Mo., 19.3.2012 Informatik EF M (Kr)

1. Klausur EF.2 (11/II)

Dauer: 2 Schulstunden Name: ___www.r-krell.de____

Hilfsmittel: --

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

1 Reihungen I

a) Eine Reihung *codierungszeile* soll 24 Buchstaben aufnehmen. Definiere & erzeuge!

b) Gegeben ist *Luftballon[] nena = new Luftballon[99];*. Welche Klasse muss zur Verfügung stehen, was (und wie viel davon) kann diese Reihung offenbar aufnehmen und wie kann die vorderste und wie die letzte Komponente angesprochen werden?

c) In einem Kino gibt es 24 Sitzreihen (Reihen 0 bis 23) mit jeweils 42 Sesseln. Von jedem Sessel interessiert, ob er entweder leer oder besetzt ist. Definiere und erzeuge eine geeignete Reihung und schreibe nur den Befehl, um den 19. Sessel in der 7. Reihe zu besetzen.

d) In einem Krankenhaus gibt es 5 Etagen (vom Erdgeschoss [0. Stock] bis zum 4. Stock). Auf jeder Etage sind 2 Stationen. In jeder Station gibt es neun Patientenzimmer und in jedem Zimmer können 3 Betten stehen. Jedes einzelne Bett kann entweder 'f' = fort, 'l'=leer oder von 'm' = einem männlichen Patienten bzw. von 'w' = einer weiblichen Patientin belegt sein.

d1) Definiere die mehrdimensionale Reihung *bettBelegung*, die für jedes der 270 Betten im Krankenhaus den passenden Kennbuchstaben aufnehmen kann.

d2) Das mittlere Bett im 3. Stock, Station 0, Zimmer 4 wurde gerade zum Abziehen und Reinigen in den Keller gerollt, weil die Patientin entlassen werden konnte. Schreibe den einen Java-Befehl, um den Kennbuchstaben für dieses Bett richtig zu setzen!

d3) Die Krankenhausverwaltung möchte gerne wissen, wie viele männliche Patienten gerade Betten belegen. Schreibe eine Methode *public int anzahlMännlPatienten()*.

2 Bei der Fußball-Bundesliga-Tabelle stehen in 18 Zeilen jeweils der aktuelle Tabellenplatz, der Vereinsname, das Torverhältnis und die Punkte. Vor diesem Wochenende war der Tabellenanfang wie abgebildet:

1	Borussia Dortmund	52 : 16	56
2	FC Bayern München	58 : 17	51
3	Borussia Mönchengladbach	37 : 15	48

Die ganze 18-zeilige Tabelle soll in der Klasse *Tabelle* in einer oder in mehreren Reihungen untergebracht werden:

```
public class Torverhaeltnis
{
  int ToreGeschossen;
  int ToreVonGegnern;

  public String zeige()
  {
    return (""+ToreGeschossen
      +" : "+ToreVonGegnern);
  }
}
```

a) Variante I: Für jede der 4 gezeigten Spalten wäre eine eigene eindimensionale Reihung möglich. Definiere die 4 Reihungen!

b) Variante II: Entscheide begründet, ob eine 2-dimensionale Reihung (..[4][18]) möglich ist.

c) Variante III: Die Daten einer/jeder Zeile stehen in einem Objekt der Klasse *Tabellenzeile*. Schreibe diese Klasse in Java mit vier Attributen, einem Konstruktor und einer Methode *zeige*, die den Inhalt aller Attribute als einen Text (also in einer Textzeile) zurück gibt.
Definiere und erzeuge dann in der neuen Klasse *Tabelle* eine Reihung für 18 *Tabellenzeile*n!

d) Vergleiche die möglichen Darstellungen und gib begründet an, welche Variante besser ist.

3 Ein Schreibwarenladen möchte noch ein paar billige Schreibgeräte (Stifte, Kulis, Füller,..) ins Angebot aufnehmen.Vom Großhändler wurde eine Katalogdatei herunter geladen, die aber auch teurere Schreibgeräte enthält. Nur die billigen Schreibgeräte (mit einem Preis bis max. 9.50 €) sollen vom Datenträger in die Reihung *preiswert* aufgenommen werden (vgl. nächste Seite).

```
public class Schreibwarenladen
{
 Schreibgeraet[] preiswert = new Schreibgeraet[999];
 int anzahl = 0;

 public void löscheErste16() // für Aufg. 3d) und 4
 { ...
```

```
public class Schreibgeraet implements
 java.io.Serializable
{
 String hersteller; // z.B. "Lamy"
 String name; // z.B. "A 231"
 String system; // z.B. "Filzer"
 double preis; // z.B. 12.99
}
```

a) In der Katalogdatei „gemischt.dat“ sind teure und billige Schreibgeräte wild durcheinander. Schreibe die Methode *ladeBilligeA*, die nur die billigen Schreibgeräte Stück für Stück in *preiswert* füllt und *anzahl* dabei entsprechend erhöht.

