PR Rechnernetze und Netzwerkprogrammierung 621.801/621.802	WS 2024/2025
Institut für Informationstechnologie (ITEC)	Rainer/Timmerer
Übungsblatt 03	

Ü 3.1 Post Office Protocol 3

Machen Sie sich mit dem POP3-Protokoll vertraut. Verwenden Sie dabei das RFC 1939 und untersuchen Sie wie man sich bei einem Mailserver anmeldet und wie man Nachrichten abruft. Beantworten Sie folgende Fragen:

- Wie ist das POP3-Protokoll grundsätzlich aufgebaut?
- Ist POP3 ein sicheres Protokoll? Argumentieren Sie warum bzw. warum nicht.
- Was ist der Unterschied zwischen *single-line* und *multi-line response*?

Vergleichen Sie POP3 mit IMAP! Beschreiben Sie die wesentlichen Unterschiede.

Ü 3.2 Java Sockets: POP3-Client

Implementieren Sie einen einfachen POP3-Client in Java. Ihr Client soll:

1. sich bei einem POP3-Server **anmelden**,
2. eine **Liste aller Nachrichten** anfordern und
3. eine **Nachricht** vom Server **abrufen**.

Achten Sie auf einen ordnungsgemäßen Verbindungsabbau.

Hinweise:

- Verwenden Sie `java.net.Socket` für die Netzwerkkommunikation und die `String.toByte()`-Methode um Strings einfach in `InputStreams` zu schreiben.
- Testen Sie Ihre Implementierung mit einem Emailserver, der unverschlüsselte Anmeldung unterstützt. Alternativ können Sie https://www.stunnel.org/ verwenden, um eine sichere Verbindung mit einem beliebigen POP3-Server Ihrer Wahl herzustellen.

Ü 3.3 Java Sockets: POP3-Server

Implementieren Sie einen POP3-Server mit minimaler Funktionalität, mit dem ein Email-Client Ihrer Wahl zusammenarbeitet. Nutzen Sie dabei die über den Moodle-Kurs bereitgestellte Datenbank-Klasse *SampleDataBase.java*, die Ihnen eine Grundmenge an Nachrichten zur Verfügung stellt. Ziel ist es, dass die Nachrichten aus der Datenbankklasse in der Inbox eines Email-Clients angezeigt werden. Achten Sie bei Ihrer Implementierung darauf, dass sich mehrere Benutzer gleichzeitig zum Server verbinden können. Verwenden Sie dazu für jeden Benutzer einen eigenen Thread.

Hinweise:

- Verzichten Sie auf User-Management und akzeptieren Sie jede beliebige Kombination aus Login-Name und Passwort.
- Implementieren Sie, im Falle von Mozilla-Thunderbird, die Kommandos CAPA, USER, PASS, STAT, TOP, RETR, LIST, DELE und QUIT.
- Konsultieren Sie für Ihre Implementierung die RFCs 1939 und 2449.

Testen Sie Ihre Implementierungen aus Ü 3.2 und Ü 3.3 mit tatsächlichen Client- und Serverimplementierungen und verfassen Sie einen Erfahrungsbericht inkl. Screenshots.

Ü 3.4 Java Sockets: HTTP-Server (single-thread)

Implementieren Sie einen HTTP-Server, der Anfragen von Clients in einem einzigen Thread behandelt. Der Einfachheit halber soll sich die Funktionalität des Servers auf HTTP-Version 0.9 (http://www.w3.org/Protocols/HTTP/AsImplemented.html) stützen, die auf das Ausliefern einzelner Dateien beschränkt ist.

Der Server soll über Kommandozeilenparameter konfiguriert werden. Dabei müssen sowohl TCP-Port als auch Basisverzeichnis (document root) des HTTP-Servers beim Start angegeben werden.

Hinweise:

- Beachten Sie, dass diese ursprüngliche HTTP-Version keine Möglichkeit bietet, Fehlermeldungen an den Client zu übertragen. Signalisieren Sie daher einen Fehler einfach dadurch, dass der Server nichts liefert und die Verbindung schließt.
- Sollte ein angegebenes Dokument nicht existieren, so schließt der Server einfach die Verbindung, ohne Daten auszuliefern.
- Sollte ein Verzeichnis angefordert werden, so soll der Server die in diesem Verzeichnis befindliche Datei index.hml zurückgeben. Ist diese nicht vorhanden, so wird wiederum nichts an den Client retourniert.
- Die Beschreibung (http://www.w3.org/Protocols/HTTP/AsImplemented.html) geht davon aus, dass nur Text zurückgeliefert wird. Die aktuellen Browser unterstützen aber auch, dass binäre Daten wie z.B. Bilder ausgeliefert werden.

Im Moodle-Kurs befinden sich ein paar Testdaten (documentRoot.zip), mit denen Sie Ihren Server mit einem Browser Ihrer Wahl testen können.

Ü 3.5 Java Sockets: HTTP-Server (multi-thread)

Erweitern Sie die Funktionalität des Servers aus Ü 3.4 wie folgt. Anstatt die Anfragen von Clients seriell in einem Thread zu behandeln, soll jede Anfrage durch einen neuen Thread behandelt werden. Erweitern Sie den Funktionsumfang Ihres Servers, dass dieser HTTP-Version 1.0 unterstützt (1.0: https://tools.ietf.org/html/rfc1945, 1.1: https://datatracker.ietf.org/doc/html/rfc9112/). Des Weiteren soll ihr Server persistente Verbindungen unterstützen (siehe hierzu das RFC zu HTTP 1.1).

Weiters ist der Server durch eine Logging-Komponente zu erweitern. Jeder Zugriff auf den Server (Zeitpunkt, Request, IP und Port des Clients) soll dabei mitprotokolliert werden. Dieses Logging soll zentral durch einen eigenen Thread realisiert werden, der in Abständen von 5 Sekunden das Zugriffsprotokoll auf die Konsole schreibt. Die Zugriffe müssen daher mit geeigneten Mechanismen von den behandelnden Threads an diesen Logging-Thread kommuniziert werden. Sorgen Sie für eine geeignete Synchronisation der Threads.

Beispiel-Ausgabe:

```
2021-09-26 15:40:10 GET /index.html 127.0.0.1 44239
2021-09-26 15:40:10 GET /images/logo.gif 127.0.0.1 44240
2021-09-26 15:40:10 GET /images/TechnikErleben.png 127.0.0.1 44241
```