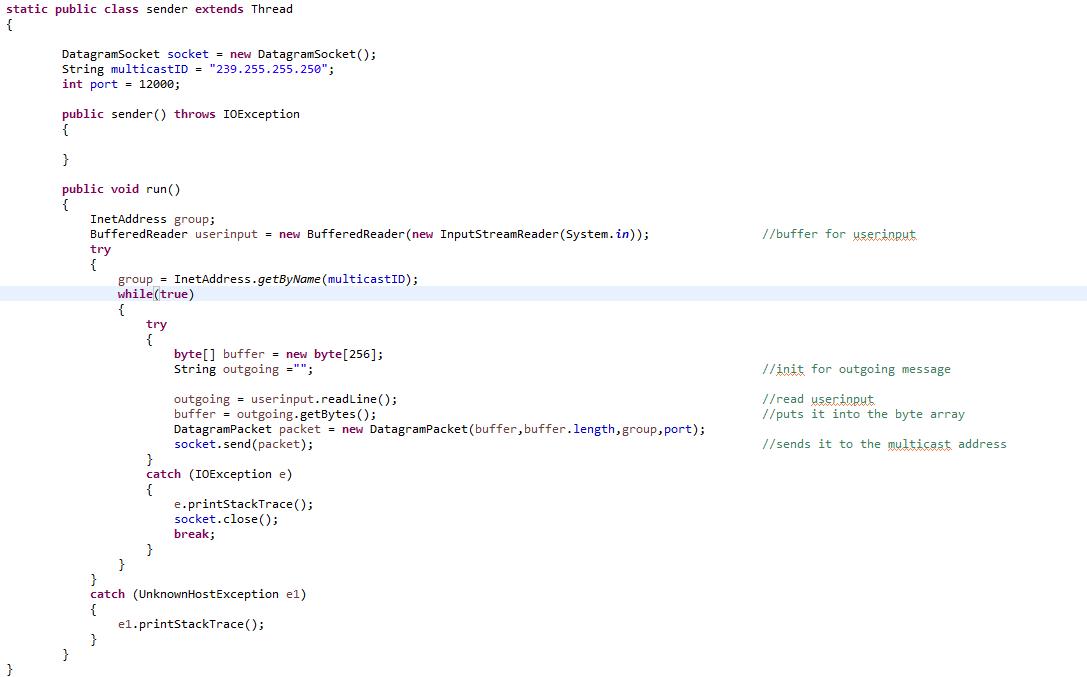
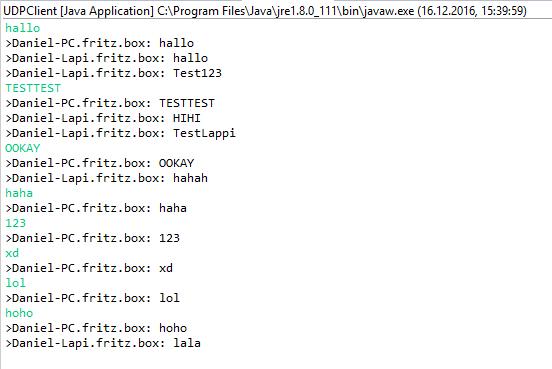
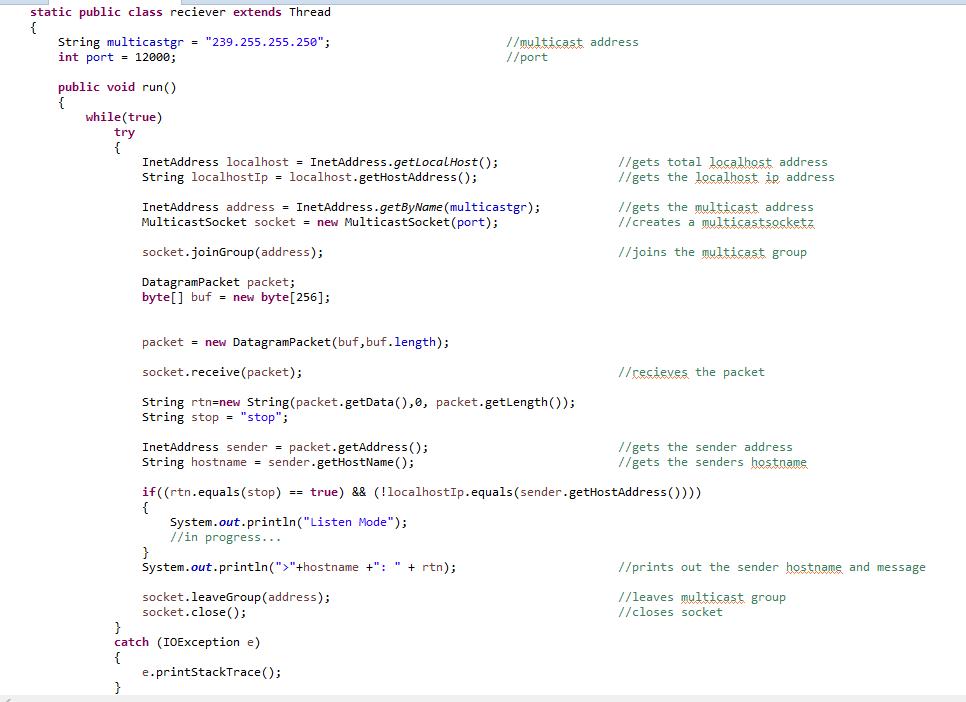
# Chat service

