

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ**

Ордена Трудового Красного Знамени

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«Московский технический университет связи и информатики»

Кафедра «Математическая Кибернетика и Информационные технологии»

Дисциплина «Информационные технологии и программирование»

Лабораторная работа №3

«Хэш-таблицы в Java»

Выполнила:

Студентка группы БВТ2303

Морозова Ольга

Цель работы:

Изучение хэш-таблиц, их возможностей и особенностей, и применение их на практике на языке программирования Java.

Ход работы:

Задание 1.

Для создания собственного класса для реализации хэш-таблицы необходимо сначала создать класс, содержащий пару ключ-значение, объекты которого мы впоследствии и будем хранить в массиве хэш-таблицы. В этом классе задаём две переменные и key и value (обе типа String для возможности более обширного использования нашей хэш-таблицы с разными типами данных при необходимости).



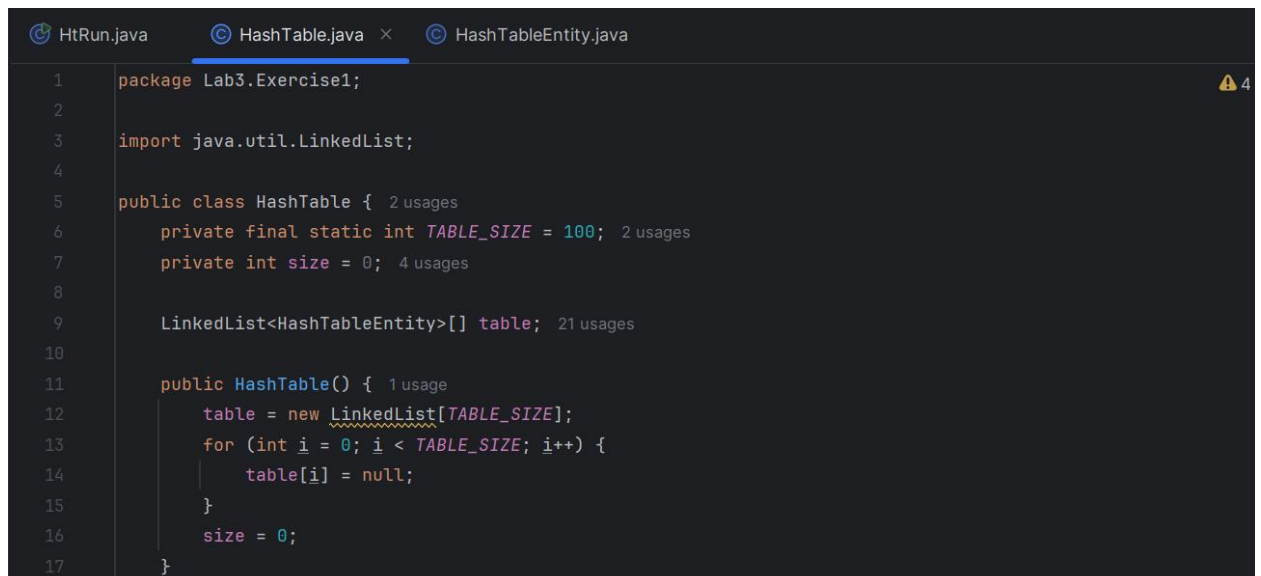
```
1 package Lab3.Exercise1;
2
3 public class HashTableEntity { 3 usages
4     private String key; 3 usages
5     private String value; 3 usages
6
7     public HashTableEntity (String key, String value) { 2 usages
8         this.key = key;
9         this.value = value;
10    }
11
12    public HashTableEntity () { no usages
13        this("", "");
14    }
15
16    public String getKey() { 4 usages
17        return key;
18    }
19
20    public void setKey(String key) { no usages
21        this.key = key;
22    }
23
24    public String getValue() { 1 usage
25        return value;
26    }
27
28    public void setValue(String value) { 1 usage
29        this.value = value;
30    }
31 }
```

Далее создаём сам класс `HashTable`, в котором и будет прописана реализация функций хеш-таблицы. Задаём константу для указания размера массива хеш-таблицы, а также поле `size` для подсчёта количества занесённых в хеш-таблицу пар ключ-значение. Изначально при создании объекта класса заполняем всю таблицу значением `null`.

Используя геттеры и сеттеры класса `HashTableEntity` прописываем метод `put()` для занесения в хеш-таблицу новой пары ключ-значение с проверкой коллизий и её разрешением методом цепочек (с помощью использования `LinkedList` в каждом элементе массива).

Прописываем методы `get()` и `remove()` для получения и удаления данных с проверкой наличия объекта с указанным ключём.

