Joseph Kelley

Table of Contents

Introduction

This assignment was one of the most difficult assignments I have ever worked on. I started it when the assignment became available and just finished it two hours before the due date.

Expectations:

Single Server that functions as a word count:
I created the Word Count Server that sends data to Cashe Server for processing

Must Be Written in Java:
I wrote this in Java

Design of Cashe Service:
I created a Cashe service as a file that only allowed the nulls to be changed. This allows for a max amount of files to be used, set a limit and have the cashe be full if too many files are there. I chose to use the Cashe service as a text file as it made sense to me and the file kind of reminded me of the database that I used at work.

Each Client Must Interface with your Word Count Server:

The client is looking at the port number that I have used looking for the word count server to say something and return it to the user. I also included a loop for the user to return multiple lines as the client will only spit one at a time. As I said in my code I feel like there's an easier way to do this but I was unable to figure it out.

Users Send Text File to the server and chose what information the server returns:

When the user sends information through the terminal that information is automatically changed into a text file.
(I was a little confused about this so I asked my friend and he said that it wouldn't make sense to submit an actual text file to the account and he assumed you were thinking of creating a text file with the user input. If you wanted an actual text file then a user entered a command of what to do that would have been so much easier and I hate my friend.)

The Server Will track totals for all inputs

After the user enters an input the server will spit back the information requested from the user then it will also total the lines, the characters and the words that the user has sent for processing.

Client Java Application that will compile and run from the command line

I created it and tested my code working from eclipse with a preset port and host and cashe size. This made it easier to run my code rather than the hassle of typing in everything. I then checked and my code runs from the command line.

Code Must Readable

This is not a requirement that you had but I commented my code very thoroughly allowing you to be able to see everything that I was trying to do with my code and I felt that it would help.

Design of Files

My code was designed with three main files Client.Java, ChangeLineInFile.Java, CasheServer.Java and WordCountServer.java. I also included an instructions.txt file and a closeFile.txt.

Client.Java

This is my most basic class and was based pretty heavily on EchoClient.java which you have provided. This class is my output window class that prints everything that WordCountServer.java returns and Outputs. The only difference that I changed was I created a variable called nLines that was the first line returned by WordCountServer and that was the amount of lines that I was going to send to the Client Class. For Example if my expected Output was hi. The Client class would receive 1 \n and hi. This made it so I was able to print the correct amount of lines.

ChangeLineInFile.Java

This is the File I copied from GeeksforGeeks or StackOverFlow. I can not remember as I have used it so many times when trying to append a file as it saves so much time. It takes the inputs of what line you would like to change. What file you want to change and also what you would like the new line to be. I still believe this should be a built-in method when working with Java or any programming language.

CasheServer.Java

This is the File that does the meat of my code and runs the assignment. This file receives an input of the user's instructions as a txt file and returns what the user has requested. For example if a user types Read 3.txt. This class will return the file read out. This class does everything that is required.

(I am not sure if I am suppose to list every method that it has I am still not good with design documents)

WordCountServer.java

This is the main server File that the Client.Java is looking for. The file runs my entire project and calls everything that is needed to work on the file. It first calls the method createCashe and is given the size of the cash as a parameter. If the cashe is not big enough to support the files already there the method will create a txt File called closeFile. The file then checks if the file close File is there. If the file is there it deletes the file and exits the program.

If the cashe size the user entered is big enough the file then creates the cashe and will fill null as the empty lines for example if a user enters 4 for the cashe and three is 1 text file already there it will return.
Textfile.txt
null
null
null

The stuff above only runs at start up while the stuff below will run every time a user enters something into the client.

The file then calls the method writeFile and is given a parameter of what the user has written and writes it in Instructions.txt.

The file then calls the method words with a parameter of the instruction file created. This updates the line, the characters and the words of what the user has written.

The file then checks to see if there are more than two words if there is not the file returns a must insert more than one word error and then adds up the inputs and returns the error as well as updates the variables line and word and character.

If the file passes the check it calls the CasheServer.Java file and runs the users request and returns if it is successful or it is not successful.

