Informatik 2 Group 2

Lab 2 Histogram

By Bartholomäus Berresheim (s0568624) & Fatima Rindert (s0571628)

Index

Introduction

In this weeks lab, the main focus was on Characters. How to read them from a file, count them and then create a histogram based on that information.

Pre-lab

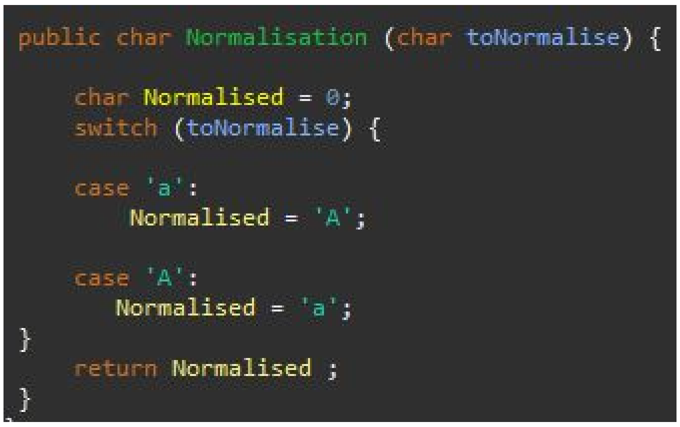
P1 In some programming languages, such as Ada, you can define an array of characters with any discrete type as the index:
someArray : ARRAY ['A' .. 'Z'] of INTEGER;

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'?

This question was confusing at first, since it stated that an array in Java doesn’t have the corresponding feature but we were still supposed to use it to solve the problem. After some discussion with Shiro, she sent me a link (s1) she had found. Using this I came to a website showing discussions on that subject. Researing on that website, we found 2 ways to answer this question, but only one of them implies the usage of an Array, while the other uses a HashMap.
The first one is to create an array of length [‘z’ + 1] and then to use the elements, that have a char as an index and to leave the others empty.
The other one, which we used later on, was to create a HashMap and directly assign a char as the index.

P2 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.

For this task, we found 3 different approaches, one more or less elegant than the others. The first one would be to use a switch and create a case for every single letter and turning it to lower or upper Case, depending on the input.

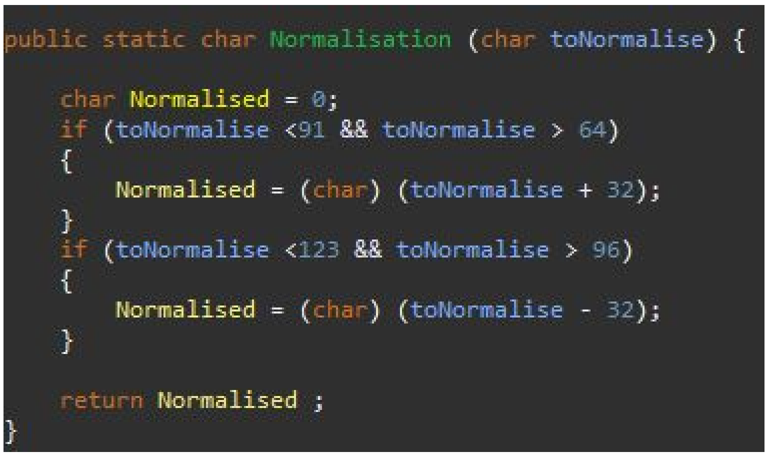


```
public char Normalisation (char toNormalise) {

    char Normalised = 0;
    switch (toNormalise) {

    case 'a':
        Normalised = 'A';

    case 'A':
       Normalised = 'a';
}
    return Normalised ;
}
```

(This screenshot only shows 2 cases, since that would be enough to get a general idea).

