Part(1, "Резистор", 1),
            Part(2, "Диод", 2),
            Part(3, "Транзистор", 1),
            Part(4, "Микрочип", 3),
        ]

        self.part_supplies = [
            PartSupply(1, 1, 100),
            PartSupply(2, 1, 200),
            PartSupply(3, 2, 150),
            PartSupply(4, 3, 50),
        ]

        self.service = PartService(self.parts, self.suppliers, self.part_supplies)

    def test_find_parts_ending_with_or(self):
        """Тест поиска деталей, заканчивающихся на 'ор'"""
        result = self.service.find_parts_ending_with_or()
        expected = [
            ("Резистор", "Алексей"),
```

```
            ("Транзистор", "Алексей")
        ]
        self.assertEqual(result, expected)

    def test_get_suppliers_avg_supply(self):
        """Тест расчета средней поставки деталей"""
        result = self.service.get_suppliers_avg_supply()
        expected = [
            ("Алексей", 150),
            ("Николай", 150),
            ("Пётр", 50)
        ]
        self.assertEqual(result, expected)

    def test_get_suppliers_starting_with_p(self):
        """Тест поиска поставщиков, начинающихся на 'П'"""
        result = self.service.get_suppliers_starting_with_p()
        expected = {
            "Пётр": ["Микрочип"]
        }
        self.assertEqual(result, expected)

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

Models.py

```
from dataclasses import dataclass

@dataclass
class Part:
    """Деталь"""
    part_id: int
    name: str
    supplier_id: int

@dataclass
class Supplier:
    """Поставщик"""
    supplier_id: int
    name: str

@dataclass
class PartSupply:
    """Поставки деталей для реализации связи многие-ко-многим"""
    part_id: int
    supplier_id: int
    quantity: int
```

services.py

```python
from typing import List, Dict, Tuple
from operator import itemgetter

from models import Part, Supplier, PartSupply

class PartService:
    def __init__(self, parts: List[Part], suppliers: List[Supplier], part_supplies:
List[PartSupply]):
        self.parts = parts
        self.suppliers = suppliers
        self.part_supplies = part_supplies

    def find_parts_ending_with_or(self) -> List[Tuple[str, str]]:
        """Список всех деталей, у которых название заканчивается на 'ор', и имена
поставщиков"""
        result = []
        for p in self.parts:
            if p.name.endswith('ор'):
                for s in self.suppliers:
                    if p.supplier_id == s.supplier_id:
                        result.append((p.name, s.name))
        return result

    def get_suppliers_avg_supply(self) -> List[Tuple[str, float]]:
        """Список поставщиков со средней поставкой деталей, отсортированный по средней
поставке"""
        result = []
        for s in self.suppliers:
            supplies = [ps for ps in self.part_supplies if ps.supplier_id ==
s.supplier_id]
            if supplies:
                total_quantity = sum(ps.quantity for ps in supplies)
                avg_quantity = round(total_quantity / len(supplies))
                result.append((s.name, avg_quantity))
        return sorted(result, key=itemgetter(1), reverse=True)

    def get_suppliers_starting_with_p(self) -> Dict[str, List[str]]:
        """Список всех поставщиков, у которых имя начинается с буквы 'П', и детали,
которые они поставляют"""
        result = {}
        for s in self.suppliers:
            if s.name.startswith('П'):
                s_parts = [p.name for p in self.parts if p.supplier_id ==
s.supplier_id]
                result[s.name] = s_parts
        return result
```

Анализ результатов

```
● → PCPL git:(main) python3 -m unittest RK2/tests/test_services.py -v
test_find_parts_ending_with_or (RK2.tests.test_services.TestPartService)
Тест поиска деталей, заканчивающихся на 'ор' ... ok
test_get_suppliers_avg_supply (RK2.tests.test_services.TestPartService)
Тест расчета средней поставки деталей ... ok
test_get_suppliers_starting_with_p (RK2.tests.test_services.TestPartService)
Тест поиска поставщиков, начинающихся на 'П' ... ok

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

OK
```