Scope/Overview

Created Design

The first thing I did when looking at this assignment was honestly freak out. I was scared and was still confused on the client host relationship.

After my freak out was done I first figured designed each of the 5 steps separately

1. Count lines words and characters from a file
2. Store File
3. Update File
4. Remove File
5. Read File

After that I then decided what to create as my storage for the final couple tasks.

I decided that my database would be a textfile.

Then Created

1. Get File Names
2. Remove File algorithm to remove also from text file

I then made everything work like that then I started working on connecting everything

After that was done I put it over to a server and set that up. That was the hardest part.

Then I created the file so the cashe was able to set at a certain number. I tried booleans but that did not work so I eventually created a closeFile.

After that was all working I then realized that I was not passing the file to the processing class So I redid my inputs and redid a lot of code. I then bug checked and read the requirements a million times hoping I was not missing anything.

Assumptions

Working through this assignment I made a couple of assumptions, the first being a user enters something into the console and then it is converted into a text file. I did not assume the user entering the name of a text file then what to do with the text file. For example, given update I assumed.

Update NameofFile What you wanna add.

I did not assume

NameofFile Update What you wanna type.

I also assumed the database would be a text file and the size would be the amount of files I did not assume actually the size of the database. For example

10 = 10 Files

Not 10 mb.

I also assumed that CasheServer is directly called without looking at a host like the way Client and WordCountServer are.

I also assumed that the files are supposed to stay in the cashe and not be deleted each time the system starts.

Criteria Establishment
This project should work if the instructions for user input are followed as given in the help me section.

Criteria Validation
The project has built in criteria validation as if the user inputs something incorrectly the file should crash. As if the user does not enter the text file in the correct place the auto file not found check will be in place. It is also validated by the fact that if the user enters something incorrectly there is no guess as it is if statements not LUIS. In my experience with LUIS it picks the best idea.

Procedures
Procedures to run the file are included in the README folder and also here!
To Run in Eclipse First Start the WordCountServer File.
Then Start Client.
This is the way I did it as it is the easiest.

To Run in the command line first comment out
hostName and portNumber in Client
portNumber and casheSize in Word Count Server
Then uncomment the args version of them

client(hostName,PortNumber)
WordCountServer(PortNumber,CasheSize)
Cashe size must be greater than 1 at the beginning because there is 1 file in the storage.

Things you can ask

To Reach Store File Name: Type What you would like to type then Save and Name
For Example- What I want to type Save RandomFileName

To Reach Update File: Type update then the file you would like to update then what you would like to add

For Example- Update FileName What I want to type

To Reach Delete File: Type the file you would like to remove
For Example- Remove RandomFileName");

To Reach All File Names in cashe: Type get or list files
For Example- Get Files or List Files");

To Reach Read Files: Type read then the file
For Example- Read Random FileName

For Help- Type (Help Me)

If you need any more help reach me at (12072812951) or JK1582@mynsu.nova.edu
I worked hard on this!

Plans for Data Analysis
I could include a text file that all the information that the processing layer did not understand was added to and we could see what should be added in the future. It could also help in including common misspellings.

Data Design

Input Data

1. Text File (That will be updated)
2. Command to Use (Store Remove etc)
3. Stuff to Add (Information added to new or old File)

ArrayList, Array

Used to store data

File

Used when accessing files

Processing Files:

This includes Scanner, BufferedWriter and so on

Socket

This is used to connect to Socket and read what is sent

Output Data:

This is all the data that is outputted to the user such as Success Messages, Files and other things

Initialization:

This requires the same port number for Client and WordCountServer and a hostName for Client and a cashe size for WordClientServer.

Termination:

This only terminates when the user decides to terminate or if the cashe is not big enough at startup. I did not include a quit function.