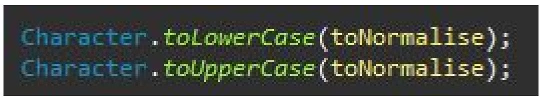
The second approach we found was something, we heard from Yuri. The idea was to change the decimal value of the characters. Looking at the ASCII table, we noticed that the difference between Upper- and

Lowercase letters was always 32, which means that it was much quicker to write than the first one, and less prone to writing mistakes.



```
public static char Normalisation (char toNormalise) {

    char Normalised = 0;
    if (toNormalise <91 && toNormalise > 64)
    {
        Normalised = (char) (toNormalise + 32);
    }
    if (toNormalise <123 && toNormalise > 96)
    {
        Normalised = (char) (toNormalise - 32);
    }

    return Normalised ;
}
```

We weren’t sure if we were allowed to use last approach since it is a Java method, but we put it here anyway for completion purposes.



```
Character.toLowerCase(toNormalise);
Character.toUpperCase(toNormalise);
```

(It is not a method from the Java String class but from the Java Character class)

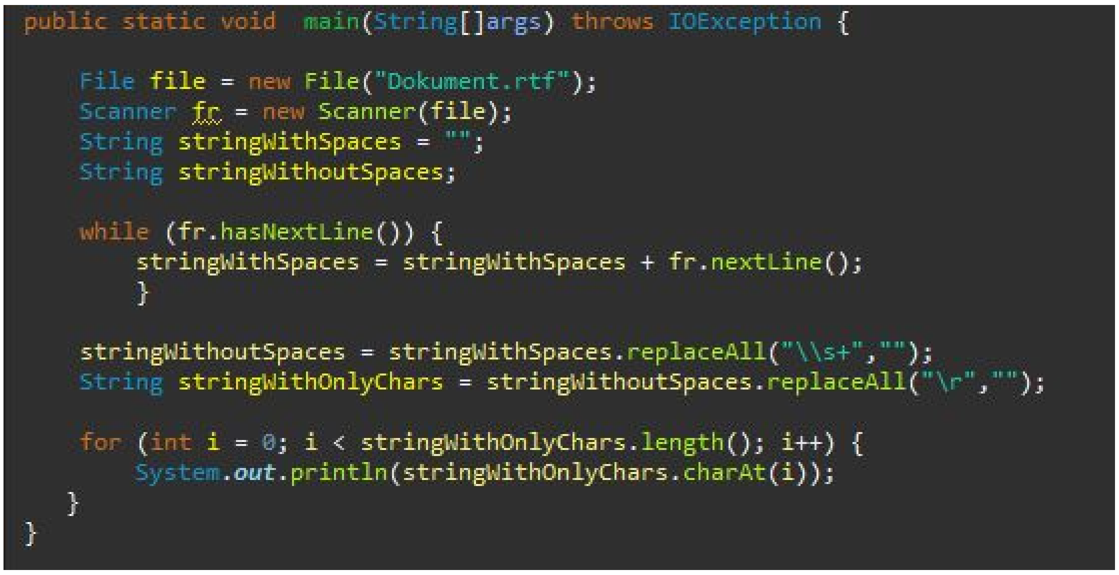
P3 What is a "carriage return"? Where does the name come from?
A “carriage return” moves the cursor to the beginning of the next line. If you want to create one in Java, you either write “\n”, “\r” or “\r\n” in your code. It comes from. The term comes from typewriters, where it signified the switching to 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.

There were a couple mistakes we did, before achieving the desired result but in the end we did get there. They weren’t caused by ignorance but rather inattentiveness, like renaming the main method and wondering why the program didn’t start or trying to read the chars from a String instead of a File. At least I can say that, for next time we won’t make these mistakes again.

