Текст программы:

```
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
    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:
    response = [
        (c, f)
        for c in components
        for f in fabrics
        if c.fabric_id == f.id
    ]
    response.sort(key=lambda item: (item[1].name, item[0].price))
    return [(c.name, f.name) for c, f in response]


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

    response_sorted = dict(sorted(response.items(), key=lambda item: item[1]))
    return [(fabric_name, sum_price) for fabric_name, sum_price in
response_sorted.items()]


def request3(components: list[Component], fabrics: list[Fabric],
fabric_components: list[FabricComponent]) -> dict:
    response = {
```

```
        f: [
            c
            for c in components
            for fc in fabric_components
            if fc.fabric_id == f.id and fc.component_id == c.id
        ]
        for f in fabrics
        if f.name.endswith("АЗ")
    }
    return {f.name: [c.name for c in comps] for f, comps in response.items()}


def main() -> NoReturn:
    components = [
        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)
    ]

    fabrics = [
        Fabric(1, "АВТОВАЗ"),
        Fabric(2, "УАЗ"),
        Fabric(3, "КАМАЗ"),
        Fabric(4, "УВЗ"),
    ]
```

```
    fabric_component = [
        FabricComponent(1, 1),
        FabricComponent(1, 2),
        FabricComponent(2, 3),
        FabricComponent(3, 4),
        FabricComponent(3, 5),
        FabricComponent(4, 6),
        FabricComponent(4, 7)
    ]



    # Запрос 1
    print(
        "Запрос №1. Список всех связанных деталей и производителей, отсортированный по производителям, сортировка по деталям по цене."
    )
    for c, f in request1(components, fabrics):
        print(f"\tДеталь: {c} | Производитель: {f}")
    print()


    # Запрос 2
    print(
        "Запрос №2. Список производителей с суммарной стоимостью деталей, отсортированный по суммарной стоимости деталей"
    )
    for fabric_name, sum_price in request2(components, fabrics):
        print(f"\tПроизводитель: {fabric_name} | Суммарная стоимость деталей: {sum_price}")
    print()
```

```
    # Запрос 3
    print(
        "Запрос №3. Список всех производителей, у которых название оканчивается на 'АЗ' и список производимых ими деталей."
    )
    for fabric, components in request3(components, fabrics, fabric_component).items():
        print(f"\tПроизводитель: {fabric} | Детали: {[c for c in components]}")
    print()



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

Test:

```
from main1 import *
import unittest



# Тестирование класса «Деталь»
class TestComponent(unittest.TestCase):
    def test_component_creation(self):
        test_comp = Component(11, "Тормозные колодки", 12000, 7)
        self.assertEqual(test_comp.id, 11)
        self.assertEqual(test_comp.name, "Тормозные колодки")
        self.assertEqual(test_comp.price, 12000),
        self.assertEqual(test_comp.fabric_id, 7)
```

```
# Тестирование класса «Производитель»
class TestFabric(unittest.TestCase):
    def test_fabric_creation(self):
        test_fabric = Fabric(10, "АВТОВАЗ")
        self.assertEqual(test_fabric.id, 10)
        self.assertEqual(test_fabric.name, "АВТОВАЗ")


# Тестирование класса для реализация связи «Многие ко многим»
class TestFabricComponent(unittest.TestCase):
    def test_map_creation(self):
        test_map = FabricComponent(1, 3)
        self.assertEqual(test_map.fabric_id, 1)
        self.assertEqual(test_map.component_id, 3)


class TestMainFunctions(unittest.TestCase):
    # Генерация тестовых данных
    def setUp(self):
        self.components = [
            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)
        ]
        self.fabrics = [
```

```
        Fabric(1, "АВТОВАЗ"),
        Fabric(2, "УАЗ"),
        Fabric(3, "КАМАЗ"),
        Fabric(4, "УВЗ"),
    ]
    self.fabric_component = [
        FabricComponent(1, 1),
        FabricComponent(1, 2),
        FabricComponent(2, 3),
        FabricComponent(3, 4),
        FabricComponent(3, 5),
        FabricComponent(4, 6),
        FabricComponent(4, 7)
    ]

# Тестирование запроса №1
def test_request1(self):
    self.assertEqual(
        request1(self.components, self.fabrics),
        [
            ("Фары", "АВТОВАЗ"),
            ("Тормозные колодки", "АВТОВАЗ"),
            ("Аккумулятор", "КАМАЗ"),
            ("Генератор", "КАМАЗ"),
            ("Заднее крыло", "УАЗ"),
            ("Дворники", "УВЗ"),
            ("Тормозные диски", "УВЗ")
        ]
    )
```

```
    # Тестирование запроса №2
    def test_request2(self):
        self.assertEqual(
            request2(self.components, self.fabrics),
            [
                ("УАЗ", 10000),
                ("УВЗ", 12000),
                ("АВТОВАЗ", 17000),
                ("КАМАЗ", 23000)
            ]
        )

    # Тестирование запроса №3
    def test_request1(self):
        self.assertEqual(
            request3(self.components, self.fabrics, self.fabric_component),
            {
                "АВТОВАЗ": ['Тормозные колодки', 'Фары'],
                "УАЗ": ['Заднее крыло'],
                "КАМАЗ": ['Генератор', 'Аккумулятор']
            }
        )


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