Прописываем методы `size()` и `isEmpty()` для получения количества записанных в хеш-таблицу пар ключ-значение и проверки, пустая ли хеш-таблица.



```
1 package Lab3.Exercise1;
2
3 import java.util.LinkedList;
4
5 public class HashTable { 2 usages
6     private final static int TABLE_SIZE = 100; 2 usages
7     private int size = 0; 4 usages
8
9     LinkedList<HashTableEntity>[] table; 21 usages
10
11     public HashTable() { 1 usage
12         table = new LinkedList[TABLE_SIZE];
13         for (int i = 0; i < TABLE_SIZE; i++) {
14             table[i] = null;
15         }
16         size = 0;
17     }
```

```

19 @ public void put(String key, String value) { 6 usages
20     int index;
21     if (key.isEmpty()) {
22         index = 0;
23     } else {
24         index = Character.getNumericValue(key.charAt(0)) % 100;
25     }
26     if (index < 0) {
27         index = 0;
28     }
29     if (table[index] == null) {
30         table[index] = new LinkedList<HashTableEntity>();
31     } else {
32         for (int i = 0; i < table[index].size(); i++) {
33             if (table[index].get(i).getKey().equals(key)) {
34                 table[index].get(i).setValue(value);
35                 return;
36             }
37         }
38     }
39     table[index].add(new HashTableEntity(key, value));
40     size++;
41 }

```

```

43 @ public String[] get(String key) { 5 usages
44     int index;
45     if (key.isEmpty()) {
46         index = 0;
47     } else {
48         index = Character.getNumericValue(key.charAt(0)) % 100;
49     }
50     if (table[index] == null) {
51         return new String[] {"None", "None"};
52     } else {
53         for (int i = 0; i < table[index].size(); i++) {
54             if (table[index].get(i).getKey().equals(key)) {
55                 return new String[] {table[index].get(i).getKey(), table[index].get(i).getValue()};
56             }
57         }
58     }
59     return new String[] {"None", "None"};
60 }

```

```

62 @      public void remove(String key) { 2 usages
63         int index;
64         if (key.isEmpty()) {
65             index = 0;
66         } else {
67             index = Character.getNumericValue(key.charAt(0)) % 100;
68         }
69         if (table[index] != null) {
70             for (int i = 0; i < table[index].size(); i++) {
71                 if (table[index].get(i).getKey().equals(key)) {
72                     table[index].remove(i);
73                     size--;
74                     return;
75                 }
76             }
77         }
78     }
79
80     public int size() { no usages
81         return size;
82     }

```

```

84     public boolean isEmpty() { 2 usages
85         for (int index = 0; index < table.length; index++) {
86             if (table[index] != null) {
87                 return false;
88             }
89         }
90         return true;
91     }
92
93 @      public Integer[] getIndexAndNum(String key) { 5 usages
94         int index;
95         if (key.isEmpty()) {
96             index = 0;
97         } else {
98             index = Character.getNumericValue(key.charAt(0)) % 100;
99         }
100         if (table[index] == null) {
101             return new Integer[] {index, 0};
102         } else {
103             return new Integer[] {index, table[index].size()};
104         }
105     }
106 }

```

Наконец прописываем класс HtRun, в котором прописываем метод main() и проверяем все прописанные ранее методы для класса HashTable.

```
HtRun.java x HashTable.java HashTableEntity.java
1 package Lab3.Exercise1;
2
3 import java.util.Arrays;
4
5 public class HtRun {
6     public static void main(String[] args) {
7         HashTable hash = new HashTable();
8
9         System.out.println(hash.isEmpty());
10        System.out.println(hash.size() + "\n");
11
12        hash.put(String.valueOf(12345), String.valueOf(13));
13        hash.put("omega", String.valueOf(8));
14        hash.put(String.valueOf(350), "MayBee");
15        hash.put("0 is for Orange", "You know?");
16        hash.put("0 is for Orange", "I know");
17        hash.put("", "well");
18
19        System.out.println(Arrays.toString(hash.get("omega")));
20        System.out.println(Arrays.toString(hash.getIndexAndNum(key: "omega")) + "\n");
21
22        System.out.println(Arrays.toString(hash.get(String.valueOf(350))));
23        System.out.println(Arrays.toString(hash.getIndexAndNum(String.valueOf(350)) + "\n");
24
25        System.out.println(Arrays.toString(hash.get("0 is for Orange")));
26        System.out.println(Arrays.toString(hash.getIndexAndNum(key: "0 is for Orange")) + "\n");
27
28        System.out.println(Arrays.toString(hash.get("")));
29        System.out.println(Arrays.toString(hash.getIndexAndNum(key: "") + "\n");
30
31        hash.remove(key: "omega");
32        hash.remove(key: "");
33
34        System.out.println(Arrays.toString(hash.get("omega")));
35        System.out.println(Arrays.toString(hash.getIndexAndNum(key: "omega")) + "\n");
36
37        System.out.println(hash.isEmpty());
38        System.out.println(hash.size() + "\n");
39    }
40 }
```

Задание 2. Вариант 5.