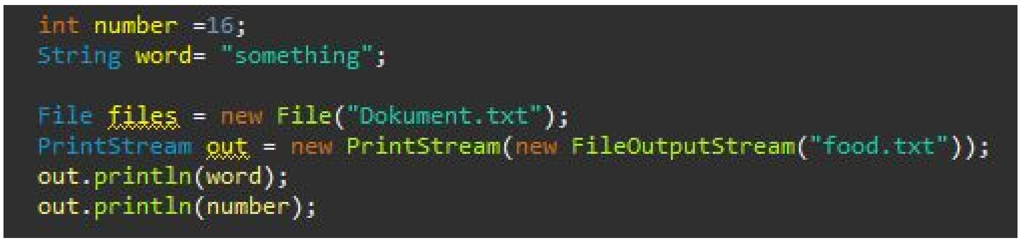
To get back on track, we looked up “File” in the Java API and other websites (s2) and found the Scanner method, which we then used to read from the File. Admittedly we could have used the FileReader seen in class for that task but in combination with the nextLine() method, we were able to convert the input from the File to a String, using a while-loop. This meant that if there was no input, then the String would stay empty.
We then choose to use the replaceAll() method to remove the spaces and carriage returns. Leaving us with the question on how to output the String char by char. We ended up creating a for-loop in which we used the charAt(index) method to output the chars.



```java
public static void  main(String[]args) throws IOException {

    File file = new File("Dokument.rtf");
    Scanner fr = new Scanner(file);
    String stringWithSpaces = "";
    String stringWithoutSpaces;

    while (fr.hasNextLine()) {
        stringWithSpaces = stringWithSpaces + fr.nextLine();
        }

    stringWithoutSpaces = stringWithSpaces.replaceAll("\\s+","");
    String stringWithOnlyChars = stringWithoutSpaces.replaceAll("\r","");

    for (int i = 0; i < stringWithOnlyChars.length(); i++) {
        System.out.println(stringWithOnlyChars.charAt(i));
    }
}
```

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?

To write anything to a File, we first needed to create a Writer. There are many different types of Writers, like a BufferedWriter or PrintWriter. We later choose to create a PrintStream, using a FileOutputStream as parameter, which in turn had the file name as parameter.
Doing it this way, allowed us to create the file at the same time as the writer. Of course, we can create a File separately by using new File (filename), as you can see in the Code of the previous task. And finally, if you want to write to the newly created file, you simply use the println() method.

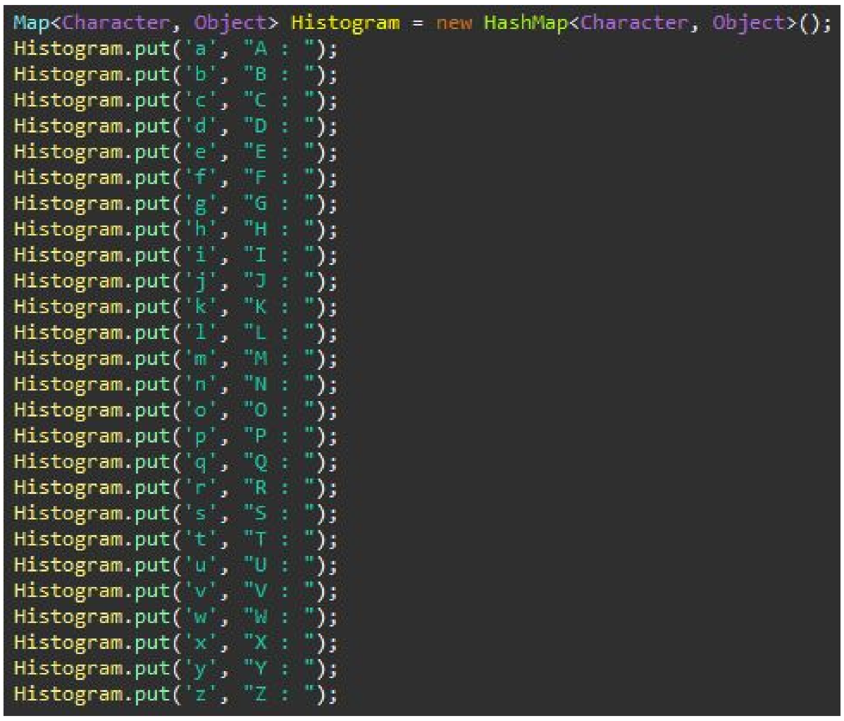


```java
int number =16;
String word= "something";

File files = new File("Dokument.txt");
PrintStream out = new PrintStream(new FileOutputStream("food.txt"));
out.println(word);
out.println(number);
```

