Ludwig-Maximilians-Universität München
Institut für Informatik
PD. Dr. Peer Kröger
Johannes Niedermayer, Andreas Züfle

München, 05.12.2012

Einführung in die Programmierung

WS 2012/13

Übungsblatt 7: Schleifen, Algorithmen

Besprechung: 12./14./17.12.2012

Ende der Abgabefrist: Dienstag, 11.12.2012 14:00 Uhr.

Hinweise zur Abgabe:
Geben Sie bitte Ihre gesammelten Lösungen zu diesem Übungsblatt in einer Datei `loesung07.zip` unter `https://uniworx.ifi.lmu.de` ab.

Aufgabe 7-1 *Terminierung von Schleifen*

Welche der folgenden Schleifen terminieren, welche terminieren nicht? Begründen Sie Ihre Antworten kurz. Geben Sie Ihre Lösung in einer Datei `schleifen.txt` oder `schleifen.pdf` ab.

(a)
```
int i = 1;
int j = 1;
do
{
  i = i + j;
  j++;
}while(i < 200);
```

(b)
```
int i = 1;
int j = 20;
while(i + j > i)
{
  i = i + 2;
  j--;
}
```

(c)
```
int i = 100;
int j = 27;
while (i != j)
{
  i = i / 20;
  j = j / 3;
}
```

(d)
```
int i = 26;
int j = 24;
for (int x = 0; x < 1000; x++)
{
  i = i / 12 + 23 * x;
  j = (x--) + j + 5;
}
```

Aufgabe 7-2 *Kontrollstrukturen* **8 Punkte**

Im folgenden sind zwei Programmausschnitte (a) und (b) gegeben. Betrachten Sie alle vorkommenden Variablen als deklariert und initialisiert.

(a)
```
int i = 0;
while(i != j) // Sei j >= 0
{
  i++;
}
```

(b)
```
while(a < b)
{
  c = a;
  a = b;
  b = c;
}
```

Überlegen Sie, was in den obigen Schleifen (a) und (b) passiert. Wie können Sie den gleichen Effekt ohne Verwendung von Wiederholungsanweisungen erzielen? Geben Sie ein entsprechendes Code-Fragment an.

(c) Schreiben Sie für die folgende **do**–**while**-Schleife jeweils eine äquivalente **while**- und **for**- Schleife.

```
int n = 1;
double x = 0;
double s;
do
{
  s = 1.0 / (n * n);
  x = x + s;
  n++;
} while (s > 0.01);
```

Geben Sie Ihre Lösungen als Datei `kontrollstrukturen.txt` oder `kontrollstrukturen.pdf` ab.

Aufgabe 7-3 *Algorithmen* **12 Punkte**

Ziel dieser Aufgabe ist es, eine Methode zu implementieren, die alle möglichen Ergebnissequenzen beim n-maligen Werfen eines k-seitigen Würfels auf der Konsole ausgibt, wobei eine Seite des Würfels zwischen 1 und k Augen hat. Die Reihenfolge der Ausgabe der Ergebnissequenzen spielt dabei keine Rolle.
Gehen Sie zur Lösung der Aufgabe folgendermaßen vor.

(a) Implementieren Sie in einer Datei `Wuerfeln.java` (die Datei kann auf der Website heruntergeladen werden) eine Methode `public static int exp(int b, int e)`, die den Wert von b^e berechnet. Eine Verwendung von `Math.pow()` ist in dieser Aufgabe *nicht* gestattet!

(b) Implementieren Sie die Methode `public static void muenzwurf(int n)` in Java, die alle möglichen Ergebnissequenzen beim n-maligen Werfen einer Münze (Also eines zweiseitigen Würfels) ausgibt.

Beispiel: Ein Aufruf der Methode `muenzwurf(3)` würde (die Reihenfolge der Zeilen spielt keine Rolle) folgende Ausgabe auf der Konsole erzeugen:
1,1,1
1,1,2
1,2,1
1,2,2
2,1,1
2,1,2
2,2,1
2,2,2

Hinweis: Verwenden Sie die Methode `System.out.print(int x)` um eine Zahl x auf der Konsole auszugeben. Die Methode `System.out.println()` kann verwendet werden um einen Zeilenumbruch zu erzeugen. Kommas können Sie mit dem Befehl `System.out.print(",")` ausgeben werden.

(c) Erweitern Sie die zuvor entwickelte Methoden in der Klasse `Wuerfeln.java` eine Methode `public static void wuerfeln(int n, int k)`, die alle möglichen Ergebnissequenzen beim n-maligen Würfeln eines k-seitigen Würfels ausgibt. Mögliche Seiten eines Würfels sollen in dieser Teilaufgabe die Beschriftung $1..k$ haben.

Beispiel: Die Ausgabe aller Sequenzen der Länge 3 eines zweiseitigen Wüfels (`wuerfeln(3,2)`) ist also äquivalent zur Ausgabe von `muenzwurf(3)`. Die Ausgabe aller Sequenzen der Länge 2 eines dreiseitigen Würfels (`wuerfeln(2,3)`) würde folgende Ausgabe erzeugen:
1,1
1,2
1,3
2,1
2,2
2,3
3,1
3,2
3,3

Geben Sie die bearbeitete Datei `Wuerfeln.java` ab. Achten Sie darauf, dass die abgegebenen Klassen kompilieren; nicht kompilierbare Klassen werden nicht bewertet!