Overview

A program to count the characters in an ASCII file and display them in another ASCII file named by the user.

Introduction

This document holds the technical design of the CSCI312\_A2\_Schmidt program. The program is intended to process ASCII files and produce another ASCII file that shows the character frequencies of the original file in a fast and efficient method using a LinkedList.

## Scope

To count all the characters in a given ASCII file using a LinkedList.

To produce a new ASCII file with the produced results.

To be fast an efficient.

### PROCESSING

The user enters an ASCII file address and a desired name for the soon-to-be created ASCII file.

The program then creates a LinkedList of type CharacterFrequency. For every character in the ASCII file, the program creates a CharacterFrequency instance object with that character and a frequency of 1. Then for every CharacterFrequency object in the LinkedList, the program runs an equality override that checks if CharacterFrequency 1 is the same as CharacterFrequency 2. If it returns true, it increments CharacterFrequency 2, breaks the loop, and the garbage collector eats CharacterFrequency 1.

After going through all the characters, the program then writes all the toStrings() onto a separate ASCII file.

### DATA

CharacterFrequency

Object, m\_Ch = byte and m\_Frequency integer.

The byte is the character the CharacterFrequency object represents; the integer is how many times that character was in the ASCII file.

m\_Ch = all possible characters, m\_Frequency > 0

Instance variables: m\_Ch = character used when instanced, m\_Frequency = 1.

### COMPONENTS

CSCI312\_A2\_Schmidt

* Holds the static main and runs the program through the process.
* Also holds static method, ProcessWithArray, to hold code for performing this with an array.

CharacterFrequency

* Data class needed by the program.

|  |
| --- |
| **CharacterFrequency** |
| -ch : char  -frequency : int |
| +getCharacter() : char  +setCharacter(in character : char)  +getFrequency() : int  +setFrequency(in frequency : int)  +increment()  +Equals(in CharacterFrequency) : bool  +toString() : string |

### TESTING

|  |  |  |
| --- | --- | --- |
| Scenario | Description | Pass/Fail |
| 1st wap.txt run | Process wap.txt | Fail |
| 2nd wap.txt run | Process wap.txt | Pass |

Scenario #1- wap.txt test

|  |  |  |
| --- | --- | --- |
| Step | Description | Input/Output |
| 1. | Enter wap.txt address | Input: .txt file to be counted |
| 2. | Enter desired file name | Output: .txt file with countings |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| EXPECTED OUTPUT | | To produce another .txt file with the results. |
| ACTUAL OUTPUT | | The program didn’t increment, every toString() had a frequency of 1. |
| RESULTS – The increment method for CharacterFrequency 2 was being ran from inside the equals method of CharacterFrequency 1 and was not able to reach the code for CharacterFrequency 2 since it was a different instance. The increment method was moved to the static main. | | FAIL |

Scenario #2 – wap.txt test

##### 

|  |  |  |
| --- | --- | --- |
| Step | Description | Input/Output |
| 1. | Enter wap.txt address | Input: .txt file to be counted |
| 2. | Enter desired file name | Output: .txt file with countings |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| EXPECTED OUTPUT | | To produce another .txt file with the results. |
| ACTUAL OUTPUT | | Produced another .txt file with the results. |
| RESULTS – The program works almost as well and as fast as using arrays. Maybe less than a second slower. Overall, the LinkedList method may have worked better because instead of allocating space for 255 objects like the Array method did, this only creates space for the objects needed thus not wasting as many system resources. | | PASS |