Создаём класс Product для хранения информации о товаре: его названии, цене и количестве. После этот класс будет использоваться в качестве значений объектов в хеш-таблице.

```
1 package Lab3.Exercise2;
2
3 public class Product { 3 usages
4     private String title; 3 usages
5     private double price; 3 usages
6     private int amount; 3 usages
7
8     public Product(String title, double price, int amount) { 2 usages
9         this.title = title;
10        this.price = price;
11        this.amount = amount;
12    }
13
14    public Product() { no usages
15        this( title: "None", price: 0, amount: 0);
16    }
17
18
19    public String getTitle() { 2 usages
20        return title;
21    }
```

```
23        public void setTitle(String title) { 1 usage
24            this.title = title;
25        }
26
27        public double getPrice() { 2 usages
28            return price;
29        }
30
31        public void setPrice(double price) { 1 usage
32            this.price = price;
33        }
34
35        public int getAmount() { 2 usages
36            return amount;
37        }
38
39        public void setAmount(int amount) { 1 usage
40            this.amount = amount;
41        }
42    }
```

В классе Store создаём объект HashMap и работаем с уже созданной хеш-таблицей. Прописываем методы для вставки, поиска и удаления продукта по штрихкоду, беря за основу уже имеющиеся методы класса HashMap. Далее прописываем метод main() для проверки работоспособности всех ранее прописанных методов.

```
Store.java x Product.java
1 package Lab3.Exercise2;
2
3 import java.util.HashMap;
4
5 public class Store {
6     public static HashMap<Integer, Product> hash; 13 usages
7
8     public static void main(String[] args) {
9         hash = new HashMap<>();
10
11         putProduct( barcode: 12345678, title: "Арбуз", price: 108.5, amount: 10);
12         putProduct( barcode: 87654321, title: "Дыня", price: 155, amount: 8);
13         putProduct( barcode: 87654321, title: "Дыня", price: 185, amount: 3);
14         putProduct( barcode: 56781234, title: "Перчик", price: 83, amount: 5);
15
16         int barcode = 12345678;
17         if (getProd(barcode) != null) {
18             System.out.println(getProd(barcode).getTitle()+": "+getProd(barcode).getPrice()+" RUB, "+getProd(barcode).getAmount()+" шт.");
19         } else {
20             System.out.println("Товара со штрихкодом " + barcode + " нет в каталоге.");
21         }
22         getProduct( barcode: 87654321);
23         getProduct( barcode: 56781234);
24         System.out.println("");
25
26         delProduct( barcode: 87654321);
27         delProduct( barcode: 87654321);
28         System.out.println("");
29
30         getProduct( barcode: 87654321);
31     }
32
33     public static void putProduct(int barcode, String title, double price, int amount) { 4 usages
34         if (hash.containsKey(barcode)) {
35             hash.get(barcode).setTitle(title);
36             hash.get(barcode).setPrice(price);
37             hash.get(barcode).setAmount(amount);
38         } else {
39             hash.put(barcode, new Product(title, price, amount));
40         }
41     }
42
43     public static Product getProd(int barcode) { 4 usages
44         return hash.getOrDefault(barcode, defaultValue: null);
45     }
46
47     public static void getProduct(int barcode) { 3 usages
48         if (hash.containsKey(barcode)) {
49             System.out.println(hash.get(barcode).getTitle()+": " + hash.get(barcode).getPrice()+" RUB, "+hash.get(barcode).getAmount()+" шт.");
50         } else {
51             System.out.println("Товара со штрихкодом " + barcode + " нет в каталоге.");
52         }
53     }
54
55     public static void delProduct(int barcode) { 2 usages
56         if (hash.containsKey(barcode)) {
57             hash.remove(barcode);
58             System.out.println("Успешно удалён товар со штрихкодом " + barcode + ".");
59         } else {
60             System.out.println("Товар со штрихкодом " + barcode + " уже отсутствовал.");
61         }
62     }
63 }
```


Вывод:

Мы изучили понятие хэш-таблиц, их возможности и особенности, а также написали программы на языке программирования Java с использованием полученных знаний.

GitHub - https://github.com/MiniLynx13/ITaP_Lab3