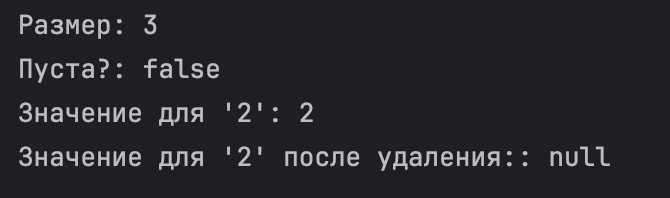
**Отчет по лабораторной работе №3**

import java.util.LinkedList;  
  
public class HashTable<K, V> {  
 private static final int *INITIAL\_CAPACITY* = 16;  
 private LinkedList<Entry<K, V>>[] table;  
 private int size;  
  
 public HashTable() {  
 table = new LinkedList[*INITIAL\_CAPACITY*];  
 size = 0;  
 }  
  
 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);  
  
 if (table[index] != null) {  
 for (Entry<K, V> entry : table[index]) {  
 if (entry.getKey().equals(key)) {  
 return entry.getValue();  
 }  
 }  
 }  
  
 return null;  
 }  
  
 public void remove(K key) {  
 int index = hash(key);  
  
 if (table[index] != null) {  
 table[index].removeIf(entry -> entry.getKey().equals(key));  
 size--;  
 }  
 }  
  
 public int size() {  
 return size;  
 }  
  
 public boolean isEmpty() {  
 return size == 0;  
 }  
  
 private int hash(K key) {  
 int hashCode = key.hashCode();  
 return (hashCode & 0x7FFFFFFF) % table.length;  
 }  
  
 private static class Entry<K, V> {  
 private 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;  
 }  
 }  
  
 public static void main(String[] args) {  
 HashTable<String, Integer> hashTable = new HashTable<>();  
 hashTable.put("one", 1);  
 hashTable.put("two", 2);  
 hashTable.put("three", 3);  
  
 System.*out*.println("Размер: " + hashTable.size());  
 System.*out*.println("Пуста?: " + hashTable.isEmpty());  
  
 System.*out*.println("Значение для '2': " + hashTable.get("two"));  
 hashTable.remove("two");  
 System.*out*.println("Значение для '2' после удаления:: " + hashTable.get("two"));  
 }  
}

1. **Поля класса:**
   * **INITIAL\_CAPACITY: Константа, устанавливающая начальную емкость хэш-таблицы.**
   * **table: Массив связанных списков для хранения пар ключ-значение.**
   * **size: Текущий размер хэш-таблицы.**
2. **Методы:**
   * **put(K key, V value): Добавляет элемент в таблицу по указанному ключу. Если элемент с таким ключом уже существует, его значение обновляется.**
   * **get(K key): Возвращает значение по ключу. Если элемент существует, возвращает его значение, иначе - null.**
   * **remove(K key): Удаляет элемент из таблицы по ключу.**
   * **size(): Возвращает текущий размер хэш-таблицы.**
   * **isEmpty(): Проверяет, пуста ли хэш-таблица.**
3. **Приватные методы:**
   * **hash(K key): Применяет хэш-функцию к ключу для определения индекса в массиве.**
4. **Внутренний класс Entry:**
   * **key: Ключ элемента.**
   * **value: Значение элемента.**
5. **Метод main:**
   * **Создаёт экземпляр хэш-таблицы с типами String в качестве ключа и Integer в качестве значения.**
   * **Добавляет несколько элементов, выводит размер, проверяет на пустоту, получает и удаляет элемент по ключу.**

**Реализация и особенности:**

* **Хэш-таблица использует массив связанных списков для разрешения коллизий.**
* **Метод hash() применяет хэш-функцию к ключу для определения индекса в массиве.**
* **При добавлении элемента проверяется, существует ли уже элемент с таким ключом, и в случае наличия происходит обновление значения.**
* **Внутренний класс Entry представляет пару ключ-значение.**
* **Методы size() и isEmpty() позволяют получить информацию о текущем размере и пустоте таблицы.**

