

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ**

Ордена Трудового Красного Знамени

**Федеральное государственное бюджетное образовательное
учреждение высшего образования**

«Московский технический университет связи и информатики»

Лабораторная работа № 3

«Создание хэш-таблиц»

Выполнила: Студентка группы БВТ2306

Максимова Мария

Москва 2024

Гитхаб: <https://github.com/Mashaaaaa7/itip-3>

Задачи:

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

Задание 2: Работа с встроенным классом HashMap.

Вариант 4: 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.

При выполнении задания 1 реализуем методы put, get, remove (key) и добавим методы Size и isEmpty.

Код к заданию 1:

```
import java.util.LinkedList;
import java.util.Objects;

public class HashTable<K, V> {

    private static class Entry<K, V> {

        private final K key;
        private V value;
        public Entry(K key, V value) {
            this.key = key;
            this.value = value;
        }

        public K getKey() {
            return key;
        }

        public V getValue() {
            return value;
        }
    }
}
```

```

    public void setValue(V value) {
        this.value = value;
    }

    @Override
    public boolean equals(Object obj) {

        if (this == obj) {
            return true;
        }

        if (!(obj instanceof Entry<?, ?> entry)) {
            return false;
        }

        return Objects.equals(getKey(), entry.getKey());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getKey());
    }
}

private LinkedList<Entry<K, V>>[] table;
private int size;

public HashTable(int capacity) {
    table = new LinkedList[capacity];
    size = 0;
}

private int hash(K key) {
    return Math.abs(Objects.hashCode(key)) % table.length;
}

public void put(K key, V value) {

    int index = hash(key);

    if (table[index] == null) {
        table[index] = new LinkedList<>();
    }

    for (Entry<K, V> entry : table[index]) {
        if (entry.getKey().equals(key)) {
            entry.setValue(value);
            return;
        }
    }

    table[index].add(new Entry<>(key, value));
    size++;
}

public V get(K key) {

    int index = hash(key);

    LinkedList<Entry<K, V>> bucket = table[index];

```

```

        if (bucket != null) {
            for (Entry<K, V> entry : bucket) {
                if (entry.getKey().equals(key)) {
                    return entry.getValue();
                }
            }
        }

        return null;
    }

    public V remove(K key) {

        int index = hash(key);

        LinkedList<Entry<K, V>> bucket = table[index];

        if (bucket != null) {
            for (Entry<K, V> entry : bucket) {
                if (entry.getKey().equals(key)) {
                    V removedValue = entry.getValue();
                    bucket.remove(entry);
                    size--;
                    return removedValue;
                }
            }
        }

        return null;
    }

    public int size() {
        return size;
    }

    public boolean isEmpty() {
        return size == 0;
    }
}

```

Этот код представляет простую имитацию хэш-таблицы с использованием класса Entry.

1. Внутренний класс Entry:

```

import java.util.LinkedList;
import java.util.Objects;

public class HashTable<K, V> { no usages

    private static class Entry<K, V> { 8 usages

        private final K key; 2 usages
        private V value; 3 usages
        public Entry(K key, V value) { 1 usage
            this.key = key;
            this.value = value;
        }

        public K getKey() { 6 usages
            return key;
        }

        public V getValue() { 2 usages
            return value;
        }

        public void setValue(V value) { 1 usage
            this.value = value;
        }
    }
}

```

Рисунок 1

1. Класс HashTable:

- Класс Entry<K,V> представляет пару ключ-значение.
- private K key;: хранит ключ типа String.
- private V value;: хранит значение типа String.
- Класс Entry содержит HashMap, в котором ключом является строка, а значением - объект класса KeyValue.
- Метод put добавляет пару ключ-значение в таблицу.
- Метод get возвращает значение по заданному ключу.
- Метод remove удаляет значение по ключу.
- Метод size возвращает размер таблицы (количество элементов).
- Метод isEmpty проверяет, пуста ли таблица.

