IMI WiSe 2019 / Informatik 2 / Group 1

Students : Anh Pham Viet
Pavel Tsvyatkov

Email : s0569867@htw-berlin.de
s0559632@htw-berlin.de

Instructor: Prof. Dr. Debora Weber Wulff

Laboratory Report

Exercise 2: Histogram

Lab-plan

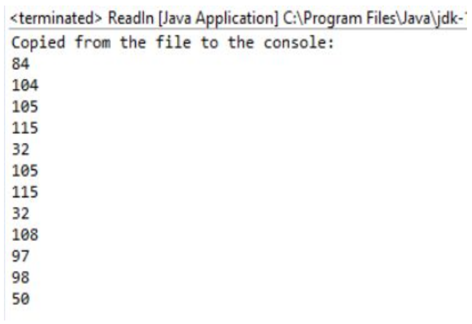
This lab was about reading in characters from a file and dealing with carriage return. We also had to create a file and write different data to it. We had to create a program that reads in a file character by character, count the frequencies of the characters and output them. We had to output a histogram of the character frequencies and tell what's the complexity of our algorithm.

Assignment

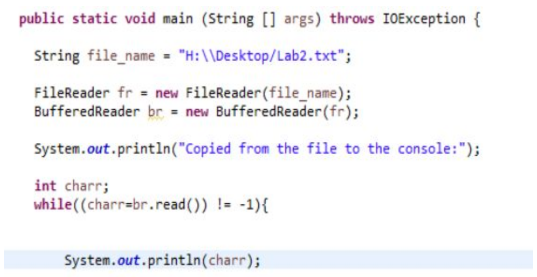
1. How do you go about reading in characters from a file? Write and test a method that returns the next character in a file. Note that you have to do something with the carriage returns - such as ignoring them - and that you have to decide what to do when there are no characters to be returned.

   First of all we created a String file_name and assigned the path of our .txt document to it. With the help of the lecture script we we created a new FileReader and BufferedReader in order read the file.

With the while loop we will be able to determine when the code should end. Looking at the Java API we searched for a method to read chars - read() seemed to be fine. Well, it only printed numbers to the console when we ran it even though we had letters written in the .txt file.



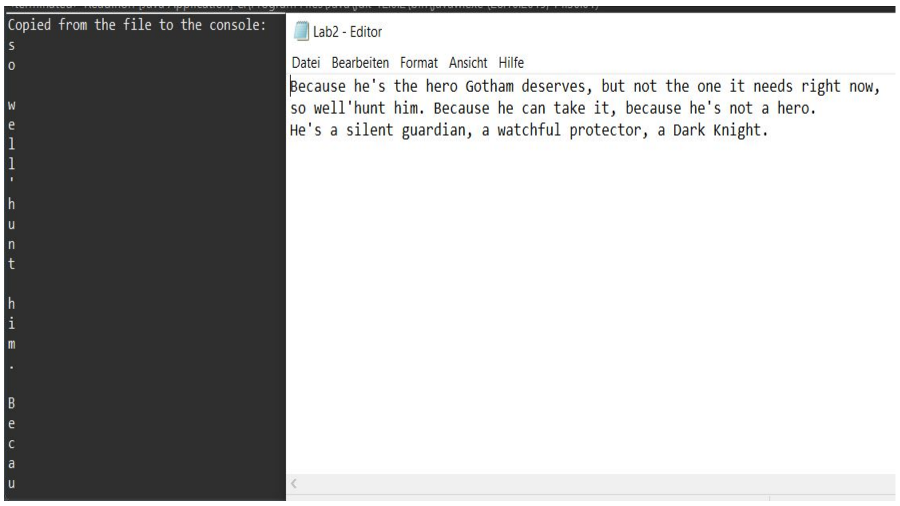
```
<terminated> ReadIn [Java Application] C:\Program Files\Java\jdk-
Copied from the file to the console:
84
104
105
115
32
105
115
32
108
97
98
50
```



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

  String file_name = "H:\\Desktop/Lab2.txt";

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

  System.out.println("Copied from the file to the console:");

  int charr;
  while((charr=br.read()) != -1){


      System.out.println(charr);
```

Our first assumption was that they must be Unicode and upon comparing to the unicode table it was indeed so. We simply had to cast our int charr to type String in order to print out what we intend to print out.

Now our next problem was that this method read the chars one by one and put out a carriage return after each sentence. Furthermore it did not print the first line.



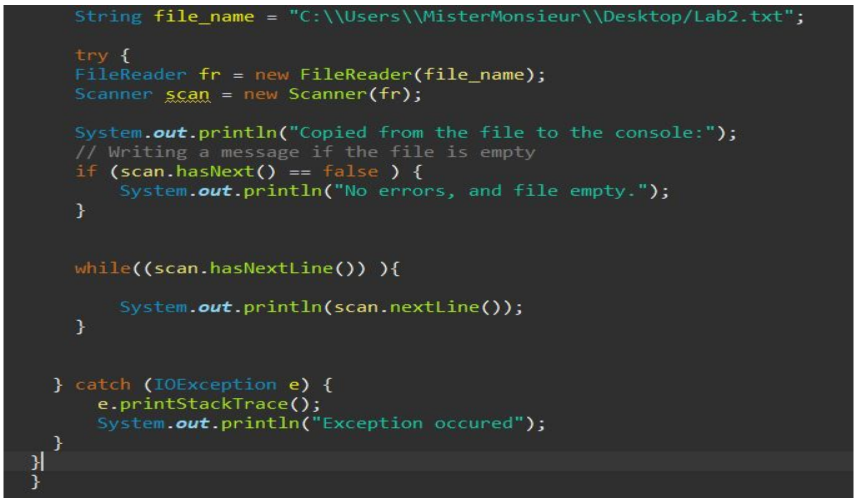
```
Copied from the file to the console:
s
o

w
e
l
l
'
h
u
n
t

h
i
m
.

