Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr
2007

46114

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Algorithmen, Datenstrukturen und Programmiermethodik**

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 9

Bitte wenden!

Thema Nr. 1

Sämtliche Teilaufgaben sind zu bearbeiten!

Aufgabe 1:

Das Java-Programm `Converter` hat die Aufgabe, Dezimalzahlen als Zahlen zur Basis 16 darzustellen. Dabei wird die Dezimalzahl fortgesetzt ganzzahlig durch 16 dividiert, der jeweilige (eventuelle) Rest ist der Wert einer Ziffer der gesuchten Darstellung.

a) Finden Sie die Fehler in der Klasse `Hex` und korrigieren Sie diese, indem Sie jeweils die fehlerhafte(n) Zeilennummer(n) und den zugehörigen korrigierten Quelltext angeben und jeweils kurz erläutern, worin der Fehler besteht.

b) Überarbeiten Sie das Programm dahingehend, dass die Ausgabe der Ziffern in der richtigen Reihenfolge geschieht.

Quelltext `Converter.java`

```
1   class Hex {
2       char[] ziffern = {'0','1','2','3','4','5','6','7',
3                         '8','9','A','B','C','D','E','F'};
4       int h = 16;
5       void out(int z) {
6           int rest;
7           while ( z > 1 ) {
8               rest = z / h;
9               z    = z % h;
10              System.out.println(z);
11          }
12      }
13  }
14
15  class Converter {
16
17      public static void main(String[] args) throws IOException {
18
19          int zahl;
20          BufferedReader stdin =
21            new BufferedReader(new InputStreamReader(System.in));
22          String inData;
23
24          System.out.println("Gib mir eine Zahl:");
25          inData = stdin.readLine();
26          zahl   = Integer.parseInt(inData);
27
28          Hex hex = new Hex();
29          hex.out(zahl);
30      }
31  }
```

Fortsetzung nächste Seite!

Aufgabe 2:

Betrachten Sie das unten angegebene Java-Programm.

a) Welche Aufgaben haben die Methoden `foo_a()`, `foo_b()` und `foo_c()` bei der Lösung des Gesamtproblems?

b) Geben Sie das Abbruchkriterium für die Rekursion in `foo_c()` an.

c) Wie lautet der Inhalt des Arrays `data` nach dem ersten Aufruf der Methode `foo_b()`?

d) Welches Verfahren wird hier umgesetzt? Erklären Sie den Algorithmus anhand des Quellcodes.

Quelltext `Aufgabe.java`

```
1   class Meldung {
2       static void print_r(int[] arr){
3           int i;
4           for (i = 0; i < arr.length; i++){
5               System.out.print(arr[i] + ", ");
6           }
7           System.out.println();
8           }
9   }
10
11  class Algorithm {
12
13      void foo_a (int[] arr, int n, int m) {
14          int h;
15          h = arr[m];
16          arr[m] = arr[n];
17          arr[n] = h;
18      }
19
20      boolean foo_b (int[] data){
21          int i;
22          boolean doit = false;
23          for (i = 0; i < data.length - 1; i++){
24              if (data[i+1] < data[i]){
25                  doit = true;
26                  foo_a(data,i,i+1);
27              }
28          }
29          Meldung.print_r(data);
30          return doit;
31      }
32
33      void foo_c (int[] data){
34          if (foo_b(data)) {
35              foo_c(data);
36          }
37      }
38  }
39
40  class Aufgabe {
41
42      public static void main(String[] args) {
43          int[] data = {23,38,14,-3,0,14,9,103,0,-56};
44          Algorithm algorithm = new Algorithm();
45          algorithm.foo_c(data);
46      }
47  }
```

Fortsetzung nächste Seite!

Aufgabe 3:

Eine Bank verfügt über zwei Arten von Konten: Sparkonten, die keinen negativen Stand haben dürfen, und Girokonten, die überzogen werden dürfen. Der Aufruf zum Auszahlen von einem Konto soll für Spar- und Girokonten einheitlich realisiert werden. Alle Konten werden in einer gemeinsamen Liste verwaltet, die z. B. als Reihung (Array) realisiert ist. Die Liste verfügt über eine Methode, die zu einer gegebenen Kontonummer das zugehörige Konto findet.

a) Geben Sie ein Klassendiagramm mit allen für die Aufgabenbeschreibung relevanten Attributen und Methoden sowie die Kardinalitäten an.

b) Implementieren Sie die Methode, die zu einer gegebenen Kontonummer das zugehörige Konto findet (als binäre Suche).

c) Geben Sie ein Hauptprogramm an, das zur Kontonummer 1234567 das zugehörige Konto findet und bei diesem eine Auszahlung von 100 EUR vornimmt. Dabei soll keine explizite Falluntersuchung zwischen Spar- und Girokonten erforderlich sein.

