Do., 26.11.2009 Informatik 12 M (Kr)

2. Klausur 12/I

Dauer: 3 Schulstunden (Abgabe 10:25 Uhr) Name: **www.r-krell.de**

Hilfsmittel: normaler Taschenrechner,

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

```
public class Aufgabe1
{
  Keller k = new Keller();
  Schlange s = new Schlange();

  public void geheimnis()
  {
    k.rein("Anna");     //1
    s.rein("Bernd");    //2
    k.rein("Claudia"); //3
    s.rein("Dennis");   //4
    s.rein(k.raus());   //5
    k.rein(s.zeige()); //6
    k.rein(s);          //7
  }
}
```

1 Gegeben ist nebenstehendes Programm. Zeichne *k* und *s* nach jeder Zeile mit ihren aktuellen Inhalten, sodass klar wird, wie sich Keller und Schlange verändern!

2 Keller mit Reihung

Geht man davon aus, dass ein Keller nie mehr als 29 Elemente aufnehmen muss, kann er auch mit einer Reihung *Object[] reihe = new Object[30];* realisiert werden. Im Unterricht hatten wir zwei Versionen überlegt: (1) das neueste/oberste Element kommt immer in *reihe[0]* oder (2) das älteste Element steht in *reihe[0]*.

a) Schreibe jeweils für beide Versionen den Anfang der Klasse Keller (nur mit den benötigten globalen Variablen/Attributen und der Methode *rein*)!

b) Begründe, welche der beiden Kellerimplementationen -- (1) oder (2) -- besser ist.

c) Sinnvollerweise füllt man im Konstruktor von Keller (1) das Feld *reihe[0]* schon mit einem Objekt wie etwa dem String "üü", der als Inhalt garantiert nicht vorkommt, der aber u.a. bei *rein* mit verschoben wird. Erläutere, wozu "üü" gebraucht wird!

3 Keller mit Knoten

a) Schreibe die komplette Klasse *Knoten* sowie den Anfang der Klasse *Keller* (nur alle globalen Variable und *rein*) in Java, wobei der Keller mit den Knoten arbeiten soll.

b) Vergleiche den Keller aus Aufgabe 3a mit dem besseren Keller aus Aufgabe 2 und bewerte!

c) Manchmal merkt man erst nach dem Einkellern eines Elements, dass der Vorgänger gar nicht hätte eingekellert werden dürfen. Für solche Fälle wäre eine Methode *public void löscheVorletzten( )* wünschenswert, die zusätzlich in die Kellerklasse aufgenommen werden soll. Schreibe die Methode in Java (du darfst darauf vertrauen, dass sie nur aufgerufen wird, wenn der Keller mindestens zwei Elemente enthält), wobei „Zeiger auf Knoten verbogen“ werden (bitte mit Skizze)

d) Manchmal würde man von einem Keller gerne die genaue Elementzahl wissen. Natürlich könnte man *rein/raus* so erweitern, dass immer ein globaler Zähler erhöht/erniedrigt wird. Hat man dies aber versäumt, muss auf Anforderung gezählt werden. Schreibe die entsprechende Methode *public int elementzahl()* wieder auf zwei Arten in Java

d1) mit Zugriff auf die Knoten

d2) nur unter Verwendung der Standardmethoden des Kellers (implementationsunabhängig)

4 Liste

Bei einer Liste kann auf jedes beliebige, mit *anDenAnfang* bzw. *weiter* ausgewählte Element zugegriffen werden.

a) Erläutere, warum wir mit *rein* ein neues Element lieber vor statt nach dem gerade ausgewählten aktuellen Element einfügen wollten (Dass anschließend das neu eingefügte Element aktuell sein sollte, muss nicht begründet werden).

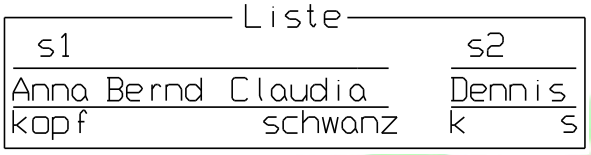
b) Eine Liste *li* soll schon die Namen "Anna" (Anfang), "Bernd", "Claudia" und "Dennis" enthalten. Dann

```
public void aufgabe4b()
{
  li.anDenAnfang();      //1
  li.weiter();           //2
  li.rein("Emre");       //3
  li.weiter();           //4
  Object x = li.raus(); //5
  li.rein(li.raus());    //6
}
```

wird das nebenstehende Programmstück ausgeführt. Beschreibe die Veränderung der Liste durch jede Zeile (bzw. zeichne die Liste nach jeder Änderung) und gib den Wert von *x* bei //5 an!

c) Die Liste soll jetzt mit zwei Schlangen *s1* und *s2* realisiert werden, wobei *weiter* immer das vorderste Element von *s2* hinten an die Schlange *s1* anfügt (d.h. *public void weiter () { s1.rein(s2.raus()); }*)

c1) Gezeichnet ist die Liste mit "Anna", "Bernd", "Claudia" und "Dennis". Begründe, ob im gezeichneten Zustand besser "Claudia" oder "Dennis" als aktuelles Element gelten soll. Beschreibe außerdem in Worten, wo/wie *zeige, rein* und *raus* funktionieren sollen.



c2) Schreibe den Beginn der Klasse *Liste* mit den benötigten globalen Variablen sowie nur mit den beiden Listen-Methoden *istLeer* und *anDenAnfang* in Java (für die Liste aus/mit zwei Schlangen)

d) Im Unterricht hatten wird die Liste direkt mit Knoten, einem Zeiger *voraktuell* und einem zusätzlichen Knoten mit Dummy-Element am Kopf der Liste realisiert. Überlege begründet, welche Listenimplementation -- die aus dem Unterricht oder die aus Aufgabe 4c -- besser ist.

e) Hätten statt zwei Schlangen wie in Aufgabe 4c auch ebenso gut oder besser/schlechter zwei Keller für die Liste verwendet werden können?

5 Zusatz: wie Aufgabe 3 c), wobei jetzt aber nur die Standardmethoden der Schlange (und keine implementationsabhängigen Details) verwendet werden