B
e
c
a
u
```

```
Because he's the hero Gotham deserves, but not the one it needs right now,
so well'hunt him. Because he can take it, because he's not a hero.
He's a silent guardian, a watchful protector, a Dark Knight.
```

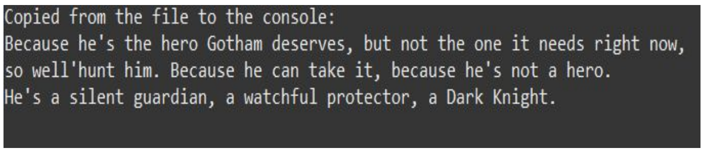
To cut it short, we tried lots of things but haven’t been able to get rid of the first line problem. So we ditched our bufferedReader in favour of the Scanner class[1].

[1] https://www.youtube.com/watch?v=IHFlAYaNfdo

With this we didn't need to worry about the carriage return issue anymore since the scanner just scans the .txt file how it supposed to be viewed by the writer. Also the first line issue doesn't exist here.



```
        String file_name = "C:\\Users\\MisterMonsieur\\Desktop/Lab2.txt";

        try {
        FileReader fr = new FileReader(file_name);
        Scanner scan = new Scanner(fr);

        System.out.println("Copied from the file to the console:");
        // Writing a message if the file is empty
        if (scan.hasNext() == false ) {
            System.out.println("No errors, and file empty.");
        }


        while((scan.hasNextLine()) ){

            System.out.println(scan.nextLine());
        }


    } catch (IOException e) {
        e.printStackTrace();
        System.out.println("Exception occured");
    }
  }
  }
```

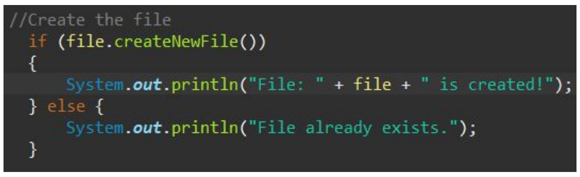


```
Copied from the file to the console:
Because he's the hero Gotham deserves, but not the one it needs right now,
so well'hunt him. Because he can take it, because he's not a hero.
He's a silent guardian, a watchful protector, a Dark Knight.
```

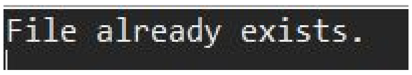
Well in the end we were not sure whether we should just print one char of the whole text, or all one by one. We asked a classmate which did the former one but we found it the most useful way to be faithful to the format written in the .txt file.

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?

   Before writing anything, we first create a file by creating a new file object and assigning it to the path, its name and type. We chose a .txt file again since we want to read it. Well it is not created yet, we used the createNewFile() method from the File object class in an if-clause to determine whether there is an identical existing or not.

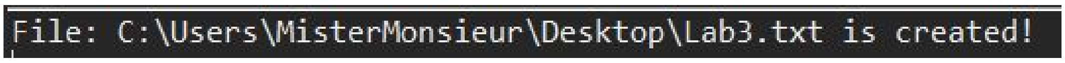


```
//Create the file
  if (file.createNewFile())
  {
      System.out.println("File: " + file + " is created!");
  } else {
      System.out.println("File already exists.");
  }
```



```
File already exists.
```

Lab2 already exists.

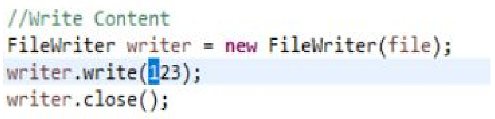


```
File: C:\Users\MisterMonsieur\Desktop\Lab3.txt is created!
```

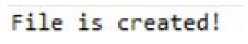
But we did not have Lab3.txt yet, thus we've been able to create it.

Now to write someting into the file, a Filewriter object was needed.[2]

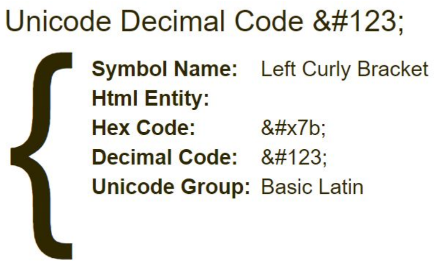
We passed in our created file as parameter and used the method writer() of Filewriter to write something into it. We chosed a simple 123 and that is what we got:



```
//Write Content
FileWriter writer = new FileWriter(file);
writer.write(123);
writer.close();
```



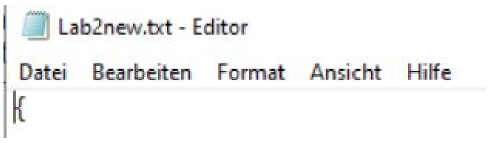
```
File is created!
```



Unicode Decimal Code &#123;

{

Symbol Name:	Left Curly Bracket
Html Entity:	
Hex Code:	&#x7b;
Decimal Code:	&#123;
Unicode Group:	Basic Latin

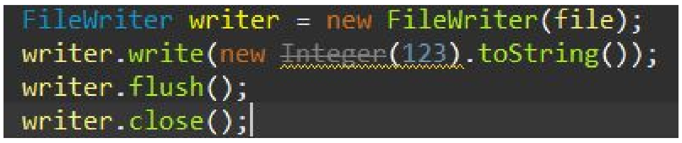


Lab2new.txt - Editor

Datei Bearbeiten Format Ansicht Hilfe

{

A curly bracket? We had something like this before, one look at unicode table reveals that 123 is { in Unicode. But we wanted to display 123, therefore the toString() was needed to convert the language.

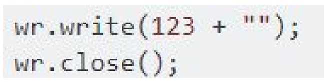


```
FileWriter writer = new FileWriter(file);
writer.write(new ~~Integer~~(123).toString());
writer.flush();
writer.close();
```

Eclipse automatically crossed out the Integer when converting it into String.

Since Integer is a wrapper class it is way more flexible than int, which is primitive. Integer can be casted / assigned to String, while you can't really do that with an int and thus can't write an int to a file.

But there is clever trick to write an int to a file. In the stackoverflow-link there was an example:



```
wr.write(123 + "");
wr.close();
```

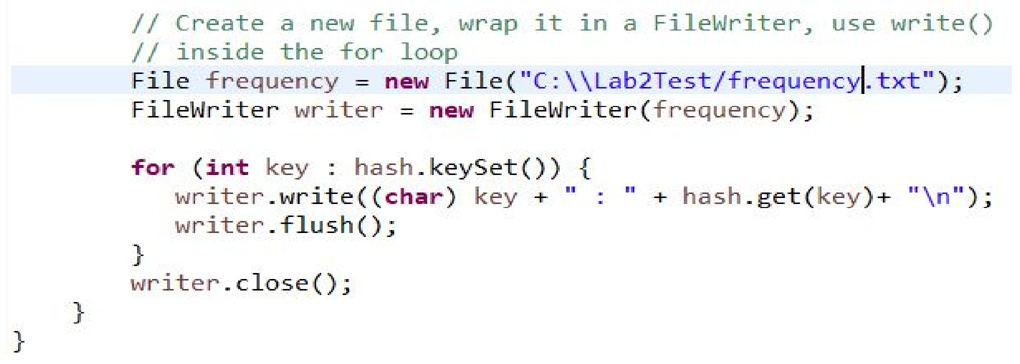
It concatenates the int to a string and therefore parses the int to a string. This hack might be even easier then using a wrapper class like Integer.

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.

2 https://stackoverflow.com/questions/7357852/write-int-to-text-file-using-writer?fbclid=IwAR3981p4thBgr69NM5Jm12NVl9BkIAXcWK4evN8nnPh00c3G9TTG6S0jp90

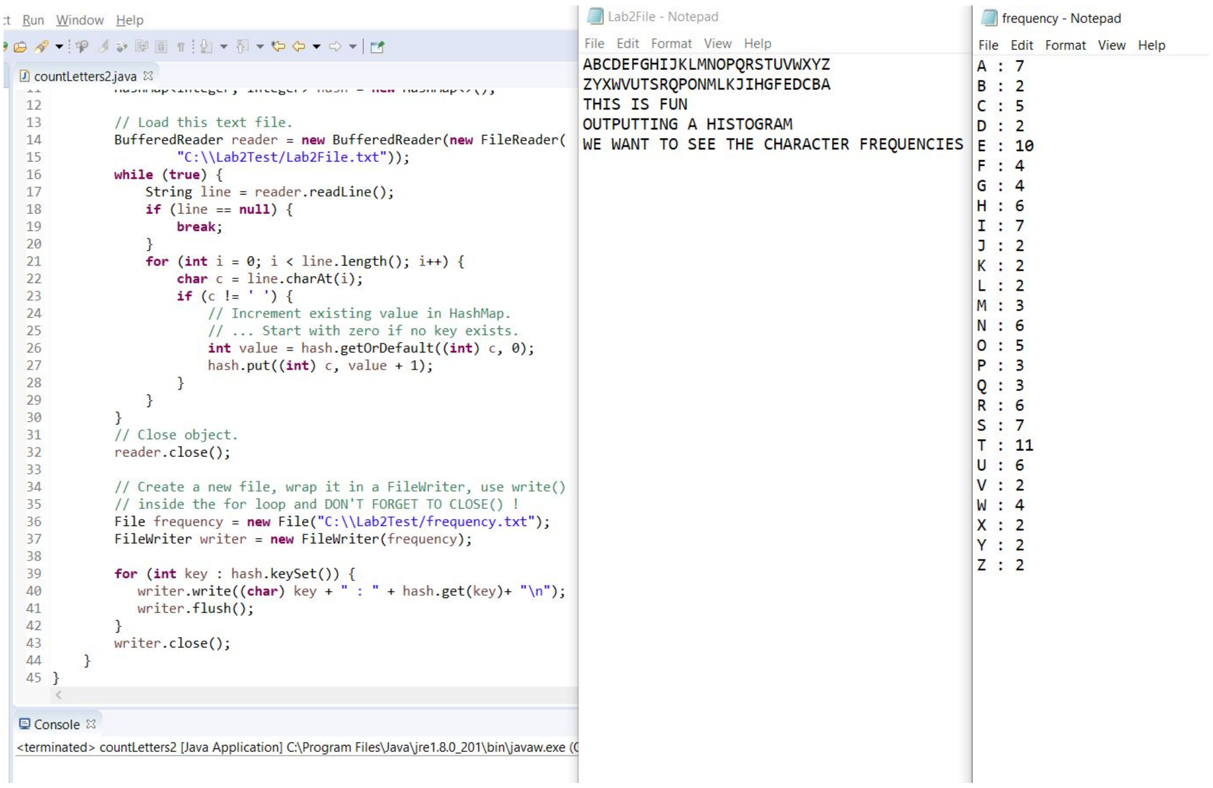
We didn’t have time to begin with this exercise in the lab, so we both agreed we are going to continue working on it and the following exercises from home. We discussed about this exercise and we had similar ideas about what our approach could be to solve it, but we weren’t exactly sure how to structure our code. Our initial idea was to create a new array[] that holds 26 elements which we are going to use for all the letters in the alphabet and then use a for loop to set them to 0. After that we would have to wrap the File we are going to read from in a Buffered Reader and read in it character by character and increase the number of the letters each time we have read them. After that we thought we can create a new frequency.txt file using the File class and a FileWriter to fill it with the characters and their frequencies. However, when we tried to write our code, we weren’t able to structure it correctly and we needed help.
At this point we started researching online for more information. We really liked the explanation with comments and the structure of the code that we found here: http://dotnetperls.com/count-letter-frequencies-java. Until this point we didn’t think about solving this task with a HashMap. We saw that in this example the HashMap gets populated with values and in the end there is a for loop that loops through the keySet and prints them out.

We tested the code and it worked, but it wasn’t doing exactly what we needed. We had to create a new File and output the frequencies of the characters in the file instead of printing them to the console.
First we had to import java.io.File and java.io.FileWriter. Then we create a new File called frequency and make sure we use “\\” and “/” correctly, because we had some issues when we tried to copy our path the first time. We then create a new FileWriter and wrap our frequency File in it. Then, instead of using System.out like in the code we found online, we still keep the for loop, but this time we use the write() method. It took us a while until we structured everything correctly. We had to add “\n” so that everything gets written on a new line and then we use flush(). At first we put the close() method inside the for loop and we were wondering why we didn’t get the expected output. After few minutes thinking, we realized that and put the close() method out of the for loop.



```
        // Create a new file, wrap it in a FileWriter, use write()
        // inside the for loop
        File frequency = new File("C:\\Lab2Test/frequency.txt");
        FileWriter writer = new FileWriter(frequency);

        for (int key : hash.keySet()) {
           writer.write((char) key + " : " + hash.get(key)+ "\n");
           writer.flush();
        }
        writer.close();
    }
}
```

After that we proceeded to test our program and it was working. When we run the program as soon as we finish reading in the file, a new frequency text file was getting created at the path we have it set to.