****

**2.**

import java.util.HashMap;  
  
public class Contact {  
 private String name;  
 private String email;  
 private String additionalInfo;  
  
 public Contact(String name, String email, String additionalInfo) {  
 this.name = name;  
 this.email = email;  
 this.additionalInfo = additionalInfo;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public String getAdditionalInfo() {  
 return additionalInfo;  
 }  
}  
  
 class PhoneBook {  
 private HashMap<String, Contact> contacts;  
  
 public PhoneBook() {  
 contacts = new HashMap<>();  
 }  
  
 public void addContact(String phoneNumber, Contact contact) {  
 contacts.put(phoneNumber, contact);  
 }  
  
 public Contact findContact(String phoneNumber) {  
 return contacts.get(phoneNumber);  
 }  
  
 public void removeContact(String phoneNumber) {  
 contacts.remove(phoneNumber);  
 }  
  
 public static void main(String[] args) {  
 PhoneBook phoneBook = new PhoneBook();  
  
 // Добавление контактов  
 Contact contact1 = new Contact("John Doe", "john@example.com", "Additional info 1");  
 Contact contact2 = new Contact("Jane Smith", "jane@example.com", "Additional info 2");  
  
 phoneBook.addContact("+1234567890", contact1);  
 phoneBook.addContact("+9876543210", contact2);  
  
 // Поиск контакта по номеру телефона  
 Contact foundContact = phoneBook.findContact("+1234567890");  
 if (foundContact != null) {  
 System.*out*.println("Found contact: " + foundContact.getName());  
 } else {  
 System.*out*.println("Contact not found.");  
 }  
  
 // Удаление контакта  
 phoneBook.removeContact("+1234567890");  
 Contact removedContact = phoneBook.findContact("+1234567890");  
 if (removedContact == null) {  
 System.*out*.println("Contact removed.");  
 } else {  
 System.*out*.println("Failed to remove contact.");  
 }  
 }  
}

**Класс Contact:**

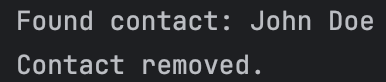
1. **Поля класса:**
   * **name: Имя контакта.**
   * **email: Адрес электронной почты контакта.**
   * **additionalInfo: Дополнительная информация о контакте.**
2. **Методы:**
   * **Contact(String name, String email, String additionalInfo): Конструктор, устанавливающий значения полей контакта.**
   * **getName(): Возвращает имя контакта.**
   * **getEmail(): Возвращает адрес электронной почты контакта.**
   * **getAdditionalInfo(): Возвращает дополнительную информацию о контакте.**

**Класс PhoneBook:**

1. **Поля класса:**
   * **contacts: Хранит соответствие между номером телефона и контактом.**
2. **Методы:**
   * **PhoneBook(): Конструктор, инициализирующий пустую телефонную книгу.**
   * **addContact(String phoneNumber, Contact contact): Добавляет контакт в телефонную книгу по указанному номеру телефона.**
   * **findContact(String phoneNumber): Ищет контакт по номеру телефона и возвращает его.**
   * **removeContact(String phoneNumber): Удаляет контакт из телефонной книги по номеру телефона.**
3. **Метод main:**
   * **Создаёт экземпляр телефонной книги.**
   * **Добавляет несколько контактов с номерами телефонов.**
   * **Ищет контакт по номеру телефона и выводит информацию о найденном контакте или сообщение о его отсутствии.**
   * **Удаляет контакт и проверяет, успешно ли он был удалён из книги.**

**Особенности и комментарии:**

* **Класс Contact представляет отдельный контакт со всей своей информацией.**
* **Класс PhoneBook использует HashMap для хранения контактов по номерам телефонов.**
* **Методы addContact, findContact и removeContact обеспечивают основные операции добавления, поиска и удаления контактов.**
* **В main методе создаётся PhoneBook, добавляются контакты, проверяется поиск и удаление контакта.**

****

**Git**

**https://github.com/Red6Wolf/JavaLab.git**