Mo., 4.6.2012 Informatik EF-M (Kr)

2. Klausur 12/II (EF.2)

Dauer: 2 Schulstunden Name: __www.r-krell.de___

Hilfsmittel: --

* *Achte auf sorgfältige Darstellung mit vollständigem, nachvollziehbarem Lösungsweg!* *
* *Kommentiere deine Programme!* *

```
for (int i = 0; i < n; i ++)
{
  for (int j = 0; j < n/2; j ++)
  {
    for (int k = 0; k < n; k++)
    {
      zeileStern(); // *
    }
  }
}
```

4 Gegeben ist nebenstehender sinnloser Programmtext. Entwickle nachvollziehbar eine Formel in Abhängigkeit von *n*, wie oft die Zeile * ausgeführt wird.

1 In einer Reihung stehen bis zu 50 Kommazahlen.

a) Schreibe den Anfang der Klasse *Aufgabe1*. Definiere darin eine geeigneten Reihe *reihe* sowie eine Variable *anzahl* für den tatsächl. Füllstand und eine Methode *fülleZufällig (z)*, die die ersten *z* Komponenten der Reihung jeweils mit einer zufälligen Kommazahl im Bereich von 0 bis 99.9999 füllt und *anzahl* entsprechend setzt (Hinweis: eine Kommazahl kann z.B. mit *double x = 100.0 * Math.random();* einen Zufallswert zwischen 0 und 99.9999 erhalten).

b) Schreibe eine Methode *istAbsteigend*, die genau dann den Wert *true* liefert, wenn die Kommazahlen in *reihe[0]* bis *reihe[anzahl-1]* absteigend sortiert sind, d.h. wenn die Zahlen immer kleiner werden oder höchstens gleich groß sind. Andernfalls soll der Rückgabewert *false* sein.

c) Jetzt sei *anzahl* = 6 und die *reihe* enthalte am Anfang 14.7 78.3 34.6 11.9 34.5 77.7. Sortiere diese Zahlen von Hand absteigend nach dem BubbleSort-Verfahren, mache jede Vertauschung deutlich und schreibe die Zahlen nach jedem Durchgang vollständig hin. Gib außerdem an, wie viele Vergleiche und wie viele Vertauschungen insgesamt ausgeführt wurden!

d) Gegeben ist nebenstehende Methode, die eine Hilfsmethode *tausche* aufruft.

```
public void aufgabe1d()
{
  for (int durchg=0; durchg < anzahl-1; durchg++)
  {
    for (int i=durchg+1; i < anzahl; i++)
    {
      if (reihe[i] > reihe[durchg])  // Vergleich
      {
        tausche (i, durchg);  // Vertauschung
      }
    }
  }
}
```

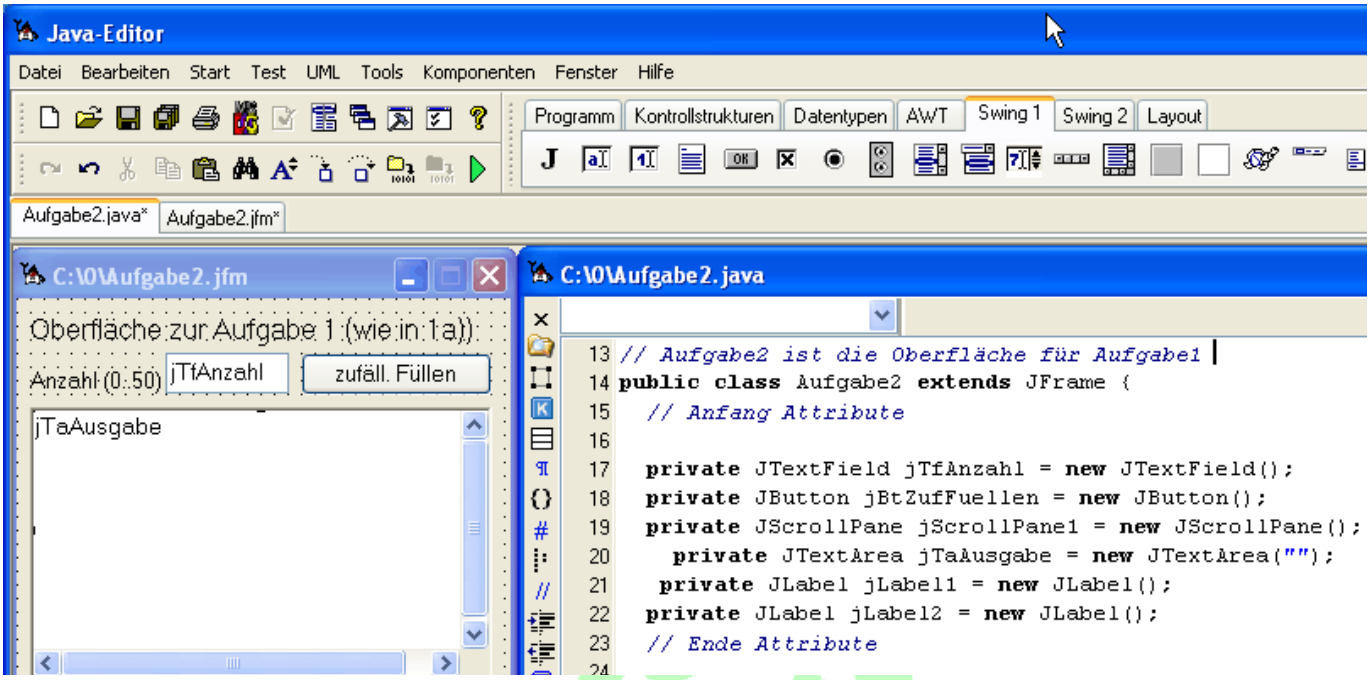
d1) Schreibe die Methode *tausche* (zum Vertauschen der zwei Kommazahlen an den angegebenen bzw. übergebenen Positionen innerhalb der *reihe* in der Klasse *Aufgabe1*)