```
        // Load this text file.
        BufferedReader reader = new BufferedReader(new FileReader(
                "C:\\Lab2Test/Lab2File.txt"));
        while (true) {
            String line = reader.readLine();
            if (line == null) {
                break;
            }
            for (int i = 0; i < line.length(); i++) {
                char c = line.charAt(i);
                if (c != ' ') {
                    // Increment existing value in HashMap.
                    // ... Start with zero if no key exists.
                    int value = hash.getOrDefault((int) c, 0);
                    hash.put((int) c, value + 1);
                }
            }
        }
        // Close object.
        reader.close();

        // Create a new file, wrap it in a FileWriter, use write()
        // inside the for loop and DON'T FORGET TO CLOSE() !
        File frequency = new File("C:\\Lab2Test/frequency.txt");
        FileWriter writer = new FileWriter(frequency);

        for (int key : hash.keySet()) {
           writer.write((char) key + " : " + hash.get(key)+ "\n");
           writer.flush();
        }
        writer.close();
    }
}
```

```
<terminated> countLetters2 [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe
```

Lab2File - Notepad

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ZYXWVUTSRQPONMLKJIHGFEDCBA
THIS IS FUN
OUTPUTTING A HISTOGRAM
WE WANT TO SEE THE CHARACTER FREQUENCIES
```

frequency - Notepad

```
A : 7
B : 2
C : 5
D : 2
E : 10
F : 4
G : 4
H : 6
I : 7
J : 2
K : 2
L : 2
M : 3
N : 6
O : 5
P : 3
Q : 3
R : 6
S : 7
T : 11
U : 6
V : 2
W : 4
X : 2
Y : 2
Z : 2
```

The frequency text file was getting filled with the characters that occured and their frequency. We knew that there are probably other and better approaches to solve this exercise, but we liked the idea to use a HashMap and use the keySet method. We made sure to go through the code and read the explanations, so that we get a better idea of what our program is doing exactly.

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.

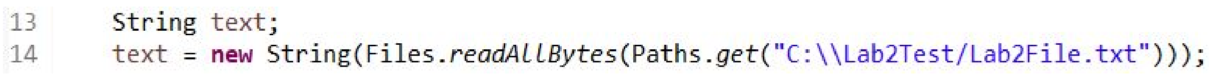
When we got to this exercise it had us thinking for quite a while, because at first we weren't able to come up with any good ideas about outputting a histogram. We first read the Histogram wiki to get a better understanding of the term and saw some examples. Then we looked at the previous exercise and realized that it's similar in a way. We thought about how can we change the code so that we print a star each time we read a character. We started explaining our thoughts and agreed that we needed an array for the letters from A to Z again and a file to read in. We also agreed that we will need another for loop to loop through the letters and a nested for loop to print a star for every occurence. We were not sure how to implement this exactly in our code since we used a HashMap for the previous exercise and we got stuck, because now we had to approach the exercise in a different way. We tried a few solutions by using a nested for loop, but we weren't able to get a positive result.

We decided to look online for an advice or guidance about how to approach this exercise. After researching for a while we found something very useful that helped us a lot to solve the task:
https://stackoverflow.com/questions/53252568/how-to-make-a-frequency-histogram-in-java

We read through the code and the explanations and we both instantly thought that this will help us find a working solution to our exercise. We really liked that the code had comments and it was easy to read and follow. We noticed that first an array of integers that holds 26 elements gets created, which is what our initial idea was as well. It was interesting to see that inside the first for loop a method toCharArray() was used so we looked it up in the Java API and saw that it converts a string to a character array. The next thing surprised us a bit, because we weren't sure why 97 was getting subtracted. We both thought about the first exercise when we got an output with numbers and decided to look up the ASCII table. We saw that 97 was the number for the character **a** and realized what was the for loop doing. We both thought that this was an interesting approach. In the code we were then checking to see if the number is bigger than 0 and smaller than 25, because in the ASCII table the small letters were from 97 to 122. We liked that the name of the variable in the first for loop was set to "convert" and in the next for loop to "convertback". In that way we were able to track the thought process and understand the code better. We saw that in the last piece of code there was a nested for loop and it was printing the stars, just as the idea we had in the beginning.

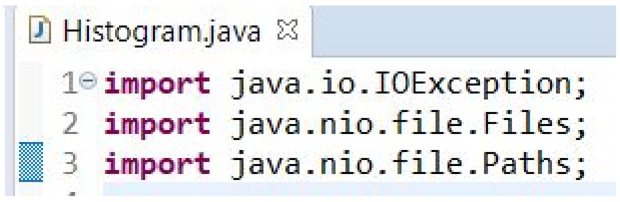
However, when we went on to test the code we realized that we have a problem. We need the File we are reading in to be of type String so we can use the toCharArray() method on it. At this point we tried multiple ways on our own, but we could not get it to work and we were out of ideas. We checked online for a solution and we found an useful example here:
https://www.java67.com/2016/08/how-to-read-text-file-as-string-in-java.html

After reading through the example we went on to try it in our program and wrote the following piece of code.



```
String text;
text = new String(Files.readAllBytes(Paths.get("C:\\Lab2Test/Lab2File.txt")));
```

