**МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Информационные технологии»

Лабораторная работа по ИТП №3

Выполнил: Студент группы

БВТ2204

Сарыков Эрбол

Москва

2023

Задания для выполнения лабораторной работы  
Задание 1:   
1. Создайте класс HashTable, который будет реализовывать хэш -  
таблицу с помощью метода цепочек.  
2. Реализуйте методы put(key, value), get(key) и remove(key), которые   
добавляют, получают и удаляют пары «ключ-значение» соответственно.   
3. Добавьте методы size() и isEmpty(), которые возвращают   
количество элементов в таблице и проверяют, пуста ли она.   
Пример реализации метода put(key, value):   
public void put(K key, V value) {   
 int index = hash(key);   
 if (table[index] == null) {   
 table[index] = new LinkedList<Entry<K, V>>();   
 }   
 for (Entry<K, V> entry : table[index]) {   
 if (entry.getKey().equals(key)) {   
 entry.setValue(value);   
 return;   
 }   
 }   
 table[index].add(new Entry<K, V>(key, value));   
 size++;   
}

Задание 2:  
Реализация хэш-таблицы для хранения информации о книгах

в библиотеке. Ключом будет ISBN книги, а значением - объект класса Book,

содержащий информацию о названии, авторе и количестве копий.

Необходимо реализовать операции вставки, поиска и удаления книги по

ISBN

Задание 1

import java.util.ArrayList;

import java.util.LinkedList;

public class HashTable<K, V> {

    private static class Entry<K, V> {

        K key;

        V value;

        Entry(K key, V value) {

            this.key = key;

            this.value = value;

        }

    }

    private ArrayList<LinkedList<Entry<K, V>>> table;

    private int size;

    public HashTable(int capacity) {

        table = new ArrayList<>(capacity);

        for (int i = 0; i < capacity; i++) {

            table.add(new LinkedList<>());

        }

        size = 0;

    }

    private int hash(K key) {

        return Math.abs(key.hashCode()) % table.size();

    }

    public void put(K key, V value) {

        int index = hash(key);

        LinkedList<Entry<K, V>> bucket = table.get(index);

        for (Entry<K, V> entry : bucket) {

            if (entry.key.equals(key)) {

                entry.value = value;

                return;

            }

        }

        bucket.add(new Entry<>(key, value));

        size++;

    }

    public V get(K key) {

        int index = hash(key);

        LinkedList<Entry<K, V>> bucket = table.get(index);

        for (Entry<K, V> entry : bucket) {

            if (entry.key.equals(key)) {

                return entry.value;

            }

        }

        return null;

    }

    public void remove(K key) {

        int index = hash(key);

        LinkedList<Entry<K, V>> bucket = table.get(index);

        for (Entry<K, V> entry : bucket) {

            if (entry.key.equals(key)) {

                bucket.remove(entry);

                size--;

                return;

            }

        }

    }

    public int size() {

        return size;

    }

    public boolean isEmpty() {

        return size == 0;

    }

    public static void main(String[] args) {

        HashTable<String, Integer> hashTable = new HashTable<>(10);

        hashTable.put("Один", 1);

        hashTable.put("Два", 2);

        hashTable.put("Три", 3);

        System.out.println("Размер: " + hashTable.size());

        System.out.println("Получаем 'Два': " + hashTable.get("Два"));

        hashTable.remove("Два");

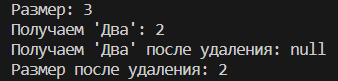
        System.out.println("Получаем 'Два' после удаления: " + hashTable.get("Два"));

        System.out.println("Размер после удаления: " + hashTable.size());

    }

}

Показ работоспособности:



1. Класс HashTable<K, V>

Этот класс представляет саму хэш-таблицу. Он обобщенный и может работать с ключами (K) и значениями (V) любых типов.

2. Вложенный класс Entry<K, V>

Этот вложенный класс представляет запись, содержащую пару ключ-значение. Каждая запись содержит ключ типа K и значение типа V.

3. Поле table

Это поле представляет список списков (LinkedList) записей, которые являются основой для хранения данных в хэш-таблице. Каждый список представляет собой "ведро" (bucket) для хранения записей с одинаковым хэшем.

4. Поле size

Это поле отслеживает текущее количество записей в хэш-таблице.