- **public boolean equals(Object obj):**

Этот метод переопределяет стандартный метод equals() из класса Object. Он определяет, равны ли два объекта.

- **if (this == obj):**

Проверяет, сравниваются ли один и тот же объект. Если да, то объекты равны.

- **if (!(obj instanceof Entry<?, ?> entry)):**

Проверяет, является ли передаваемый объект obj экземпляром класса Entry. Если нет, то объекты не равны. Entry<?, ?> — это обобщенный тип Entry, который может содержать любые типы ключей и значений.

- **return Objects.equals(getKey(), entry.getKey());:**

Сравнивает ключи двух объектов Entry. getKey() — это метод для получения ключа из объекта Entry. Objects.equals() — это статический метод класса Objects, который безопасно сравнивает два объекта на равенство, обрабатывая случаи null.

- **private LinkedList<Entry<K, V>>[] table;:**

Этот код объявляет массив ссылок на связанные списки (LinkedList).

- `LinkedList<Entry<K, V>>`: Каждый элемент массива будет ссылаться на связанный список, хранящий элементы типа `Entry<K, V>`. `Entry<K, V>` — это класс, который, скорее всего, используется для хранения пары ключ-значение (key-value) в хэш-таблице.
- `[]`: Означает, что `table` — это массив.
- `private`: Означает, что это поле **приватное** и доступно только внутри класса `HashTable`.

- **`private int size;`**

Это поле **целого типа** и хранит **количество элементов** в `HashTable`. Оно тоже является **приватным** и доступно только внутри класса.

- **`public HashTable(int capacity):`**

Это конструктор класса `HashTable`. Он принимает один аргумент `capacity`, который задает **начальную емкость** хэш-таблицы (количество элементов в массиве `table`).

- **`table = new LinkedList[capacity];`**

Внутри конструктора создается массив `table` заданной емкости.

- **`size = 0;`**

Изначально `size` устанавливается в 0, так как в хэш-таблице еще нет элементов.

- **`private int hash(K key):`**

Это приватный метод, который принимает ключ `key` и возвращает хеш-код этого ключа. Хеш-код используется для определения позиции элемента в массиве `table`.

- **`Math.abs(Objects.hashCode(key)):`**

- `Objects.hashCode(key)`: Вызывает стандартный метод `hashCode()` для получения хеш-кода переданного ключа `key`.
- `Math.abs(...)`: Возвращает абсолютное значение хеш-кода, чтобы получить неотрицательное число.

- **% table.length:**

Берет остаток от деления хеш-кода на длину массива table. Этот остаток используется как индекс в массиве, определяя положение элемента.

Далее, перейдем к заданию 2.

Мой код:

```
import java.util.HashMap;

class Book { //определяет класс, который представляет книгу
    private String title;
    private String author;
    private int copies;

    public Book(String title, String author, int copies) { //инициализирует
        объект Book
        this.title = title;
        this.author = author;
        this.copies = copies;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    public int getCopies() {
        return copies;
    }

    // Метод для изменения количества копий
    public void setCopies(int copies) {
        this.copies = copies;
    }
}

public class Library { //определяет класс Библиотека
    private HashMap<String, Book> bookMap; // Явное объявление HashMap

    public Library() {
        bookMap = new HashMap<>(); //создает поле bookMap типа HashMap,
        которое будет хранить книги с ключом ISBN.
    }

    public void addBook(String isbn, Book book) {
        bookMap.put(isbn, book); // Добавление книги в хэш-таблиц
    }

    public Book findBook(String isbn) {
        return bookMap.get(isbn);
    }
}
```



```

public void removeBook(String isbn) {
    bookMap.remove(isbn);
}

public static void main(String[] args) {
    Library library = new Library(); //создает объект Library

    // Добавление книг
    library.addBook("978-0143034231", new Book("1984", "George Orwell",
5));
    library.addBook("978-0141439501", new Book("Pride and Prejudice",
"Jane Austen", 3));

    // Поиск книги
    Book foundBook = library.findBook("978-0143034231");
    if (foundBook != null) { //проверяет, найдена ли книга.
        System.out.println("Найдена книга: " + foundBook.getTitle() + "
by " + foundBook.getAuthor());
    } else {
        System.out.println("Книга не найдена.");
    }

    Book foundBook2 = library.findBook("978-0141439501");
    if (foundBook2 != null) {
        System.out.println("Найдена книга: " + foundBook2.getTitle() + "
by " + foundBook2.getAuthor());
    } else {
        System.out.println("Книга не найдена.");
    }

    // Удаление книги
    library.removeBook("978-0141439501");
    library.removeBook("978-0141439501");

    // Поиск удаленной книги
    foundBook = library.findBook("978-0141439501");
    if (foundBook != null) {
        System.out.println("Книга найдена.");
    } else {
        System.out.println("Книга не найдена.");
    }

    foundBook2 = library.findBook("978-0141439501");
    if (foundBook2 != null) {
        System.out.println("Книга найдена.");
    } else {
        System.out.println("Книга не найдена.");
    }
}
}

