

Modul Verteilte Software WS 2022/2023

Aufgaben 5

URL-Connection

1. Entwickeln Sie eine Anwendung die in regelmäßigen Abständen die Verfügbarkeit einer Webseite prüft. Ergebnis der Überprüfung: Abfragezeitpunkt, Hostname, Größe der Webseite (Länge des Strings) sowie der Verfügbarkeitsstatus sollen ausgegeben werden. (Ist die Seite nicht verfügbar, erfolgt die Ausgabe innerhalb eines catch-Blockes). Verwenden Sie dazu ein Thread.

Hinweise:

Verwenden Sie zum Zugriff auf die Web-Seite die Klassen URL und URLCollection:

```
URL url = new URL(path);
URLConnection source = url.openConnection();
source.setUseCaches(false);
source.connect();
```

2. Ergänzen Sie Ihre Lösung um die Ausgabe in eine Logdatei. Zum Speichern in die Logdatei verwenden Sie die Klassen FileWriter und BufferedWriter.

Sockets

1. Entwickeln Sie eine Client-Server-Anwendung die wie die Anwendung der vorhergehenden Aufgabe die Verfügbarkeit einer Webseite prüft. Die Überprüfung findet dabei in Server-Teil statt.

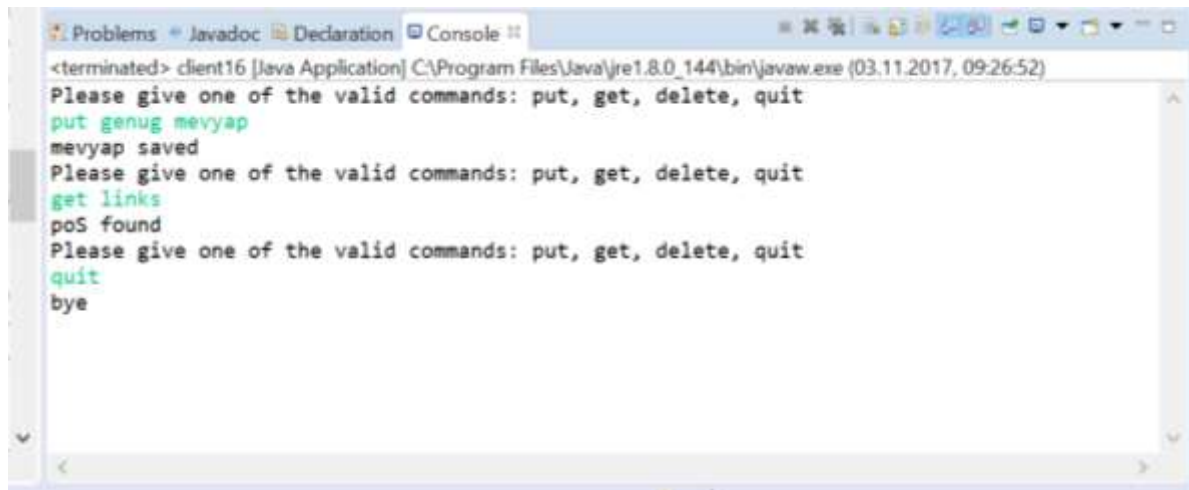
Die Client-Anwendung liest die URL der zu testenden Webseite und sendet diese zum Server als **DatagramPacket** über ein **DatagramSocket**, empfängt das Ergebnis des Testes und zeigt es an.

Das Serverprogramm überprüft die Erreichbarkeit der Seite und sendet das Ergebnis der Überprüfung an den Client. Der Server soll in der Lage sein in einer endlosen Schleife mehrere Anfragen des Clients zu bedienen.

Verwenden Sie als Adresse für den Remote Server „localhost“ und beachten Sie, dass nur eine Anwendung gleichzeitig an einem Port „empfangen“ kann, d.h. verwenden Sie zum Empfangen bei Server und Client verschiedene Ports.

2. Implementieren Sie ein Server-basiertes Wörterbuch sowie einen Client, der die Benutzeranfragen einliest und an den Server weiterleitet. Das Wörterbuch soll zu einem deutschen Wort die Übersetzung z. B. in Klingonisch liefern (bzw. erfassen) und verfügt über vier Befehle, mit denen die auf dem Server gespeicherten Daten manipuliert werden können (s. Abbildung):

- **put <de> <kli>**: erstellt ein neues Werte Paar oder überschreibt ein existiertes Paar auf dem Server, z. B. **put genug mevyap**
- **get <de>**: fragt den entsprechenden Wert mit einem Schlüsselwort ab, z. B. **get genug**
- **delete <de>**: löscht ein existiertes Wertepaar mit dem angegebenen Schlüsselwort, z. B. **delete genug**
- **quit**: beendet die aktuelle Verbindung mit dem Server und speichert das Wörterbuch in eine Datei.



```
<terminated> client16 [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (03.11.2017, 09:26:52)
Please give one of the valid commands: put, get, delete, quit
put genug mevyap
mevyap saved
Please give one of the valid commands: put, get, delete, quit
get links
poS found
Please give one of the valid commands: put, get, delete, quit
quit
bye
```

Hinweise:

- Der Server nutzt zum „Lauschen“ auf die Anfrage die Klasse `java.net.ServerSocket`
- Der Server startet für jede Client-Anfrage ein Thread, das zur Kommunikation mit Client die Klasse `java.net.Socket` nutzt.
- Der Client nutzt zum Verbindungsaufbau und zur Kommunikation die Klasse `java.net.Socket`.
- Verwenden Sie zum Speichern der Wortepaare ein Objekt des Collection-Type (z. B. `ConcurrentHashMap<String,String>`), das an die Threads übergeben wird.
- Verwenden Sie zum Lesen/Schreiben von/in die Sockets die mit Socket-Input- bzw. Socket-Output-Stream initialisierte `PrintWriter` bzw. `BufferedReader/InputStreamReader`. Das bequeme Schreiben in die Sockets ist mit der Operation **`println`** der Klasse `PrintWriter`, das Lesen einer ganzen Zeile mit **`readLine`** von `BufferedReader` möglich.
- Verwenden Sie beim Client zur Eingabe des Befehls Klasse `Scanner`:
`Scanner scanner = new Scanner(System.in);`
- Einige Programmfragmente finden Sie in der Datei `zuAufgaben5.txt`