

Name: \_\_\_\_\_ Vorname: \_\_\_\_\_  
Matrikelnummer: \_\_\_\_\_ Testat: \_\_\_\_\_

### Algorithmen und Programmierung 3 Praktikum 4

**Aufgabe 1:** Erläutern Sie den folgenden Programmabschnitt genau. Wozu macht man diesen Zirkus?

```
while (true) {  
    Socket client = listener.accept();  
    new Thread(new Handler(client)).start();  
    System.out.println("New client no."+id+  
        " from "+ listener.getInetAddress()+  
        " on client's port "+client.getPort());  
}
```

**Frage:** Wo findet man heraus, auf welche Nachrichten ein Webserver antworten sollte, wie diese aufgebaut sind und was bei der Antwort zu beachten ist?

**Aufgabe 2:** Vervollständigen Sie den vorgegebenen einfachen Web-Server. Beachten Sie dabei die Kommentare und die TODO-Hinweise.

Mitspielende Klassen:

- **Main:** Startet halt alles. **Kommandozeilenparameter beachten!!**
- **SingleThreadServer:** Lässt Request in einem Thread durch den **Worker** ausführen.
- **SimplePoolServer:** Verwaltet einen Pool von Threads, die **Worker** arbeiten lassen.
- **ExecutorServer:** Benutzt `java.util.concurrent..`
- **AbstractHttpWorker:** Erledigt für den richtigen **Worker** schon mal die HTTP-Sachen.
- **Worker:** Macht die ganze Arbeit
- **Request:** Analysiert die Anfrage und stellt die nötige Information zur Verfügung (Parser).
- **RequestScanner:** Ist ein ganz normaler Scanner für http.
- **MimeTypes:** Gibt für jede Endung die richtige Mime-Information zurück.
- **Parameter:** Verwaltet die Konfiguration (einstellbar über die Datei `server.properties`)

Im ersten Schritt sollte der `SingleThreadServer` laufen. Sie sollten demonstrieren können, dass es sich lohnt, mehrere parallele Threads an der Arbeit zu halten (Sie können ja das Ausliefern von Dateien künstlich verlangsamen). Die Klassen `SimplePoolServer` und `ExecutorServer` sollen bessere Alternativen implementieren (im 2. Fall unter Nutzung der Java-Bibliothek). Im `SimplePoolServer` soll nicht bei jedem Request ein neuer Thread erzeugt werden (das wäre langsam), sondern einer der untätigen Threads sollte sich den „Request“ holen und ausführen. Fertigen Sie bitte auch ein Klassendiagramm an, an dem Sie den Ablauf der Abwicklung eines HTTP-Requests erläutern können.

**Hinweis:** Einen Webserver kann man mit einem Browser testen. Es ist zu beachten, dass hinter der URL (z.B. `localhost`) die Portnummer angegeben werden kann (z.B. `localhost:8080/index.html`).

**Aufgabe 3** Schreiben Sie auf der Basis von RMI ein Telebanking System bestehend aus einem Client- und einem Serverprogramm. Die gewünschte Funktionalität ist: Anlegen eines Kontos (mit Kontonummer), Einzahlen, Abheben, Überweisen und Abfragen des Kontostands. Der Client kann einen ganz einfachen Textdialog führen. Der Server soll ebenfalls ganz einfach sein, also keine Datenbank haben. Achten Sie auf korrekte Synchronisation der Zugriffe auf gemeinsam genutzte Datenstrukturen. Wie ist es mit der Überweisung? Kann dabei evtl. ein Deadlock entstehen (Können Sie dies selbst hervorrufen?).