

Prüfung vom 30. April 2013

Teil 1: 30 Minuten

Name, Vorname:

Binder, Luzi

Allgemeine Hinweise:

- 1) Diese Prüfung besteht aus zwei Teilen.
- 2) Für diesen ersten Teil der Prüfung sind keine Unterlagen erlaubt.
- 3) Bitte beantworten Sie die Fragen dieses ersten Teiles direkt auf dem Aufgabenblatt.
- 4) Für diesen ersten Teil haben Sie 30 Minuten Zeit.

Viel Erfolg!

Aufgabe 1: Network Programming Basics

3

(4 Punkte)

Markieren Sie für jede Aussage ob sie richtig oder falsch ist. 0.5 Punkte Abzug pro falsche Antwort, min. 0 Punkte.

Aussage	Richtig	Falsch
Ein WSDL-File ist ein XML File welches eine REST-Schnittstelle beschreibt.		X
Der <messages> Teil in einem WSDL-File beschreibt die Struktur der Meldungen, die bei Aufrufen hin oder her geschickt werden.	X	
Der <portType> Teil in einem WSDL-File beschreibt Interfaces, also eine Menge von Operationen.	X	
Beim Use-Style „Encoded“ werden in den XML-Dokumenten, die hin und her geschickt werden, die XML-Datentypen angegeben.	X	
Das Java-Tool ws-gen kann verwendet werden, um auf Klientenseite die Proxy-Klassen für einen durch ein WSDL-File definierten Service zu generieren.		X
Bei einem SOAP Aufruf muss im HTTP Request ein SOAPAction Header vorhanden sein.	X	
Ein Java-Prozess kann mehrere ServerSockets offen haben.	X	X
Beim HTTP Chunked-Encoding werden die Daten komprimiert übertragen.		X
Bei einem HTTP GET Request können im Body des Requests keine Daten mitgeschickt werden.	X	X
Ein Parameter vom Typ Holder<T> in einer generierten Web-Service-Schnittstelle steht entweder für einen OUT Parameter oder einen INOUT (also Referenz-) Parameter.	X	

Aufgabe 2: HTTP**(3+1 = 4 Punkte)**

Wir starten telnet mit dem Befehl

```
telnet www.baz.ch 80
```

und geben folgende Daten ein (abgeschlossen mit zwei CRLF, ohne die Zeilennummern (1) und (2)):

- (1) GET / HTTP/1.1
Accept: */*
Accept-Language: de
User-Agent: Mozilla/4.0
Host: localhost
- (2) Connection: Keep-Alive

Der Server antwortet mit folgenden Daten:

```
HTTP/1.1 200 OK
Set-Cookie: SERVERINFO=R1895111747; path=/; expires=Sat, 20-Apr-2013 09:21:36 GMT
Date: Fri, 19 Apr 2013 09:16:35 GMT
Server: Apache
Vary: Accept-Encoding,User-Agent
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

- (3) 95
<html>
<head><title>baz Hosting</title></head>
<body>
<h1 align="center">Für diesen Host "localhost" ist kein Webserver konfiguriert</h1>
</body>
</html>
0

a) Geben Sie an, was die Zeilen (1) bis (3) genau bedeuten.

- (1) Wir möchten Inhalt anfordern (=GET), nämlich direkt die als Index konfigurierte Datei auf dem "/"-Verzeichnis, mit der HTTP-Version 1.1 ✓
- (2) Server soll Verbindung offen halten und nicht wieder schließen. ✓
- (3) Inhalt ist mit chunked unterteilt, es folgen 95 Zeichen. ✓
hex

3

b) Erklären Sie, warum die Anfrage auf die Seite der Basler Zeitung nicht die erwartete HTML Seite des BaZ-Portals zurückliefert.

Wir möchten auf den Host localhost bei baz zugreifen, was nicht möglich ist. ✓

1

4

Aufgabe 3: Sockets

(1.5+1.5+1 = 4 Punkte)

In dieser Aufgabe betrachten wir einen Server, welcher Dateien komprimiert. Der Server öffnet dazu einen ServerSocket und startet bei jeder eingehenden Socket-Verbindung einen neuen Thread, der in seiner run-Methode die Daten liest und über einen mit einem GZIP-Filter versehenen Output-Stream zurückschreibt:

```
InputStream in = socket.getInputStream();
GZIPOutputStream out = new GZIPOutputStream(socket.getOutputStream());
byte[] buffer = new byte[1024];
int bytesRead;
while ((bytesRead = in.read(buffer)) != -1) { out.write(buffer, 0, bytesRead); }
out.finish(); // Flush bytes from GZIPOutputStream
socket.close();
```

- a) Ein Klientenprogramm schickt Daten einer Datei an diesen Server. Sobald alle Daten an den Server geschickt sind, wird der Socket mit `s.shutdownOutput()` halb geschlossen. Was passiert, wenn nach dem Aufruf von `s.shutdownOutput()`

a1) weitere Daten auf den Socket geschrieben werden?

Daten werden nicht mehr verschickt ✓

a2) Daten vom Socket auf der Serverseite gelesen werden?

Werden weiterhin verarbeitet ✓

a3) Daten auf der Serverseite auf den Socket geschrieben werden?

Funktioniert weiterhin normal ✓

- b) Erst nach dem Aufruf von `s.shutdownOutput()` liest der Client die komprimierten Daten vom Server. Wird dieses Programm verwendet um eine grosse Datei zu komprimieren, so landet man in einem Dead-lock. Erklären Sie, wie dieses Problem zustande kommt.

Bei Sockets Buffer unterlegt → wenn voll → Buffer auf Clientseite läuft voll → Server kann keine Daten schreiben → ist blockiert → liest & schreibt nicht mehr
out.write ist eventuell blockiert da das OS nur eine gewisse Anzahl Verbindungen zulässt, Linux (Ubuntu) z.B. nur 5
kann dann bei grossen Dateien ziemlich schnell zu der blockierenden Situation kommen.
hier ist nur 1 Thread und 1 Server involviert, d.h. 1 Connection.

- c) Wie kann das unter b) beschriebene Problem gelöst werden, damit dieser Komprimierer für beliebig grosse Dateien funktioniert? Beschreiben Sie ihre Lösung, Code ist nicht nötig.

Man könnte die Daten vorher auf den Filesystem als Datei zwischenspeichern, bevor sie verarbeitet werden.

Aufgabe 4: REST

(2+2 = 4 Punkte)

Sie haben in den Übungen auch eine REST Schnittstelle zur Bank implementiert. Ein sinnvoller Ansatz dazu ist, für die Bank und für jedes Konto eine eigene Ressource zu definieren:

/bank/accounts	GET (1) POST	get all accounts create a new account
/bank/accounts/{id}	GET HEAD (2) PUT DELETE	get account details (getBalance, getOwner) check account head (isActive) change balance (setBalance) remove account (removeAccount)

- a) Markieren Sie in obiger Liste alle Methoden welche *nicht* idempotent sind und begründen Sie ihre Auswahl (d.h. erklären Sie, warum die markierten Methoden *nicht* idempotent sind).

(1) nicht idempotent. Es wird jeweils eine neue Kontonummer produziert, auch wenn Input immer "Hans" ist. ✓
 (2) Je nachdem wenn z.B. Kontostand Null ist und man 10 Fr beziehen möchte, wird kein Geld abgeboben. ✓
 je, ist aber trotzdem idempotent
 Alle außer POST idempotent
 1

- b) Die Umsetzung der Methode GET auf /bank/accounts kann grosse Resultate liefern welche das Netz belasten. Der Datenverkehr kann mit einem Conditional GET reduziert werden. Beschreiben Sie, wie dies funktioniert und welche Daten dabei vom Klienten an den Server und umgekehrt vom Server an den Klienten geschickt werden.

Man könnte z.B. einen Hash-Code aus allen aktuell verfügbaren Accounts generieren und dann jeweils mit "If-Match" überprüfen, ob die Daten noch die gleichen sind. Falls schon, muss nicht nochmals die ganze Liste mit allen Accounts versendet werden. If-None-Match
 Also Client sendet Anfrage GET mit Hashcode der aktuellen Accounts im Client. Server überprüft Hashcode und sendet nur Accounts, falls nicht übereinstimmt. ✓ 2/3

Prüfung vom 30. April 2013

Teil 2: 60 Minuten

Name, Vorname:

Bruder, Luzi

Allgemeine Hinweise:

- 1) Bitte starten Sie jede Aufgabe auf einem neuen Blatt. Schreiben Sie auf jedes Blatt Ihren Namen.
- 2) Pro Aufgabe darf höchstens ein gültiger Lösungsversuch abgegeben werden. Ungültige Lösungsversuche müssen klar durchgestrichen sein!
- 3) Lesen Sie eine Aufgabe genau durch bevor Sie sie zu lösen beginnen.
- 4) Insgesamt haben Sie für diesen Teil 60 Minuten Zeit.

Viel Erfolg!

Aufgabe 5: Timer Server

(9+3+2 = 14 Punkte)

In dieser Aufgabe programmieren Sie einen kleinen Timer-Server, der jede Sekunde an alle registrierten Klienten die aktuelle Zeit (in Form eines Long-Wertes) verschickt. Für die Aktualisierung der Zeit und die Notifikation der Klienten verwendet der Server einen Timer-Task. Die Struktur der Klasse sieht wie folgt aus:

```
public class TimerServer {
    private static long time = System.currentTimeMillis() / 1000;
    public static void main(String[] args) throws Exception {
        Timer t = new Timer();
        t.scheduleAtFixedRate(new TickTask(), 0, 1000); // nach diesem Aufruf wird
                                                         // TickTask.run jede Sekunde aufgerufen.
        // ...
    }

    static void notifyClients() { /* ... */ }
    static class TickTask extends TimerTask {
        public void run() { time++; notifyClients(); }
    }
}
```

Der Server nimmt neue Klienten über Port 1234 entgegen. Sobald sich ein Klient über Port 1234 angemeldet hat, erhält er jede Sekunde den aktuellen Wert. Dieser Wert wird dabei über einen `DataOutputStream` mit `writeLong` verschickt. Falls das Ausliefern dieses Wertes zu einer `IOException` führt, dann wird dieser Klient in Zukunft nicht mehr mit Daten beliefert.

Der Wert des Timers (also des Feldes `time`) kann von einem Klienten über Port 1235 auf dem Server auch geändert (d.h. neu gesetzt) werden. Ein Klient muss dazu einen Socket auf Port 1235 öffnen und kann dann den neuen Wert über einen `DataOutputStream` mit `writeLong` setzen.

Aufgaben:

- a) Vervollständigen Sie die Klasse `TimerServer` und implementieren Sie die beiden Funktionen
 - Notifikation von Klienten die sich auf Port 1234 anmelden, und
 - Änderung der Zeit auf dem Server über Port 1235.
- b) Implementieren Sie einen Klienten, der auf der Konsole jeweils den aktuellen Wert des `TimeServers` ausgibt. Der Klient muss sich dazu beim Server auf Port 1234 anmelden und dann die erhaltenen Long-Werte ausgeben.
- c) Nehmen wir an, dass die Informatikdienste Sie dazu zwingen, die Lösung mit HTTP zu realisieren. Beschreiben Sie, wie in diesem Fall der Time-Server implementiert werden muss, d.h. beschreiben Sie, wie der Time-Server von Aufgabe a) und insbesondere der Klient aus Aufgabe b) realisiert werden müssen.

Aufgabe 6: REST

(4+8+4 = 16 Punkte)

In gewissen Fernsehsendungen von SRF1 gibt es Zuschauer-Wettbewerbe mit Verlosung von Bargeldpreisen. Die Zuschauer können ohne Beantwortung einer Wettbewerbsfrage per Anruf oder per WAP teilnehmen. Die URL für die WAP-Teilnahme lautet: <http://wap.srf.ch/1gegen100/>. Wenn man diese Seite auf einem Handy anzeigt erscheint das rechts abgebildete Formular.

In dieser Aufgabe sollen Sie einen Server implementieren, der Teilnahmen, die über dieses Formular erfasst werden, registriert. Die HTML Version des Formulars ist unten angegeben (wobei wir das Captcha weglassen).

Bemerkungen:

- Es werden nur Requests von einem Android-Handy akzeptiert (als Annahme für diese Aufgabe). Dazu muss das HTTP Header-Feld

User-Agent: Android

gesetzt sein. Falls dieses Feld nicht diesen Wert hat, dann soll beim Absenden des Formulars die Meldung „Illegal User Agent“ (als Text) zusammen mit dem Status 412 (Precondition failed) zurückgegeben werden, ansonsten der Status 200 OK.

- Das HTML Formular sieht wie folgt aus:

```
<form name="form1" method="post" action="/1gegen100/wap.php">
<input type="hidden" name="PHPSESSID" value="1ecf78b4"/>
Name: <br/>
  <input type="text" name="txtName" size="25" tabindex="1"/> <br/>
Handynummer: <br/>
  <input type="text" name="txtPhone" size="25" tabindex="5"/><br />
<INPUT TYPE="SUBMIT" VALUE="ABSENDEN">
</form>
```

Aufgaben:

- Geben Sie an, wie der HTTP Request aussieht, der an den Server geschickt wird, wenn beim oben dargestellten Formular auf einem Android-Handy die Felder ausgefüllt und auf „ABSENDEN“ geklickt wird (vollständig mit Headern und Body).
- Implementieren Sie mit REST / Jersey einen Server, der die mit dem oben dargestellten Formular erfassten Teilnahmen registriert. Der Einfachheit halber soll, falls der User-Agent korrekt angegeben ist, Name und Handynummer auf der Konsole ausgegeben werden. Es genügt, wenn Sie die Ressource ausprogrammieren. Der Server ist gegeben und sieht wie folgt aus:

```
public class Server {
  public static void main(String[] args) throws Exception {
    final String baseUrl = "http://localhost:80/";
    ResourceConfig rc = new PackagesResourceConfig("ch.srf.wap");
    HttpServer httpServer = GrizzlyServerFactory.createHttpServer(baseUrl, rc);
  }
}
```

- Implementieren Sie einen Klienten, der eine Teilnahme mit dem Namen „FHNW“ und der Telefonnummer „0562027700“ an den Server schickt. Der User-Agent muss natürlich explizit gesetzt werden. Verwenden Sie dazu das Jersey-Client-API.

Aufgabe 5) a) try (ServerSocket serv = new ServerSocket(1234)) { Lizzi Brucke

while (true) {

Socket s = serv.accept(); 1/2

this.writer = new PrintWriter(s.getOutputStream(), true); 1/2

{ catch (IOException e) { s.close(); } ✓

3). start();

Thread t2 = new Thread(@Override run() {

try (ServerSocket serv = new ServerSocket(1235)) { ✓

while (true) {

Socket s = serv.accept(); 1

data output also?

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

String long input = in.readLine(); 1

while (input != null) {

~~3~~ this.time = input;

3). start();

s.close();

Ich brauche noch folgendes Feld für die Klasse TimeServer:

private static PrintWriter writer; ✓ 0

static void notifyClients() {

this.writer.println(time);


}

1

(lös bei Fehler fehl.)

Soll mehrere Klien werden?

5/2

Bitte wenden! 

G. fehlt eine Anzeige
den Cl. Polle von $\frac{1}{2}$
auf 1/ser Tche. 1 2

```
system.out.println("Time: " + in.readLine());
```

a) POST /lgagen100/wap.php HTTP/1.1

Accept: */*
 Referer: http://wap.srt.ch/1gagen100
 Accept-Language: de-ch
 Content-Type: application/x-www-form-urlencoded
 Accept-Encoding: gzip, deflate
 User-Agent: Android
 Host: wap.srt.ch
 Content-Length: 123
 Connection: keep-alive
 Cache-Control: no-cache

PHPSESSID = 1ec178b4&txtName = FHNW & txtPhone = 0562027700

b) package ch.srt.wap;

@Singleton
 @Path("/")
 public class SrtGame {}

@POST

@Path("/lgagen100/wap.php")

@Consumes("application/x-www-form-urlencoded")

@Produces("text/html")

public String doLgagen100game(@FormParam("PHPSESSID") String PHPSESSID,
 @FormParam("txtName") String txtName, @FormParam("txtPhone") String
 txtPhone, @HeaderParam("User-Agent") String agent, @Context Request request) {}

Response Builder builder = request.evaluatePreconditions(agent, "Android");

if (builder != null) {}

return builder.build();

if ("Android".equals(agent)) return Response.status(412).build();

System.out.println(txtName + " - " + txtPhone);

builder = Response.ok("Ch>Danke fuer die Teilnahme!", "text/html");

return builder.build();

c) public class WettbewerbsteilnehmerClient {}

public static void main(String[] args) {}

Client create = Client.create();

WebResourceService = create.resource("http://wap.srt.ch/1gagen100/wap.php");

Service.post("PHPSESSID", "1ec178b4").post("txtName", "FHNW");

post("txtPhone", "0562027700").headerParam("User-Agent", "Android");

type("application/x-www-form-urlencoded")

