Informatik II (D-ITET)
FS 2011
Dr. S. Santini
A. Bernauer (bernauer@inf.ethz.ch)

ETH Zürich
Institut für Pervasive Computing
Gruppe für Verteilte Systeme
http://www.vs.inf.ethz.ch

Übungsserie Nr. 1

Ausgabe: 02. März 2011
Abgabe: 09. März 2011

Unterlagen

Für diese Serie benötigen Sie das Archiv
http://www.vs.inf.ethz.ch/edu/I2/downloads/u1.zip

1. Aufgabe: (4 Punkte) Altägyptische Multiplikation

In der Vorlesung wurde die Korrektheit des altägyptischen Multiplikationsschemas für alle $a, b \in \mathbb{N}^+$ mittels Induktion über b bewiesen.

(1a) *(1 Punkt)* Könnte man die Korrektheit auch durch Induktion über a zeigen? Geben Sie eine Begründung für Ihre Antwort.

(1b) *(1 Punkt)* Beweisen Sie, dass der Algorithmus für alle erlaubten Eingabewerte terminiert.

(1c) *(2 Punkte)* Wie sind die Beweise aus a) und b) zu ändern, wenn das Programm auf Folie 1.34 des Skripts so modifiziert wird, dass es statt

```
if (b == 1) return a;
```

heisst:

```
if (b == 0) return 0;
```

2. Aufgabe: (6 Punkte) Laufzeitkomplexität

In der Vorlesung wurden folgende Methoden eines funktionalen Programms diskutiert:

```
static boolean gerade(int x) {
    if (x == 0) return true;
    return !gerade(x - 1);
}

static int verdopple(int x) {
    if ( x == 0 ) return 0;
    return 2 + verdopple(x-1);
}

static int halbiere(int x) {
    if (x == 0) return 0;
    if (x == 1) return 0;
    return 1 + halbiere(x - 2);
}

static int f(int a, int b) {
    if (b == 0) return 0;
    if (gerade(b)) return f(verdopple(a), halbiere(b));
    return a + f(verdopple(a), halbiere(b));
}
```

(2a) *(1 Punkt)* Wie oft rufen sich die Methoden *gerade*, *verdopple* und *halbiere* bei Eingabe eines Wertes x jeweils selbst rekursiv auf?

(2b) *(2 Punkte)* Wie viele Aufrufe der drei Methoden gibt es insgesamt in Abhängigkeit von *a* und *b* bei **einem** Aufruf von *f*? Ignorieren Sie dabei zunächst, dass f sich selbst rekursiv aufruft.

(2c) *(3 Punkte)* Nun sollen Sie berücksichtigen, dass $f(a, b)$ wiederum $f(2a, b/2)$ aufruft bis $b == 0$ erreicht wird. Schätzen Sie die Gesamtzahl aller Methodenaufrufe in Abhängigkeit von a und b unter Vernachlässigung von Rundungsfehlern ab.

3. Aufgabe: (6 Punkte) Überprüfung von Benutzereingaben

Im Skript auf Folie 1.33 wird der gültige Wertebereich für die Parameter der altägyptischen Multiplikation bewusst auf die positiven ganzen Zahlen beschränkt. Die Implementierung auf der folgenden Folie überprüft diese Einschränkung allerdings nicht. Bei einer versehentlichen Fehlbedienung ist daher unklar, was geschieht. Um so etwas zu vermeiden, sollte eine gute Implementierung ihre Eingaben auf ihre Gültigkeit hin überprüfen und ggf. definiert reagieren.

Im Paket *u1a3* befindet sich die Klasse `Mult` mit der bekannten Implementierung der altägyptischen Multiplikation in der privaten Funktion `f`. Die öffentliche Funktion `mult` ist dazu gedacht, die Eingaben zu überprüfen und nur gültige Werte an `f` weiter zu geben.

(3a) *(2 Punkte)* Erweitern Sie die Funktion `mult`, so dass beide Parameter auf ihre Gültigkeit hin untersucht werden. Bei ungültigen Parametern soll die Funktion eine `IllegalArgument-Exception` auslösen.

(3b) *(2 Punkte)* Die Implementierung von `f` enthält einen Fehler. Finden Sie diesen mit Hilfe der mitgelieferten Tests und korrigieren Sie ihn.

(3c) *(2 Punkte)* Neben der definierten Reaktion auf ungültige Eingaben braucht es eine passende Dokumentation, damit man die gültigen Werte nicht aus dem Code herauslesen muss. Java bietet für diesen Zweck Javadoc[1].

Dokumentieren Sie die Funktion `mult` mit Javadoc. Es muss dabei klar werden, was die Funktion berechnet, welche Parameter mit welchen Wertebereichen erwartet werden, was die Funktion zurück gibt und wann sie welche Ausnahmen auslöst. Hinweis: Ihre Java-Entwicklungsumgebung bietet Ihnen sehr wahrscheinlich Unterstützung beim Schreiben von Javadoc an.

Geben Sie ein Archiv des Packets *u1a3* mit Ihren Änderungen bei Ihrem Tutor ab.

[1] `http://java.sun.com/j2se/javadoc/writingdoccomments/index.html`