

---

**Programmieren in Java**

<http://proglang.informatik.uni-freiburg.de/teaching/java/2013/>

---

**Java-Übung Blatt 7 (Servlets, Logging)**

2013-06-04

**Hinweise**

- Schreiben Sie Identifier *genau so*, wie sie auf dem Blatt stehen (inklusive Groß- und Kleinschreibung), nicht nur ungefähr.
- Identifier und Kommentare bitte auf *englisch*!
- Schreiben Sie *sinnvolle* Kommentare
- Laden Sie Ihre Lösungen mit subversion (svn) ins Übungssystem hoch. Den entsprechenden Pfad finden Sie online.
- Das Übungssystem kann überprüfen, ob Sie Ihr Quelltext den Anforderungen genügt und ob Sie alle Klassen erstellt haben, etc. Nutzen Sie dies!
- Sollte das Übungssystem Ihre Lösungen ablehnen, dann werden sie *nicht* korrigiert!

## Akzeptanzkriterien:

- Compiliert erfolgreich
- Checkstyle bringt keine Fehler
- Alle Packages, Klassen, Interfaces, Methoden, Typen, Argumente sind exakt wie auf dem Übungsblatt gefordert.
- Ihr korrigierender Tutor wird die Korrektur Ihrer Abgabe in Ihr svn-Repository unter dem Namen `Feedback-<login>-ex<XX>.txt` comitten. (<login> ist dabei Ihr myAccount name und <XX> die Kennziffern des Übungsblatts)
- Zusätzlich können Sie Ihre Gesamtpunktzahl im Übungsportal einsehen.

Abgabe: Freitag, 21. Juni 2013, um 23.59 Uhr.

**Hinweise zum Eclipse Setup:** Um die Aufgaben dieses Blattes zu bearbeiten, müssen Sie in Eclipse eine Web-Entwicklungsumgebung konfigurieren. In der Vorlesung wurden die dazu notwendigen Schritte teilweise erklärt und es gibt eine Anleitung auf der Website der Vorlesung:

<http://proglang.informatik.uni-freiburg.de/teaching/java/2013/eclipse-jee.html>

Sollten die Projekte im Template in Eclipse Fehler anzeigen, dann überprüfen Sie die Projektkonfiguration, wie in der oben verlinkten Anleitung beschrieben.

**Achtung:** Sie müssen die Projektkonfiguration anpassen wenn sie das Template benutzen. Sonst erkennt Eclipse nicht, dass es sich um ein Dynamic Web Project handelt. Die notwendigen Schritte finden Sie auf der Anleitung zum Eclipse Setup unter dem Punkt “Projektsetup”.

**Aufgabe 1 (Teletype, 10 Punkte)**

Projekt: `ex07_1`. Package: `teletype`. Implementieren sie ein Servlet, welches es erlaubt, den Text, der in einem Writer-Client geschrieben wird, auf einen Reader-Client darzustellen.

Im Package `teletype.clients` befinden sich Beispielimplementierungen für die Clients, die aber noch keine Kommunikation vorsehen. Erweitern Sie Reader- und Writer-Client entsprechend der Aufgabe. Die Clients ähneln dem Chat-Client aus Aufgabe `ex07_1`; insbesondere implementiert der Writer-Client Tipp-Events.

Es sollen der Text im Editorfenster und die Tipp-Events vom Writer-Client auf den Reader-Client übertragen werden: der Reader-Client kann abfragen, ob der Writer gerade tippt und was bei ihm gerade im Editor-Fenster steht. Der Reader sollte diese Informationen korrekt, regelmäßig und halbwegs zeitnah anzeigen.

Sie müssen ihre Implementierung nicht mit JUnit testen; ausprobieren reicht.

**Hinweise:**

- Achten Sie auf eine robuste Kommunikation. Benutzen Sie die in der Vorlesung gezeigten Mittel zur Daten-Speicherung und halten Sie sich an die vorgestellten HTTP Konventionen.
- Im Template zu dieser Aufgabe finden Sie auch die Message Queue Implementierung MQ aus der Vorlesung. Sie können diese verwenden, die Aufgabe lässt sich aber auch ohne lösen.

- In der Vorlesung wurden Daten, die von Client zu Server gesendet werden, in der URL codiert. Dies ist für diese Aufgabe auch erlaubt. Da die Länge der URL aber im Allgemeinen beschränkt ist, können Sie alternativ auch auf dem folgendem Weg Daten zum Server übertragen:

---

```

1 // This is an excerpt of client code uploading data to a server over HTTP
2 HttpURLConnection con = ...;
3 con.setDoOutput(true); // enable upload to server
4 // create a PrintWriter to upload... it resides in java.io
5 PrintWriter w = new PrintWriter(con.getOutputStream());
6 // A PrintWriter can be used like System.out or response.getWriter()
7 w.println(...)
8 // ...
9 // after the upload is finished, we can get the response
10 System.out.println(con.getResponseCode());
11 // from here on, no more uploading is allowed

```

---

Eigentlich müssten für diese Upload-Methode weitere Vorsichtsmaßnahmen ergriffen werden, damit der Client den Server nicht übermäßig belasten kann. Wir sehen in dieser Aufgabe aber von solchen Problemen ab.

## Aufgabe 2 (GuessANumber, 11 Punkte)

Projekt: `ex07_2`. Package: `guessanumber`.