b) Jetzt soll die Katalogdatei „sortiert.dat“ aufsteigend nach Preisen geordnet sein: das erste Schreibgerät kostet nur 0.15 €, das letzte, teuerste ist ein Cheffüller für 349 €. Schreibe für diese Datei eine etwas schneller ausführbare Methode *ladeBilligeB*, die wieder nur die billigen Schreibgeräte von Anfang an (Platz 0) in die Reihung *preiswert* lädt.

c) Gib mit kurzer Begründung an, ob *ladeBilligeA* und *ladeBilligeB* besser in getrennte, eigene oder in eine neue gemeinsame oder in eine der beiden angegeben Klassen (welche?) geschrieben werden sollten!

d) Der Geschäftsführer des Schreibwarenladens meint, dass Schreibegräte unter 1.50 € nicht lohnen. Deshalb will er sie wieder aus dem Angebot nehmen. In der [wie in b) gefüllten] Reihung *preiswert* stehen die 16 ganz billigen Geräte alle hintereinander am Anfang. Sie sollen dort ersatzlos gelöscht werden, wobei die restlichen Schreibgeräte aufrücken müssen. Schreibe die Methode *löscheErste16*.

e) Beschreibe nur in Deutsch ohne Programmtext, wie vorzugehen ist, wenn die 16 ganz billigen Geräte irgendwie wahllos über *preiswert* verteilt sind [wie nach a)]. Vergleiche außerdem den Aufwand des neuen Verfahrens mit dem aus d).

4 Um die Aufgaben 3b) und d) leicht ausführen zu können, wird eine zusätzliche Klasse *SchreibwarenGUI* geschrieben mit nebenstehendem Aussehen. Zu den Attributen dieser Oberflächen-Klasse wurde *Schreibwarenladen sl = new Schreibwarenladen();* hinzu gefügt. Beim Druck auf den zweiten Knopf (mit Namen *jBtLade* und der Beschriftung „Lösche erste 16“) soll die entsprechende Methode der Schreibwarenladen-Klasse ausgeführt und der Erfolg im Textfeld *jTfAusgabe* angezeigt werden. Ergänze dazu

public void jBtLade_ActionPerformed(ActionEvent evt) {

5 Reihungen II (soweit noch Zeit)

Für den Test eines Würfels soll 10000 mal gewürfelt werden (simuliert durch *Hilfe.zufall(1,6);*) und anschließend auf der Konsole ausgegeben werden, wie oft jede der sechs Zahlen 1 bis 6 kam.

a) Schreibe ein vollständiges Programm (aber ohne Startdatei) mit Reihung

b) Erläutere den Vorteil einer Reihung. Wie hätte man ohne Reihung vorgehen müssen?

Anlage
(wird auch in der Klausur beigefügt)

Mo., 5.3.2012 Informatik IF EF M (Kr)

Typische Konstrukte für das Schreiben und Lesen von (Objekt-)Dateien

Hier geht's um Dateien für beliebige Objekte, z.B. um Schueler (wobei die Klasse *Schueler* als Bauplan für die in der Datei gespeicherten oder zu speichernden Objekte mit dem Zusatz **implements java.io.Serializable** definiert werden muss – sofern die Klasse noch weitere Unterklassen benutzt, müssen auch diese den Zusatz **implements java.io.Serializable** erhalten!).

Zu Beginn der Klassen (z.B. der Klasse *SchuelerListe*) mit den folgenden Operationen ist **import java.io.*;** nötig!

Schreiben (auf Disk)

```
try
{
 ObjectOutputStream datei = new ObjectOutputStream(
   new FileOutputStream ("name.dat"));

 //folgenden Befehl wiederholen, um mehrere Objekte zu
 //schreiben (person sei ein Objekt vom Typ Schueler)

 datei.writeObject (person); //schreibt ein Objekt



 datei.close();
}
catch (IOException ex)
{..}
```

Lesen (von Disk)

```
try  //Fehler, falls Datei nicht vorhanden oder close ver-
sagt
{
 ObjectInputStream datei = new ObjectInputStream(
   new FileInputStream ("name.dat"));
 try //Fehler, wenn Datei zu Ende
 {
   //folgenden Lesebefehl immer wieder wiederholen:
   person = (Schueler) datei.readObject();  //liest Objekt
 }
 catch (EOFException eex) //EOF = End of File =..
 {                        //..= Dateiende
   datei.close();  //Datei schließen, da zu Ende
 }
}
catch (Exception ex) //für IO- oder ClassNotFound-Ex.
{..}
```

Die Art der Wiederholstrukturen richtet sich danach, wie die einzelnen Komponenten zur Verfügung stehen. Passende Variable (wie *person*) müssen vorher definiert (und vorm Schreiben mit Inhalt gefüllt) sein, während sie beim Lesen gefüllt werden.