**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 РК №2
Вариант запросов: В
Вариант предметной области: 19

Выполнил:
студент группы ИУ5-33Б
Нагапетян Валерий

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2023 г.

Вариант запросов В.

1. «Деталь» и «Производитель» связаны соотношением один-ко-многим. Выведите список всех деталей, которые начинаются с «Тормоз» и названия их производителей.

2. «Деталь» и «Производитель» связаны соотношением один-ко-многим. Выведите список производителей с минимальной стоимостью деталей, отсортированный их по минимальной стоимости.

3. «Деталь» и «Производитель» связаны соотношением многие-ко-многим. Выведите список всех связанных деталей и производителей, отсортированный по производителям, сортировка деталей по цене.

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).

Листинг программы main.py:

```
from typing import Union, NoReturn


class Component:
    """Класс Деталь"""

    def __init__(self, id: int, name: str, price: Union[int, float], fabric_id: int):
        self.__id = id
        self.__name = name
        self.__price = price
        self.__fabric_id = fabric_id

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

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

    @property
    def price(self) -> Union[int, float]:
        return self.__price

    @property
    def fabric_id(self) -> int:
        return self.__fabric_id


class Fabric:
    """Класс Производитель"""

    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 FabricComponent:
    """Класс детали производителя"""

    def __init__(self, fabric_id: int, component_id: int):
        self.__fabric_id = fabric_id
        self.__component_id = component_id

    @property
```

```python
    def fabric_id(self) -> int:
        return self.__fabric_id

    @property
    def component_id(self) -> int:
        return self.__component_id


def request1(components: list[Component], fabrics: list[Fabric]) -> list[tuple[str, str]]:
    response = [
        (c, f)
        for c in components
        for f in fabrics
        if c.fabric_id == f.id and c.name.startswith("Тормоз")
    ]
    return [(c.name, f.name) for c, f in response]


def request2(components: list[Component], fabrics: list[Fabric]) -> list[tuple[str, int]]:
    response = {}
    for fabric in fabrics:
        minimal_price = min([c.price for c in components if c.fabric_id == fabric.id])
        response[fabric.name] = minimal_price

    response_items = list(response.items())
    response_items.sort(key=lambda item: item[1])

    return response_items


def request3(components: list[Component], fabrics: list[Fabric], fabric_components:
list[FabricComponent]) -> list[tuple[str, str]]:
    response = [
        (f, c)
        for fc in fabric_components
        for f in fabrics
        for c in components
        if fc.fabric_id == f.id and fc.component_id == c.id
    ]

    response.sort(key=lambda item: (item[0].name, item[1].price))
    return [(fabric.name, component.name) for fabric, component in response]


def components_data_test() -> list[Component]:
    return [
        Component(1, "Тормозные колодки", 12000, 1),
        Component(2, "Фары", 5000, 1),
        Component(3, "Заднее крыло", 10000, 2),
        Component(4, "Генератор", 15000, 3),
        Component(5, "Аккумулятор", 8000, 3),
        Component(6, "Тормозные диски", 9000, 4),
        Component(7, "Дворники", 3000, 4)
    ]
```

```python
def fabrics_data_test() -> list[Fabric]:
    return [
        Fabric(1, "АВТОВАЗ"),
        Fabric(2, "УАЗ"),
        Fabric(3, "КАМАЗ"),
        Fabric(4, "УВЗ"),
    ]


def fabrics_components_data_test() -> list[FabricComponent]:
    return [
        FabricComponent(1, 1),
        FabricComponent(1, 2),
        FabricComponent(2, 3),
        FabricComponent(3, 4),
        FabricComponent(3, 5),
        FabricComponent(4, 6),
        FabricComponent(4, 7)
    ]


def main() -> NoReturn:
    components = components_data_test()
    fabrics = fabrics_data_test()
    fabrics_components = fabrics_components_data_test()

    response1 = request1(components, fabrics)
    response2 = request2(components, fabrics)
    response3 = request3(components, fabrics, fabrics_components)

    print('Запрос №1. Список деталей, которые начинаются с "Тормоз", и их производителей.')
    for (component_name, fabric_name) in response1:
        print(f"    Деталь: {component_name} | Производитель: {fabric_name}")
    print()

    print("Запрос №2. Список производителей с минимальной стоимостью деталей.")
    for (fabric_name, minimal_price) in response2:
        print(f"    Производитель: {fabric_name} | Минимальная стоимость детали: {minimal_price}")
    print()

    print(
        "Запрос №3. Список всех связанных деталей и производителей, отсортированный по производителям. Сортировка деталей по цене."
    )
    for (fabric_name, component_name) in response3:
        print(f"    Производитель: {fabric_name} | Деталь: {component_name}")
    print()


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

Листинг программы tests.py:

```
import unittest
from RK2.main import *


class TestComponent(unittest.TestCase):
    def test_component_init(self):
        component = Component(1, "shock absorber", 5871, 1)
        self.assertEqual(component.id, 1)
        self.assertEqual(component.name, "shock absorber")
        self.assertEqual(component.price, 5871)
        self.assertEqual(component.fabric_id, 1)


class TestFabric(unittest.TestCase):
    def test_fabric_init(self):
        fabric = Fabric(1, "АВТОВАЗ")
        self.assertEqual(fabric.id, 1)
        self.assertEqual(fabric.name, "АВТОВАЗ")


class TestFabricComponent(unittest.TestCase):
    def test_fabric_component_init(self):
        fc = FabricComponent(1, 2)
        self.assertEqual(fc.fabric_id, 1)
        self.assertEqual(fc.component_id, 2)


class TestRequest(unittest.TestCase):
    def setUp(self) -> None:
        self._components = components_data_test()
        self._fabrics = fabrics_data_test()
        self._fabrics_components = fabrics_components_data_test()

    def test_request1(self):
        expected_result = [
            ("Тормозные колодки", "АВТОВАЗ"),
            ("Тормозные диски", "УВЗ")
        ]
        actual_result = request1(self._components, self._fabrics)
        self.assertEqual(actual_result, expected_result)

    def test_request2(self):
        expected_result = [
            ("УВЗ", 3000),
            ("АВТОВАЗ", 5000),
            ("КАМАЗ", 8000),
            ("УАЗ", 10000)
        ]
        actual_result = request2(self._components, self._fabrics)
        self.assertEqual(actual_result, expected_result)

    def test_request3(self):
        expected_result = [
```

```
            ("АВТОВАЗ", "Фары"),
            ("АВТОВАЗ", "Тормозные колодки"),
            ("КАМАЗ", "Аккумулятор"),
            ("КАМАЗ", "Генератор"),
            ("УАЗ", "Заднее крыло"),
            ("УВЗ", "Дворники"),
            ("УВЗ", "Тормозные диски"),
        ]
        actual_result = request3(self._components, self._fabrics, self._fabrics_components)
        self.assertEqual(actual_result, expected_result)


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

Результат выполнения:

```
"C:\Users\valer\OneDrive\Рабочий стол\3 семестр\ПиКЯП\bmstu-pcpl\venv\Scripts\python.exe"
test_component_init (tests.TestComponent.test_component_init) ... ok
test_fabric_init (tests.TestFabric.test_fabric_init) ... ok
test_fabric_component_init (tests.TestFabricComponent.test_fabric_component_init) ... ok
test_request1 (tests.TestRequest.test_request1) ... ok
test_request2 (tests.TestRequest.test_request2) ... ok
test_request3 (tests.TestRequest.test_request3) ... ok

----------------------------------------------------------------------
Ran 6 tests in 0.000s

OK

Process finished with exit code 0
```