CS2101

MINI PROJECT 1

November 4, 2015

Student ID: 150002227
St. Andrews University
Department of Computer Science

INTRODUCTION

The objective of the project was to create a file sharing system the allowed the user to share to and download from any other people using the application.

FUNCTIONALITY OVERVIEW

FileSharingMain
In my main method I first declared the final variables which both sides of my program, the server and the clients sides, would rely on to communicate. I also created a monitor object to use to make the method wait until it is called to continue. In the main method I set the directories specified to be final as i am not allowing them to be changed at any point. I then checked the directories existed and were directories as they are crucial to all parts of the program, I ended the method if they were not. I created and started a client thread and a server thread, the use of threads is to allow the program to send requested files and download files simultaneously. After they start, I call a waiter method, this stops the main method continuing but still allows the threads it has created to run until the unlockWaiter method is called. Once the unlock method is called I shut both the server and client threads down and end the program.

ServerThread
In the ServerThread constructor I assign all the parameter variables to be thread local variables. When the thread is started I create and start a multicast server thread. I then set up the server socket. I check the server has been created and if so I enter a loop to accept clients. The reason for a loop is to allow the program to accept multiple clients simultaneously. In the loop the server accepts the client connection and creates and starts an associated clientProcessingThread to do as its name suggests before returning to waiting to accept the next client. When the loop is closed I stop the multicast server thread and close the server socket.

ClientProcessingThread
In the clientProcessingThread constructor I again assign the parameter variables to be thread local variables. I then set up the input and output streams I would use throughout the thread. It calls another method to receive the clients request, check it is valid and to call a method to send the correct response back. I made it so that this is the only place the server will receive an input and that the server would return to waiting for a client request once the message

has been sent. This is so that I could keep the server stateless, if in the future I made it so that the client would be able to connect to multiple servers I wouldn't have to keep track at which state the client is in at each server(this is hard to do I'd imagine).
The server can either send a list of the files it has or send a file itself. For the list files the server simply sends each file name one by one then indicates it has reached the end of the list. To the deliver the file to download the client sends the name of the file when it is sending the request as I don't want to ask separately for its name, that would mean the server isn't stateless. I check the file is in the directory and, if it is, I send it back in byte form, this allows me to send files of any type.

ClientThread
I again, assign the parameters to the thread variables in the constructor. I then go to what is effectively the main menu, I list the possible things the client can do, I take the user input, check it is valid and call the method associated with it. The client does not need to be stateless as it doesn't need to send any data to the server beyond requests for data and does most of the processing of that data.
The first task the server can do is list the servers that are online that you can receive files from. This method creates a mulicastClientThread, obtains a list of available host names and displays them.
The second task it can do is connect to a server. Once connected you are brought to another menu where you can choose to list the files on that server or to download a file. Both send the relevant request to the server. The list files simply prints out the files it receives. The receive file asks for the name of the file and then concatenates it to the request before sending it so it satisfies the protocol. The file, if available is then downloaded as as bytes to the specified directory. The third task it can do is search for a file across all the available servers. To do this is sends a request to list all the active servers and then connects to them one by one and requests it sends them a list of their files. It searches through the files and if the file matches the one it is looking for it ass the server name to a list, when finished it returns the list.
Finally it can list the downloaded files. To do this it gets a list of files from the download directory and displays them.

MulticastServerThread
When the multcast servers constructor is called it first assigns parameters to the thread variables. In the run method it sets up a multicast server socket and creates a data gram message to send, the message is created outside of the coming loop as it is going to be a consistent

