Lab Report

EXERCISE 2: HISTOGRAM

	Name	Titel des Kurses	Datum
Juri Wiechmann Joel Preik	571085 571627	Prof. Dr. Weber-Wulff Info 2 Group 2	24.10.2019

Index

Introduction

This week’s exercise was about reading and writing from and into a file. Which was introduced to us in the lectures. The first task of our prelab was the hardest to understand because the example was in Ada, which most people in our course didn’t know about. We have a total of 5 versions for this lab so we will concentrate to mention the first one then move fast on till our final one to answer each assignment.

PRE-LAB

1. **In some programming languages, such as Ada, you can define an array of characters with any discrete type as the index:**

```
someArray : ARRAY ['A' .. 'Z'] of INTEGER;
```

You can then access the array, for example, using a value of character type: someArray['T']. Java does not have this feature. How would you go about making an array in Java for representing counters for the letters 'A' to 'Z'?

The solution was not an array. The task is talking about arrays so the first idea was to create two arrays even if it’s against the task rule`s to only use one. The first one being in Integer and the other one in Character with the same size, so that we can call a char and can relate to the integer counter. Later we will see how we can solve this more easily.

2. **Normalization of Strings means transforming all Strings to either uppercase or lowercase before comparing them. Write a method that takes a character as a parameter and returns a normalized version of the character without using the methods available in the Java String class.**

You take a look at the Unicode and if you pay attention you will see, that the decimal difference between the small letters and the big once is exactly 32. So, to change a small letter to a big one you just need to subtract 32 from the small letter char to get the Big one. The Task does not allow us to use the String method that can do that for us behind the scene. Because we are lazy students we used the Character method:

```
Character.toUpperCase(ListNotNorm[i]);
```

3. **What is a "carriage return"? Where does the name come from?**

In Windows, a new line is denoted using “\r\n”, sometimes called a **Carriage Return** and Line Feed, or CRLF. Adding a new line in **Java** is as simple as including “\n” or “\r” or “\r\n” at the end of our string. The name is coming from the Typewriter when you use a mechanism to write into a new line.

Assignments

1. **How do you go about reading in characters from a file? Write and test a method that returns the next character in a file. Note that you have to do something with the carriage returns - such as ignoring them - and that you have to decide what to do when there are no characters to be returned.**

We already had this one in the pre-lab so we tried to think about a way to optimize our reader method. Our first reader method returns us a Character-Array with all Characters even with the carriage returns. In our first Semester we got introduced to ArrayLists. ArrayList automatically expand their size and that makes them perfect to safe all Characters from a File because we don't know how many we will get.

```
public static char[] reader (String file_name) throws IOException {

	FileReader fr = new FileReader(file_name);
	BufferedReader br = new BufferedReader(fr);

	System.out.println("Copied from the file to the console:");
	System.out.println("File = " + file_name);

	ArrayList<Character> File = new ArrayList<Character>();

	while(br.ready()) {
		File.add((char) br.read());
	}

	char[] Array = new char[File.size()];

		for(int i = 0; i < Array.length; i++) {
			Array[i] = File.get(i);
		}
	return Array;
}
```

First version.

If there are no Characters to read, we simply get an empty Array back.

Later we get some issues because of the carriage return. If we want to place them into our Hashmap, he will not be able to find it and we will get a NullPointerException. That's why we changed a bit in our reader-method. Instead of reading a whole file directly into a char-array line by line we now create for each line an array and then add each cell to our ArrayList. With that we get rid of the carriage returns.

```
public static char[] reader (String file_name) throws IOException {

        FileReader fr = new FileReader(file_name);
        BufferedReader br = new BufferedReader(fr);

        System.out.println("Copied from the file to the console:");
        System.out.println("File = " + file_name);

        ArrayList<Character> File = new ArrayList<Character>();
        String line;
        char[] lineChar;

        while(br.ready()) {
                line = br.readLine();
                lineChar = line.toCharArray();

                for(int i = 0; i< lineChar.length;i++) {
                        File.add(lineChar[i]);
                }
        }
        char[] Array = new char[File.size()];

                for(int i = 0; i < Array.length; i++) {
                        Array[i] = File.get(i);
                }
        return Array;
}
```

Final version.

