Westfälische Hochschule

Prof. Dr. Manfred Meyer
Fachbereich 6 / Bocholt
Professur BWL / Wirtschaftsinformatik

Algorithmen und Datenstrukturen
(Studiengang Wirtschaftsinformatik 2. Semester)

Aufgabenstellung für das Praktikum am 13.4.2016

Aufgabe 1:

Erstellen Sie ein Java-Programm zur Ausgabe aller vollkommenen Zahlen (auch perfekte Zahlen genannt) zwischen 1 und einer einzugebenden Obergrenze n.

Testen Sie Ihr Programm und analysieren Sie es im Hinblick auf Terminierung, Korrektheit und Komplexität (Speicher- und Laufzeitverhalten in Abhängigkeit von n).

Aufgabe 2:

Erstellen Sie ein Java-Programm zur Berechnung aller Primzahlen zwischen 1 und einer einzugebenden Obergrenze n, das (nur ausnahmsweise und nur zu Übungszwecken!) nur partiell korrekt (also nicht total korrekt) ist.

Aufgabe 3:

Untersuchen Sie die folgenden in Java formulierten Algorithmen hinsichtlich Terminierung und Komplexität (Laufzeit und Speicherplatz). Was ist zu deren Korrektheit zu sagen?

Um Ihre Überlegungen oder Vermutungen zu bestätigen, messen Sie mit Hilfe der Methode `System.currentTimeMillis()` die Laufzeit der Algorithmen bei unterschiedlichen größeren Werten für n – für zu kleine Werte sind die Laufzeiten möglicherweise zu klein, so dass Sie keine hilfreichen Ergebnisse erhalten! Ermitteln Sie außerdem die Anzahl der Durchläufe insgesamt (Anzahl der Aufrufe der Operation **`System.out.println()`**) für unterschiedliche Werte für n.

a)
```
static void f1(int n)
{
  for (int i=0; i<n; i++)
  {
      for (int j=n; j>0; j--)
          System.out.println(j++);
  }
}
```

Hebt sich bei n < 0 gegenseitg auf und erzeugt somit eine Endlosschleife!

b)
```
static void f2(int n)
{
  for (int i=0; i<n; i++)
  {
      for (int j=n; j>0; j--)
          System.out.println(j+i);
  }
}
```

Algorithmus ist Korrekt und Terminiert sobald (i = n-1) erreicht.

c)
```
static void f3(int n)
{
  for (int i=0; i<n; i++)
  {
      int[] a=new int[n];
      while (i<n)
      {
          a[i]=i;
          System.out.println(i++)
      }
  }
}
```

Es wird lediglich jedes 2te Feld des Algorithmus gefüllt.
Er Terminierrt sobald i=n-1 erreicht.

d)
```
static void f4(int n)
{
  for (int i=0; i<n; i++)
  {
      for (int j=0; j<n; j++)
      {
          int[] a=new int[n];
          a[j]=j;
          System.out.println(j);
      }
  }
}
```

Algorithmus ist Korrekt und Terminiert sobald (i = n-1) erreicht.

e)
```
static void f5(int n)
{
  int i=0;
  int j=0;
  int k=0;
  while (i<n)
  {
      while (j<n)
      {
          while (k<n)
          {
              System.out.println(k++);
          }
          j++;
      }
      i++;
  }
}
```

Algorithmus ist Korrekt und Terminiert sobald (i=n-1) erreicht.

f)
```
static void f6(int n)
{
  int i=0;
  int j=0;
  while (i<n)
  {
      System.out.println(i++);
      i=i*j;
  }
}
```

Solange n!=0 ist wird die Schleife endlos lange wiederholt.
Dieser Algorithmus terminiert nie, da j=0!
(Ausser in o.g. Fall)

g)
```
static void f7(int n)
{
  int i=1;
  int j=1;
  int k=1;
  while (i<n)
  {
      while (j<n)
      {
          int[] a=new int[i*j*k];
          k=0;
          while (k<n)
          {
              System.out.println(k++);
              a[i-1]=j;
          }
          j++;
      }
      i++;
  }
}
```

In Annahme das n>1 ist läuft dieser Algorithmus 1mal durch und gibt "0" aus.
Im zweiten Durchlauf terminiert er aufgrund einer "ArrayOutOfBounds" Exception, da das Array hier eine länge von 0 hat. (2*2*0=0)

Viel Spaß und Erfolg bei der Bearbeitung der Aufgaben!