(Since the question didn't say that, we had to write a method to showcase how we did it, we weren't sure if we were supposed to do it but in the end, we wrote this to underline what we described up above)

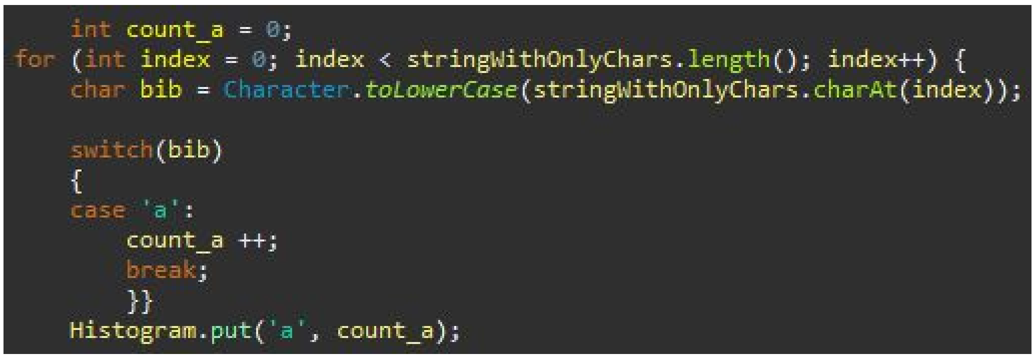
3&4. Now the fun begins! Write a Java application to read in a file character by character, counting the frequencies with which each character occurs. When there are no more characters, create a file frequency.txt and output the frequencies for each character.
*Output a **histogram** of the character frequencies. One simple kind of histogram has horizontal lines proportional to the magnitude of the number it represents.*

For this Task we wrote 2 Versions: the First One was rather inelegant and writing it was a Nightmare, since we made a couple mistakes that took us ages to find the Cause for. It had the downside of creating a histogram that only accounted for the Letters of the Alphabet and not the other Characters. The second one is much tidier and works for all the elements from the ASCII table from 0 to 127. They both are the same at their Core, but they have different approaches for each Task.
The Approach for the first Version could be described of doing "manual labor" instead of letting a loop do the work for you. What I mean by that is that we created a HashMap and then used the put()-method to create an Element for each letter of the Alphabet.



```java
Map<Character, Object> Histogram = new HashMap<Character, Object>();
Histogram.put('a', "A : ");
Histogram.put('b', "B : ");
Histogram.put('c', "C : ");
Histogram.put('d', "D : ");
Histogram.put('e', "E : ");
Histogram.put('f', "F : ");
Histogram.put('g', "G : ");
Histogram.put('h', "H : ");
Histogram.put('i', "I : ");
Histogram.put('j', "J : ");
Histogram.put('k', "K : ");
Histogram.put('l', "L : ");
Histogram.put('m', "M : ");
Histogram.put('n', "N : ");
Histogram.put('o', "O : ");
Histogram.put('p', "P : ");
Histogram.put('q', "Q : ");
Histogram.put('r', "R : ");
Histogram.put('s', "S : ");
Histogram.put('t', "T : ");
Histogram.put('u', "U : ");
Histogram.put('v', "V : ");
Histogram.put('w', "W : ");
Histogram.put('x', "X : ");
Histogram.put('y', "Y : ");
Histogram.put('z', "Z : ");
```

