Lehrstuhl für Informatik VI
Prof. Dr. Frank Puppe
Am Hubland
97074 Würzburg

Dipl.-Inform. Rainer Herrler
Dipl.-Inform. Martin Schuhmann
swt@informatik.uni-wuerzburg.de

Softwaretechnik SS 2004

ÜBUNGSBLATT NR. 1
Ausgabe: Mi. 28.04.2004
Abgabe: Mi. 05.05.2002 (bis 10.00 Uhr)

VORBEMERKUNGEN:

1. Jeweils aktuelle Programmbeispiele und andere wichtige Informationen finden Sie auf unserer Softwaretechnik–Seite im Internet unter

http://ki.informatik.uni-wuerzburg.de/
→ Lehre: Vorlesungen (SS 2004)
→ Softwaretechnik bzw. Übungen zur Softwaretechnik

Bitte schauen Sie dort regelmäßig vorbei!

2. (Übungsblätter) Zur Vorlesung wird es jede Woche ein Übungsblatt geben. Das Aufgabenblatt finden Sie auf der Seite zu den Übungen zur Softwaretechnik (http://ki.informatik.uni-wuerzburg.de/teach/ss-2004/softwaretechnik/). Normalerweise ist dies ab Beginn der Vorlesung am Mittwoch zugänglich und ist bis zum Mittwoch der folgenden Woche zu bearbeiten. Mindestens 2 und maximal 3 Personen können ein Übungsblatt gemeinsam abgeben. Bitte suchen Sie sich frühzeitig Übungspartner – der Studierende neben Ihnen könnte bereits ein geeigneter Übungspartner sein ☺

3. (Abgabe Übungsblätter) Sofern nicht anders angegeben sind die Übungsblätter schriftlich abzugeben. Im Untergeschoß des Informatik-Gebäudes findet ihr die Briefkästen für eure Übungsgruppe. Jede Abgabe muss zusammengeheftet und deutlich mit Namen und Matrikelnummern der Bearbeiter gekennzeichnet sein.

ÜBUNG 1
8 Punkte (4+2+2)

a) In welchen der folgenden Programmierstilen (funktional, relational, regelbasiert, logikbasiert, imperativ, objektorientiert, Programmieren durch Beispiele) muss der Programmierer Kontrollstrukturen explizit oder implizit bzw. gar nicht festlegen?
b) Erläutern Sie, wie die Ablaufumgebung einer regelbasierten Sprache explizites Kontrollwissen für die Programm-Interpretation berücksichtigen kann.
c) Diskutieren Sie die Auswirkungen von expliziten Kontrollstrukturen auf die Eignung von visuellem Programmieren.

ÜBUNG 2
8 Punkte (2+4+2)

Gegeben sei folgende Funktion:

```
F(x) = 3*x*F(x-1) + (x-2),     für x > 0
F(x) = 1,                      für x ≤ 0
```

a) Schreiben sie ein funktionales JAVA-Programm, das diese Funktion mithilfe einer einfachen Rekursion berechnet
b) Formen sie diese Funktion um, so dass es eine end-rekursive Funktion darstellt.
c) Schreiben sie die Funktion als iteratives Programm mithilfe einer for-Schleife. Begründen sie, warum dieses Programm nicht mehr dem funktionalen Programmierparadigma entspricht.

ÜBUNG 3
8 Punkte (5+3)

a) Schreiben Sie ein funktionales Programm, das die Kubikwurzel nach dem Newtonschen Iterationsverfahren berechnet. (5)
b) Geben Sie alle Stellen an, wo in Ihrem Programm Komposition, Selektion und Rekursion verwendet wird. (3)

Def.: y ist die Kubikwurzel von x ($y = \sqrt[3]{x}$), wenn $y^3 = x$.
Vorgehen: 1. Setze geeigneten Startwert für y.
2. Für den Näherungswert y an die Kubikwurzel von x erhält man einen besseren Näherungswert mit der Formel:

$$\frac{x/y^2 + 2y}{3}.$$

ÜBUNG 4

10 Punkte (2+2+2+2+2)

Geben Sie die Zeitkomplexität folgender Funktionen in O-Notation an und begründen Sie diese. Machen sie sich dazu Gedanken, wie oft die Funktion maximal aufgerufen wird und setzen sie diesen Wert in Abhängigkeit zur Eingabegröße x.

a)
```
int f(final int x) {
   if (x < 0) return 1;
   else return x * f(x-2);
}
```

b)
```
int g(final int x) {
   if (x < 1) return 1;
   else return g(x/3) + g(x/3);
}
```

c)
```
int h(final int x) {
   if (x < 0) return 1;
   else return h(x-1) + h(x-2) + 3;
}
```

d)
```
int i(final int x) {
   if (x < 0) return 1;
   else return i(x-1) + i(x/2 - 1);
}
```

e)
```
int j(final int x) {
   if (x < 0) return 1;
   else return j(x/2) + j(2*(x/3)) + j(3*(x/4));
}
```