2. **How do you write a String to a file? How do you write an Integer to a file? An int? How do you create a file, anyway?**

For this we wrote our writer-method. We use the FileWriter, it works but overwrote everything. We wanted to add some text to the file, instead it just overwrote our contend. So, we asked and got a hint to take a look at the API of the FileWriter-Constructors. We found out that we just needed to set a true as additional parameter. Our first writer-Method is our final as well because we like to have more than one or two ways to write into a file so we gave us the opportunity to.

```
        /*first param: Name of File u want 2 write in
        *sec param: Text u want 2 write in the file
        *third param:
        *x = 1 Overwriting everything,
        *x = 2 Add Text simple behind last Char,
        *x = 3 Add Text at the bottom in a new Line,
        *x = 4 Add Text at the bottom behind the last char
        */
        public static void writer (String file_name, String Text, int x) throws IO-
Exception {

                switch(x) {
                case 1:
                        FileWriter fw = new FileWriter(file_name);
                        fw.write(Text);
                        fw.close();
                        break;
                case 2:
                        FileWriter fw1 = new FileWriter(file_name, true);
                        fw1.write(Text);
                        fw1.close();
                        break;
                case 3:
                        FileWriter fw2 = new FileWriter(file_name, true);
                        BufferedWriter bw = new BufferedWriter(fw2);
                        PrintWriter pw = new PrintWriter(bw);
                        pw.println(Text);
                        pw.close();
                        break;
                case 4:
                        FileWriter fw3 = new FileWriter(file_name, true);
                        BufferedWriter bw1 = new BufferedWriter(fw3);
                        PrintWriter pw1 = new PrintWriter(bw1);
                        pw1.print(Text);
                        pw1.close();
                        break;
                }

        }
```

First and final Version of this method.

3. **Now the fun begins! Write a Java application to read in a file character by character, counting the frequencies with which each character occurs. When there are no more characters, create a file frequency.txt and output the frequencies for each character.**

First we create a method to create a File:

```
public static void CreateFile(String file_name) throws IOException {
        File file = new File(file_name);

        file.createNewFile();
}
```

First and final Version of this method.

We tried at first to create a method to count the Letters. To do so we need some kind of storage with a Char as index and an Integer that we can count up. We really dismissed the idea to use two Arrays. Bartholomäus told me one lecture about maps. I have heard about them but had no idea what they are so we watched the API and some Videos how to use them(S2). We understood fast and implemented them.

```
static HashMap<Character, Integer> Histo = new HashMap<>();
```

Now we need to fill up this map with the letters as Index that we can count them up in the Integercell. The “65” and “91” represent A-Z in decimalform.

```
public static void FillMap() {
        for (int i = 65; i<91;i++) {
                Histo.put((char) i, 0);
        }
}
```

First version

```
public static void FillMap() {
        for (int i = min; i<max;i++) {
                Histo.put((char) i, 0);
        }
}
```

Final version

Later we found out, that we need those numbers really often so we put them to fields. This way we can easier change them later as well if we want to count more chars.

Now to count the small letters too we need to change them to big once. To not get in trouble later with other chars by switching their decimal number we used the Character.toUpperCase Method.

```
public static char[] Arraynorm(char[] ListNotNorm) {

        char[] notnorm = new char[ListNotNorm.length];

        for(int i = 0; i < notnorm.length; i++) {
                notnorm[i] = Character.toUpperCase(ListNotNorm[i]);
        }
        return notnorm;
}
```

First and final Version of this method.

With this out of our mind now we can put them into our Hashmap. We not really putting them into our Hashmap because they are already as Index in there. We now go through the whole normalized array that we got now and count them up. If we find the char that we looking at the value goes up by one.

```
public static void reduce(char[] BigCharArray) {

        for(int i = 0; i<BigCharArray.length ; i++) {
                if(Histo.containsKey(BigCharArray[i])) {
                        int value = Histo.get(BigCharArray[i]);
                        Histo.put((BigCharArray[i]), value+1);
                }
        }
}
```

First Version.

