

# Laboratory 2: Histogram

---

(<http://people.f4.htw-berlin.de/~weberwu/info2/labs/Exer2.shtml>)

- 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.**

At first we create a new Java project. Now we create a new class FileTasks.

Then we create a new method readFromFileAndPrint(), which should fulfill the tasks given in the question:

```
public static void readFromFileAndPrint(File file) {
    try {
        FileInputStream fis = new FileInputStream(file);
        int current;

        while (fis.available() > 0) {
            current = fis.read();
            System.out.print(current);
        }

        fis.close();
    } catch (IOException e) {}
}
```

A FileInputStream fis reads the given file and prints each character as long as fis finds some. We test everything: It works!

- 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?**

Now we create a new method writeStringToFile():

```
public static void writeStringToFile(String text, File file) {
    try {
        BufferedWriter bw = new BufferedWriter(new FileWriter(file));
        bw.write(text);
        bw.close();
    } catch (IOException io) {}
}
```

A `BufferedWriter` `bw` writes an inputted `String` text to the `FileWriter`, which opens the file. After this `bw` gets closed. If something goes wrong, a `IOException` gets caught.

Now, to create a new file we simply have to call the method `createNewFile()` of the file class, which is what we did in the new method `createFile()`:

```
public static void createFile(String path) {  
    File newFile = new File(path);  
    try {  
        if(newFile.createNewFile() == false)  
            System.out.println("File already exists.");  
    } catch(IOException io) {}  
}
```

We test our new methods and everything works fine.

**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.**

In the new method `characterFrequenciesToFile()` we first read from a file char by char, as we already did before. Now to the tricky part:

We create an array `ascii` with 126 Integers. Each index of the array represents a char in the ASCII table (for example: A = 65). Now if an A gets read, we count index 65 one up.

Now a for-loop builds the output `String`, which contains the counted chars. Method `writeStringToFile()` then writes the output `String` to a new file `frequency.txt`.

```
public static void characterFrequenciesToFile(File readFile, File printFile, boolean histogram)
{
    try {
        FileInputStream fis = new FileInputStream(readFile);
        int ascii[] = new int[126];
        String fileContent = "";

        //Count all the characters and save the result to the array.
        while (fis.available() > 0) {
            int current = fis.read();
            ascii[current]++;
        }

        if(!histogram) {
            //Make a String out of the results
            for(int i = 0; i < ascii.length; i++) {
                if(ascii[i] > 0 && i != 10) {
                    fileContent = fileContent + "" + (char) i + ": " + ascii[i] + "\n";
                }
            }
        }
        else {
            //Make a String out of the results as a histogram
            for(int i = 0; i < ascii.length; i++) {
                if(ascii[i] > 0 && i != 10) {
                    fileContent = fileContent + "" + (char) i + " : " + createStars(ascii[i]) + "\n";
                }
            }
        }

        writeStringToFile(fileContent, printFile);

        fis.close();
    } catch (IOException e) {}
}
```

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 : *****
```

Everything works exactly the same way as in task 3. Only the output String has to be modified with a method createStars():

```
private static String createStars(int amount)
{
    String stars = "";
    for(int i = 0; i < amount; i++)
        stars = stars + "*";

