**Lab 3 – Chatterbox**

Internationale Medieninformatik

2nd Semester

Alexander Jäger / 578538 / s0578538@htw-berlin.de

Tuan Dai Fischer / *s0576066* / *Tuan.Fischer@Student.HTW-Berlin.de*

GitHub-Link: <https://github.com/Aczia/Lab03/tree/main/ChatterboxExercise>

Informatik 2 by Cay Horstmann, Eva Schütze & Jonas Fichtmüller

Apr 28, 2021

**Einleitung**

Dieser Report wurde von Alexander Jäger und Tuan Dai Fischer erstellt. Das Design wurde von Lab-Reports des letzten Semesters inspiriert. Die Aufgaben wurden von Prof. Dr. Weber-Wulff erstellt und durch Jonas Fichtmüller erklärt und zur Verfügung gestellt.

**Aufgaben**

(Quelle: <https://people.f4.htw-berlin.de/~weberwu/info2/labs/Exer1.shtml> & <https://moodle.htw-berlin.de/pluginfile.php/1198800/mod_assign/introattachment/0/Changes%20to%20Lab%203.pdf?forcedownload=1> )

1) Start your chatterbox by writing a method that listens on a port. This is your chatterbox server. (127.0.0.1 is the ip for the local host)

2) Now write a client that writes to that port.

3) Test your methods on your own machine. For now, just echo what you have read to the console to see it working.

4) Once your client is working, connect to any port between 8050 and 8062 on

<ipOfMyLaptop> (will be given during the lab) and start chatting with me and your colleagues.

**Report**

1)

Zuerst haben wir der Chatterbox den Port 8051 zugewiesen und die Methode startServer ausgeführt:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung  
(screensot 1.1: main() Methode in Chatterbox)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung  
(screenshot 1.2: Methode startServer() in Chatterbox)

Zum einen wird ein neuer ServerSocket geöffnet an unserem Port (8051). Wenn dieser ServerSocket akzeptiert wird, werden auch die Streams initialisiert.

Außerdem wird ein loop gestartet, dass einkommende Nachrichten empfängt und in die Konsole schreibt (wenn eine Nachricht ankommt).

2)

Als erstes erstellen wir einen neunen Client mit der IP vom local host (127.0.0.1) und einem Port (in diesem Besipiel: 8050).

Ein Bild, das Text, Uhr enthält.

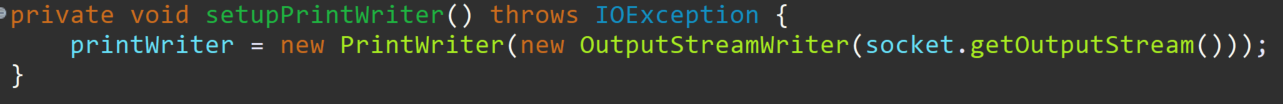
Automatisch generierte Beschreibung  
(screenshot 1.1: main() methode in Client)

Danach wird der Client gestartet:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung  
(screenshot 1.2: Methode startClient() in Client)

Wenn der Client gestartet wird, wird auch einmalig der PrintWriter initialisiert:

  
(screenshot 2.3: Methode setupPrintWriter() in Client)

Wie von der Aufgabe verlangt soll der Client ununterbrochen alle 4 Sekunden „Hello World“ schreiben:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung  
(screenshot 2.4: erweiterter Screenshot von main() Methode in Client)

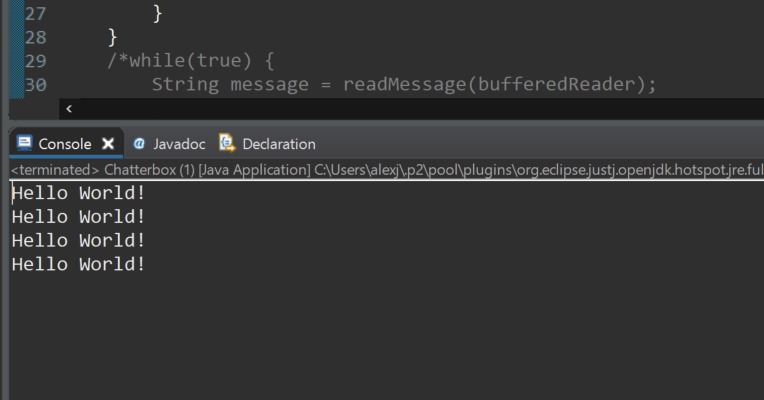
Dadurch wird mithilfe des PrintWirters die Nachricht in der Methode writeMessage geschrieben und danach “geflushed“:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung  
(screenshot 2.5: Methode write Message in Client)

3)

Wie erwartet, wurde nun alle 4 Sekunden „Hello World“ von der Chatterbox empfangen und in die Konsole geschrieben.



(screenshot 3.1: Test am eigenen PC)

4)

Leider hatten wir es zeitlich nicht mehr geschafft, den Code zu vollenden und ihn danach zu testen.

Wir haben die angeforderten Aufgaben versucht zu bewältigen, indem wir einen Advanced Client haben, der nicht nur Nachrichten printed, sondern auch mit Hilfe eines buffers auch einkommende Nachrichten liest.