Later for the 7th task we had to change a bit. Because our outcome would now print all chars its getting out of control if we have a lots more. So we searched for a solution and found pretty fast one (S3). We put it behind the for-loop

```
Histo.values().removeAll(Collections.singleton(0));
```

All we had to do now was to write a method that print line by line our histogram with the right number of value. So we created a String with the staring letter first and then we go through every value with a loop and add every time a "*".

```
public static void printHisto() throws IOException {
        String line;

        for(int i = 65; i < 91;i++) {
                line = String.valueOf((char) i);
                for(int j = 0; j < Histo.get((char) i);j++) {
                        line = line + "*";
                }
                writer(Histogramm,line,3);
        }
}
```

Here too we changed the numbers to our fields and in our final version we needed to find a new way to iterate through the Hashmap. Later I will explain why we needed to.

```
public static void printHistoHori() throws IOException {
        String line;

        for (Entry<Character, Integer> entry : Histo.entrySet()) {
            int power = entry.getValue();
            line = String.valueOf(entry.getKey());

            for(int j = 0; j < power;j++) {
                     line = line + "*";
              }
            writer(Historows,line,3);
        }
}
```

4. **Output a histogram of the character frequencies. One simple kind of histogram has horizontal lines proportional to the magnitude of the number it represents. For example:**

```
A : **********
B : *****
C : ******
```

Well, everything we were supposed to do in the fourth task is in 3. We slightly misunderstood the third Task and shot over the target.

5. **What is the complexity of your algorithm?**

To answer this question we will take a look at our main method from the current state and at the final state. Before we take a look we will basically explain how our main works. We got 3 Files. One is our Source("Quelle"), from here we gain the Text that we want to analysed. We copy it into a new File("ZuZählendeWörter") just to count them not from the Source. Just to mention it, I don't know how complex some behind the scene methods are so I just will count my once.Current state:

```
	public static void main(String[] args) throws IOException {
		Histogramm ="(some pathing stuff i dont want to show /frequency.txt";
		ZuZählendeWörter ="(pathing) /Gedicht.txt";
		Quelle = "(pathing)/Source.txt";


		FillMap();


		CreateFile(ZuZählendeWörter);

		Addtext = new String(reader(Quelle));
		writer(ZuZählendeWörter,Addtext, 1);


		reduce(Arraynorm(reader(ZuZählendeWörter)));

		printHisto();
	}
```

So first we fill the Map. Only one for-loop so O(N).

Then we Create a File. Some behind the scene stuff that I don't know.

Then we call "reader" with 2 loops in int so O(2N).

The writer-method have no loops so O(c) I guess.

Now we need to clear that from behind, so we start with the reader. We already know its O(N).

Now the Arraynorm-Method, again just a simple for loop so O(N).

Reduce is also O(N).

But finally we got our first method "printHisto()" which contains a for loop in a for loop. First we thought its O(N^2) but if you think about it we just go through all Characters here and every additional Character don't increase it it by the power of 2. Its linear. So we would guess its O(N) again.

All in all we got **O(7N+c).** rounded to O(N)

In the same way we went through the final version:

```
public static void main(String[] args) throws IOException {
	Historows = "(pathing) /frequency.txt";
	ZuZählendeWörter = "(pathing) /Gedicht.txt";
	Quelle = "(pathing) /Source.txt";


	FillMap();


	CreateFile(ZuZählendeWörter);

	Addtext = new String(reader(Quelle));
	writer(ZuZählendeWörter,Addtext, 1);


	reduce(Arraynorm(reader(ZuZählendeWörter)));

	//printHistoHori();
	printHistoVerti();
}
```

The only thing that have been changed is our printHistoVerti-method. Our normal PrintHisto-method changed to printHistoHori. So we are at O(6N+c). printHistoVert contains 2 simple loops and 2 for in for loops. But it's kind of the loop in loop before just with a little difference, now we are creating a lot " " (spaces) with an average double the amount an chars. So finally we got **O(2*2*8N+c)** and if we round it we are back to O(N). Now in 6 we will see the code for this so we won't show them here.

6. **Make your histogram application display the histogram with vertical lines and input the file name as a parameter.**