```

Этот код представляет собой пример использования HashMap в Java для хранения книг в библиотеке.

В методе main создается объект класса Library и добавляются две книги. Затем находится книга по ISBN, выводится информация о ней. Книга удаляется по ISBN и затем снова производится поиск этой книги, результат которого выводится в консоль.

Класс Book:

```
class Book { //определяет класс, который представляет книгу 6 usages
    private String title; 2 usages
    private String author; 2 usages
    private int copies; 3 usages

    public Book(String title, String author, int copies) { //инициализирует объект Book с указанными названием, автором и количеством копий
        this.title = title;
        this.author = author;
        this.copies = copies;
    }

    public String getTitle() { 1 usage
        return title;
    }

    public String getAuthor() { 1 usage
        return author;
    }

    public int getCopies() { no usages
        return copies;
    }

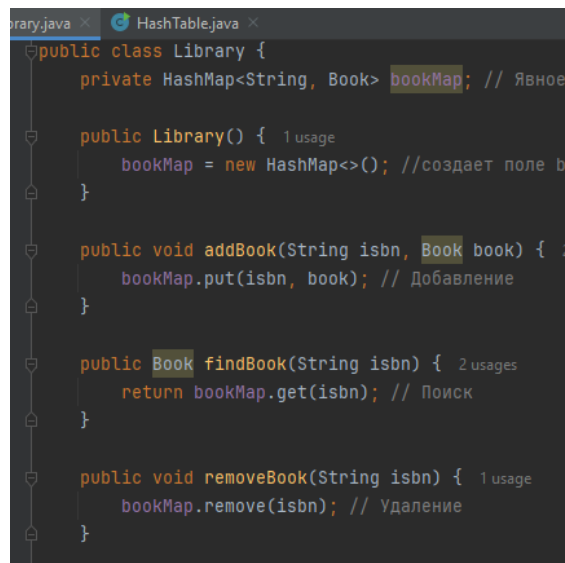
    // Метод для изменения количества копий
    public void setCopies(int copies) { no usages
        this.copies = copies;
    }
}
```

Рисунок 6

Класс Book представляет книгу с полями заголовок, автор и количество копий. В классе Library создается HashMap bookMap, где ключом является ISBN книги, а значением объект книги.

- class Book: определяет класс, который представляет книгу с названием, автором и количеством копий.
- private String title;: хранит название книги.
- private String author;: хранит имя автора.
- private int copies;: хранит количество копий книги.
- Book(String title, String author, int copies): конструктор, который инициализирует объект Book с указанными названием, автором и количеством копий.
- getTitle(), getAuthor(), getCopies(): геттеры для получения информации о книге.
- setCopies(int copies): сеттер для изменения количества копий.

