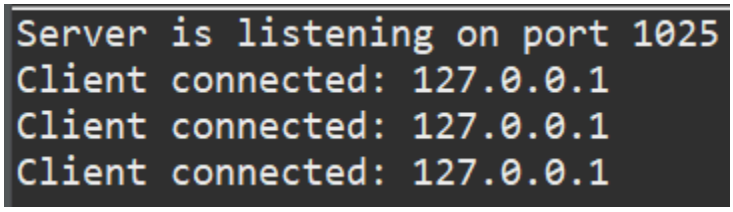
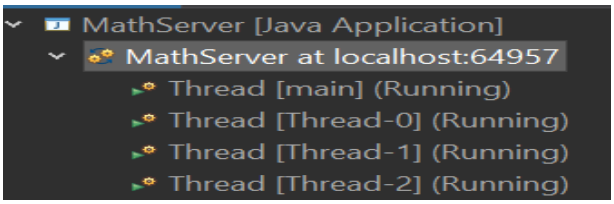
MathClient–MathServer File Transfer

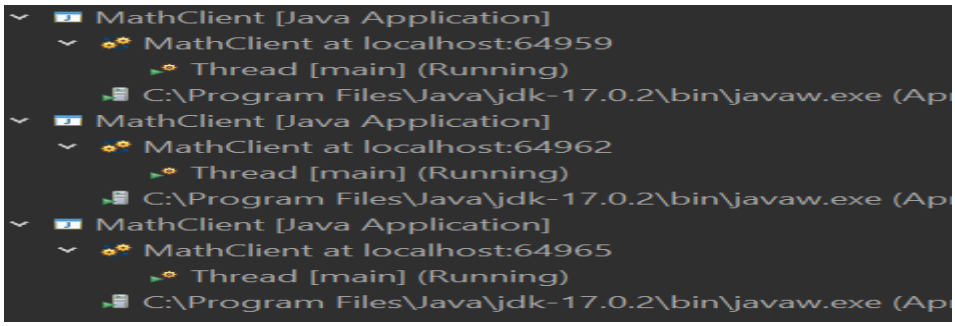
The program consists of two files named *MathServer.java* and *MathClient.java*, both of which need to be compiled and run. The program runs entirely on the user's local machine and uses a TCP socket to let the user upload files to, and download files from, the MathServer using the MathClient. The MathServer uses threads of the ClientHandler class to allow for multiple concurrent users to connect on the same port. The MathClient allows users to transfer files to and from the MathServer by allowing the user to input 1 of 4 commands into the command-line: List, Upload, Download, or Exit. Security was completely ignored because the program does not in any way connect to other machines, and thus the potential harm caused by a security vulnerability is minimal if it even exists. Communication between MathClient and ClientHandler is done using UTF encoded strings that are sent through data streams.

On Startup: MathServer

When MathServer is run, it creates a ServerSocket that listens on port 1025 and immediately accepts any client attempting to connect to the socket. After the connection, a thread of the ClientHandler class is created to handle all the requests of the client while the server continuously listens to the port for new clients, allowing for concurrency. MathServer also prints to the console, notifying the user of when the server is listening, when a client has connected, and when an error has occurred. The entirety of MathServer is executed within a try-catch block in order to easily deal with any errors, such as another server listening on that same port.







After the thread is created, it is started, and the ClientHandler is run by implementing the run() method of the Runnable interface. Within each thread, an input and output stream are created for server–client communication and a string containing instructions for the user is

written to the server-side output stream and sent to the client-side input stream. All strings in the input and output stream are UTF-encoded.

On Startup: MathClient

When MathClient is run, it attempts to connect to MathServer by connecting to port 1025 on the local machine. It also instantiates an input and output data stream for server–client communication, as well as a BufferedReader for receiving user input. MathClient is executed within a try-catch block in order to easily deal with any errors, such as MathServer not currently running. After creating the data streams, MathClient reads from the client-side input stream and prints it to the console, providing instructions to the user.

Upon providing instructions, MathClient uses a while-loop to continuously wait for the user to provide commands through the console. The commands are read using the previously instantiated BufferedReader. Accepted commands are: “List”, “Upload”, “Download”, and “Exit”. All commands are case-sensitive.

List Command

When the user enters “List” into the console, MathClient writes the word “List” to the client-side output stream. Then ClientHandler reads the string “List” from the server-side input stream, calls the getFiles() method, and then writes the string returned by the getFiles() method to the server-side output stream. Finally, MathClient reads the string from the client-side input stream and prints it to the console.

getFiles()

The getFiles() method of the MathServer class takes all the files in MathServer’s local directory, appends them all to a File array, transforms the File array into a string, and then returns the string.

Upload Command

When the user enters “Upload” followed by a filename into the console, MathClient writes to the client-side output stream. The string contains the name of the file–obtained using the parseFileName() method, the contents of the file–obtained using the getFileContent() method, and length of the file–obtained by applying the length() method to the getFileContent() method. ClientHandler then reads the string from the server-side input stream and passes it as a parameter to the uploadFile() method. If the uploadFile() method returns true, ClientHandler writes “Upload complete” to the server-side output stream. Then MathClient prints a string containing the filename and file contents to the console before finally printing “Upload Complete.”

If uploadFile() returns false, ClientHandler writes “File already exists” to the server-side output stream and MathClient prints “File already exists” to the console.

parseFileName()

The parseFileName() method of the MathServer class takes in a string as a parameter. If the string starts with the letter ‘D’, this indicates the string includes a file for downloading, and therefore the filename starts at the string’s 9th index. Otherwise, the string is presumed to contain a file for uploading, and therefore the filename starts at the string’s 7th index. parseFilename() then returns the filename contained in the string.

getFileContent()

The getFileContent() method of the MathServer class takes in a filename as a parameter. The method instantiates a FileReader and a string named “content”. The FileReader opens the file indicated by the filename and appends every character of that file to “content”. “content” is then returned.

uploadFile()

The uploadFile() method of the MathServer class takes in a string containing the information of a file to be written to MathServer’s directory. uploadFile() first checks whether the file is in the server directory or not using the fileExists() method. If fileExists() returns true, uploadFile() returns false. Otherwise, uploadFile() uses a FileWriter to write the contents of the file to the directory before returning true.

fileExists()

The fileExists() method of the MathServer class takes in a filename as a parameter. It creates a File array of all the files in MathServer’s directory and then iterates through the array to see if any of the filenames in the directory match the one given as a parameter. If a match is found, fileExists() returns true, else it returns false.

Download Command

When the user enters “Download” followed by a filename into the console, MathClient writes the user’s input to the client-side output stream. ClientHandler then checks if the file exists in the MathServer’s directory using the fileExists() and parseFileName() methods. If the file does not exist, ClientHandler writes “File not found” to the server-side output stream. MathClient then prints “File not found” to the console.

If the file does exist, ClientHandler uses the getFileContent() and parseFileName() methods to write the contents of the file to the server-side output stream. MathClient then creates a file with the same filename and contents as the file in the MathServer directory in MathClient’s

directory. MathClient then prints the contents of the file to the console, as well as a message saying “Download complete”.

Exit Command

When the user enters “Exit” into the console, MathClient terminates the data streams, the BufferedReader, and the socket. It also prints a message to the console informing the user that the connection has been terminated before ending execution, thus killing this particular ClientHandler thread.