This wasn't that hard at all if you solved the fourth task properly. The main issue was to think about how to represent it in code that you can print it. One Solution is to create a two dimensional char array which contains every position of x and y that we need to represent A-Z and the maximum value. So the width is A-Z and the high is the maximum value + one for the letter at the bottom.

```
public static void printHistoVerti() throws IOException {

        int maxLetter = 0;

        for (int i = 0; i<Histo.size();i++) {
              if(maxLetter < Histo.get((char)(i+min))) {
                    maxLetter = Histo.get((char)(i+min));
              }
        }
        // Histo.size == Columns; maxLetter == Rows
        char [][] Histogramm = new char[Histo.size()][maxLetter+1];

        for(int i = min; i < max;i++) {
              Histogramm [i-min][0] = (char) i;
        }
        for(int i = 1; i <= maxLetter;i++) {
              for(int j = min; j < max;j++) {

                    if(i<= Histo.get((char)j)) {
                          Histogramm[j-min][i] = '*';
                    }
                    else {
                          Histogramm[j-min][i] = ' ';
                    }
              }
        }
        writer(Historows,"",1);
        for(int i = maxLetter; i >= 0;i--) {
              writer(Historows,"",3);
              for(int j = 0; j < max-min;j++) {
                    writer(Historows,"  ",4);
                    writer(Historows,String.valueOf(Histogramm[j][i]),4);

              }
        }
}
```

First we want to find out the maximum value to know our height which we search in the followed loop. Then we creating our 2D-array. Then we fill up the letters. Our first row are the letters and our last row is at the maximum value. So, we filled up fast the first row. Now it is getting a bit more complicated. We iterate in our outer loop every row. By starting at the second row we don't overwrite our letters. Now we iterate the columns. We compare the value of each letter that we got with the amount of rows where we at to see if we need to put "*" or " " (space) in. Here I was talking about doubling the amount of Characters.

We call writer to reset the file. Now we can start to Print this Array just by go through it. But of course we start with the row at the top and that's why we need to count down the rows. Everything we enter a new row we write nothing in a new line just to force the following writer-calls to write here. We create some space each column that it looks more pretty and then just print the right value.

7. **Look up Unicode - there are a lot more characters here! How many more? Assume you have a text with an unknown number of different (and bizarre) Unicode characters. How can you make a histogram for such a text?**

So our first idea is just to increase the min and max of our chars:

```
static int min = 32;
static int max = 591+1;
```

Let's say…it works not well but it did. But we got 2 big problems that wasn't user friendly. First we got now nearly 600 Characters in our File that all are listed even if there is a value of for them. And the second one is, if we get more Characters to read our program need like 20 minutes for 15.000 Characters.

To solve our first problem we had to delete every Index and its value when there value is 0. We searched a bit in the www and found on stackoverflow a solution. The funny thing is, not the recommended work but another one with less up votes (s3).

```
Histo.values().removeAll(Collections.singleton(0));
```

Now we caused another huge problem. Our literation for our Histo wont work now anymore because they depends on the Character-value itself (A=65) and don't allow missing once. So we needed to find a way to iterate a Hashmap by a forEach-loop. We searched again and found one (S4):

```
for (Entry<Character, Integer> entry : Histo.entrySet()) {

}
```

The good thing, it works, the bad thing, even till today we don't know why. We only know that we need to use Entry.something to get things from our Hashmap "Histo". So we did and changed every for loop that iterate our Hashmap. By filling up the Arrays you still need some kind of index that starts at 0 and its counting up every time so we creating one outside the loop and reset him back to 0 after the loop to use him more than once.

Now the second problem was an easy fix but hard to find. We found out, that the writing method is taking a lot of time, especially if you call it for every Character in our 2D-array. So with a bit moving and adding some "carriage returns" we managed to change our printing loop, that we only need to call it once. So we changed

From this:

```
writer(Historows,"",1);
      for(int i = maxLetter; i >= 0;i--) {

            writer(Historows,"",3);

            for (Entry<Character, Integer> entry : Histo.entrySet()) {
                  writer(Historows," ",4);
                  writer(Historows,String.valueOf(Histogramm[index][i]),4);

                  index++;
                  }
            index=0;
            }
```