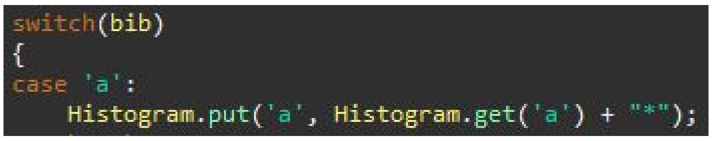
We then kept the method we had for the reading of the File in and only changed the for-loop at the End. Instead of outputting the chars to the Console, we used the toLowerCase() method to normalize the Characters and then used a switch to count the frequency of the letters. For the Assignment 3 we created a local variable for each letter, which would get counted up if the corresponding Case happened and then after the Loop, it would be put in the HashMap.



```
    int count_a = 0;
for (int index = 0; index < stringWithOnlyChars.length(); index++) {
    char bib = Character.toLowerCase(stringWithOnlyChars.charAt(index));

    switch(bib)
    {
    case 'a':
        count_a ++;
        break;
        }}
    Histogram.put('a', count_a);
```

(3)

For the assignment 4 we added a "*" to the corresponding HashMap element instead of a counter.



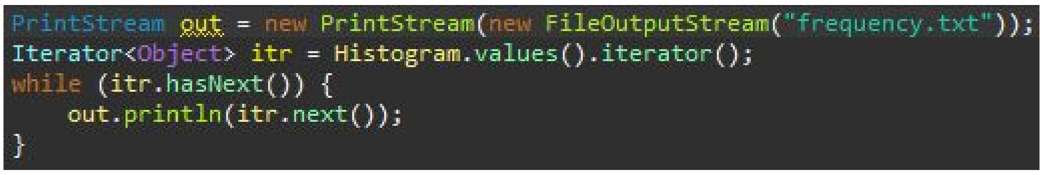
```
switch(bib)
{
case 'a':
    Histogram.put('a', Histogram.get('a') + "*");
```

(4)

(We didn't screenshot all of it, but you should get the idea of how it works)

As you may have noticed, this way of writing the code is not a good idea, since it is very easy to overlook something, if you made a Mistake...Which we did. For whatever reason, we forgot to but breaks at the end of some cases, causing some of the Frequencies shown in the histogram to have double or triple their real size.

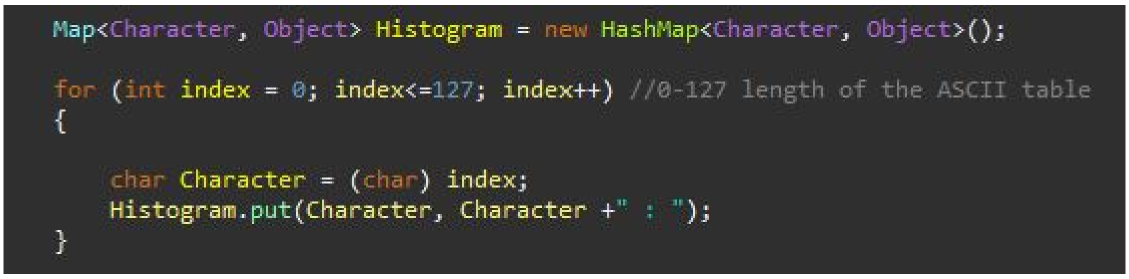
To finish things off, we wrote an iterator for the values we had in the HashMap, which used a while-loop and the hasNext() method to write to the file via the PrintStream we had seen earlier.



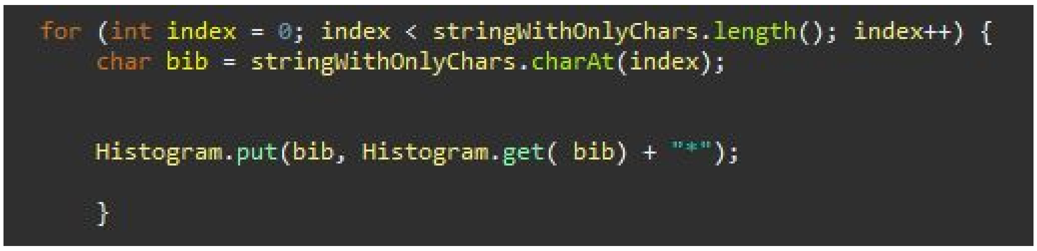
```
PrintStream out = new PrintStream(new FileOutputStream("frequency.txt"));
Iterator<Object> itr = Histogram.values().iterator();
while (itr.hasNext()) {
    out.println(itr.next());
}
```

After being relatively happy with our work, I sent it to Yuri to get his Opinion on it. He then told me that just looking at it gave him brain seizures and that using loops and the decimal values of Characters would look much better. Although he had a point, finishing the code was quite irritating, so we didn't want to change it again, just for the looks of it. He then pointed out that the task stated “Characters” and not “letters”, which meant our code was faulty since it ignored the other characters. Which lead us to the Creation of the second Version.

