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

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе № 3

Выполнил: студент группы БВТ2402

Косякова Олеся Дмитриевна

Москва, 2025

Цель работы: изучить принципы работы хэш-таблиц и методов класса Object в Java, а также приобрести практических навыков реализации собственной структуры данных.

Задание:

1) Создайте класс HashTable, который будет реализовывать хэш-таблицу с помощью метода цепочек.
2) Реализация хэш-таблицы для хранения информации о то варах 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.

Ход работы:

Задание 1.

Создаем обобщенный класс, и массив хранящий списки пар ключ-значение. Прописываем внутренний класс для хранения этих пар.

```java
import java.util.LinkedList;
//Класс является обощенным и может работать с любым типом данных
//Если это не использовать то ключ и значение будут обрабатываться как объекты типа Object
//И придется явно преобразовывать типы

//K - параметр типа для ключа V - параметр типа для значения
public class HashTable<K, V> {
    // Создаем массив из связзаных списков, в каждом списке хранится пара ключ -
    // значение
    // Entry - внутренний класс созданный для хранения пары ключ-значение
    // Если убрать [] то это будет один связный список без индексации по хэшу
    private LinkedList<Entry<K, V>>[] table;
    // Создаем параметры хранящие текущее количество пар и вместимость таблицы
    // Сразу задаем значения для упрощения кода, чтобы не писать явный конструктор
    private int size = 0;
    private int capacity = 16;

    // Внутренние классы используются для группировки тесно связанных объектов с
    // внешним классом
    // В нашем случае это структура для хранения пары ключ-значение
    // Она не используется вне класса хэштейбл поэтому логично сделать ее внутренним
    // классом
    // Специально не делаем сеттер для ключа так как его менять нельзя и задать
    // можно только при создании записи
    private static class Entry<K, V> {
```

```java
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;
    }
}

// Конструктор
//При создании нового объекта конструктор выделяет память под массив ссылок типа LinkedList
//и длинной capacity
//Если конструктор будет пустой то массив не создастся
public HashTable() {
    table = new LinkedList[capacity];
}
```

Далее реализуем хэш-методы, такие как put, get, remove, size, isEmpty

```java
//Реализуем хеш-функцию. Если ключ равен нул то мы возвращаем 0 -
//значит все пары с ключом нул будут в 0 корзине
//Берем модуль хешированого ключа и делим на наше количество корзин это и будет номер корзины
// Хеш-код - числовое представление объектов
private int hash(K key) {
    if (key == null) {
        return 0;
    } else {
        return Math.abs(key.hashCode()) % capacity;
    }
}

public void put(K key, V value) {
    //Вычисляем индекс нашей корзины в которой будет хранится пара
    int index = hash(key);
    //Проверяем есть ли по этому индексу созданная корзина если нет создаем
    if (table[index] == null) {
        table[index] = new LinkedList<>();
    }
    //Проходим по всем парам в этой корзине и проверяем совпадают ли ключи
    //Если да то записываем этому ключу значение и выходим из метода
    for (Entry<K, V> entry : table[index]) {
        if (entry.getKey().equals(key)) {
            entry.setValue(value);
            return;
        }
    }
    //Если ключ не найден в корзине то создаем новую пару и увеличиваем счетчик
    table[index].add(new Entry<>(key, value));
    size++;
}

public V get(K key) {
    int index = hash(key);
    if (table[index] == null) {
        return 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) {
            return;
        }
        for (Entry<K, V> entry : table[index]) {
            if (entry.getKey().equals(key)) {
                table[index].remove(entry);
                size--;
                return;
            }
        }
    }

    public int size() {
        return size;
    }

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

}
```

Задание 2.

Создаем класс Product, который будет хранить название, описание, цену и количество продуктов. Прописываем конструктор, геттеры, сеттеры и переопределяем метод toString, чтобы при выводе данных из хэш-таблицы мы получили детальную информацию об объекте

```
    @Override
    public String toString() {
        return "название: '" + title + '\'' +
                ", описание: '" + description + '\'' +
                ", цена: " + price +
                ", количество: " + quantity;
    }
}
```

Далее создаем класс ProductAdd и в нем переменную хранящую пары ключ-значение. Далее прописываем методы этого класса используя функции хэш-таблицы.

```
import java.util.HashMap;

public class ProductAdd {
    //Создаем переменную хранящую пары ключ-значение
    private HashMap<Integer, Product> productMap;

    //Создаем пустой объект HashMap который будет хранить пары
    public ProductAdd() {
        productMap = new HashMap<>();
    }

    public void addProduct(int number, Product product) {
        productMap.put(number, product);
    }
```

```java
    public Product findProduct(int number) {
        return productMap.get(number);
    }

    public void removeProduct(int number) {
        Product removedProduct = productMap.get(number);
        if (productMap.containsKey(number)) {
            productMap.remove(number);
            System.out.println("Продукт " + removedProduct + " удален");
        } else {
        System.out.println("Продукт " + removedProduct + " не найден"); }
    }

    Run | Debug
    public static void main(String[] args) {
        ProductAdd table = new ProductAdd();
        Product a = new Product(title:"Шампунь", description:"для сухих волос", price:300, quantity:5);
        Product b = new Product(title:"Пальто", description:"шерстяное", price:10000, quantity:2);

        table.addProduct(number:1233, a);
        table.addProduct(number:1533, b);

        System.out.println("Найден продукт: " + table.findProduct(number:1233));
        table.removeProduct(number:1233);
    }
}
```

Вывод консоли:

```
PS C:\Users\Олеся\it>  & 'C:\Program Files\Java\jdk-25\bin\java.exe' '-XX:+ShowCodeDetailsInExceptio
nMessages' '-cp' 'C:\Users\Олеся\AppData\Roaming\Code\User\workspaceStorage\7548f2c86350adc58ce52ec0
4885fe19\redhat.java\jdt_ws\it_d94a6b3\bin' 'lab3.ProductAdd'
Найден продукт: название: 'Шампунь', описание: 'для сухих волос', цена: 300.0, количество: 5
Продукт название: 'Шампунь', описание: 'для сухих волос', цена: 300.0, количество: 5 удален
PS C:\Users\Олеся\it>
```

Вывод: в ходе лабораторной работы мы изучили принципы работы хэш-таблиц и методов класса Object в Java, а также приобрели практические навыки реализации собственной структуры данных.

Ссылка на репозиторий: https://github.com/swydiz/information_technology