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»

Лабораторная работа №3

Работа с коллекциями

Выполнил: Студент группы

БВТ2303

Кунецкий Владислав

Москва

2024

Цели работы:

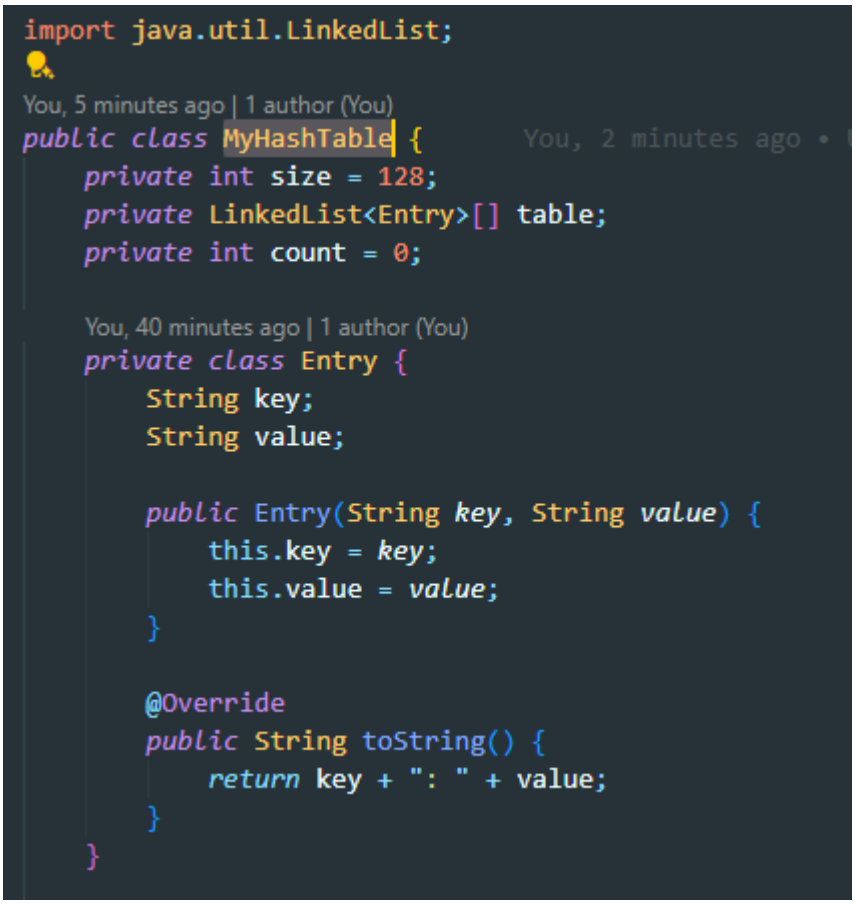
- Познакомиться с коллекциями, представленными в языке Java.

- Реализовать свой аналог HashTable

- Поработать со встроенной HashMap.

-Ход работы:

После изучения принципов хеш-таблиц я приступил к реализации своей. Для начала я создал основной класс MyHashTable. Я решил использовать массив связанных списков, в которые будут помещаться объекты, содержащие ключ-значение. Массив будет динамическим, изменение размера будет производиться в зависимости от количества элементов.



Скрин 1 – Поля класса

Поле size - начальный размер массива, массив списков это собственно и есть наша основная структура данных. Count будет отслеживать количество элементов внутри нашей мапы, а класс Enty описывает объекты, которые мы будет помещать внутрь.

Далее я написал свою хеш-функцию, преобразующую ключ в индекс массива.

```
private int hash(String key, int size) {
    int hash = 17;
    for (int i = 0; i < key.length(); i++)
        hash = hash * 113 + key.charAt(i);
    if (size == 0)
        setSize(size:128);

    return Math.abs(hash) % size;
}
```

Скрин 2 – хеш-функция

Для лучшего распределения использовал простые числа. Сдвиги, хоть и являются довольно быстрыми, но обеспечивают плохую распределяемость, поэтому от них я отказался.

