Objektorientierte Programmierung & Modellierung

Universität Koblenz-Landau
Institut für Informatik

Prof. Dr. Ralf Lämmel
Dr. Manfred Jackel

WS 2007/8

1. Teilklausur

07.12.2007

Gruppe B

Bitte in Druckschrift leserlich ausfüllen!

Name

Vorname

E-Mail-Adresse @uni-koblenz.de

Matrikelnummer

Studiengang:
☐ Computervisualistik (Diplom)
☐ Informatik (Diplom)
☐Computervisualistik (BSc)
☐Informatik (BSc)
☐ Anglistik & Medienmanagement (BSc)
☐ B Ed _____
☐ Informationsmanagement (BSc)
☐ Diese Prüfungsleistung melde ich verbindlich als Freiversuch im Sinne der Prüfungsordnung an (nur alte PO).
☐ Diese Prüfungsleistung ist mein 2. Versuch (Nachklausur). Erstversuch im _____ (Semester).

Auswertung:

	1	2	3	4	GESAMT
Punkte					

Aufgabe 01 (12 Punkte)

Diese Aufgabe umfasst 4 Multiple-Choice Cluster mit je 4 Ankreuzfragen. Für jedes Cluster gilt: wenn alle 4 Kreuze an der richtigen Stelle stehen, gibt es 4 Punkte für das Cluster. Ein falsches Kreuz gibt einen Punkt Abzug. Wer 2 richtige und 2 falsche Kreuzchen in einem Cluster macht, erhält 1+1-1-1=0 Punkte. Zum Trost: es gibt keine negativen Gesamtpunktzahlen, jedes Cluster bringt 0 bis 4 Punkte.

		Ja	Nein	Frage
a)				**Grundlagen**
	1.	☐	☒	Einige wenige Algorithmen benötigen unbedingt eine Sprache mit Goto-Anweisungen.
	2.	☐	☒	Klassendiagramme beschreiben das Verhalten von Objekten.
	3.	☐	☒	Jede Assoziation ist auch eine Aggregation.
	4.	☒	☐	UML kennt Mehrfachvererbung, Java nicht.
b)				
	1.	☐	☒	Nicht jede rekursive Java-Funktion kann in eine nicht-rekursive Funktion umgeschrieben werden.
	2.	☐	☒	Der Ausdruck 9/2/2 liefert den Wert 2.25.
	3.	☒	☐	“a“ ist ein String und ’b’ ist ein Character-Literal.
	4.	☐	☒	for (int i; i<10; i--) System.out.print(i); ist korrekter Java-Code.
c)				
	1.	☐	☒	`long int real;` ist korrekter Java-Code.
	2.	☐	☒	Jede nicht abstrakte Java-Klasse muss eine main-Methode besitzen.
	3.	☒	☐	Ein Java-Objekt kann aus dem Speicher entfernt werden, wenn keine Referenz mehr auf das Objekt zeigt.
	4.	☐	☒	Wenn eine Klasse keinen Konstruktor hat, meldet der Compiler einen Fehler.

Aufgabe 02 (12 Punkte)

Implementieren Sie eine Klasse **`ZeitDauer`,** die eine Zeitspanne in Tagen, Stunden und Minuten festlegt. Die interne Speicherung sei normiert, d.h. die Werte von Minuten liegen zwischen 0 und 59, die Stunden zwischen 0 und 23. Tag kann beliebig groß (im Wertebereich von long) werden. Wir betrachten nur positive Zeitspannen. Wir machen folgende Vorgaben:

```
public class ZeitDauer {
     protected int m; // Minuten 0<=m<60
     protected int h; // Stunden 0<=h<24
     protected long d; // Tage d>=0
```

a) Implementieren Sie einen Konstruktor, der die Zeitspanne der Dauer d Tage, h Stunden, m Minuten erzeugt: Denken Sie an den Aufruf der Methode normiere() (siehe b)), damit auch nicht normierte Übergabewerte verarbeitet werden können.

```
public ZeitDauer (long d, int h, int m) { // hier ergänzen
     this.m=m;
     this.h=h;
     this.d=d;
     normiere();
}
```

b) Vervollständigen Sie die Methode normiere, die die Zeitwerte wie anfangs beschrieben normiert.

```
void normiere() {
     // normiere Minuten
     while (m>=60) {
          m -=60;
          h++;
     }

     // hier ergänzen: normiere Stunden
          while (h>=24) {
          h -=24;
          d++;
          }
}
```

.

c) Implementieren Sie eine Methode add, die eine Zeitspanne aufaddiert. Denken Sie an die notwendige Normierung! Beispiel ZeitDauer(0,0,20).add(0,0,100) liefert dieselbe Zeitspanne wie ZeitDauer(0,2,0).

```
public void add(long d1, int h1, int m1) {
          m +=m1;
     h +=h1;
     d +=d1;
     normiere();

}
```

Aufgabe 03 (12 Punkte)

Programmieren Sie die Methode **`public static void printMinMax`** aus. Die Methode soll den Wert des kleinsten und des größten Elementes im Array b mit Hilfe von System.out.print ausgeben. Denken Sie auch an den Fall eines leeren Feldes.

```
public class Aufgabe1 {

  public static void printMinMax(int[] b) { // jetzt sind Sie dran!
              if (Blength==0) System.out.println(„Array ist leer“);
        else {
          int max=b[0], min=b[0];
          for (int i=1;i<b.length;i++) {
            if (b[i]<min) min = b[i];
            if (b[i]>max) max = b[i];
            }
         System.out.println(“Min = ”+min+”, Max = “+max);
        }

}
```

Aufgabe 04 (12 Punkte)

Die rekursive Funktion f(n) = 2*n-1 + f(n-1) mit der Initialisierung f(1) = 1 berechnet für positive ganze Zahlen n: f(n) = n*n.

Schreiben Sie eine iterative (nicht-rekursive!) Funktion `linSquare(int n)`, die durch fortgesetztes Aufaddieren von 2*i-1 (i=1..n) n*n berechnet!

Den Rahmen geben wir wieder vor:

```
/**
 * Iterative Berechnung von n*n
 */

public static int linSquare(int n) {
// hier können Sie die Lösung codieren
     int sum=0;
   for (int i=1; i<=n; i++) sum += i+i-1;
    return sum;

}
```

```
}
```