Information

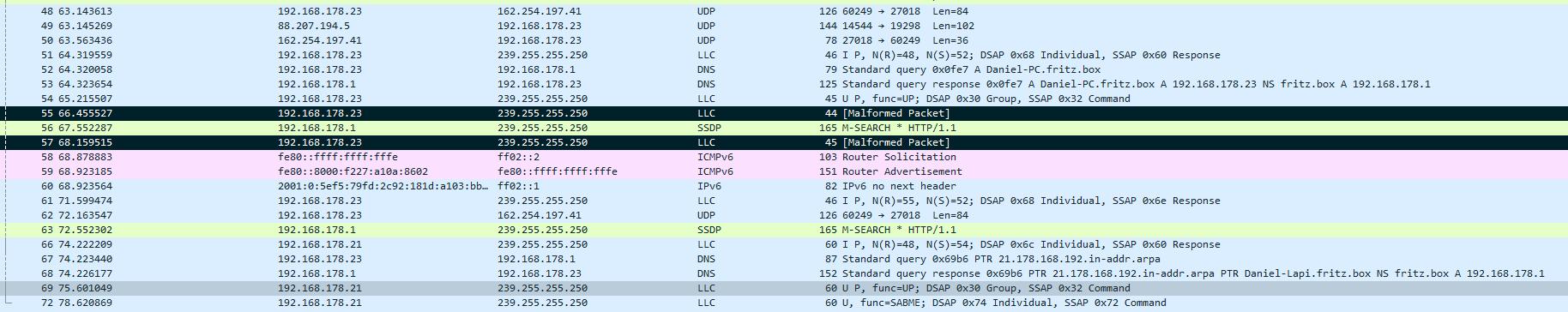
The design of the chat service is pretty simple, we only use a command line in order to communicate between each other. There is no need for a GUI we just need to insert some text hit return and the message is transmitted to each peer.

Our two main components are the class “receiver” and the class “sender”. In order to transmit the messages to multiple peers we’ve used multicast and created a multicast socket to receive these packages. We’ve send all the messages over the multicast address and each receiver connected himself to the multicast group.



Encountered problems

The main problem was how to send the messages to each peer so we tried using broadcast which didn’t work and so we’ve ended up with multicast.



Sometimes we’ve also get malformed packages (wireshark capture) and these aren’t shown in our chat application. On our first test run with 3 PC’s there was only one pc which made malformed packages now also the 2 other PC’s make it. We’ve didn’t manage to find the problem of this strange behaviour, but else it works.

Potential future work

* Find the problem with the malformed packages.
* Maybe create a GUI
* Custom username
* Polishing

# File transfer

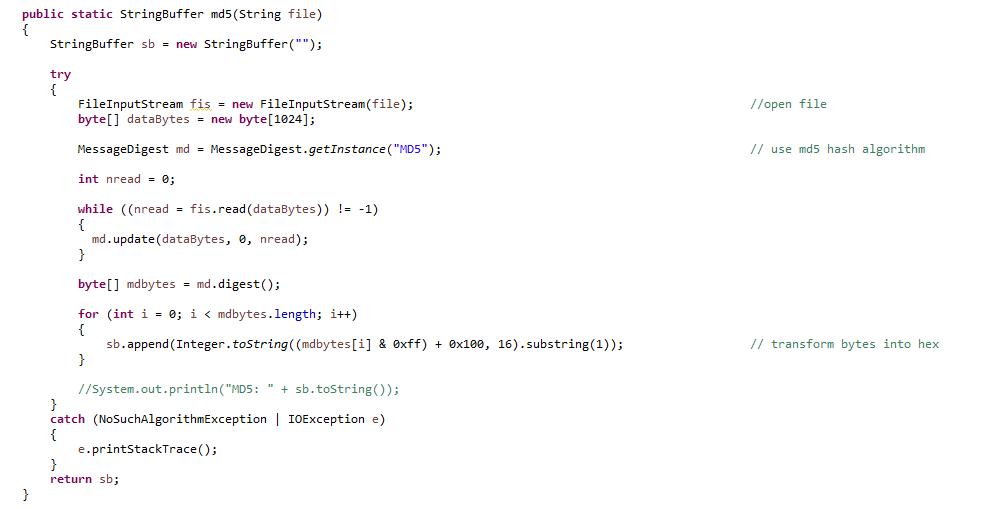
Information

The design for the file transfer also only a command line, we need to indicate the filename which we want to transmit and then IP-Address of the destination.

Here we’ve got the “server” thread who receives the file. The client thread who’s sending the file. We’ve also got a method md5() which calculates a hash value from the sended/received file.

We’ve got the validateHash() method which sends a message to the sender with the md5 hash value The hashVal() method checks the received hash value with the own hash value and resends the file if necessary.

We’ve decided us for the hashing method because it would be the simplest and the efficient. If the file has been transferred the hash value would be calculated and send to the sender of the file. The sender of the file would now calculate it’s own hash value of his local file. If now both hash values are the same it is ok. If not the sender of the file would resend the file automatically.



Encountered problems

The big main problem was here that we didn’t know how we would need to send the packages and how to receive them. We’ve basically tried it with embedded sequence numbers in the packages but that didn’t work well. After long trying we’ve decided that checksums would also do the job, if not even better, so we can profit from the small overhead of UDP and didn’t need to bother with sequences.

Potential future work

* Get rid of some bugs
* Create a user interface