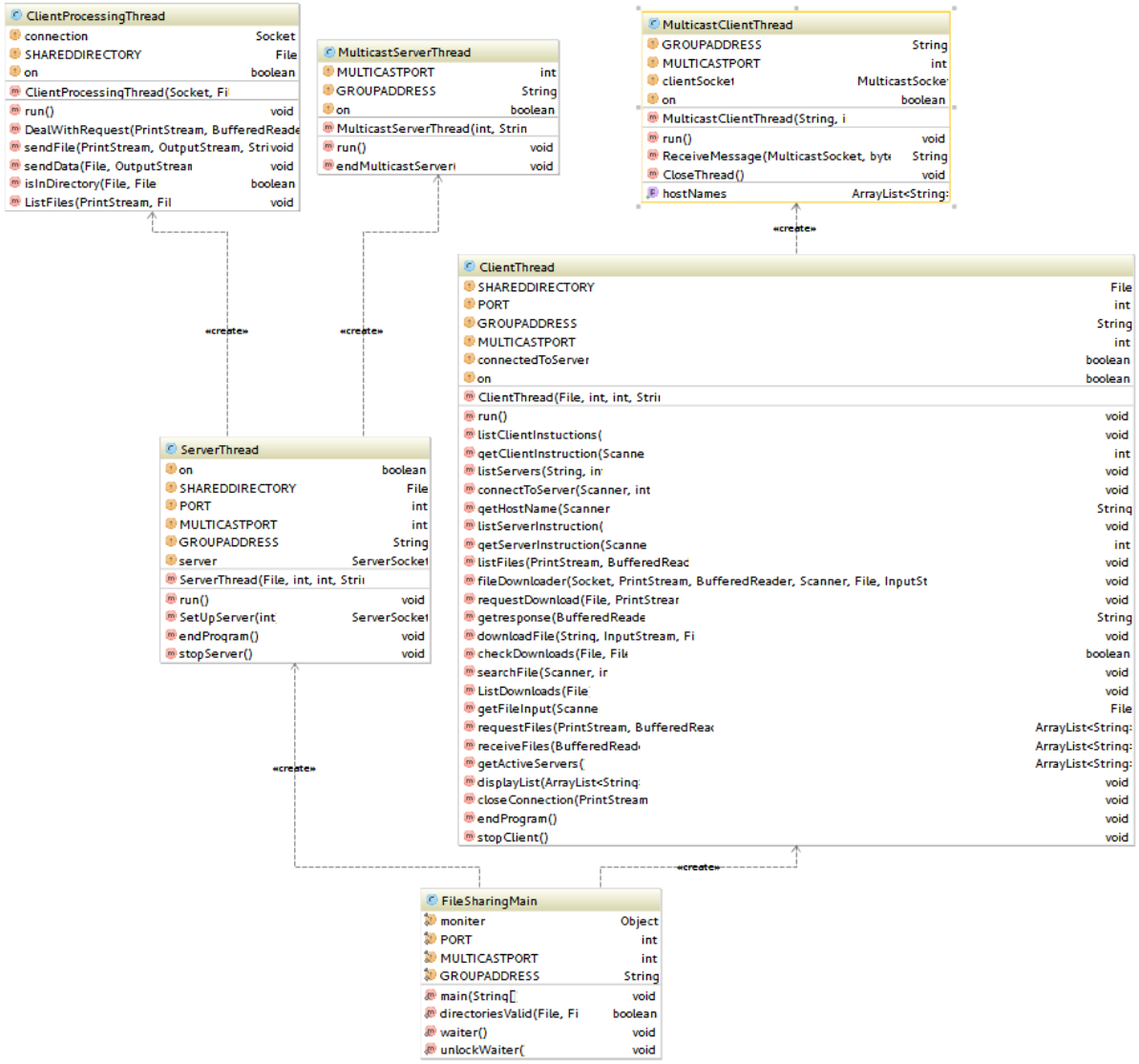
message. The message is the name of the server. It then enters a loop where, while the server is active, every 0.5 seconds it sends the message out to all the listening client sockets.

MulticastClientThread
As ever,when the multcast client constructor is called it first assigns parameters to the thread variables. In the run method, I create a list where I'm going to store the host names. Then while it is on, every time it receives a host name it would check it wasn't already in the list and if not it would add it to it. Every time I called it from the client method I made the client server sleep for 1 second while it collected the host names before assigning a variable to reference the list and closing the server. The times I pocked for the multicast server and client were quite arbitrary and based off the examples I see in the CS2003 lectures, it is very possible there are better times that could be chosen but they worked well in testing.

MY WORK

Most of the work was my own. The parts I adapted for my own use from either the class lectures or external sources were the socket connection, the multithread socket connection and the waiter method in the main menu. The rest of the project was mainly my own work (although if stuck I may have looked up small hints, I cant remember exactly when).