Aufgabe 4:

Erläutern Sie, wie Sie ein Klassendiagramm des objektorientierten Entwurfs (z.B. UML-Klassendiagramm) in Code einer objektorientierten Programmiersprache Ihrer Wahl übersetzen. Berücksichtigen Sie hierbei folgende Elemente von Klassendiagrammen: Klassen, Attribute, Methoden, Vererbungsbeziehungen, Aggregationen, Assoziationen. Unterscheiden Sie bei Assoziationen die Fälle 1:1-, 1:n- und n:m-Beziehung.

Aufgabe 5:

Gegeben seien zwei Beispiele für einfache arithmetische Ausdrücke, in denen Variablen $a, b, c, \ldots$ sowie Operatoren $+, -, *, /$ vorkommen und die vollständig geklammert sind.

1. $((a + b) * c)$
2. $((x * y)/(x - y))$

a) Geben Sie ein Verfahren an, das mit Hilfe eines Stapels (Stack) beliebige, wohlgeformte arithmetische Ausdrücke auswertet. Verwenden Sie eine gängige höhere Programmiersprache Ihrer Wahl oder einen Pseudocode. Gehen Sie davon aus, dass die Stapeloperationen Push und Pop zur Verfügung stehen.

b) Für das 1. Beispiel soll das Verfahren folgende Belegungen des Stapels erzeugen. (Eine Stapelbelegung steht in einer Zeile, das oberste Stapelelement steht rechts.)

a	
$a+$	
$a + b$	
Z_1	($a + b$ wird zu Z_1 ausgewertet)
Z_1*	
$Z_1 * c$	
Z_2	($Z_1 * c$ wird zu Z_2 ausgewertet)

Welche Folge von Belegungen des Stapels wird für das 2. Beispiel erzeugt?

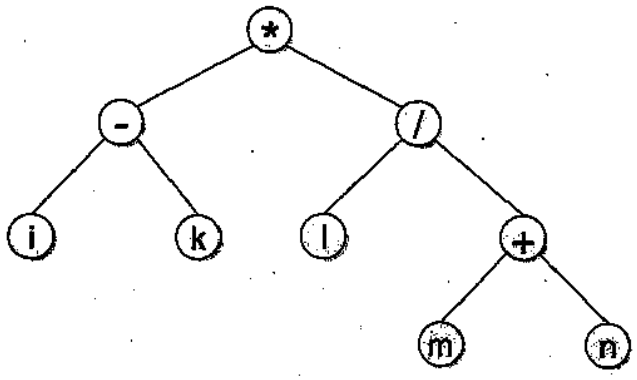
Fortsetzung nächste Seite!

Aufgabe 6:

Ein arithmetischer Ausdruck lautet in Infixnotation $((a + b) * c)$ und in Postfixnotation $ab + c*$. Bei der Postfixnotation kann auf Klammern verzichtet werden.

a) Geben Sie den Ausdruck $((x * y)/(x - y))$ in Postfixnotation an.

b) Ein arithmetischer Ausdruck kann in einem Binärbaum gespeichert werden. Traversieren Sie



den Binärbaum in

- Inorder,
- Postorder.

Geben Sie jeweils den arithmetischen Ausdruck an, der ausgegeben wird. Begründen Sie, ob und nach welcher Strategie Klammern gesetzt werden.

c) Geben Sie ein Verfahren an, das einen arithmetischen Ausdruck, der als Binärbaum vorliegt, in

- Infixnotation,
- Postfixnotation.

ausgibt. Verwenden Sie eine gängige höhere Programmiersprache oder einen entsprechenden Pseudocode.
Hinweis:
Sie können wahlweise davon ausgehen, dass der Binärbaum als Reihung (Array) oder mit Zeigern realisiert ist und die entsprechenden Operationen zur Verfügung stehen. Achten Sie darauf, dass Klammern gesetzt werden, falls erforderlich.

Thema Nr. 2

Sämtliche Teilaufgaben sind zu bearbeiten!

Aufgabe 1:

In der folgenden Aufgabe soll ein einfaches Programm zur Verwaltung eines Fußballvereins entwickelt werden.

Entwickeln Sie zunächst die Klasse Spieler. In Objekten dieser Klasse soll die folgende Information gespeichert sein:

*Name
*Vorname
*Geburtsdatum
*Anzahl gespielter Spiele
*Anzahl geschossener Tore
*Spielernummer

Die Klasse soll einen Konstruktor besitzen, der ein neues Objekt anlegt. Dem Konstruktor soll der Vor- und Nachnahme des Spielers und dessen Geburtsdatum übergeben werden. Die Spielernummer soll vom Programm selbst vergeben werden. Dabei muss darauf geachtet werden, dass kein Spieler eine bereits vergebene Nummer erhält.