Class Names

1. Client
2. Word CountServer
3. ChangeLineInFile
4. CasheServer

Method Names/MethodsCalled

1. Main
2. words
3. writeFile
4. createCashe
5. changeALineInATextFile

6. numberOfLinesInFile
7. turnFileIntoArrayOfStrings
8. editLineinContent
9. writeToFile
10. readFile
11. Help
12. updateFileName
13. storeFileName
14. readFileName
15. getFileNames
16. removeFileName
17. createCashe

Files Changed

1. Text files Created
2. instructions
3. cashe

Interface Design

SampleOutPut

```
help me
Insructions:
To Reach Store File Name: Type  What you would like to type then Save and Name
For Example- What I want to type Save RandomFileName
To Reach Update File: Type update then the file you would like to update then what you would like to add
For Example- Update Random FileName What I want to type
To Reach Delete File: Type  the file you would like to remove
For Example- Remove RandomFileName
To Reach All File Names in cashe: Type get or list files
For Example- Get Files or List Files
To Reach Read Files: Type read then the file
For Example- Read Random FileName
For Help- Type (help me)
Number of Words Inputed: 2
Number of Characters Inputed: 8
Number of Lines Inputed: 1
list files
File Names Are: counting.txt
Number of Words Inputed: 4
Number of Characters Inputed: 19
Number of Lines Inputed: 2
1 2 3 save test
File has been saved
Number of Words Inputed: 9
Number of Characters Inputed: 35
Number of Lines Inputed: 3
1 2 3 save test
File already exists under that name and is unable to save it please repeat with a new name
Number of Words Inputed: 14
Number of Characters Inputed: 51
Number of Lines Inputed: 4
read test
File: test.txt
1 2 3
Number of Words Inputed: 16
Number of Characters Inputed: 61
Number of Lines Inputed: 5
update test 4 5 6
File has been updated
Number of Words Inputed: 21
Number of Characters Inputed: 79
Number of Lines Inputed: 6
read test
File: test.txt
1 2 3
4 5 6
Number of Words Inputed: 23
Number of Characters Inputed: 89
Number of Lines Inputed: 7
```

```
update no a a s
File does not exist:
Changes Have Not Been Made Please Try again
Number of Words Inputed: 28
Number of Characters Inputed: 105
Number of Lines Inputed: 8
list files
File Names Are: counting.txt
test.txt
Number of Words Inputed: 30
Number of Characters Inputed: 116
Number of Lines Inputed: 9
save 1
File has been saved
Number of Words Inputed: 32
Number of Characters Inputed: 123
Number of Lines Inputed: 10
save 2
File has been saved
Number of Words Inputed: 34
Number of Characters Inputed: 130
Number of Lines Inputed: 11
save 3
File has been saved
Number of Words Inputed: 36
Number of Characters Inputed: 137
Number of Lines Inputed: 12
save 4
File has been saved
Number of Words Inputed: 38
Number of Characters Inputed: 144
Number of Lines Inputed: 13
save 5
File has been saved
Number of Words Inputed: 40
Number of Characters Inputed: 151
Number of Lines Inputed: 14
save 6
File has been saved
Number of Words Inputed: 42
Number of Characters Inputed: 158
Number of Lines Inputed: 15
save 7
File has been saved
Number of Words Inputed: 44
Number of Characters Inputed: 165
Number of Lines Inputed: 16
save 8
Cashe is Full Unable to Save it
Number of Words Inputed: 46
Number of Characters Inputed: 172
Number of Lines Inputed: 17
list files
File Names Are: counting.txt
```

```
test.txt
1.txt
2.txt
3.txt
4.txt
5.txt
6.txt
7.txt
Number of Words Inputed: 48
Number of Characters Inputed: 183
Number of Lines Inputed: 18
remove 7
File Has Been Deleted and Removed from Cashe
Number of Words Inputed: 50
Number of Characters Inputed: 192
Number of Lines Inputed: 19
list files
File Names Are: counting.txt
test.txt
1.txt
2.txt
3.txt
4.txt
5.txt
6.txt
Number of Words Inputed: 52
Number of Characters Inputed: 203
Number of Lines Inputed: 20
remove 7
File Not Found
Number of Words Inputed: 54
Number of Characters Inputed: 212
Number of Lines Inputed: 21
read counting
File: counting.txt
1
2
Number of Words Inputed: 56
Number of Characters Inputed: 226
Number of Lines Inputed: 22
```