**Reflection**

Alex:

Diese Woche haben wir Socket Programming behandelt. Da wir noch nicht wirklich Vorlesungen/Übungen zu diesem Thema hatten, war dieses Lab relativ schwierig zu lösen.   
Aber zum Glück gibt es zahlreiche Dokumentationen zu Socket Programming im Internet, die sehr hilfreich waren. Außerdem hat uns Jonas weitergeholfen, als wir ein Syntax-Problem hatten.  
Nach diesem Lab habe ich Socket Programming so langsam verstanden, aber bei einigen Stellen sind auf jeden Fall noch Lücken vorhanden.  
Ich hoffe, dass wir in Zukunft auch Labs/Übungen zum selben Thema haben, wie in der Info2 Vorlesung. Dadurch sollten die Labs effizienter und reibungsloser laufen, da man dann weniger mit Recherche beschäftigt ist.  
Die Gruppenarbeit mit Dai hat sehr gut geklappt, da wir direkt am Anfang ein GitHub-Repo erstellt haben, um Datenaustausch zu vereinfachen. Außerdem konnten wir uns gegenseitig bei der Problemlösung unterstützen und auf Fehler/ weitere Fragen hinweisen.   
Leider sind wir mit den Aufgaben nicht in der Übungszeit fertig geworden und konnten somit die letzte Aufgabe nicht wirklich testen (Verbindung mit der IP des Laptops von Jonas).   
Trotzdem fand ich das Thema sehr interessant und bin schon auf das Thema vom nächsten Lab gespannt.

Dai:

In dieser Woche wurde das socketing behandelt, was für mich Neuland war und auch im Nachhinein nicht sehr viel klarer wurde.

Nachdem wir das Grundgerüst von Jonas übernommen haben und die Methoden eingebaut haben, hat und Jonas mit ein paar Syntax Problemen und Import/Field etc. Fehlern ausgeholfen, wodurch dann die Aufgabe 1-3 beendet wurde. Da wir aber sehr viel Zeit dafür aufgewendet haben, da wir nicht direkt wussten wo unsere Fehler waren, war dann auch die Lab Stunde selbst vorbei, wodurch wir unsere Aufgabe 4 dann asynchron über GitHub versucht haben zu beenden. Durch fehlendes Testen, war es sehr anstrengend gewesen herauszufinden, ob das was wir geschrieben haben uns gelungen ist oder nicht.

Ich hoffe, dass das Socketing entweder in der Vorlesung nochmal näher behandelt wird oder wir ein Lab nochmal haben, in der man weniger (zumindest fühlte es sich für mich so an) in das kalte Wasser geworfen wird.

Ich habe nach dem Lab grundlegend die Dinge wie Streams und Setups verstanden, dennoch würde ich nicht behaupten, dass ich ohne Weiteres Aufgaben aus dem Bereich einfach bewältigen könnte. Dennoch ist das Thema Serverprogrammierung interessant und hoffe das es nochmal aufgearbeitet wird.

**Zeiteinteilung**

|  |  |  |
| --- | --- | --- |
|  | Dai | Alex |
| Pre-Lab | ca. 30 min | ca. 20 min |
| Laboratory | ~4½ Stunden | |
| Report | ca. 60 min | ca. 1¾ h |

**Quellen**

Aufgaben: <https://people.f4.htw-berlin.de/~weberwu/info2/labs/Exer1.shtml> & <https://moodle.htw-berlin.de/pluginfile.php/1198800/mod_assign/introattachment/0/Changes%20to%20Lab%203.pdf?forcedownload=1>

Code von Jonas Fichtmüller (Grundgerüst + Hilfe)

<https://www.geeksforgeeks.org/socket-programming-in-java/>

<https://docs.oracle.com/cd/E19120-01/open.solaris/817-4415/6mjum5som/index.html>

<https://www.edureka.co/blog/socket-programming-in-java/>

**Pre-Lab**

Alex:

Pre-Lab

1. Socket Programming in Java  
<https://www.geeksforgeeks.org/socket-programming-in-java/>

<https://docs.oracle.com/cd/E19120-01/open.solaris/817-4415/6mjum5som/index.html>

<https://www.edureka.co/blog/socket-programming-in-java/>

<https://www.baeldung.com/a-guide-to-java-sockets>

2. public static String readConnection(){

Socket echoSocket = new Socket(hostName, portNumber);

PrintWriter out = new PrintWriter(echoSocket.getOutputStream(), true);

}

(<https://docs.oracle.com/javase/tutorial/networking/sockets/readingWriting.html>)

3. public static String writeConnection(){

BufferedReader in =   
 new BufferedReader(new InputStreamReader(echoSocket.getInputStream()));

BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in))

}

(<https://docs.oracle.com/javase/tutorial/networking/sockets/readingWriting.html>)

Dai:

Pre Lab:

Read and Write methods:

<https://docs.oracle.com/javase/tutorial/networking/urls/readingWriting.html>

Socket verbinden:

<https://javabeginners.de/Netzwerk/Socketverbindung.php>

https://www.baeldung.com/a-guide-to-java-sockets

**Anhang**

Code: https://github.com/Aczia/Lab03/tree/main/ChatterBoxSimple