In this Version we didn’t initiate the values of the map for each letter manually but we wrote a loop using the ASCII decimals from 0-127 and created a corresponding entry in our Map.


```
Map<Character, Object> Histogram = new HashMap<Character, Object>();

for (int index = 0; index<=127; index++) //0-127 length of the ASCII table
{

    char Character = (char) index;
    Histogram.put(Character, Character +" : ");
}
```

We did something similar for the counting of the Frequency and since there was no mention of Normalization in this assignment, we removed the toLowerCase() method.


```
for (int index = 0; index < stringWithOnlyChars.length(); index++) {
    char bib = stringWithOnlyChars.charAt(index);


    Histogram.put(bib, Histogram.get( bib) + "*");

    }
```

The iterator at the end didn’t need any changing.

5. What is the complexity of your algorithm?
Since our code doesn’t contain any nested loops, we think that the complexity is linear.

Reflections

Fatima:
In this weeks lab I learned that sometimes it’s good to asks other people for their Opinion, to improve your work. Although I found it quite cool to work on “Files” it was more complicated than I first thought. I start to get quite frustrated when things are not working and I had to learn to manage that. I learned that I first have to plan the things I want to do and not just start typing something, which in the end is nothing but a mistake.

Bartholomäus:
This weeks lab was quite a mess to be honest. It wasn't difficult but due to the mistakes I made and having to correct them, it got quite taxing mentally. Going the "manual labor" path might have been less complex (although not much), it ended up causing multiple instances, during which I just wanted to flip my table over and walk away (which I didn't do). So if I have to say that I learned something from this Lab, then it was that "simpler" doesn't mean less work. And that next time I should think more or rather inform myself better, if at the beginning I had thought about using the ASCII decimal, it would have spared me from so much work and frustration. But I guess you first have to make mistakes to learn from them.