d2) Führe für die Methode *aufgabe1d* mit den Werten aus 1c) einen Bleistifttest durch, d.h. gib die Belegung der Variablen *durchg, i* und *reihe* an und mache alle Vertauschungen deutlich.

d3) Nenne außerdem die Anzahl der Vergleiche sowie die Anzahl der Vertauschungen in d2), beschreibe die Grundidee von *aufgabe1d* in Worten und gib kurz begründet an, an welches der Verfahren aus dem Unterricht *aufgabe1d* am ehesten erinnert. Begründe außerdem, ob *aufgabe1d* oder das aus dem Unterricht bekannte ähnliche Verfahren besser ist.

e) Gib begründet an, ob die Methoden *istAbsteigend* aus 1b), *tausche* aus 1d1) und *aufgabe1d* am besten in eigenen, verschiedenen Klassen aufgeschrieben werden oder ob sie in eine gemeinsame Java-Klasse gehören (welche?, warum?).

2 In einem Objekt nach dem Bauplan von Aufgabe 1a) soll die *reihe* mit der Methode *fülleZufällig* per Knopfdruck aus der Swing-Oberfläche Aufgabe2 heraus gefüllt werden. Die Oberfläche wurde mit dem Java-Editor per drag & drop erzeugt:



a) Notiere zunächst nur, welcher Java-Text in Zeile 16 von Hand eingefügt werden muss, damit die Oberfläche ein Objekt nach dem Bauplan der Klasse Aufgabe1 verwenden kann.

b)

```
69  // Anfang Methoden
70  public void jBtZufFuellen_ActionPerformed(ActionEvent evt) {
71    // TODO hier Quelltext einfügen
72
73
74  }
75
76  // Ende Methoden
```

Notiere jetzt den Java-Text, der ab Zeile 72 von Hand eingefügt werden muss, damit durch Druck auf die Schaltfläche *jBtZufFuellen* das Füllen der Reihe mit der zuvor im Textfeld *jTfAnzahl* eingetippten Anzahl von Zufallszahlen ausgelöst wird und im Ausgabe-Bereich *jTaAusgabe* die Meldung „Reihung mit xx Zufallszahlen gefüllt“ (mit der richtigen Anzahl xx) erscheint.

3 Dass alle 50 Kommazahlen in *reihe* aus Aufgabe 1 nach zufälliger Füllung absteigend sind, ist ziemlich unwahrscheinlich. Aber unterschiedlich lange absteigende Teilstücke sind zu erwarten. Beispielsweise sind in 14.7 78.3 34.6 11.9 34.5 77.7 45.0 41.5 39. 4 8.5 26.7 22.3 81.3... in den unterstrichenen Stücken mal drei, mal fünf und dann zwei Zahlen absteigend sortiert (Stücke der Länge 1 sind immer absteigend sortiert). Eine Methode *längstesAbsteigendesStück* soll angeben, aus wie vielen Zahlen das größte absteigend sortierte Teilstück besteht (im Beispiel wäre das Ergebnis 5). Beschreibe in Deutsch, wie die Methode bei nur einem Durchgang arbeiten kann. Benutze dazu u.a. die Variablen *bisherLängstesStück* und *aktuelleLänge*, nachdem du erklärt hast, was darin gespeichert werden sollte (falls du noch Zeit hast, sind zusätzlich ein Struktogramm und/oder Javatext möglich).