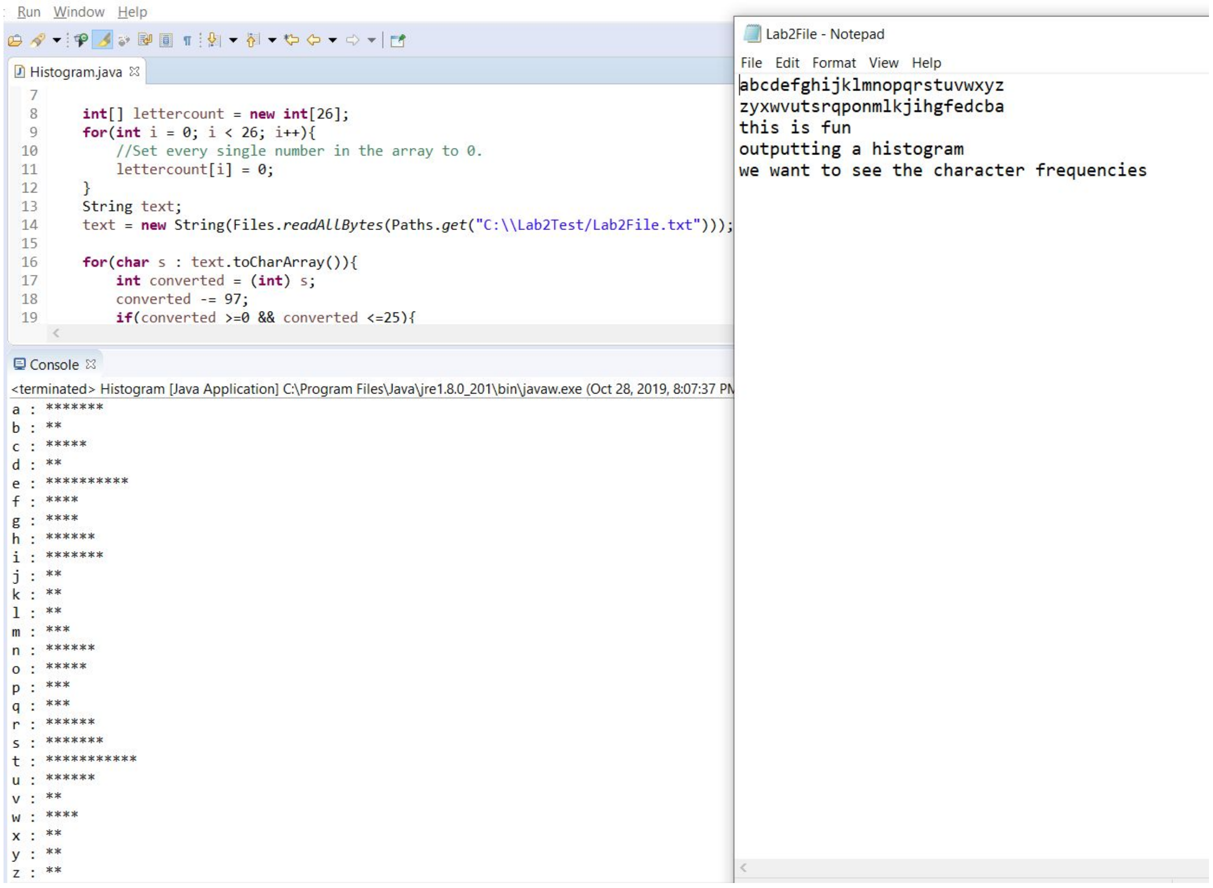
When we added the two lines to our program we saw multiple errors like "Paths cannot be resolved" and "Files cannot be resolved", but we were able to quickly fix them. We had to import the following:



```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
```

We also had to make sure our method throws an IOException and then we were not getting an error anymore.

We filled the file we are reading in from with some text and proceeded to test our program. In the end we commented out the last part of the System.out from the original code and we left it to just output the letter and the stars so that it looks like a Histogram.



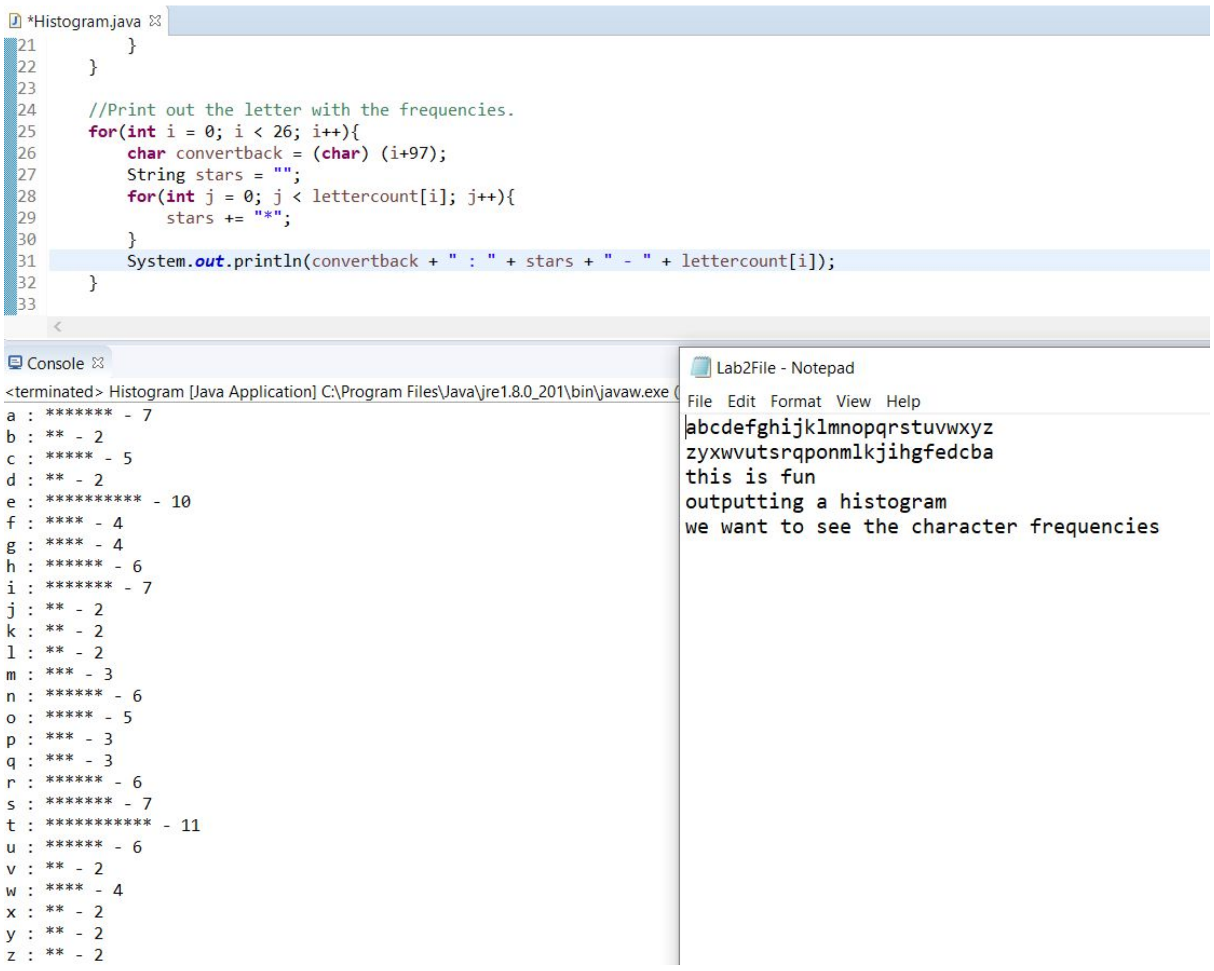
```java

    int[] lettercount = new int[26];
    for(int i = 0; i < 26; i++){
        //Set every single number in the array to 0.
        lettercount[i] = 0;
    }
    String text;
    text = new String(Files.readAllBytes(Paths.get("C:\\Lab2Test/Lab2File.txt")));

    for(char s : text.toCharArray()){
        int converted = (int) s;
        converted -= 97;
        if(converted >=0 && converted <=25){
```

```
abcdefghijklmnopqrstuvwxyz
zyxwvutsrqponmlkjihgfedcba
this is fun
outputting a histogram
we want to see the character frequencies
```

```
<terminated> Histogram [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Oct 28, 2019, 8:07:37 PM
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 : **
```