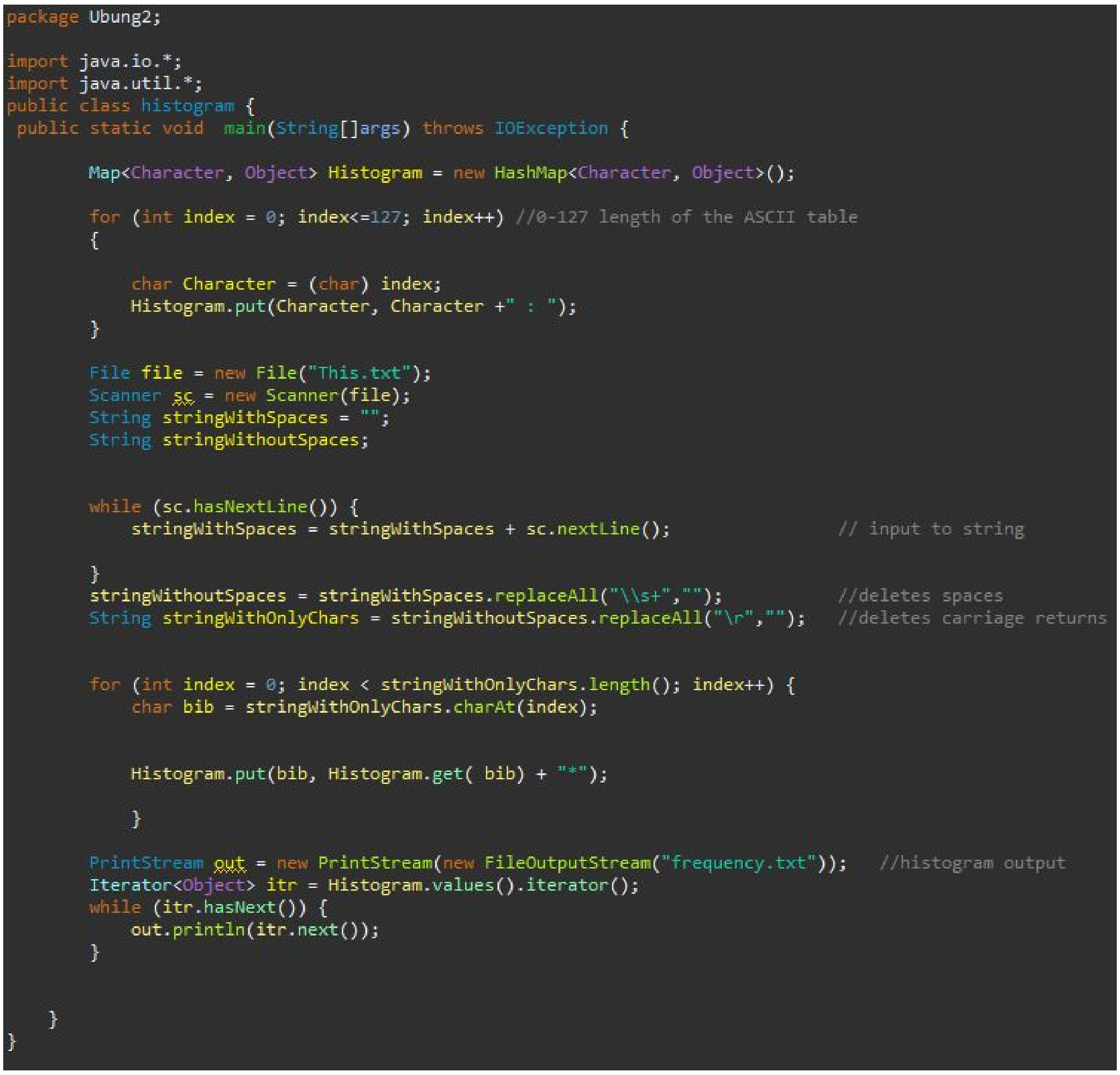
Appendix

Sources:

(s1) :
https://stackoverflow.com/questions/11069609/java-create-an-array-with-letter-characters-as-index#11069635
(s2) : https://www.w3schools.com/java/java_files.asp
https://javaconceptoftheday.com/remove-white-spaces-from-string-in-java/

Fatima Rindert 571628 25.10.2019
Bartholomäus Berresheim 568624 Informatik 2 Group 2

Code:



```java
package Ubung2;

import java.io.*;
import java.util.*;
public class histogram {
 public static void  main(String[]args) throws IOException {

        Map<Character, Object> Histogram = new HashMap<Character, Object>();

        for (int index = 0; index<=127; index++) //0-127 length of the ASCII table
        {

            char Character = (char) index;
            Histogram.put(Character, Character +" : ");
        }

        File file = new File("This.txt");
        Scanner sc = new Scanner(file);
        String stringWithSpaces = "";
        String stringWithoutSpaces;


        while (sc.hasNextLine()) {
            stringWithSpaces = stringWithSpaces + sc.nextLine();                // input to string

        }
        stringWithoutSpaces = stringWithSpaces.replaceAll("\\s+","");           //deletes spaces
        String stringWithOnlyChars = stringWithoutSpaces.replaceAll("\r","");   //deletes carriage returns


        for (int index = 0; index < stringWithOnlyChars.length(); index++) {
            char bib = stringWithOnlyChars.charAt(index);


            Histogram.put(bib, Histogram.get( bib) + "*");

            }

        PrintStream out = new PrintStream(new FileOutputStream("frequency.txt"));   //histogram output
        Iterator<Object> itr = Histogram.values().iterator();
        while (itr.hasNext()) {
            out.println(itr.next());
        }


    }
}
```