Außerdem soll die Klasse eine Methode besitzen, die die Zahl der Spiele um eins erhöht. Dieser wird die Anzahl der Tore übergeben, die der Spieler in diesem Spiel erzielt hat. Zuletzt soll die Klasse noch über eine Methode verfügen, die alle relevanten Informationen über den Spieler in einer Zeile ausgibt. Dazu gehören: Name, Vorname, Geburtsdatum, Anzahl Spiele, Anzahl Tore und die durchschnittliche Anzahl von Toren pro Spiel.

A) Spezifizieren Sie die Klasse Spieler, indem Sie die Signaturen des Konstruktors und der Methoden in einer objektorientierten Programmiersprache Ihrer Wahl angeben. Überlegen Sie sich außerdem, welche Werte für die jeweiligen Parameter zulässig sind, bzw. bei welchen Parametern eine Ausnahme generiert werden soll. (15 Minuten)

B) Implementieren Sie diese Klasse. Achten Sie darauf, dass die Implementierung zu der von Ihnen erstellten Spezifikation aus Teilaufgabe A passt. Insbesondere sollen ungültige Parameter erkannt und mit einer Ausnahme behandelt werden. (20 Minuten)

Des Weiteren soll eine Klasse Mannschaft implementiert werden. Eine Mannschaft besteht aus bis zu elf Stammspielern und einer beliebigen Anzahl von Ersatzspielern. Sie soll über einen Konstruktor verfügen, der eine leere Mannschaft erzeugt. Mit Hilfe einer Methode sollen der Mannschaft Spieler hinzugefügt werden. Hat die Mannschaft weniger als 11 Stammspieler, wird der neu hinzugefügte Spieler zum Stammspieler, andernfalls zum Ersatzspieler. Eine zweite Methode dient dazu, einen Ersatzspieler mit einem Stammspieler zu tauschen: Der Stammspieler wird zum Ersatzspieler und umgekehrt.

Zuletzt soll eine Methode entwickelt werden, die alle Informationen über die Mannschaft in einer Tabelle auflistet.

Fortsetzung nächste Seite!

Dazu werden zunächst alle Stammspieler aufgelistet, anschließend alle Ersatzspieler. Außerdem wird zuletzt die durchschnittliche Anzahl von Toren pro Spieler und Spiel ausgegeben.

C) Spezifizieren Sie die Klasse Mannschaft (analog zu A). (15 Minuten)

D) Implementieren Sie die Klasse entsprechend ihrer Spezifikation. Erweitern Sie evtl. auch die Klasse Spieler, falls dies notwendig sein sollte. (20 Minuten)

Aufgabe 2:

Arithmetisches Mittel eines Feldes

Gegeben sei ein Feld von Fließkommazahlen der Länge n. Gesucht ist das arithmetische Mittel M dieser Zahlen, definiert durch:
M = Summe der Elemente / n

Für diese Aufgabe sollen zwei Algorithmen entwickelt und anschließend miteinander verglichen werden.

A) Entwickeln Sie zunächst die Funktion MIterativ, welche die Aufgabe mit einem iterativen Programmieransatz löst. (10 Minuten)

B) Entwickeln Sie die Funktion MRekursiv, welche die Aufgabe mittels einer rekursiven Funktion löst, die das Feld in jedem Rekursionsschritt in zwei Hälften teilt, jeweils das Mittel der beiden Hälften berechnet und anschließend das Mittel der beiden Ergebnisse bildet. Überlegen Sie sich zunächst ein geeignetes Rekursionsende. Der Einfachheit halber dürfen Sie annehmen, dass die Länge des Feldes eine Zweierpotenz ist. Falls dies nicht der Fall ist, soll eine Fehlermeldung ausgegeben werden. (20 Minuten)

C) Entwickeln Sie ein Hauptprogramm, das

* solange Werte von der Standardeingabe liest, bis das Dateiende erreicht wird,
* diese in einem Feld ablegt,
* das arithmetische Mittel mit Hilfe der obigen Funktionen ermittelt,
* dieses auf der Standardausgabe ausgibt.