We were surprised, but it was working well. At this point we were not sure if we should just print the Histogram to the console, but we knew we could use a FileWriter to output the Histogram in a file like we did in the previous exercise.

We were still curious to see the output with the last part of the System.out line so we decided to try it and we really liked the result. It was interesting to play around with the program and test different things.



```java
            }
        }

        //Print out the letter with the frequencies.
        for(int i = 0; i < 26; i++){
            char convertback = (char) (i+97);
            String stars = "";
            for(int j = 0; j < lettercount[i]; j++){
                stars += "*";
            }
            System.out.println(convertback + " : " + stars + " - " + lettercount[i]);
        }
```

```
<terminated> Histogram [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (
a : ******* - 7
b : ** - 2
c : ***** - 5
d : ** - 2
e : ********** - 10
f : **** - 4
g : **** - 4
h : ****** - 6
i : ******* - 7
j : ** - 2
k : ** - 2
l : ** - 2
m : *** - 3
n : ****** - 6
o : ***** - 5
p : *** - 3
q : *** - 3
r : ****** - 6
s : ******* - 7
t : *********** - 11
u : ****** - 6
v : ** - 2
w : **** - 4
x : ** - 2
y : ** - 2
z : ** - 2
```

```
abcdefghijklmnopqrstuvwxyz
zyxwvutsrqponmlkjihgfedcba
this is fun
outputting a histogram
we want to see the character frequencies
```

We were able to output the letter, the stars and the number of occurrences and we thought that was really nice. However, in the end we kept the print line with the letter and the stars only.

5. What is the complexity of your algorithm?

We both talked about the way we solved the last exercise and we thought that the important piece of code are the last two for loops where we loop through the characters and print a star every time there is an occurrence. We were still not entirely sure about the complexity of our algorithm, but we thought it should be **O(N^2)**, because we reviewed our notes from the lecture and saw the examples we did in class with the multiple nested loops.

Personal Reflection

Anh: This exercise made me be more aware of the different data types, their possibilities and limits. Again there were numerous ways to find a solution for one problem. I think the more precise you are with the things you want, the easier it is to find the best possible methods / solutions.

Pavel: This assignment was very interesting, but I also found it quite difficult. I needed to research and test different pieces of code to come up with working solutions, which helped me learn a lot. I learned about the toCharArray method and how to save a file in a variable of type String, which helped us solve the last exercise. It was interesting to see how writing to a file works and the different outputs we were getting when we were both reading in and writing to a file.

Appendix

Sources:

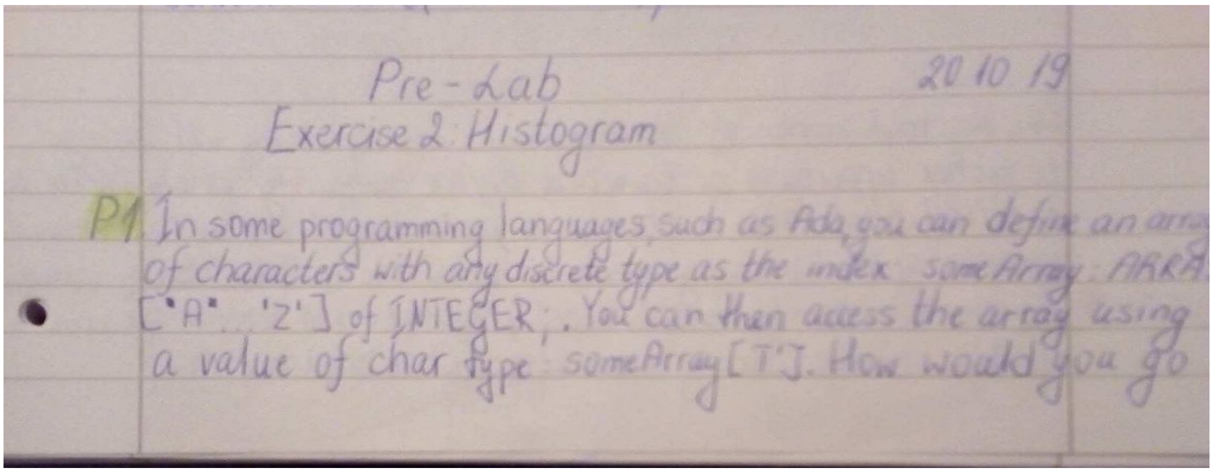
https://stackoverflow.com/questions/7357852/write-int-to-text-file-using-writer

http://dotnetperls.com/count-letter-frequencies-java

https://stackoverflow.com/questions/53252568/how-to-make-a-frequency-histogram-in-java

