МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю 2

Выполнил:
Студент группы ИУ5-34Б
Буш И. Е.
Преподаватель:
Нардид Анатолий Николаевич

Москва 2025

Рубежный контроль №2 по курсу ПиК ЯП

Содержимое: рефакторинг программы РК №1 для модульного тестирования + 3 модульных теста (unittest) и результаты запуска.

1) Рефакторинг программы (модуль rk2_program.py)

```
"""Рубежный контроль №1 (рефакторинг для РК №2) — Вариант В
Предметная область: CD-диск — Библиотека CD-дисков

Сущности и связи:
- CDDisc (CD-диск) -> Library (Библиотека): связь один-ко-многим (library_id) -
CDDisc <-> Library: связь многие-ко-многим через DiscLibrary (таблица связей)

Запросы:
1) (1:N) Список всех CD-дисков, у которых название начинается с буквы «А», и названия их библ
2) (1:N) Список библиотек с минимальной ценой дисков в каждой, отсортированный по
   минимальной3) (M:N) Список всех связанных дисков и библиотек, отсортированный по дискам
   (по названию дис
"""

from __future__ import annotations
from dataclasses import dataclass
from typing import Iterable, List, Dict, Tuple


@dataclass(frozen=True)
class Library:     id:
int     name: str


@dataclass(frozen=True)
class CDDisc:     id:
int     title: str
    price: int  # количественный признак для запроса №2
library_id: int  # для связи один-ко-многим


@dataclass(frozen=True)
class DiscLibrary:
disc_id: int
library_id: int


def create_test_data() -> tuple[list[Library], list[CDDisc], list[DiscLibrary]]:
    """Тестовые данные (3–5 записей) с корректными ключами."""
libraries = [
        Library(1, "Центральная библиотека"),
        Library(2, "Филиал №1"),
        Library(3, "Медиа-центр"),
    ]

    discs = [
        CDDisc(1, "Альфа", 500, 1),
        CDDisc(2, "Бета", 350, 1),
        CDDisc(3, "Астра", 250, 2),
        CDDisc(4, "Гамма", 400, 2),
        CDDisc(5, "Ария", 600, 3),
    ]

    disc_libraries = [
        DiscLibrary(1, 1),
        DiscLibrary(1, 2),
```

```
        DiscLibrary(2, 1),
        DiscLibrary(3, 2),
        DiscLibrary(3, 3),
        DiscLibrary(4, 2),
        DiscLibrary(5, 3),
        DiscLibrary(5, 1),
    ]
    return libraries, discs, disc_libraries


def _lib_by_id(libraries: Iterable[Library]) -> Dict[int, Library]:
return {lib.id: lib for lib in libraries}


def query1_discs_starting_with_a(discs: Iterable[CDDisc], libraries: Iterable[Library]) -> Li
    """(1:N) Диски, название которых начинается с «A», и название их библиотеки."""
libs = _lib_by_id(libraries)     return [
        (d.title, libs[d.library_id].name)
for d in discs
        if d.title.startswith("A") and d.library_id in libs
]


def query2_libraries_min_price(discs: Iterable[CDDisc], libraries: Iterable[Library]) -> List
"""(1:N) Библиотеки с минимальной ценой дисков в каждой, сортировка по минимальной цене."
    # Сгруппировать цены по library_id     prices_by_lib:
Dict[int, List[int]] = {}     for d in discs:
prices_by_lib.setdefault(d.library_id, []).append(d.price)

    # Сформировать результат только для библиотек, у которых есть хотя бы один диск
result = [
        (lib.name, min(prices_by_lib[lib.id]))
for lib in libraries
        if lib.id in prices_by_lib and prices_by_lib[lib.id]
    ]
    return sorted(result, key=lambda x: x[1])


def query3_many_to_many_pairs(
discs: Iterable[CDDisc],
libraries: Iterable[Library],
links: Iterable[DiscLibrary],
) -> List[Tuple[str, str]]:
    """(M:N) Все пары (диск, библиотека), отсортированные по дискам (по названию)."""
disc_by_id = {d.id: d for d in discs}     lib_by_id = _lib_by_id(libraries)

    pairs = [
        (disc_by_id[lnk.disc_id].title, lib_by_id[lnk.library_id].name)
for lnk in links
        if lnk.disc_id in disc_by_id and lnk.library_id in lib_by_id
    ]
    return sorted(pairs, key=lambda x: x[0])


def main() -> None:     libraries, discs, disc_libraries
= create_test_data()

    print("Запрос №1: Диски на «A» и их библиотеки")     for title, lib_name in
query1_discs_starting_with_a(discs, libraries):         print(f"- {title} —
{lib_name}")     print("\nЗапрос №2: Библиотеки и минимальная цена дисков (по
возрастанию)")     for lib_name, min_price in query2_libraries_min_price(discs,
libraries):         print(f"- {lib_name}: {min_price}")
```

```
    print("\nЗапрос №3: Связи диск-библиотека (M:N), сортировка по дискам")     for
title, lib_name in query3_many_to_many_pairs(discs, libraries, disc_libraries):
print(f"- {title} — {lib_name}")


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

2) Модульные тесты (модуль test_rk2_program.py)

