Московский государственный технический
университет им. Н.Э. Баумана

Факультет информатика и системы управления
Кафедра системы обработки информации и управления

Курс «Парадигмы и конструкции языков программирования»
Отчет по рубежному контролю №1
Вариант Б6

Выполнил:
студент группы ИУ5-32Б:
Кунев В.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2025 г.

Текст программы

```
"""Решение Рубежного контроля №1 по курсу ПиК ЯП.

Вариант предметной области: 6 (Дом, Улица)
Вариант запросов: Б

Скрипт создает модели данных для 'Дома' и 'Улицы', реализует связи "один-ко-многим" и "многие-ко-многим",
а затем выполняет три запроса из варианта 'Б'.
"""

from operator import itemgetter


class House:
    """Представление дома.

    Attributes:
        identifier (int): Уникальный идентификатор дома.
        number (str): Номер дома (например, '10а', '22').
        population (int): Количество жильцов в доме.
        street_id (int): Внешний ключ, ссылающийся на ID улицы (для связи 1:М).
    """

    def __init__(self, identifier: int, number: str, population: int, street_id: int):
        """Инициализирует экземпляр дома.

        Args:
            identifier (int): Уникальный идентификатор дома.
            number (str): Номер дома.
            population (int): Количество жильцов.
            street_id (int): ID связанной улицы.
        """
        self.identifier = identifier
        self.number = number
        self.population = population
        self.street_id = street_id


class Street:
    """Представление улицы.

    Attributes:
        identifier (int): Уникальный идентификатор улицы.
        name (str): Название улицы.
    """

    def __init__(self, identifier: int, name: str):
        """Инициализирует экземпляр улицы.
```

```
        Args:
            identifier (int): Уникальный идентификатор улицы.
            name (str): Название улицы.
        """
        self.identifier = identifier
        self.name = name


class HouseStreet:
    """
    Ассоциативная сущность для связи "многие-ко-многим" между Домом и Улицей.

    Attributes:
        street_id (int): Внешний ключ, ссылающийся на ID улицы.
        house_id (int): Внешний ключ, ссылающийся на ID дома.
    """

    def __init__(self, street_id: int, house_id: int):
        """Инициализирует связь M:M.

        Args:
            street_id (int): ID связанной улицы.
            house_id (int): ID связанного дома.
        """
        self.street_id = street_id
        self.house_id = house_id


# Тестовые данные
streets: list[Street] = [
    Street(1, "ул. Ленина"),
    Street(2, "ул. Пушкина"),
    Street(3, "просп. Мира"),
]
houses: list[House] = [
    House(1, "10a", 50, 1),
    House(2, "12", 120, 1),
    House(3, "5a", 80, 2),
    House(4, "76", 200, 3),
    House(5, "22", 75, 2),
]
houses_streets: list[HouseStreet] = [
    HouseStreet(1, 1),
    HouseStreet(2, 1),
    HouseStreet(1, 2),
    HouseStreet(2, 3),
    HouseStreet(3, 4),
    HouseStreet(2, 5),
]
```

```python
def main():
    """
    Основная функция программы.

    Выполняет подготовку данных (join) и решает три задачи варианта 'Б'.
    """

    # Подготовка данных

    # Соединение "один-ко-многим" (Дом -> Улица)
    one_to_many: list[tuple[str, int, str]] = [
        (h.number, h.population, s.name)
        for s in streets
        for h in houses
        if h.street_id == s.identifier
    ]
    # Соединение "многие-ко-многим"
    many_to_many_temp: list[tuple[str, int, int]] = [
        (s.name, hs.street_id, hs.house_id)
        for s in streets
        for hs in houses_streets
        if s.identifier == hs.street_id
    ]
    many_to_many: list[tuple[str, int, str]] = [
        (h.number, h.population, s_name)
        for s_name, s_id, h_id in many_to_many_temp
        for h in houses
        if h.identifier == h_id
    ]

    # Выполнение запросов

    print("Задание Б1")
    # Вывести список всех связанных домов и улиц, отсортированный по домам
(номеру дома).
    res_1: list[tuple[str, int, str]] = sorted(one_to_many, key=itemgetter(0))
    print(res_1)

    print()
    print("Задание Б2")
    # Вывести список улиц с количеством домов в каждой, отсортированный по
количеству домов.
    res_2_unsorted: list[tuple[str, int]] = []
    for s in streets:
        s_houses: list[tuple[str, int, str]] = list(
            filter(lambda item: item[2] == s.name, one_to_many)
        )

        if len(s_houses) > 0:
            s_count: int = len(s_houses)
            res_2_unsorted.append((s.name, s_count))
```

```
    res_2: list[tuple[str, int]] = sorted(res_2_unsorted, key=itemgetter(1))
    print(res_2)

    print()
    print("Задание Б3")
    # Вывести список всех домов, у которых номер заканчивается на «а», и названия
их улиц.
    res_3: list[tuple[str, str]] = [
        (h_num, s_name) for h_num, _, s_name in many_to_many if
h_num.endswith("a")
    ]
    print(res_3)


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

Скриншот работы приложения

```
(.venv) cunev@ROGStrixG814JIR:~/VSCode/ParadigmsAndConstructsOfProgrammingLanguages$ /home/cunev/VSCode/ParadigmsAndConstructsOfProgramm
ngLanguages/.venv/bin/python /home/cunev/VSCode/ParadigmsAndConstructsOfProgrammingLanguages/rk1/main.py
Задание Б1
[('10а', 50, 'ул. Ленина'), ('12', 120, 'ул. Ленина'), ('22', 75, 'ул. Пушкина'), ('5а', 80, 'ул. Пушкина'), ('7б', 200, 'просп. Мира')]

Задание Б2
[('просп. Мира', 1), ('ул. Ленина', 2), ('ул. Пушкина', 2)]

Задание Б3
[('10а', 'ул. Ленина'), ('10а', 'ул. Пушкина'), ('5а', 'ул. Пушкина')]
```

Рисунок 1. Вывод результатов программы