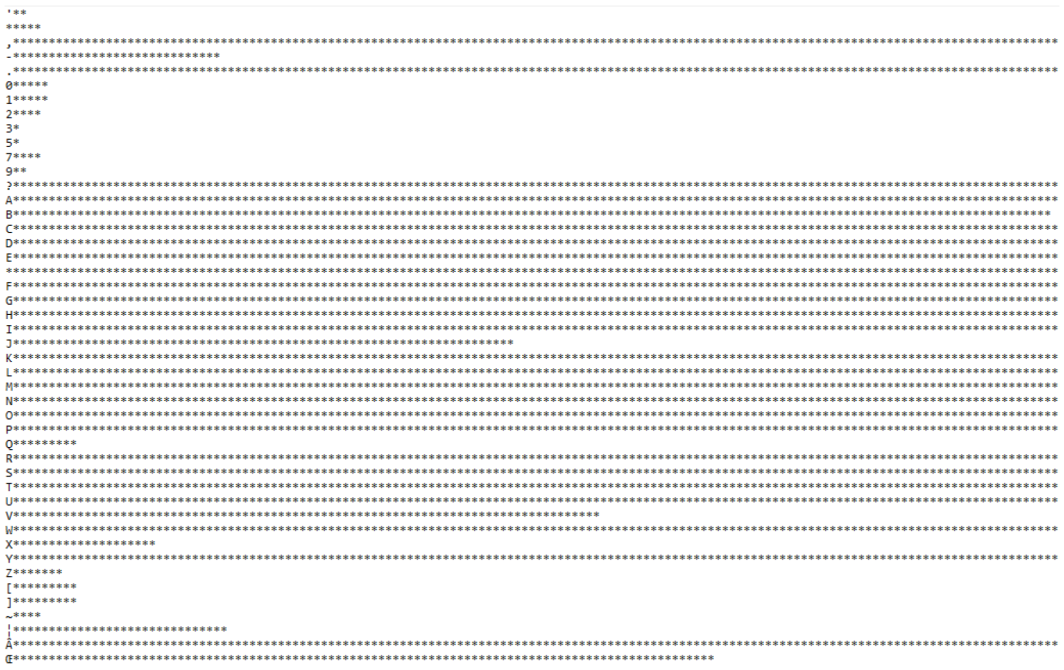
To this:

```
      writer(Historows,"",1);
      String Text = "";
      for(int i = maxLetter; i >= 0;i--) {

            Text = Text + "\r\n";
            for (Entry<Character, Integer> entry : Histo.entrySet()) {
                  Text = Text + String.valueOf(Histogramm[index][i]) + " ";
                  index++;
            }
            index=0;
      }
      writer(Historows,Text,3);
```

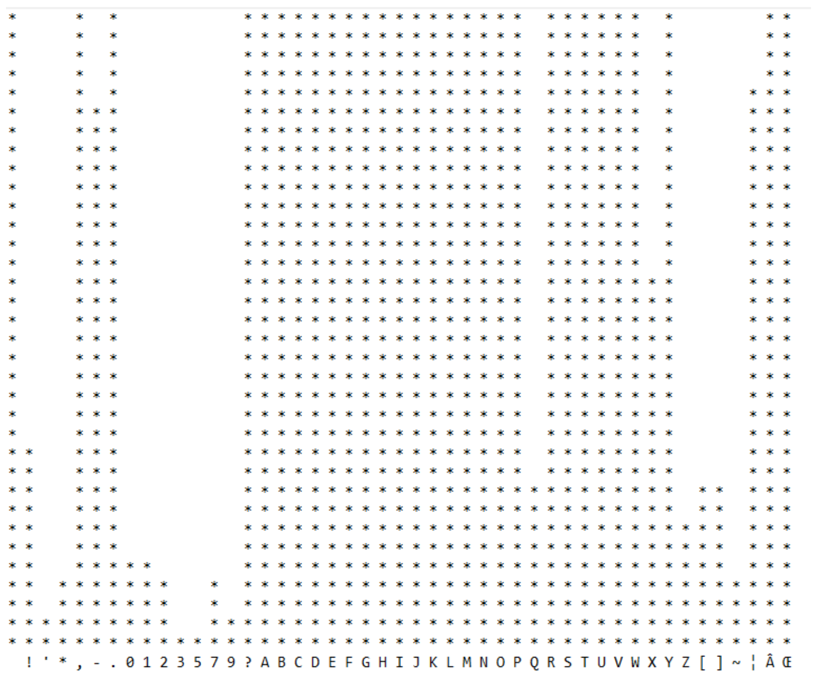
The final outcome looks like this:

Horizontal:



```
'**
*****
,**************************************************************************************************************************************************************************
-*****************************
.**************************************************************************************************************************************************************************
0*****
1*****
2****
3*
5*
7****
9**
?**************************************************************************************************************************************************************************
A**************************************************************************************************************************************************************************
B*************************************************************************************************************************************************************************
C**************************************************************************************************************************************************************************
D**************************************************************************************************************************************************************************
E**************************************************************************************************************************************************************************
***************************************************************************************************************************************************************************
F**************************************************************************************************************************************************************************
G**************************************************************************************************************************************************************************
H**************************************************************************************************************************************************************************
I**************************************************************************************************************************************************************************
J*********************************************************************************
K**************************************************************************************************************************************************************************
L**************************************************************************************************************************************************************************
M**************************************************************************************************************************************************************************
N**************************************************************************************************************************************************************************
O**************************************************************************************************************************************************************************
P**************************************************************************************************************************************************************************
Q*********
R**************************************************************************************************************************************************************************
S**************************************************************************************************************************************************************************
T**************************************************************************************************************************************************************************
U**************************************************************************************************************************************************************************
V***********************************************************************************************
W**************************************************************************************************************************************************************************
X********************
Y**************************************************************************************************************************************************************************
Z*******
[*********
]*********
~****
¦*******************************
Â**************************************************************************************************************************************************************************
Œ******************************************************************************************************************
```

Vertical:



```
*       *   *               * * * * * * * * * * * * * * * * *   * * * * * *   *           * *
*       *   *               * * * * * * * * * * * * * * * * *   * * * * * *   *           * *
*       *   *               * * * * * * * * * * * * * * * * *   * * * * * *   *           * *
*       *   *               * * * * * * * * * * * * * * * * *   * * * * * *   *           * *
*       *   *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * *   *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
*       * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
* *     * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
* *     * * *               * * * * * * * * * * * * * * * * *   * * * * * * * *         * * *
* *     * * *               * * * * * * * * * * * * * * * * * * * * * * * * * *   * *   * * *
* *     * * *               * * * * * * * * * * * * * * * * * * * * * * * * * *   * *   * * *
* *     * * *               * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   * * *
* *     * * *               * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   * * *
* *     * * * * *           * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   * * *
* *   * * * * * * *     *   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *   * * * * * * *     *   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * *     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  ! ' * , - . 0 1 2 3 5 7 9 ? A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ ] ~ ¦ Â Œ
```

Reflections

JURI

In this weeks`s lab I learned how to handle Files with Java. I really liked the Assignments this week because I was be able to use nearly everything that I know about java. I refreshed my knowledge about ArrayLists and get to know Hashmaps. It was interesting how you can manage Hashmaps. It was a lot of fun to code and to learn how to create Files. I also got more confident with Eclipse and some Hotkeys. Strg+shift+O import automatically every java component that you need.

JOEL

In this week's lab I learned a lot about using methods to write into and read from a file. Also, I learned a bit about Hotkeys in Eclipse which was really helpful for many tasks. One being the Shortcut Menu: Strg+Shift+L and the other one being Java auto import: Strg+Shift+O. I'm getting more used to Eclipse and I now actually prefer it over blueJ. Also, in lab we managed to do most of the task. Juri was really pleasant to work with in and out of the lab.

Sources

(S1):
https://en.wikipedia.org/wiki/Carriage_return#targetText=A%20carriage%20return%2C%20sometimes%20known,of%20a%20line%20of%20text.

(S2): https://www.youtube.com/watch?v=j442WG8YzM4

(S3): https://stackoverflow.com/questions/2594059/removing-all-items-of-a-given-value-from-a-hashmap

(S4): https://www.geeksforgeeks.org/traverse-through-a-hashmap-in-java/

Appendix: The final Version(all would be like more then 1200 rows of code)