```
import unittest

from rk2_program import (
create_test_data,
    query1_discs_starting_with_a,
query2_libraries_min_price,
query3_many_to_many_pairs, )


class TestRK1Queries(unittest.TestCase):     def setUp(self):
self.libraries, self.discs, self.links = create_test_data()

    def test_query1_discs_starting_with_a(self):
        # Ожидаем 3 диска на "А": Альфа(Центральная), Астра(Филиал №1), Ария(Медиа-центр)
result = query1_discs_starting_with_a(self.discs, self.libraries)
self.assertEqual(             result,
            [
                ("Альфа", "Центральная библиотека"),
                ("Астра", "Филиал №1"),
                ("Ария", "Медиа-центр"),
            ],
        )

    def test_query2_libraries_min_price_sorted(self):
        # Центральная: min(500,350)=350; Филиал №1: min(250,400)=250; Медиа-центр: min(600)=6
result = query2_libraries_min_price(self.discs, self.libraries)         self.assertEqual(
result,
            [
                ("Филиал №1", 250),
                ("Центральная библиотека", 350),
                ("Медиа-центр", 600),
            ],
        )

    def test_query3_many_to_many_sorted_by_disc_title(self):
        # Проверим, что пары отсортированы по названию диска и содержат все связи.
     result = query3_many_to_many_pairs(self.discs, self.libraries, self.links)

        expected = [
            ("Альфа", "Центральная библиотека"),
            ("Альфа", "Филиал №1"),
            ("Ария", "Медиа-центр"),
            ("Ария", "Центральная библиотека"),
            ("Астра", "Филиал №1"),
            ("Астра", "Медиа-центр"),
            ("Бета", "Центральная библиотека"),
            ("Гамма", "Филиал №1"),
        ]
        self.assertEqual(result, expected)


if __name__ == "__main__":
    unittest.main(verbosity=2)
```

3) Результаты выполнения программы (для проверки)

```
Запрос №1: Диски на «А» и их библиотеки
- Альфа — Центральная библиотека
- Астра — Филиал №1
- Ария — Медиа-центр

Запрос №2: Библиотеки и минимальная цена дисков (по возрастанию)
- Филиал №1: 250
- Центральная библиотека: 350
```

```
- Медиа-центр: 600

Запрос №3: Связи диск–библиотека (M:N), сортировка по дискам
- Альфа — Центральная библиотека
- Альфа — Филиал №1
- Ария — Медиа-центр
- Ария — Центральная библиотека
- Астра — Филиал №1
- Астра — Медиа-центр
- Бета — Центральная библиотека
- Гамма — Филиал №1
```

4) Результаты запуска тестов (unittest -v)

```
test_query1_discs_starting_with_a (test_rk2_program.TestRK1Queries.test_query1_discs_starting
test_query2_libraries_min_price_sorted (test_rk2_program.TestRK1Queries.test_query2_libraries
test_query3_many_to_many_sorted_by_disc_title (test_rk2_program.TestRK1Queries.test_query3_ma

----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```