Давыдов Кирилл Игоревич
Группа ИУ5-34Б

РК1, вариант 19, вариант запросов В

Код:

```python
from dataclasses import dataclass
from typing import List, Tuple, Dict

@dataclass
class Manufacturer:
    id: int
    name: str


@dataclass
class Part:
    id: int
    name: str
    price: int
    manufacturer_id: int


@dataclass
class PartManufacturer:
    part_id: int
    manufacturer_id: int

manufacturers: List[Manufacturer] = [
    Manufacturer(1, "АЛЬФА-Мех"),
    Manufacturer(2, "БетаПром"),
    Manufacturer(3, "Гамма-Сталь"),
    Manufacturer(4, "АвтоДеталь"),
    Manufacturer(5, "Аксис"),
]

parts: List[Part] = [
    Part(1, "Амортизатор", 5200, 4),
    Part(2, "Антенна", 900, 2),
    Part(3, "Болт", 30, 1),
    Part(4, "Адаптер", 700, 5),
    Part(5, "Клапан", 1350, 3),
    Part(6, "Фильтр", 850, 4),
]

part_manufacturer_links: List[PartManufacturer] = [
    PartManufacturer(1, 4),
    PartManufacturer(1, 1),
    PartManufacturer(2, 2),
    PartManufacturer(2, 5),
    PartManufacturer(3, 1),
    PartManufacturer(3, 3),
    PartManufacturer(4, 5),
    PartManufacturer(5, 3),
```

```python
    PartManufacturer(6, 4),
]

id_to_manufacturer: Dict[int, Manufacturer] = {m.id: m for m in manufacturers}

from collections import defaultdict
part_to_manufs: Dict[int, List[int]] = defaultdict(list)
for link in part_manufacturer_links:
    part_to_manufs[link.part_id].append(link.manufacturer_id)

def query_v1(parts: List[Part]) -> List[Tuple[str, str]]:
    return sorted(
        [
            (p.name, id_to_manufacturer[p.manufacturer_id].name)
            for p in parts
            if p.name.startswith("A")
        ],
        key=lambda t: (t[0].lower(), t[1].lower()),
    )

def query_v2(parts: List[Part]) -> List[Tuple[str, int]]:
    man_id_to_prices: Dict[int, List[int]] = defaultdict(list)
    for p in parts:
        man_id_to_prices[p.manufacturer_id].append(p.price)

    tuples = [
        (id_to_manufacturer[mid].name, min(prices))
        for mid, prices in man_id_to_prices.items()
        if prices
    ]

    return sorted(tuples, key=lambda t: (t[1], t[0].lower()))

def query_v3(parts: List[Part]) -> List[Tuple[str, List[str]]]:
    id_to_part_name = {p.id: p.name for p in parts}

    name_to_manuf_names: Dict[str, List[str]] = {
        id_to_part_name[pid]: sorted(
            [id_to_manufacturer[mid].name for mid in mids],
            key=str.lower,
        )
        for pid, mids in part_to_manufs.items()
    }

    return sorted(name_to_manuf_names.items(), key=lambda t: t[0].lower())



if __name__ == "__main__":
    print("=== B.1: Детали, начинающиеся на 'A', и их производители (1:M) ===")
    for part_name, manuf_name in query_v1(parts):
        print(f"{part_name} — {manuf_name}")

    print("\n=== B.2: Минимальная цена детали по каждому производителю (1:M) ===")
```

```
    for manuf_name, min_price in query_v2(parts):
        print(f"{manuf_name}: {min_price} ₽")

    print("\n=== B.3: Все связи Деталь–Производитель (M:N), отсортировано по деталям
===")
    for part_name, manuf_names in query_v3(parts):
        mans_str = ", ".join(manuf_names)
        print(f"{part_name}: {mans_str}")
```

Результат выполнения программы:

```
=== B.1: Детали, начинающиеся на 'А', и их производители (1:M) ===
Адаптер — Аксис
Амортизатор — АвтоДеталь
Антенна — БетаПром

=== B.2: Минимальная цена детали по каждому производителю (1:M) ===
АЛЬФА–Мех: 30 ₽
Аксис: 700 ₽
АвтоДеталь: 850 ₽
БетаПром: 900 ₽
Гамма–Сталь: 1350 ₽

=== B.3: Все связи Деталь–Производитель (M:N), отсортировано по деталям ===
Адаптер: Аксис
Амортизатор: АвтоДеталь, АЛЬФА–Мех
Антенна: Аксис, БетаПром
Болт: АЛЬФА–Мех, Гамма–Сталь
Клапан: Гамма–Сталь
Фильтр: АвтоДеталь
```