Java Code:

```
package Übung2_4;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map.Entry;

public class Histofgram {

	static String Historows;
	static String ZuZählendeWörter;
	static String Quelle;

	static String Addtext;
	static HashMap<Character, Integer> Histo = new HashMap<>();
	static int min = 32;
	static int max = 591+1;

	public static void main(String[] args) throws IOException {
		Historows = "(pathing)/frequency.txt";
		ZuZählendeWörter = ""(pathing)//Gedicht.txt";
		Quelle = "(pathing)/Source.txt";


		FillMap();


		CreateFile(ZuZählendeWörter);

		Addtext = new String(reader(Quelle));
		writer(ZuZählendeWörter,Addtext, 1);
		reduce(Arraynorm(reader(ZuZählendeWörter)));

		//printHistoHori();
		printHistoVerti();
	}
	public static char[] reader (String file_name) throws IOException {

		FileReader fr = new FileReader(file_name);
		BufferedReader br = new BufferedReader(fr);

		System.out.println("Copied from the file to the console:");
		System.out.println("File = " + file_name);

		ArrayList<Character> File = new ArrayList<Character>();
		String line;
		char[] lineChar;

		while(br.ready()) {
			line = br.readLine();
			lineChar = line.toCharArray();

			for(int i = 0; i< lineChar.length;i++) {
				File.add(lineChar[i]);
			}
		}
		char[] Array = new char[File.size()];

			for(int i = 0; i < Array.length; i++) {
				Array[i] = File.get(i);
			}
		return Array;
	}

	/*first param: Name of File u want 2 write in
	*sec param: Test u want 2 write in the file
	*third param:
	*x = 1 Overwriting everything,
	*x = 2 Add Text simple behind last Char,
	*x = 3 Add Text at the bottom in a new Line,
	*x = 4 Add Text at the bottom behind the last char
	*/
	public static void writer (String file_name, String Text, int x) throws IOException {

		switch(x) {
		case 1:
			FileWriter fw = new FileWriter(file_name);
			fw.write(Text);
			fw.close();

			break;
		case 2:
			FileWriter fw1 = new FileWriter(file_name, true);
			fw1.write(Text);
			fw1.close();

			break;
		case 3:
			FileWriter fw2 = new FileWriter(file_name, true);
			BufferedWriter bw = new BufferedWriter(fw2);
			PrintWriter pw = new PrintWriter(bw);

			pw.println(Text);
			pw.close();

			break;
		case 4:
			FileWriter fw3 = new FileWriter(file_name, true);
			BufferedWriter bw1 = new BufferedWriter(fw3);
			PrintWriter pw1 = new PrintWriter(bw1);

			pw1.print(Text);
			pw1.close();

			break;
		}


	}

	public static void reduce(char[] BigCharArray) {

		for(int i = 0; i<BigCharArray.length ; i++) {
			if(Histo.get(BigCharArray[i]) != null) {
				int value = Histo.get(BigCharArray[i]);
				Histo.put((BigCharArray[i]), value+1);
			}
		}
		Histo.values().removeAll(Collections.singleton(0));


	}

	public static char[] Arraynorm(char[] ListNotNorm) {

		char[] notnorm = new char[ListNotNorm.length];

		for(int i = 0; i < notnorm.length; i++) {
			notnorm[i] = Character.toUpperCase(ListNotNorm[i]);
		}
		return notnorm;
	}

	public static void FillMap() {
		for (int i = min; i<max;i++) {
			Histo.put((char) i, 0);
		}
	}

	public static void CreateFile(String file_name) throws IOException {
		File file = new File(file_name);

		file.createNewFile();
	}

	public static void printHistoHori() throws IOException {
		String line;

		for (Entry<Character, Integer> entry : Histo.entrySet()) {
		    int power = entry.getValue();
		    line = String.valueOf(entry.getKey());

		    for(int j = 0; j < power;j++) {
					line = line + "*";
				}
		    writer(Historows,line,3);
		}
	}

	public static void printHistoVerti() throws IOException {

		int maxLetter = 0;

		for (Entry<Character, Integer> entry : Histo.entrySet()) {
			if(maxLetter<entry.getValue()) {
				maxLetter= entry.getValue();
			}

		}
		// Histo.size == Columns; maxLetter == Rows
		char [][] Histogramm = new char[Histo.size()][maxLetter+1];

		int index = 0;


		for (Entry<Character, Integer> entry : Histo.entrySet()) {
		    Histogramm[index][0] = entry.getKey();
		    index++;
		}
```

```
				index=0;

				for(int i = 1; i <= maxLetter;i++) {

					for (Entry<Character, Integer> entry : Histo.entrySet()) {

						if(i <= entry.getValue()) {
							Histogramm[index][i] = '*';
						}
						else {
							Histogramm[index][i] = ' ';
						}
						index++;
					}
					index = 0;
				}

				writer(Historows,"",1);
				String Text = "";
				for(int i = maxLetter; i >= 0;i--) {


					Text = Text + "\r\n";
					for (Entry<Character, Integer> entry : Histo.entrySet()) {
						Text = Text + String.valueOf(Histogramm[index][i]) + " ";

						index++;
					}
					index=0;
				}
				writer(Historows,Text,3);
			}
		}
```