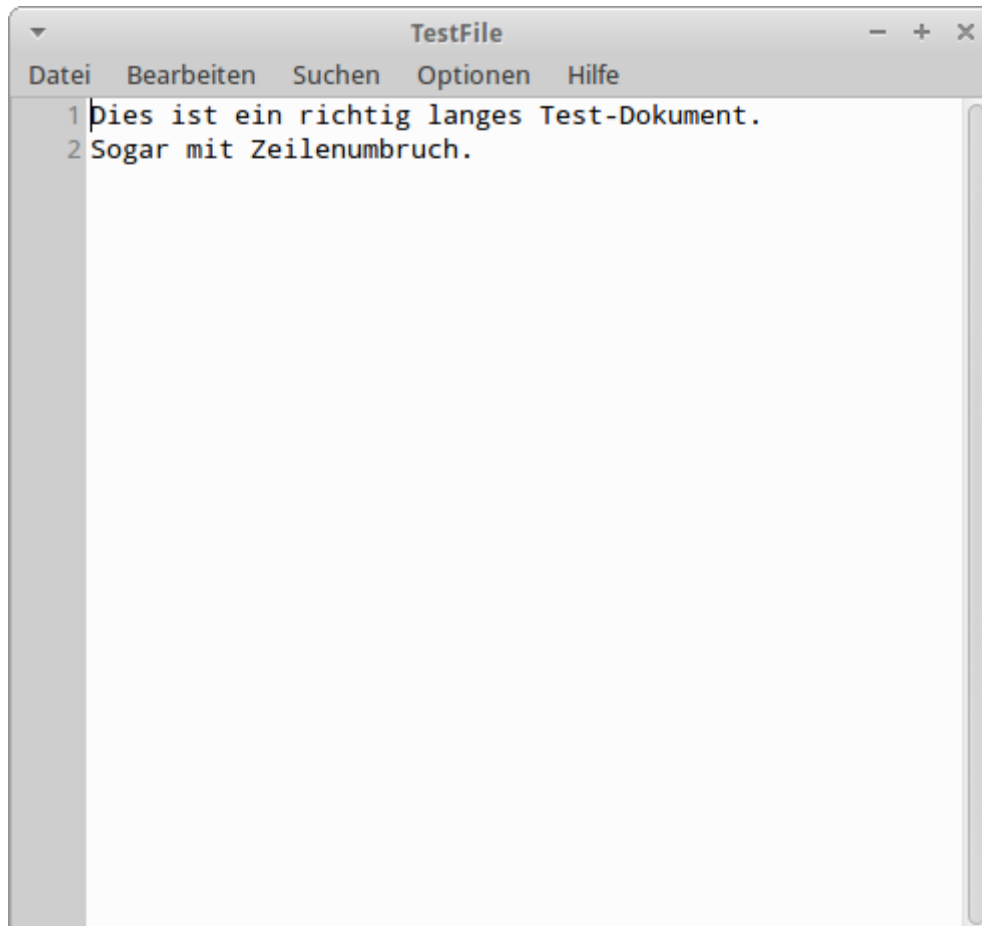
    return stars;
}
```

This method prints a number of stars instead only the number.

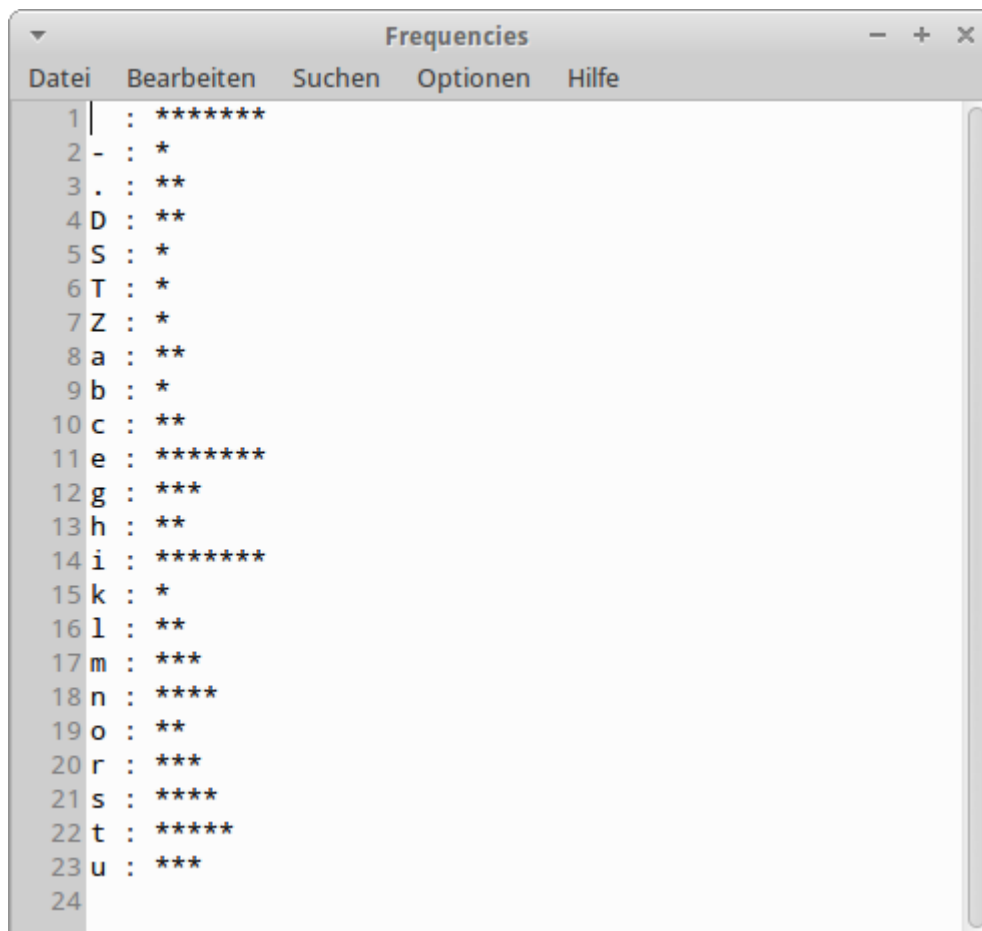
The main-method calls the method characterFrequenciesToFile() to start the program:

```
public static void main(String[] args) {  
    File testFile = new File("TestFile");  
  
    characterFrequenciesToFile(testFile, new File("Frequencies"), true);  
}
```

Now we test the code with a file named TestFile.txt:



Our results look like this:



| Datei | Bearbeiten | Suchen | Optionen | Hilfe |
|-------|------------|--------|----------|-------|
| 1     |            | :      | *****    |       |
| 2     | -          | :      | *        |       |
| 3     | .          | :      | **       |       |
| 4     | D          | :      | **       |       |
| 5     | S          | :      | *        |       |
| 6     | T          | :      | *        |       |
| 7     | Z          | :      | *        |       |
| 8     | a          | :      | **       |       |
| 9     | b          | :      | *        |       |
| 10    | c          | :      | **       |       |
| 11    | e          | :      | *****    |       |
| 12    | g          | :      | ***      |       |
| 13    | h          | :      | **       |       |
| 14    | i          | :      | *****    |       |
| 15    | k          | :      | *        |       |
| 16    | l          | :      | **       |       |
| 17    | m          | :      | ***      |       |
| 18    | n          | :      | ****     |       |
| 19    | o          | :      | **       |       |
| 20    | r          | :      | ***      |       |
| 21    | s          | :      | ****     |       |
| 22    | t          | :      | *****    |       |
| 23    | u          | :      | ***      |       |
| 24    |            |        |          |       |

The different chars get counted and everything gets printed into the new file Frequencies.txt!

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

The complexity expresses the growth rate as the number of items investigated increases. We use two nested for-loops, so the complexity of our algorithm is quadratic ( $N^2$ ).

We worked with networking in Java for the first time, which was a complete new experience for both of us. So it took us some time to understand how the two classes communicate with each other.

All in all the lab took us round about 6 hours of work.

## Source code:

### Class FileTasks:

```
import java.io.*;

public class FileTasks {

    public static void main(String[] args) {
        File testFile = new File("TestFile");

        characterFrequenciesToFile(testFile, new File("Frequencies"), true);
    }

    /**
     * Counts the appearance (frequencies) of characters in a certain file and
     * prints the results to another one.
     * @param readFile : The file object of the file we want to create a
     * frequency table from.
     * @param printFile : The file object we want to print the frequency table
     * to.
     * @param histogram : If true, the statistics will be shown as a histogram.
     */
    public static void characterFrequenciesToFile(File readFile, File printFile,
boolean histogram) {
        try {
            FileInputStream fis = new FileInputStream(readFile);
            int ascii[] = new int[126];
            String fileContent = "";

            //Count all the characters and save the result to the array.
            while (fis.available() > 0) {
                int current = fis.read();
                ascii[current]++;
            }

            if(!histogram) {
                //Make a String out of the results
                for(int i = 0; i < ascii.length; i++) {
                    if(ascii[i] > 0 && i != 10) {
```

```

                                fileContent = fileContent + "" + (char) i +
": " + ascii[i] + "\n";
                                }
                                }else {
                                //Make a String out of the results as a histogram
                                for(int i = 0; i < ascii.length; i++) {
                                    if(ascii[i] > 0 && i != 10) {
                                        fileContent = fileContent + "" + (char) i +
" : " + createStars(ascii[i]) + "\n";
                                    }
                                }
                                }

                                writeStringToFile(fileContent, printFile);

                                fis.close();
                                } catch (IOException e) {}
                                }

/**
 * Reads and prints the content of a file.
 * @param file : The object of the file.
 */
public static void readFromFileAndPrint(File file) {
    try {
        FileInputStream fis = new FileInputStream(file);
        int current;

        while (fis.available() > 0) {
            current = fis.read();
            System.out.print(current);
        }

        fis.close();
    } catch (IOException e) {}
}

/**
 * Prints a single character to a certain file.
 * @param c : The character to be written.
 * @param file : The file object of the file.
 */
public static void writeCharToFile(char c, File file) {
    try {
        FileOutputStream fos = new FileOutputStream(file);
        fos.write(c);
        fos.close();
    }catch(IOException io) {}
}

/**
 * Writes text into a File.
 * @param text : The text that should be written.
 * @param file : The object of the file.
 */
public static void writeStringToFile(String text, File file) {
    try {
        BufferedWriter bw = new BufferedWriter(new FileWriter(file));
        bw.write(text);
        bw.close();
    }catch(IOException io) {}
}

/**
```

```
    * Creates a File to a certain path.
    * @param path : The complete path including file name.
    */
    public static void createFile(String path) {
        File newFile = new File(path);
        try {
            if(newFile.createNewFile() == false)
                System.out.println("File already exists.");

        }catch(IOException io) {}
    }

    /**
    * Creates a String with a certain amount of '*' (stars) and returns it
    * @param amount : The amount of stars
    */
    private static String createStars(int amount) {
        String stars = "";
        for(int i = 0; i < amount; i++)
            stars = stars + "*";

        return stars;
    }
}
```