https://www.java67.com/2016/08/how-to-read-text-file-as-string-in-java.html

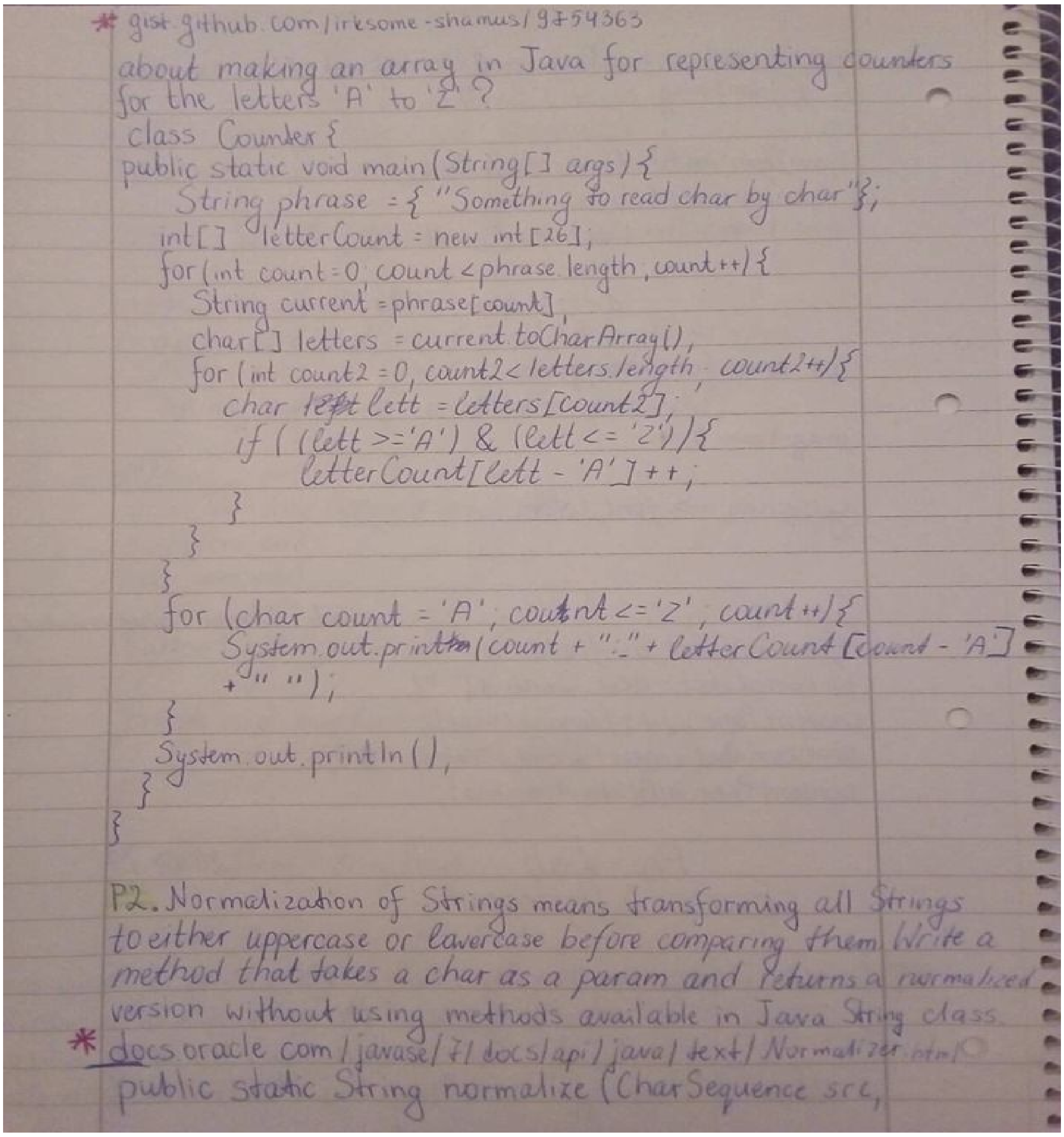
Pre-Lab



Pre-Lab 2010 19

Exercise 2 Histogram

P1. In some programming languages, such as Ada, you can define an arra
of characters with any discrete type as the index. someArray : ARRA
['A'.. 'Z'] of INTEGER;. You can then access the array using
a value of char type: someArray['T']. How would you go