5. Конструктор HashTable(int capacity)

Этот конструктор создает экземпляр хэш-таблицы с заданным начальным размером (capacity) и инициализирует список списков ведер (buckets).

6. Метод hash(K key)

Этот метод принимает ключ и возвращает его хэш-код, который используется для определения индекса ведра в списке.

7. Метод put(K key, V value)

Этот метод добавляет запись в хэш-таблицу. Он вычисляет хэш ключа, определяет ведро по хэшу и добавляет запись в это ведро. Если запись с таким ключом уже существует в ведре, то значение этой записи обновляется.

8. Метод get(K key)

Этот метод получает значение по ключу из хэш-таблицы. Он вычисляет хэш ключа, определяет ведро по хэшу и выполняет поиск записи с заданным ключом в этом ведре. Если запись с таким ключом найдена, возвращается ее значение, в противном случае возвращается null.

9. Метод remove(K key)

Этот метод удаляет запись с заданным ключом из хэш-таблицы. Он вычисляет хэш ключа, определяет ведро по хэшу и выполняет поиск записи с заданным ключом в этом ведре. Если запись с таким ключом найдена, она удаляется, и размер хэш-таблицы уменьшается на 1.

10. Метод size()

Этот метод возвращает текущий размер хэш-таблицы (количество записей).

11. Метод isEmpty()

Этот метод проверяет, пуста ли хэш-таблица (размер равен 0).

Задание 2

import java.util.\*;

class Book {

    String title;

    String author;

    int copies;

    Book(String title, String author, int copies) {

        this.title = title;

        this.author = author;

        this.copies = copies;

    }

    @Override

    public String toString() {

        return "Название: " + title + ", Автор: " + author + ", Копий: " + copies;

    }

}

public class LibraryBooks {

    private Map<String, Book> booksTable;

    public LibraryBooks() {

        booksTable = new HashMap<>();

    }

    public void insertBook(String isbn, Book book) {

        booksTable.put(isbn, book);

    }

    public Book findBook(String isbn) {

        return booksTable.get(isbn);

    }

    public void deleteBook(String isbn) {

        booksTable.remove(isbn);

    }

    public static void main(String[] args) {

        LibraryBooks library = new LibraryBooks();

        Book book1 = new Book("Война и мир", "Л. Толстой", 3);

        library.insertBook("1234567890", book1);

        Book foundBook = library.findBook("1234567890");

        if (foundBook != null) {

            System.out.println("Книга найдена: " + foundBook + "\n");

        } else {

            System.out.println("Книга не найдена");

        }

        library.deleteBook("1234567890");

        foundBook = library.findBook("1234567890");

        if (foundBook == null) {

            System.out.println("Книга успешно удалена");

        } else {

            System.out.println("Не удалось удалить книгу");

        }

    }

}

Показ работоспособности:



1. Класс Book

Этот класс представляет книгу и содержит следующие поля:

title (название книги)

author (автор книги)

copies (количество копий книги в библиотеке)

Конструктор класса Book принимает параметры для инициализации полей.

Переопределен метод toString для удобного вывода информации о книге в виде строки.

2. Класс LibraryBooks

Этот класс представляет библиотеку и использует Map (конкретно HashMap) для хранения книг. Ключами в Map являются ISBN книги (уникальный идентификатор), а значениями - объекты класса Book.

Конструктор класса LibraryBooks создает пустую библиотеку (пустой HashMap) при инициализации.

Метод insertBook позволяет добавлять новую книгу в библиотеку по указанному ISBN.

Метод findBook ищет книгу в библиотеке по указанному ISBN и возвращает объект Book, если книга найдена, или null, если книга отсутствует.

Метод deleteBook удаляет книгу из библиотеки по указанному ISBN.

3. Метод main

В методе main демонстрируется использование классов LibraryBooks и Book.

Создается объект LibraryBooks, представляющий библиотеку.

Создается объект Book и добавляется в библиотеку с использованием метода insertBook.

Выполняется поиск книги в библиотеке по ISBN с использованием метода findBook. Если книга найдена, она выводится на экран, в противном случае выводится сообщение о том, что книга не найдена.

Выполняется удаление книги из библиотеки с использованием метода deleteBook. Затем выполняется повторный поиск книги для проверки, была ли книга успешно удалена.