МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАГИИ
Ордена Трудового Красного Знамени федеральное государственное бюджетное учреждение высшего образования «Московский технический университет связи и информатики»

Кафедра «Информационные технологии»
Предмет «Математические Основы Баз Данных»

Лабораторная работа №3

Создание хэш таблицы и работа с HashMap

Вариант 2

Выполнил:
студент гр. БПИ2402
Поляков Н.А.

Москва
2025

Содержание

1 Цель работы **2**

2 Индивидуальное задание **2**

3 Выполнение **2**

3.1 Задание 1 2

3.2 Задание 2 6

4 Вывод **8**

5 Github **8**

1 Цель работы

Освоить работу с хэш-таблицами в Java. Научиться писать тип HashTable самостоятельно с помощью метода цепочек

2 Индивидуальное задание

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.

Рис. 1

2. Реализация хэш-таблицы для хранения информации о товарах в интернет-магазине. Ключом является артикул товара, а значением — объект класса Product, содержащий поля наименование, описание, цена и количество на складе. Необходимо реализовать операции вставки, поиска и удаления товара по артикулу.

Рис. 2: Задание 2. Работа с HashMap

3 Выполнение

3.1 Задание 1

Сначала я создал класс HashTable. В примере из листинга 3.3 использовались типы дженерики K и V для типов ключа и значения.

```java
public class HashTable<K, V> {
  private final int default_length = 10;
  private Node<K, V>[] table;
  private int size;

  private static class Node<K, V> {
    K key;
    V value;
    Node<K, V> next;

    Node(K key, V value, Node<K, V> next) {
      this.key = key;
      this.value = value;
      this.next = next;
    }
  }

  public HashTable(int length) {
    if (length > 0)
      this.table = new Node[length];
    else
      this.table = new Node[default_length];
  }
  public HashTable() {
  this.table = new Node[default_length];
  }

```

Рис. 3: Класс HashTable

Далее создадим методы hash, put, get и remove:

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

Рис. 4: метод hash

```java
public void put(K key, V value) {
  int index = hash(key);
  Node<K, V> current = table[index];

  while (current != null) {
    if (current.key.equals(key)) {
      current.value = value;
      return;
    }
    current = current.next;
  }

  table[index] = new Node<K,V>(key, value, table[index]);
  size++;
}
```

Рис. 5: метод put

```java
public V get(K key) {
  int index = hash(key);
  Node<K, V> current = table[index];

  while (current != null) {
    if (current.key.equals(key))
      return current.value;
    current = current.next;
  }

  return null;
}
```

Рис. 6: метод get

```java
public void remove(K key) {
 int index = hash(key);
  Node<K, V> current = table[index];
  Node<K, V> previous = null;

  while (current != null) {
    if (current.key.equals(key)) {
      if (previous == null)
        table[index] = current.next;
      else
        previous.next = current.next;
      size--;
      return;
    }
    previous = current;
    current = current.next;
  }
}
```

Рис. 7: метод remove

Дополнительно, по заданию, реализуем методы isEmpty, getSize:

```
31    public int getSize() {
32      return this.size;
33    }
34    public boolean isEmpty() {
35      return this.size == 0;
36    }
```

Рис. 8

Для удобства вывода всех значений я написал метод show, который показывает их:

```
83  public void show() {
84    System.out.println("Key\tValue--------------------------");
85    for (int i = 0; i < table.length; i++) {
86      System.out.printf("Index: %d\n", i);
87      Node<K, V> current = table[i];
88      while (current != null) {
89        System.out.printf("%s\t%s\n", current.key, current.value);
90        current = current.next;
91      }
92    }
93    System.out.println("End-----------------------------------");
94  }
```