Класс Library:



```
public class Library {  
    private HashMap<String, Book> bookMap; // Явное  
  
    public Library() { 1 usage  
        bookMap = new HashMap<>(); //создает поле bo  
    }  
  
    public void addBook(String isbn, Book book) { 2  
        bookMap.put(isbn, book); // Добавление  
    }  
  
    public Book findBook(String isbn) { 2 usages  
        return bookMap.get(isbn); // Поиск  
    }  
  
    public void removeBook(String isbn) { 1 usage  
        bookMap.remove(isbn); // Удаление  
    }  
}
```

Рисунок 7

- `public class Library`: определяет класс Library.
- `private HashMap<String, Book> bookMap`:: создает поле bookMap типа HashMap, которое будет хранить книги с ключом ISBN.
- `Library()`: конструктор, который инициализирует bookMap пустым HashMap.

Методы Library:

- `public void addBook(String isbn, Book book)`: добавляет книгу в библиотеку по ее ISBN.
- `public Book findBook(String isbn)`: ищет книгу по ее ISBN и возвращает ее объект, если она найдена, иначе возвращает null.
- `public void removeBook(String isbn)`: удаляет книгу из библиотеки по ее ISBN.

main() метод:

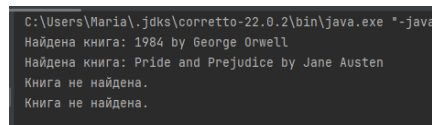
```
public static void main(String[] args) {  
    Library library = new Library(); //создает объект Library  
  
    // Добавление книг  
    library.addBook( isbn: "978-0143034231", new Book( title: "1984", author: "George Orwell", copies: 5));  
    library.addBook( isbn: "978-0141439501", new Book( title: "Pride and Prejudice", author: "Jane Austen", copies: 3));  
  
    // Поиск книги  
    Book foundBook = library.findBook( isbn: "978-0143034231");  
    if (foundBook != null) { //проверяет, найдена ли книга.  
        System.out.println("Найдена книга: " + foundBook.getTitle() + " by " + foundBook.getAuthor());  
    } else {  
        System.out.println("Книга не найдена.");  
    }  
}
```

Рисунок 8

- `public static void main(String[] args)`: точка входа в программу.
- `Library library = new Library();`: создает объект `Library`.
- `library.addBook("978-0143034231", new Book("1984", "George Orwell", 5));`: добавляет книгу “1984” в библиотеку.
- `library.addBook("978-0141439501", new Book("Pride and Prejudice", "Jane Austen", 5));`: добавляет книгу “Pride and Prejudice” в библиотеку.
- `Book foundBook = library.findBook("978-0143034231");`: ищет книгу по ISBN и сохраняет ее в переменную `foundBook`.
- `Book foundBook2 = library.findBook("978-0141439501");`: ищет книгу по ISBN и сохраняет ее в переменную `foundBook`.
- `if (foundBook != null) { ... }`: проверяет, найдена ли книга. Если да, то выводит информацию о ней, иначе выводит сообщение о том, что книга не найдена.

- `library.removeBook("978-0141439501");`: удаляет книгу по ISBN.
- `library.removeBook("978-0141439501");`: удаляет книгу по ISBN.
- `foundBook = library.findBook("978-0141439501");`: ищет удаленную книгу.
- `foundBook2 = library.findBook("978-0141439501");`: ищет удаленную книгу.
- `if (foundBook != null) { ... }`: проверяет, найдена ли книга. Если да, то выводит сообщение о том, что книга найдена, иначе выводит сообщение о том, что книга не найдена.

Вывод в консоли:



```
C:\Users\Maria\.jdk\corretto-22.0.2\bin\java.exe "-java
Найдена книга: 1984 by George Orwell
Найдена книга: Pride and Prejudice by Jane Austen
Книга не найдена.
Книга не найдена.
```

Рисунок 9