

Dp2530

Lab # 5

Deadline

Please see the dropbox.

Names of the students

- 1) Ryan Horwood

Lab Title:

Enhanced Word Frequency Counter using Hash Table

Points to be noted.

- 1) This lab is partially created by chatGPT.
- 2) Please don't submit any code or other files; copy and paste your code and output at the end of this file.
- 3) Extensions in the deadline will not be given unless permitted.

Objective:

In this programming lab, students must modify and improve an existing Java program that uses a simple hash function and hash table to store words. The goal is to build a word frequency counter that reads a text file and counts the frequency of each unique word in the file. In addition, the lab tests the student's understanding and implementation of hash tables, hash functions, collision handling techniques, and file handling in Java.

Problem Description:

You have been given a Java program (HashFunctionExample) that uses a simple hash function to store words in a hash table. Your task is to modify and improve the program to create a word frequency counter that reads a text file and computes the frequency of each unique word in the file using a hash table. You should also implement a more effective hash function and use separate chaining to handle collisions.

Requirements:

Modify the existing Java program (HashFunctionExample) to read a text file (input.txt) containing words separated by spaces and newline characters.

Implement a more effective hash function to map the words to their respective indices in the hash table.

Use separate chaining (linked lists) to handle collisions in the hash table.

Compute the frequency of each unique word in the file and store it in the hash table.

Display the words and their frequencies in the hash table in descending order of frequency.

The program should be case-insensitive, i.e., treat "Word" and "word" as the same word.

Input:


A text file (input.txt) containing words separated by spaces and newline characters.

Output:


A list of unique words and their frequencies in descending order of frequency.

Sample Input:

```
apple apple orange
banana orange
apple
```

 Copy code**Sample Output:**

```
apple: 3
orange: 2
banana: 1
```

 Copy code

Notes:

- Ensure that your program can handle large text files efficiently.
- Test your program with text files containing various words and word frequencies.
- As a bonus, you can implement resizing of the hash table to improve its efficiency when the number of elements increases or decreases.
- The lab should be accessible to medium-difficult, depending on the student's familiarity with hash tables, hash functions, separate chaining, and file handling in Java. This lab will help students strengthen their understanding of hash tables and collision handling techniques while also giving them experience n working with text files and modifying existing code.

Code design

This is the code design you should use to implement your code. Please implement the non-implemented code. You can add more methods if you wish and make modifications to achieve the goal

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.LinkedList;
```

```

public class WordFrequencyCounter {
    public static void main(String[] args) {
        String fileName = "input.txt";
        CustomHashTable hashTable = new CustomHashTable(10);
        try {
            BufferedReader reader = new BufferedReader(new
FileReader(fileName));
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\s+");
                for (String word : words) {
                    word = word.toLowerCase();
                    hashTable.incrementFrequency(word);
                }
            }
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Print the contents of the hash table
        System.out.println("Word Frequencies:");
        hashTable.printFrequenciesDescending();
    }
}

class CustomHashTable {
    private LinkedList<WordFrequency>[] table;

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

    public void incrementFrequency(String word) {
        // TODO: Implement this method
    }

    public void printFrequenciesDescending() {
        // TODO: Implement this method
    }
}

```

```
private int hash(String key) {  
    // TODO: Implement a more effective hash function  
    return key.length() % table.length;  
}  
}  
  
class WordFrequency {  
    String word;  
    int frequency;  
  
    public WordFrequency(String word) {  
        this.word = word;  
        this.frequency = 1;  
    }  
  
    public void incrementFrequency() {  
        frequency++;  
    }  
  
    public String toString() {  
        return word + ": " + frequency;  
    }  
}
```

Contents of the input.txt file

These are the contents of the input.txt file. Create the text file and put it inside the same folder as your source file. Copy these contents.

```
apple apple orange  
banana orange  
apple  
strawberry mango  
kiwi grape  
orange kiwi  
banana grape  
mango grape  
kiwi strawberry
```

Your solution

Please copy your code here. Don't Submit the code file separately.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.LinkedList;
import java.util.Collections;
import java.util.Comparator;
import java.util.*;

class WordFrequencyCounter {
    public static void main(String[] args) {
        String fileName = "input.txt";
        CustomHashTable hashTable = new CustomHashTable(10);
        try {
            BufferedReader reader = new BufferedReader(new
                FileReader(fileName));
            String line;
            while ((line = reader.readLine()) != null) {
                String[] words = line.split("\\s+");

                for (String word : words) {
                    word = word.toLowerCase();
                    hashTable.incrementFrequency(word);
                }
            }
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        // Print the contents of the hash table
        System.out.println("Word Frequencies:");
        hashTable.printFrequenciesDescending();
    }
}
```

```

}

class CustomHashTable {
    private LinkedList<WordFrequency>[] table;

    public CustomHashTable(int size) {
        table = new LinkedList[size];

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

    public void incrementFrequency(String word) {
        // TODO: Implement this method
        int index = hash(word);
        LinkedList<WordFrequency> list = table[index];
        for (WordFrequency wf : list) {
            if (wf.word.equals(word)) {
                wf.incrementFrequency();
                return;
            }
        }
        list.add(new WordFrequency(word));
    }

    public void printFrequenciesDescending() {
        // TODO: Implement this method
        LinkedList<WordFrequency> frequencies = new LinkedList<>();
        for (LinkedList<WordFrequency> list : table) {
            for (WordFrequency wf : list) {
                frequencies.add(wf);
            }
        }

        Collections.sort(frequencies, new Comparator<WordFrequency>() {
            public int compare(WordFrequency wf1, WordFrequency wf2) {

```

```

        return Integer.compare(wf2.frequency, wf1.frequency);
    }

    });

    for (WordFrequency wf : frequencies) {
        System.out.println(wf);
    }
}

private int hash(String key) {
    // TODO: Implement a more effective hash function

    int hash = 0;
    for (int i = 0; i < key.length(); i++) {
        hash = (hash * 31 + key.charAt(i)) % table.length;
    }
    return hash;
}

}

class WordFrequency {
    String word;
    int frequency;

    public WordFrequency(String word) {
        this.word = word;
        this.frequency = 1;
    }

    public void incrementFrequency() {
        frequency++;
    }

    public String toString() {
        return word + ": " + frequency;
    }
}

```


Your output

Copy your output here.

Word Frequencies:

apple: 3

orange: 3

kiwi: 3

grape: 3

mango: 2

banana: 2

strawberry: 2