Hinweis:
Sie dürfen auf die Teilergebnisse A und B auch dann zurückgreifen, wenn Sie diese Aufgaben noch nicht vollständig gelöst haben. (20 Minuten)

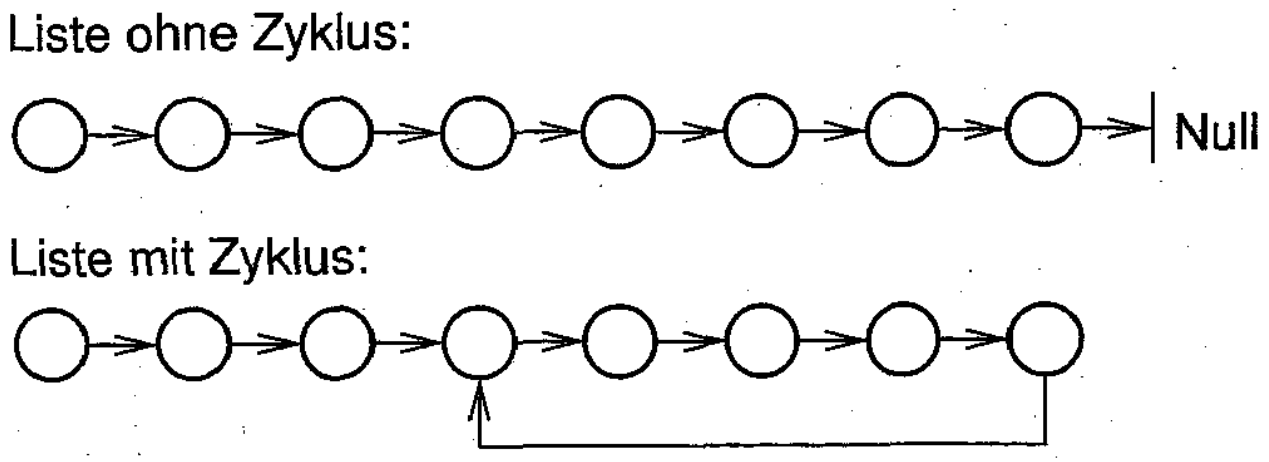
D) Vergleichen Sie die Lösungen aus Aufgabe A) und B).
Welche Vor- und Nachteile gibt es?
Für welche Lösung würden Sie sich entscheiden? Warum? (10 Minuten)

Fortsetzung nächste Seite!

Aufgabe 3:

Zyklen in verketteten Listen

Gegeben sei eine verkettete Liste, die einen Zyklus enthalten kann. Der Hase-Igel-Algorithmus überprüft, ob dies der Fall ist und soll im Folgenden implementiert werden.



Beim Hasen bzw. Igel handelt es sich jeweils um eine Referenz auf ein Listenelement. Beim Durchlauf durch die Liste wird der Igel um jeweils eine Position, der Hase jedoch um zwei Positionen vorgerückt.

Die Liste bietet also folgende Methoden an:

boolean kannIgel();//	prüft, ob der Igel um eine Position vorrücken kann oder ob das Ende der Liste bereits erreicht ist.
boolean kannHase();//	prüft, ob der Hase um zwei Positionen vorrücken kann oder ob das Ende der Liste bereits erreicht ist.
ListItem bewegeIgel();//	rückt den Igel um eine Position vor und liefert das entsprechende Listenelement. Beim ersten Aufruf wird das erste Listenelement geliefert.
ListItem bewegeHase();//	rückt den Hasen um zwei Positionen vor und liefert das entsprechende Listenelement. Beim ersten Aufruf wird das zweite Listenelement geliefert.

A) Implementieren Sie die oben angegebenen Methoden der Klasse List. Gehen Sie dabei davon aus, dass die Klasse über zwei Referenzen auf Objekte vom Typ ListItem verfügt, die den Hasen und den Igel implementieren. Zu Beginn sind beide Referenzen mit null initialisiert. Außerdem gibt es noch die Referenz start, die auf das erste Element der Liste zeigt. Die Klasse ListItem können Sie als gegeben betrachten. Sie verfügt über folgende Methode:

ListItem next();// liefert das nächste Listenelement oder null, falls das Ende der Liste erreicht ist.

Andere Klassen brauchen/dürfen nicht verwendet werden. (20 Minuten)

Fortsetzung nächste Seite!

B) Beim Hase-Igel-Algorithmus wird der Igel auf das erste Listenelement gesetzt, der Hase auf das zweite. Hase und Igel rücken dann jeweils zwei bzw. ein Feld vor. Der Algorithmus terminiert, falls:

- einer der Referenzen das Listenende erreicht. In diesem Fall besitzt die Liste keinen Zyklus.
- der Igel vom Hasen eingeholt wird, d.h., beide Referenzen zeigen auf dasselbe Listenelement. In diesem Fall besitzt die Liste einen Zyklus.

Implementieren Sie den Hase-Igel-Algorithmus unter Verwendung der Ergebnisse aus A! (20 Minuten)

C) Wie ist die Zeitkomplexität des Hase-Igel-Algorithmus?
Wie ist die Speicherkomplexität? (10 Minuten)