* gist.github.com/irksome-shamus/9754363

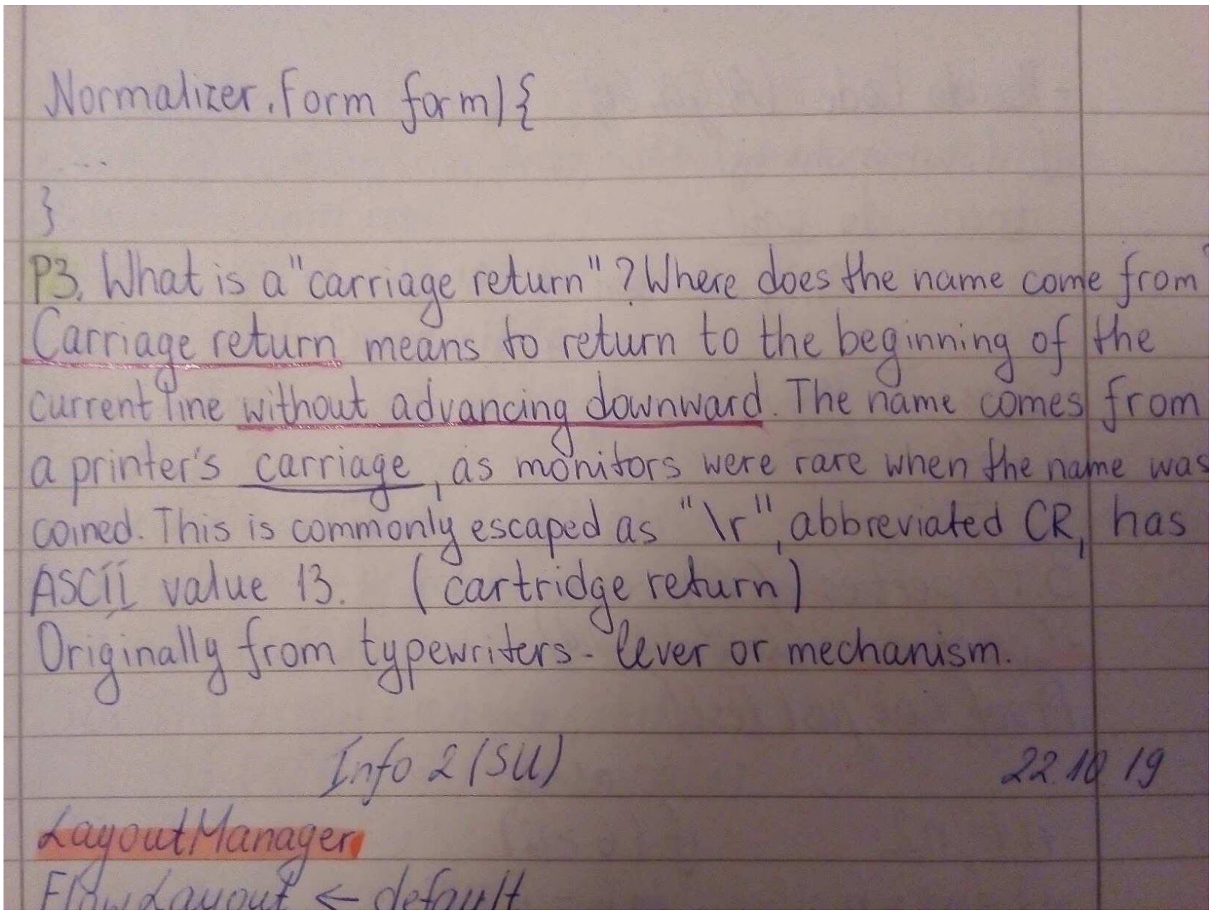
about making an array in Java for representing counters
for the letters 'A' to 'Z'?

```
class Counter {
public static void main(String[] args){
    String phrase = {"Something to read char by char"};
   int[] letterCount = new int[26];
   for(int count=0; count <phrase.length, count++){
      String current =phrase[count],
      char[] letters =current.toCharArray(),
      for (int count2 =0, count2< letters.length, count2++){
        char ~~left~~ lett =letters[count2],
         if ((lett >='A') & (lett <= 'Z')){
             letterCount[lett - 'A']++;
         }
      }
   }
   for (char count = 'A', count <='Z', count++){
       System.out.print(count + ":" + letterCount [count - 'A']
       + " ");
   }
   System.out.println(),
  }
}
```

P2. Normalization of Strings means transforming all Strings
to either uppercase or lowercase before comparing them. Write a
method that takes a char as a param and returns a normalized
version without using methods available in Java String class.

* docs.oracle.com/javase/7/docs/api/java/text/Normalizer.html

public static String normalize (CharSequence src,


Normalizer.Form form){
...
}
P3. What is a "carriage return"? Where does the name come from
Carriage return means to return to the beginning of the current line without advancing downward. The name comes from a printer's carriage, as monitors were rare when the name was coined. This is commonly escaped as "\r", abbreviated CR, has ASCII value 13. (cartridge return)
Originally from typewriters - lever or mechanism.

Info 2 (SU) 22.10.19

LayoutManager
FlowLayout ← default

Code

Exercise 1 Code

```
package lab2;

import java.io.*;
import java.util.Scanner;

        public class ReadIn {

          public static void main (String [] args)   {

                String file_name = "C:\\Users\\MisterMonsieur\\Desktop/Lab2.txt";

                try {
                FileReader fr = new FileReader(file_name);
```

```
            Scanner scan = new Scanner(fr);

            System.out.println("Copied from the file to the console:");
            // Writing a message if the file is empty
            if (scan.hasNext() == false ) {
               System.out.println("No errors, and file empty.");
            }


            while((scan.hasNextLine()) ){

               System.out.println(scan.nextLine());
            }


       } catch (IOException e) {
             e.printStackTrace();
             System.out.println("Exception occured");
       }
      }
      }
```

Exercise 2 code

```
package lab2;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;


public class FileWriterHome {

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

             File file = new File("C:\\Users\\MisterMonsieur\\Desktop/Lab3.txt");

            //Create the file
             if (file.createNewFile())
             {
                System.out.println("File: " + file + " is created!");
             } else {
                System.out.println("File already exists.");
             }
```

```
                //Write Content
                FileWriter writer = new FileWriter(file);
                writer.write(new Integer(123).toString());
                writer.flush();
                writer.close();
}
}
```

Exercise 3 code

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;

public class countLetters2 {
  public static void main(String[] args) throws IOException {

    HashMap<Integer, Integer> hash = new HashMap<>();

    // Load this text file.
    BufferedReader reader = new BufferedReader(new FileReader(
            "C:\\Lab2Test/Lab2File.txt"));
    while (true) {
      String line = reader.readLine();
      if (line == null) {
        break;
      }
      for (int i = 0; i < line.length(); i++) {
        char c = line.charAt(i);
        if (c != ' ') {
          // Increment existing value in HashMap.
          // ... Start with zero if no key exists.
          int value = hash.getOrDefault((int) c, 0);
          hash.put((int) c, value + 1);
        }
      }
    }
    // Close object.
    reader.close();

    // Create a new file, wrap it in a FileWriter, use write()
    // inside the for loop and DON'T FORGET TO CLOSE() !
    File frequency = new File("C:\\Lab2Test/frequency.txt");
```

```
        FileWriter writer = new FileWriter(frequency);

        for (int key : hash.keySet()) {
          writer.write((char) key + " : " + hash.get(key)+ "\n");
          writer.flush();
        }
        writer.close();
    }
}
```

Exercise 4 Code

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class Histogram {
        public static void main(String[] args) throws IOException {

        int[] lettercount = new int[26];
        for(int i = 0; i < 26; i++){
            //Set every single number in the array to 0.
            lettercount[i] = 0;
        }
        String text;
        text = new String(Files.readAllBytes(Paths.get("C:\\Lab2Test/Lab2File.txt")));

        for(char s : text.toCharArray()){
            int converted = (int) s;
            converted -= 97;
            if(converted >=0 && converted <=25){
                lettercount[converted] += 1;
            }
        }

        //Print out the letter with the frequencies.
        for(int i = 0; i < 26; i++){
            char convertback = (char) (i+97);
            String stars = "";
            for(int j = 0; j < lettercount[i]; j++){
                stars += "*";
            }
            System.out.println(convertback + " : " + stars); // + " - " + lettercount[i]);
        }
}
}
```