UML CLASS DIAGRAM



PROTOCOL

If the server receives:

- SEND-FILE + file name ==> Server sends file of that name over network
- LIST-FILES ==> Sends a list of the the files in the shared directory
- CLOSE-CONNECTION ==> Closes the connection to the client

If the client receives:

- FILENAME-RECEIVED ==> The client knows the server is about to send the file
- FILENAME-NOT-FOUND ==> The client knows the server has not found a file of that name at the server
- END-OF-FILELIST ==> The client knows there are not more file names to come

FEATURES OF MY CODE

I tried to factor out the functionality on the client side into small pure components wherever possible. I'm not sure exactly what the limits of that are, for example I'm not sure if having side-effects (such as System.err.println) when error handling affects the purity of the component. Having pure functions allows you to reason through programs with greater ease. Also it made it easier for me to create new functions for my client using the functions I had already defined. It is possible this can optimize the program when compiling, especially useful in functional languages, although I don't think this applies to my program.
I tried to use error handling to make it so that the client or the server the file sharing system is connected to could disconnected abruptly and the program would continue running. I achieved this although I was not able to be sure what happens I the server or file cuts out in the middle of sending a file. But if my analysis of my code is correct, if a client disconnects it will throw a I/O exception, print there was a problem and continue. If a server disconnects the input will fall to 0 and so the file will stop downloading as if the file had finished, you will have a half downloaded, possibly corrupted file but the program will go back to the file sharing menu.
The use of threads in my code was very useful, firstly it allowed me to run the client, the server and the multithread server all at the same time. This allows me to have an independent server and client as if there wasn't two threads you would only be able to do one task, download or upload, at a time. This would reduce the functionality of my program and I'm not sure it would meet the specification. It was also useful in collecting the names of servers as after opening the multicast client thread I was able to make the client class wait when it collected names before accessing the list and closing it again.
I have a very big client class, I could possibly have made it into multiple classes but many of the tasks relied on the same functions so I wanted to keep them in the same class and I didn't feel it made it a problem to follow. This did mean though my code and list of methods was very long, so I made divisions for the methods exclusive each task and a place to store all the methods used by more than one task.I used javadoc to separate them, I feel my method for

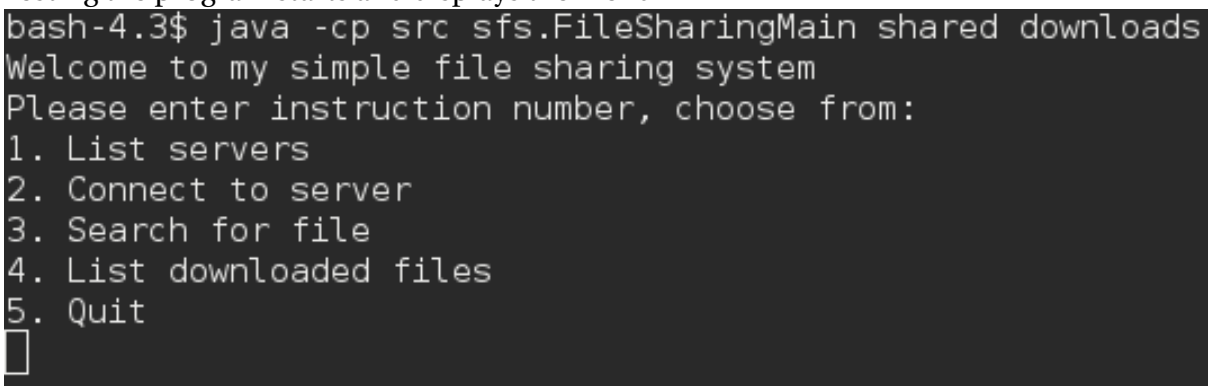
partitioning them is quite clear and easy to follow.
I feel there is a better way to lay out my client class, if you weren't going to already, could you please comment on its structure.

TESTING

I'm very sorry about the next section, I'm trying to use LaTex and I'm obviously not very good at formatting it.

