Diego Sabaris Montes: S21859.

Points 1, 2, and 3 were done from the project “System ściągania i udostępniania plików TORrent” . **The protocol used is TCP.**

Description:

This is a host-to-host torrent program. When any of the scripts below is ran, two directories are created: D:\\TORent\_0 and D:\\TORent\_1 containing one .txt example file on each

The below scripts are compiling successfully having installed jdk1.8.0\_271 with the PATH updated accordingly in the windows environment variables. In other scenarios its behavior is unknown.

The script **0MainConsoleOverview.bat** runs two applications (app 0 and app 1) and shows the main menu. This menu is a console ready to take user input (Possible input: 1, 2 or 3, and are equivalent to running the below mentioned scripts).

The script **1PullFile.bat** will pull an example file by app 1 from app 0.

The script **2FilesandMD5infoExhange.bat** will exchange between the two applications, the files which are available on each directory along with their respective MD5 checksums.

The script **3PushFile.bat** will push an example file from app 1 to app 0.

I used the TCP protocol socket connection to exchange information between the two applications. Each app ( 0 or 1 ) is client and server in the same time. The client socket of the app 0 is connected to the server Socket of app 1 and vice versa. In each app, both client and server sides are constantly reading from a stream, each of them in a separate thread. In this way, the exchange of information can happen in both ways and without interruptions.

More details:

**This is how the pushFile method works: (Server.java, line 39. Parameters: String containing a file path, or a String =“exitexit”)**

If we are sending the other app “exitexit” we are closing the connection.

If we are sending a file path, then the following is happening:

App 0 (the sender in this example) sends the path file that is being transferred to app 1 (receiver) and then this latter saves the file name using .getFileName() method.

These are the instances of the classes that are implemented in the exact order from both ends connected via Socket: BufferWriter > OutputStreamWriter > OutputStream **>** InputStream > InputStreamReader > BufferReader.

After that, the sender writes the binary information from the file which is being sent into a byte array. The content of this latter arrives to the receiver, who will then use it to create a copy of that file in the destination folder, using the file name that was saved before. FileInputStream > OutputStream > InputStream **>** FileOutputStream .

Applicable both ways <->. (Meaning that the receiver can be a sender and vice versa)

**This is how the pullFile method works (Client.java, line 49, Parameters: String containing a file path, or a String =“exitexit”)**

String “exitexit” has the same functionality as described before.

On the other hand, if we are taking as a parameter a file path, the app 1(receiver) sends the file path that wants to download from app 0 (the sender). BufferWriter > OutputstreamWriter > OutputStream > InputStream > InputStreamReader > Bufferreader

Finally, the sender performs the “pushFile” method explained before.

Applicable both ways <->.

**This is how the checkFiles method works(Client.java, line 58, No parameters):**

Using the same principal mechanisms mentioned before, the app 0(receiver) asks for the list of files to app 1(the sender), and this latter will list the files it has on D:\\TORent\_1 into a .txt. which is sent through “pushFile” method to the receiver. This one will fetch the md5 chechsums and the whole content is then printed in the console of the receiver.

Applicable both ways <->.