Algorithmen und Programmierung V

WS 10/11

Klausur am 22.2.2011, 12.15-13.45 Uhr

Maximal erreichbare Punktzahl: 80 Erforderliche Punktzahl: 40

Die jeweilige *Punktzahl* entspricht in etwa dem angesetzten Bearbeitungsaufwand in *Minuten*.

Eine typische Aufgabenlösung umfasst nur wenige Zeilen. Verrennen Sie sich also nicht in lange Abhandlungen oder umfangreiche Programme! Der Platz auf dem Papier reicht aus. Verwenden Sie die Rückseiten für das „Schreiben ins Unreine“.

- Bitte schreiben Sie auf jedes Blatt Ihre Matrikelnummer.
- Bitte geben Sie die Klausur komplett und geheftet wieder ab.
- Bitte schreiben Sie mit schwarzem oder blauem Stift; Bleistift wird ignoriert.
- Eventuell erforderliches zusätzliches Papier steht zur Verfügung.

Aufgaben	**Punkte**	erreicht
Aufgabe 1 – Server-Programmierung	12	
Aufgabe 2 – Verteilungsabstraktion	16	
Aufgabe 3 – Java RMI	8	
Aufgabe 4 – HTTP und HTML	6	
Aufgabe 5 – Web-Technologien	8	
Aufgabe 6 – JavaScript	14	
Aufgabe 7 – PHP und JavaScript	16	
Summe:	80	

Aufgabe 1 – Server-Programmierung *(12 Punkte)*

Wir betrachten einen nichtsequentiell arbeitenden Server, der über TCP angesprochen wird und für jede neue Verbindung einen eigenen Thread zur Verfügung stellt. Formulieren Sie diesen Server als Java-main-Methode und vervollständigen Sie die angegebene Klasse! Das Verhalten der Threads ist durch die unten vorgegebene `run`-Methode bestimmt. (Jegliche Ausnahmebehandlung können Sie ignorieren.)

```
//  usage:  java Server port &
public class Server {
public static void main(String arg[]) { .....
```

```
    public void run() { ...
            InputStream in = socket.getInputStream();
            ...  }
```

Aufgabe 2 – Mangelhafte Verteilungsabstraktion *(16 Punkte)*

Wir betrachten eine Methode `f` in einer Java-ähnlichen Sprache, die neben Wertparametern *(call-by-value)* auch Variablenparameter *(call-by-reference)* kennt:

```
class C {
void f(A a, ref int i) { i++; i += a.get(); }
}
class A {
int value = 0;
int get() { return value; }
A test(C c) { c.f(this, value); return this; }
}
```

a ist ein Wertparameter, i ist ein Variablenparameter. Man überzeugt sich leicht davon, dass für beliebige C-Objekte c folgendes gilt:

```
new A().test(c).value == 2
```

Wie setzen nun voraus, dass unsere Sprache Fernaufrufe unterstütze, und zwar – anders als bei RMI – ohne dass am obigen Code etwas geändert werden müsste. Wenn c ein entferntes Objekt ist, ist `c.f` ein Fernaufruf und das anschließende `a.get` (als Callback) ebenfalls. In diesem Fall hat der obige Ausdruck einen anderen Wert als 2 – welchen und warum? *(Zur Erinnerung:* Ein Variablenparameter fungiert bei einem Fernaufruf als Wert-Ergebnis-Parameter *(call-by-value-result)* !)

Aufgabe 3 – Java RMI *(8 Punkte)*

Ich möchte in meinem Java-Code ein Objekt x mit Schnittstelle X lokal zur Hand haben. Ich kenne zwar keine Klasse, die X implementieren würde, wohl aber eine entfernte `factory` für *Serializable* Objekte mit Schnittstelle X; ich erhalte das gewünschte Objekt per Fernaufruf `x = factory.createX()`. Beschreiben Sie, was das Fernaufrufsystem unter Einsatz des Netzes im einzelnen leistet, damit mein Programm tatsächlich lokal a) über die Daten, b) über den Code des Objekts x verfügen kann!

Aufgabe 4 – HTTP und HTML *(6 Punkte)*

Sie füllen ein einfaches Web-Formular www.dot.com/form aus und schicken es mit *submit* ab. Sie erhalten eine *neue Seite*, stellen aber fest, das im Adressfeld des Browsers *keine Änderung* zu sehen ist, insbesondere auch nicht der erwartete *query string* erscheint. Sie fragen sich, ob das in Ordnung ist.

a) (4 P.) Welche naheliegenden Erklärungen gibt es dafür?
b) (2 P.) Wie können Sie Ihre Erklärungen überprüfen?

(Bemerkung: Es gibt auch weniger naheliegende Erklärungen. Wenn Sie sich dafür entscheiden wollen, nur zu!)

Aufgabe 5 – Web-Technologien *(8 Punkte)*

Wenn Sie eine lokal selbst erstellte Webseite mit einem Browser betrachten wollen, müssen Sie das nicht notwendig mit einer http://-URL über einen Server tun. Sie können auch mit einer file://-URL den Browser direkt auf die Datei verweisen. Die Frage ist allerdings, ob der Effekt in beiden Fällen genau der gleiche ist. Beantworten Sie diese Frage – *mit Begründung* – für Dateien der folgenden Arten:

a) .txt
b) .html
c) .php
d) .js

Aufgabe 6 – JavaScript *(14 Punkte)*

Eine Webseite enthält eine Textpassage, die nicht jeder gleich sehen soll (z.B. einer, der gerade einen Blick über meine Schulter wirft). Der HTML-Text enthält

```
Read me:     <font color="white" ...>
             ... (geheimer Text) ...
             </font>
```

und daher sieht man im Browser-Fenster nur dies:

```
Read me:
```

Der geheime Text soll sichtbar werden, wenn die Maus sich über ihm befindet. Außerdem soll beim Anklicken des Textes der Inhalt des Fensters durch ein Bild ersetzt werden (mit `<img src="...">`). Erweitern Sie das obige HTML-Fragment zu einem vollständigen HTML-Dokument!

Hinweise: `document.write(...)` veranlasst den Browser zur Umsetzung des als Argument angegebenen HTML-Textes (!). Eine Manipulation des DOM-Baums ist *nicht* erforderlich.

(Einschlägige Ereignisse für HTML-Elemente:

`onmouseover`	wenn die Maus ins Element kommt
`onmouseout`	wenn die Maus das Element verlässt
`onclick`	wenn das Element angeklickt wird.)

Aufgabe 7 – PHP und JavaScript *(16 Punkte)*

Die Besucher Ihrer dynamischen Webseite my.php sollen erfahren, wie viel Zeit (in Millisekunden) vom Beginn des Aufbaus der Seite beim Server *bis zum Ende des Aufbaus der visuellen Darstellung (!)* beim Browser vergangen ist. Dies kann mit einfachem PHP-Code in Verbindung mit JavaScript-Code erreicht werden. (Großzügigerweise setzen wir voraus, dass die Uhren beim Server und beim Browser synchron laufen.) Wie muss my.php aussehen, wenn die Zeitangabe entweder a) am Ende der Seite oder b) am Anfang der Seite (5 Zusatzpunkte!) plaziert werden soll?

Hinweis zu PHP: `time()` liefert die aktuelle Zeit in *Sekunden*;
zu JavaScript: `(new Date()).getTime()` liefert die aktuelle Zeit in *Millisekunden*. `parseInt(number,10)` liefert den `int`-Wert, der durch die Dezimalziffernfolge `number` dargestellt wird. `document.write(...)` veranlasst den Browser zur Umsetzung des als Argument angegebenen HTML-Textes (!).