Важным этапом стало написание метода, для изменения длинны массива. В нем мы производим копирование старого массива и создаем новый, заменяющий старый. После в цикле копируем значения, пересчитывая их индекс.

```
private void resize(int newSize) {
    LinkedList<Entry>[] oldTable = table;
    table = (LinkedList<Entry>[]) new LinkedList[newSize];
    for (int i = 0; i < newSize; i++)
        table[i] = new LinkedList<Entry>();

    for (int   lab3.MyHashTable.Entry
        for (Entry entry : oldTable[i]) {
            int index = hash(entry.key, newSize);
            table[index].addLast(entry);
        }
    }
    setSize(newSize);
}
```

Скрин 3 – Метод переопределения длинны массива

После реализовал основную логику.

```
public String get(String key) {
    if (key == null)
        return null;

    int index = hash(key, size);
    for (Entry entry : table[index]) {
        if (entry.key.equals(key))
            return entry.value;
    }
    return null;
}
  You, 1 second ago • Uncommitted changes
public int getCount() {
    return count;
}

public void put(String key, String value) {
    if (key == null || key.equals(anObject:"") )
        throw new NullPointerException();

    count++;
    if (count > size)
        resize(size * 2);

    int index = hash(key, size);
    table[index].addLast(new Entry(key, value));
}
```

Скрин 4 – Вставка и получение значий.

При вставке мы генерируем индекс элемента с помощью хеш-функции и производим вставку объекта в конце связанного списка по индексу. В случае, когда количество элементов превышает размер, увеличиваем его.

В гетер мы получаем также хеш, а далее проходимся циклом по элементам списка и ищем нужный ключ.

```
public String remove(String key) {
    if (key == null)
        return null;

    int index = hash(key, size);
    for (Entry entry : table[index]) {
        if (entry.key.equals(key)) {
            table[index].remove(entry);
            count--;
            return entry.value;
        }
    }
    count--;
    if (count < size / 4)
        resize(size / 2);
    return null;
}

public void clear() {
    for (int i = 0; i < size; i++) {
        table[i].clear();
    }
    resize(newSize:128);
    count = 0;
}

public boolean contains(String key) {
    if (key == null)
        return false;

    int index = hash(key, size);
    for (Entry entry : table[index]) {
        if (entry.key.equals(key))
            return true;
    }
    return false;
}
```

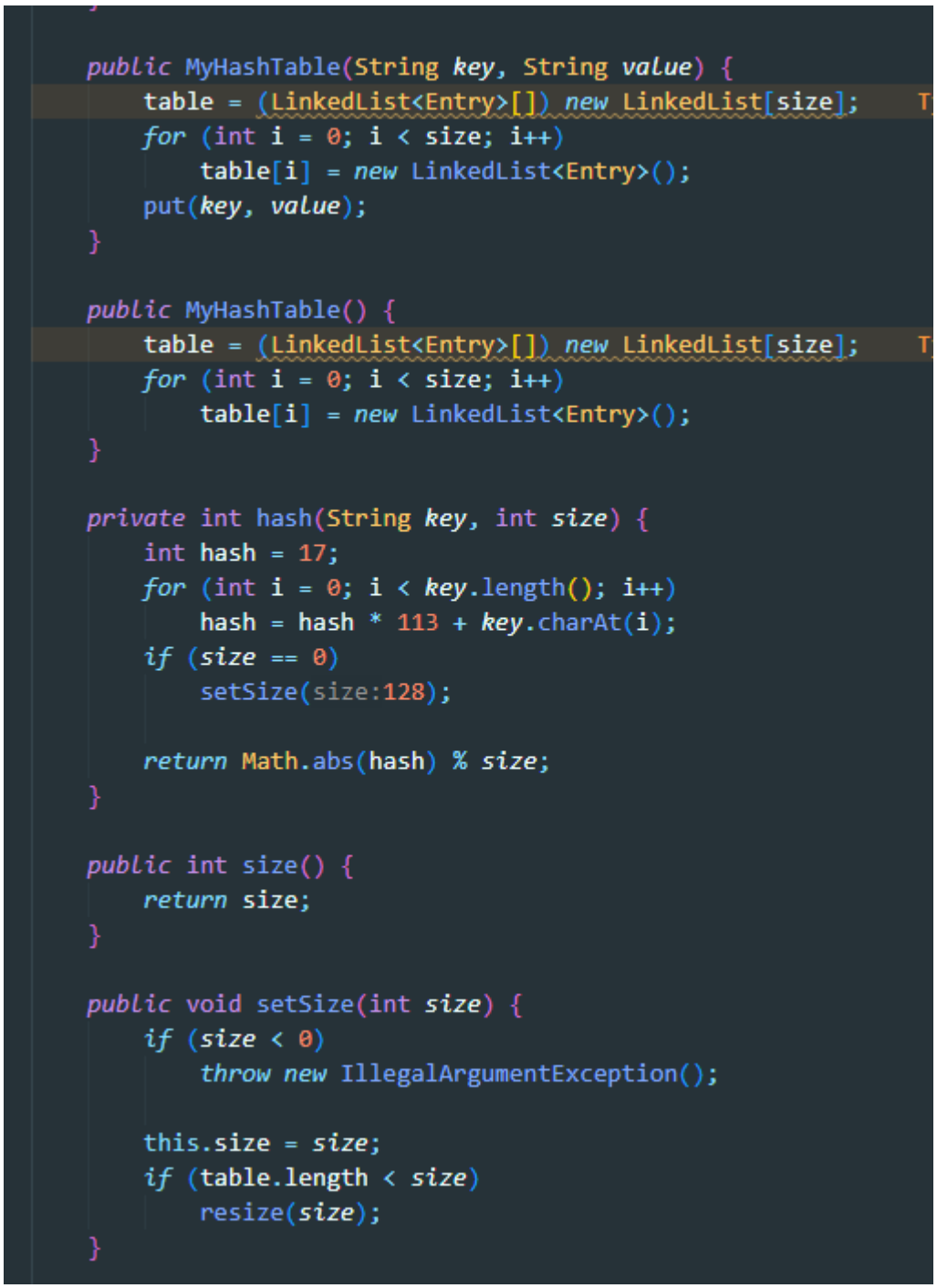
Скрин 5 – Удаление и проверка наличия.