TESTING BASIC FUNCTIONALITY

Testing the program starts and displays the menu



Testing servers are listed (this is the only computer running the program)

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
1
pc3-013-l.cs.st-andrews.ac.uk
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

Testing servers are listed when there a non-local servers

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
1
pc3-013-l.cs.st-andrews.ac.uk
pc3-004-l.cs.st-andrews.ac.uk
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

Testing the program can connect to a server and display the connected menu

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
2
Please enter the host name of the server:
pc3-013-l.cs.st-andrews.ac.uk
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
```

Testing it can connect to a non-local server

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
2
Please enter the host name of the server:
pc3-004-l.cs.st-andrews.ac.uk
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
```

Testing the program can list the files at the server

```
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
1
new-york-panorama.jpg
example.txt
main.txt
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
```

Testing the program can download text files

```
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
2
Please enter the file name:
example.txt
Finished downloading
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
```

Testing it can download other files

```
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
2
Please enter the file name:
new-york-panorama.jpg
Finished downloading
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
```

Testing it can download form a non-local server

```
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
2
Please enter the file name:
main.txt
Finished Downloading
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
```

Testing it can successfully disconnect from the server

```
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
3
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

Testing it successfully find a file and display host name of the server its at

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
3
Please enter the file name:
example.txt
pc3-013-l.cs.st-andrews.ac.uk
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

Testing it can find files at non-local servers

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
3
Please enter the file name:
new-york-panorama.jpg
pc3-013-l.cs.st-andrews.ac.uk
pc3-004-l.cs.st-andrews.ac.uk
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

Testing it can display the file that have been downloaded

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
4
example.txt
new-york-panorama.jpg
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

Testing you can exit the program

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
5
bash-4.3$
```

TESTING WRONG INPUT HANDLING

Checking what happens when you put an invalid input into the menu

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
gsrtsvb
Please enter number
542
Please enter valid number
```

Checking what happens when you put in a invalid server name

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
2
Please enter the host name of the server:
gaerh
Problem connecting to the server
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

Checking what happens when you enter an invalid input at the server menu

```
Please enter the host name of the server:
pc3-013-l.cs.st-andrews.ac.uk
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
rev
Please enter number
345
Please enter valid number
```

Checking what happens when you enter a invalid file to download name

```
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
2
Please enter the file name:
dgfpokg
Could not find file
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
```

Checking what happens when you enter a invalid file name (It doesn't print that it is at any servers)

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
3
Please enter the file name:
gs
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

Checking what happens when you enter a already downloaded file as the file to download (it doesn't print anything, it just goes back to the menu)

```
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
2
Please enter the file name:
example.txt
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
```

TESTING UNEXPECTED DISCONNECT HANDLING

Testing unexpected server disconnect

```
Please enter the host name of the server:
pc3-004-l.cs.st-andrews.ac.uk
Input server instruction
1. List Files
2. Download File
3. Disconnect from server
1
Server disconnect
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

Testing what happens when a client unexpectedly disconnects(here i have it printing out when it disconnected, it normally wouldn't)

```
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
Client disconnect
1
pc3-013-l.cs.st-andrews.ac.uk
Please enter instruction number, choose from:
1. List servers
2. Connect to server
3. Search for file
4. List downloaded files
5. Quit
```

IMPROVEMENTS

One small problem with my code is the protocol is not build into the program as intrinsically as I would like, I could possible have a place where I stored constants, which both the client and the server sides could access, which would define what the protocol is. When needed the server and client could access that to get the correct message to write across the server. This would guarantee the protocol is kept consistent on both sides.
There are areas where I could print out useful messages to the client, for example if the file was already found in the directory, explicitly say if no servers contained the file you were searching for, explicitly say if there were no files when you asked the server to list the files it contains.
I could also possibly check the format of the host name when entered, if there is a generic formula, and also let the client know that they can enter the IP address of the server they are trying to connect to.
I could allow the client to connect to multiple/all available servers as they come online. I would change the list files method to list all the files of all the servers and the send file to download file method to look through the servers and download it from the first server it finds it in.
I could stop allowing it to connect to its own server although I don't see this as a problem, I could possibly indicate it is there own.

EVALUATION

I feel I matched the basic specification to a very high standard. I have done all the extensions bar the GUI again to a good standard although my search method is not exactly as specified in the project instruction sheet. Given more time I would have liked to implement a GUI, but with no prior experience making them I felt it was a very hard extension to do. i would also have like to implement some of the improvements especially the multiple servers one although this would require a massive overhaul of my structure.