Рис. 9

Теперь напишем тестовую программу, где добавим несколько значений в разные хэш-таблицы: Здесь я сначала создаю таб-

```java
public class Main {
  public static void main(String[] args) {
    HashTable<String, Integer> myHashTable = new HashTable<String, Integer>(17);
    myHashTable.put("Test Key", 114);
    myHashTable.show();
    myHashTable.put("Orange", 17);
    myHashTable.show();
    myHashTable.put("Organism", 45);
    myHashTable.show();

    System.out.println("Значение по ключу Orange: " + myHashTable.get("Orange"));
    System.out.println("Значение по ключу asdhkjthkjv: " + myHashTable.get("asdhkjthkjv"));

    myHashTable.remove("Organism");

    System.out.println("Значение по ключу Organism: " + myHashTable.get("Organism"));
    myHashTable.show();

    HashTable<Integer, Boolean> isPrimeTable = new HashTable<Integer, Boolean>(101);
    for (int i = 0; i < 1000; i++) {
      isPrimeTable.put(i, Primes.isPrime(i)); // метод взял из первой лабораторной
    }

    System.out.println("Является ли число 101 простым: " + isPrimeTable.get(101));
  }
}
```

Рис. 10

лицу с ключом-строкой и значением-числом, записываю в него случайные данные. Далее удаляю и смотрю, как будут меняться значения.

После я создал таблицу isPrimeTable с ключом-числом и значением-числом. В нём я храню все числа от 0 до 999 и храню информацию о том, являются ли они простыми. Метод isPrime я взял из лабораторной работы

3.2 Задание 2

Для выполнения второго задания я для начала создал класс Product, в котором будет храниться вся информация о конкретном продукте. А также класс ProductManager, который будет хранить в себе хэш-таблицу с ключом-строкой и значением-объект класса product. Это распространённая практика управлением данными в Java:

```java
class Product {
  private String name;
  private String description;
  private double price;
  private int quantity;

  public String getName() { return name; }
  public String getDesc() { return description; }
  public double getPrice() { return price; }
  public int getQuantity() { return quantity; }

  public void setName(String name) { this.name = name; }
  public void setDesc(String description) { this.description = description; }
  public void setPrice(double price) { this.price = price; }
  public void setQuantity(int quantity) { this.quantity = quantity; }

  @Override
  public String toString() {
      return String.format("Название: %s | Описание: %s | Цена: %.2f | На складе: %d",
              name, description, price, quantity);
  }

  public Product(String name, String description, double price, int quantity) {
    this.name = name;
    this.description = description;
    this.price = price;
    this.quantity = quantity;
  }
}
```

Рис. 11: класс Product

```java
import java.util.HashMap;

class ProductManager {
  private final HashMap<String, Product> productMap = new HashMap<String, Product>();

  public void addProduct(String article,
                         String pName,
                         String pDescription,
                         double pPrice,
                         int pQuantity) {
    productMap.put(article, new Product(pName, pDescription, pPrice, pQuantity));
  }
  public void addProduct(String article, Product product) {
    productMap.put(article, product);
  }
  public Product getProduct(String article) {
    return productMap.get(article);
  }
  public Product removeProduct(String article) {
    return productMap.remove(article);
  }
  public void showProducts() {
    System.out.println("Все товары: ");
    for (String article : productMap.keySet()) {
      Product p = productMap.get(article);
      System.out.println(p.toString());
    }
  }
}
```

Рис. 12: класс ProductManager

Всё! Теперь напишем простую программу, в которой будем использовать все методы ProductManager:

```java
public class Main {

  public static void main(String[] args) {
    ProductManager manager = new ProductManager();
    manager.addProduct("Ф001", new Product("Яблоко", "Красные яблочки, цена за килограмм", 120.00, 40));
    manager.addProduct("Ф002", new Product("Апельсин", "Обычные апельсины, цена за килограмм", 170.00, 50));
    manager.addProduct("Ж001", new Product("Молоко", "Молоко простоквашино 900 л", 93.00, 12));

    System.out.println("Существует ли продукт с артикулом Ф002:" + (manager.getProduct("Ф002") != null));
    System.out.println("Существует ли продукт с артикулом А002:" + (manager.getProduct("А002") != null));

    manager.showProducts();
  }
}
```

Рис. 13: Пример работы с ProductManager

4 Вывод

Я освоил работу с хэш-таблица в Java, научился создавать свои собственные таблицы с помощью метода цепочек. А также дополнительно обучился работе с дженериками.

5 Github

https://github.com/KLARKOFF/IT-and-Programming-labs