In dieser Aufgabe implementieren wir ein GuessANumber-Spiel: Ziel des Spiels ist es, eine Zahl zwischen 1 und 100 mit möglichst wenig versuchen zu erraten. Mögliche Antworten nach einem Rateversuch sind:

- Zahl ist zu groß
- Zahl ist zu klein
- Zahl ist korrekt

Das Projekt teilt sich in einen Server und einen Client auf. Der Server basiert auf einem Servlet, welches das Spiel modelliert und entsprechende Methoden zum Senden eines Rateversuches und zum Neustart des Spiels zur Verfügung stellt. Das Servlet muss nicht zwischen verschiedenen Verbindungen bzw. Clients unterscheiden.

Der Client ist für die Eingabe des Rateversuches, der Darstellung des Ergebnisses und der Kommunikation mit dem Server zuständig.

1. Implementieren Sie ein Servlet, welches eine zufällige Nummer zwischen 1 und 100 generiert (`new Random().nextInt(100) + 1`) und zwei verschiedene Anfragen versteht:
  - Rateversuch mit einer Zahl zwischen 1 und 100. Wenn die Zahl zu groß, zu klein oder richtig ist, wird eine entsprechende Antwort zurück gegeben.
  - Durch einen Neustart kann eine neue zufällige Zahl gesetzt werden.
2. Implementieren Sie ein Klasse, welche die Schnittstelle `IGuessANumberProvider` implementiert und eine Kommunikationsschnittstelle zum Server zur Verfügung stellt. Im Template zur Aufgabe finden Sie eine Klasse `guessanumber.client.GUI` welche eine GUI zur Verfügung stellt. Sorgen Sie dafür, dass die Methode `createProvider` in dieser Klasse eine Instanz Ihres `GuessANumberProvider` zurück gibt.

Sie müssen ihre Implementierung nicht mit JUnit testen; ausprobieren reicht.

## Zusatzaufgaben (ohne Punkte)

Wenn sie noch ein paar Herausforderungen brauchen, können Sie versuchen folgende Zusatzaufgaben zu lösen. Diese werden allerdings nicht korrigiert.

- Sorgen Sie dafür, dass mehrere Clients gleichzeitig gestartet werden können und diese mit unterschiedlichen Nummern spielen. Eine neue Sitzung und darin gespeicherte Werte können Sie mit folgendem Code anlegen:

---

```
1 HttpSession session = request.getSession();
2 request.getSession().putValue("key", value);
3 request.getSession().getValue("key");
```

---

Beim Client wird im Header ein Cookie gesetzt, welches sie bei der nächsten Anfrage wieder mitsenden müssen.

- Zählen Sie die Anzahl Versuche die Sie brauchen um die gesuchte Zahl zu finden und zeigen Sie diese an.
- Schreiben Sie einen GuessANumber-Solver, der automatisch mit möglichst wenig versuchen die Zahl ermittelt. (Einfach von 1-100 durchprobieren zählt nicht ;-))

## Aufgabe 3 (Logging, 12 Punkte)

Projekt: `ex07_3`. Package: `logging`.

Wir implementieren einen Datenbankserver der auf Servlets basiert. In der Datenbank werden Schlüssel-Wert-Paare gespeichert. Der Datenbankserver soll folgende Anfragen unterstützen:

- Schreiben eines Schlüssel-Wert-Paares (vorhandene Paare werden überschrieben)
- Lesen eines Werts indem der Schlüssel angegeben wird. Ist der Wert nicht vorhanden wird nichts zurück gegeben.
- Lesen aller gespeicherten Schlüssel
- Löschen der gesamten Datenbank

Im Template für die Aufgabe finden Sie eine Testklasse `DatabaseTest` und einem Datenbankprovider der die Verbindung zu den (noch zu implementierenden) Servlets herstellt. Implementieren Sie ein oder mehrere Servlets und eine Datenhaltung (z.B. mit Hilfe einer Map) damit die Tests durchlaufen.

Um besser den Überblick zu behalten was auf dem Server gerade passiert, bietet sich Logging<sup>1</sup> an. Ein Logger erlaubt es, Protokollnachrichten verschiedenen Schweregrades (Warnung, Fehler, Info) zu unterscheiden und auf die Konsole oder in eine Datei zu schreiben.

In der Klasse `database.server.LogServlet` finden Sie ein Beispiel. Ein Logger wird über ein statisches Feld pro Klasse definiert und kann dann Nachrichten über Aufrufe dieser Form protokollieren: `logger.info("This is a message text.");`

Wenn Sie das Template mit *Run on server* starten und das Servlet mit `http://localhost:8080/ex07_3-skel/log` aufrufen, dann sollten Sie in der Konsole in Eclipse unter anderem folgende Ausgabe sehen:

```
Jun 04, 2013 9:19:48 PM database.server.LogServlet doGet
INFO: This is a message text.
Jun 04, 2013 9:19:48 PM database.server.LogServlet doGet
WARNING: Be careful something strange has happened
Jun 04, 2013 9:19:48 PM database.server.LogServlet doGet
SEVERE: Some really nasty error occurred
```

Nutzen Sie Logging in Ihrer Implementierung!

---

<sup>1</sup><http://docs.oracle.com/javase/6/docs/technotes/guides/logging/overview.html>