Далее я реализовал удаление, очистку и проверку на наличие(скрин 5). Логика везде одинаковая

Вспомогательные методы

```
42
43      public void showAll() {
44          for (int i = 0; i < size; i++) {
45              System.out.print(i + ": ");
46              for (Entry entry : table[i]) {
47                  System.out.print(entry.toString());
48                  System.out.print(s:", ");
49              }
50              System.out.println();
51          }
52      }
53
54      public void show() {
55          for (int i = 0; i < size; i++) {
56              if (table[i].size() > 0) {
57                  System.out.print(i + ": ");
58                  for (Entry entry : table[i]) {
59                      System.out.print(entry.toString());
60                      System.out.print(s:", ");
61                  }
62                  System.out.println();
63              }
64          }
65      }
66  }
```

Скрин 6 – показ содержимого таблицы



```
    public MyHashTable(String key, String value) {
        table = (LinkedList<Entry>[]) new LinkedList[size];    T
        for (int i = 0; i < size; i++)
            table[i] = new LinkedList<Entry>();
        put(key, value);
    }

    public MyHashTable() {
        table = (LinkedList<Entry>[]) new LinkedList[size];    T
        for (int i = 0; i < size; i++)
            table[i] = new LinkedList<Entry>();
    }

    private int hash(String key, int size) {
        int hash = 17;
        for (int i = 0; i < key.length(); i++)
            hash = hash * 113 + key.charAt(i);
        if (size == 0)
            setSize(size:128);

        return Math.abs(hash) % size;
    }

    public int size() {
        return size;
    }

    public void setSize(int size) {
        if (size < 0)
            throw new IllegalArgumentException();

        this.size = size;
        if (table.length < size)
            resize(size);
    }
```

Скрин 7 - Конструкторы и методы для изменения/получения размера

Во второй части работы мне предстояло написать хэш-таблицу для хранения контактов в телефонной книге.

Я наследовался от класса HashMap, определил объект Contact и определил основные методы.

```java
package lab3;

import java.util.HashMap;

public class PhoneBook extends HashMap<String, Object> {
    public class Contact {
        String name;
        String surname;
        String email;

        public Contact(String name, String surname, String email) {
            this.name = name;
            this.surname = surname;
            this.email = email;
        }

        @Override
        public String toString() {
            return name + " " + surname + " " + email;
        }
    }

    public PhoneBook() {
        super();
    }

    public PhoneBook(String mobile, Contact contact) {
        put(mobile, contact);
    }

    public void addContact(String mobile, Contact contact) {
        put(mobile, contact);
    }

    public Contact getContact(String mobile) {
        return (Contact) get(mobile);
    }

    public void removeContact(String mobile) {
        remove(mobile);
    }
}
```

Скрин 8 – Вторая часть работы