Objektorientierte Programmierung & Modellierung

Universität Koblenz-Landau
Institut für Informatik

Prof. Dr. Ralf Lämmel
Dr. Manfred Jackel

WS 2007/8

Nachklausur

08.04.2008

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. Erstversuch im _____ (Semester).

Auswertung:

	1	2	3	4	GESAMT
Punkte					

Aufgabe 01 (12 Punkte)

Einfaches Programm

Ein Palindrom ist eine Zeichenkette, die gleich ihrer Umkehrung ist. Beispiele „otto“ „einnegermitgazellezagtimregennie“. Das nachfolgende Java-Programm liest mit dem Scanner-Objekt einen String ein. Programmieren Sie die Methode `boolean isPali(String s)` aus, so dass `true` zurückgegeben wird, wenn s ein Palindrom ist.

```
import java.util.Scanner;

public class TestPali {

	public static boolean isPali(String s) {
	// hier ausprogrammieren!

		boolean itis = true;
		int i = 0, j = s.length() - 1;
		while (itis & i <= j) {
			itis &= s.charAt(i) == s.charAt(j);
			i++;
			j--;
		}
		return itis;

	}

	/**
	 * @param args
	 */
	public static void main(String[] args) {

		Scanner input = new Scanner(System.in);

		System.out.print("Bitte geben Sie ein Palindrom ein: ");
		String s = input.next();

		if (isPali(s))
			System.out.println(s + " ist ein Palindrom");
		else
			System.out.println(s + " ist kein Palindrom");

	}

}
```

Aufgabe 02 (12 Punkte)

DBC

Aus dem Aufgabenblatt 5 kennen Sie binäres Suchen in einem sortierten Vektor von Integern. Die nachfolgende Klasse implementiert einen sortierten Vektor von Integerzahlen mit binärem Suchen und binärem Einfügen. Sie sollen die Klasseninvariante dieser Klasse implementieren.

```
import java.util.Collections; import java.util.Vector;

public class SortedVector {

     Vector<Integer> vektor;

     public int such(int testWert, int untereGrenze, int obereGrenze) {
          …..
     }

     void insert(int val, int untereGrenze, int obereGrenze) {
          Inv();
          ……
          InV();
     }

     public SortedVector() {

          vektor = new Vector<Integer>();
          ….
          Inv();
     }

     protected void Inv() {
          assert invariant();
     }

     protected boolean invariant() { // hier bitte ausprogrammieren

          for (int i=0; i<vector.size()-1;i++)
               if (vector.elementAt(i)>=vector.elementAt(i+1))
                    return false;

               return true;

     }
}
```

Der Code der einzelnen Methoden interessiert hier nicht. Die Klasseninvariante wird als `protected void Inv()` zur Verfügung gestellt und ruft die Methode `boolean invariant()` auf. Mathematisch formuliert ist diese Invariante:
vektor.elementAt(i)<vektor.elementAt(i+1) für alle i= 0.. vektor.size()-2. Auch ein leerer Vektor soll als sortiert gelten.
Programmieren Sie die Methode `protected boolean invariant`() aus, um die Klasseninvariante zu implementieren!

Aufgabe 03 (12 Punkte)

Aufwand

```
(1) int i=0, j, sum=0;   1
(2) while (i<n) {        n+1
(3)    j=0;              n*1
(4)    while (j<i) {     (1+2+…+ n-1 + n)*1 = n*(n+1)/2
(5)       j++;           (0+1+…+ n-2 + n-1)*1 = (n-1)*n/2
(6)       sum++;         (0+1+…+ n-2 + n-1)*1 = (n-1)*n/2
       }
(7)    i++;             n
    }
```

a) Berechnen Sie den exakten Aufwand A(n), wenn jede ausgeführte Zeile 1 Aufwandseinheit erfordert.

A(n) = 3n+2 + n*(n+1)/2 + (n-1)*n = $\frac{3}{2}n^2 + \frac{5}{2}n + 2$

b) Zeigen Sie: A(n) < $10n^2$

$\frac{3}{2}n^2 + \frac{5}{2}n + 2 < 3n^2 + 5n + 2 < 3n^2 + 5n^2 + 2n^2 = 10n^2$

c) Geben Sie ein n an, so dass A(n) > 10n gilt.

n>=5

d) Was wird in der Variablen sum berechnet? Beschreiben Sie das in Worten oder durch eine mathematische Formel.

Die Summer der Zahlen von 1 bis n-1. $\frac{(n-1)n}{2}$

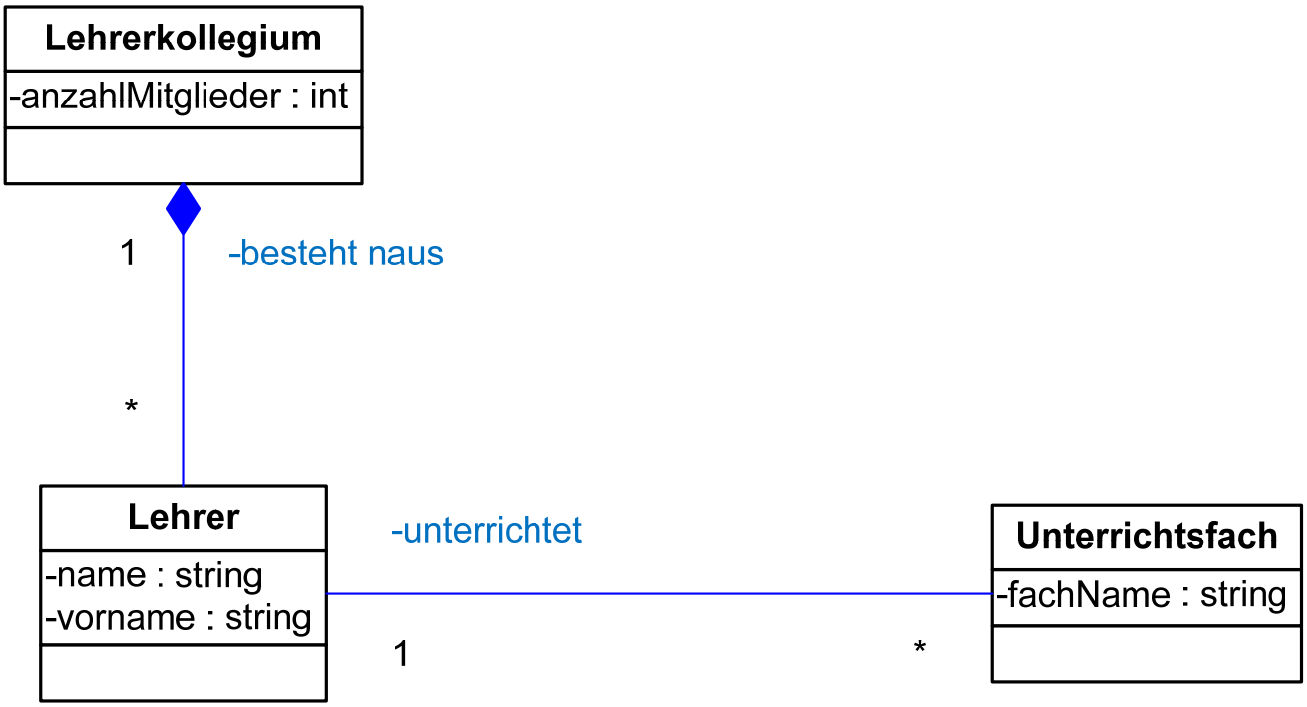
Aufgabe 04 (12 Punkte)

Diese Aufgabe umfasst 3 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	
a)				**OO**
	1.	☐	☒	Eine Klasseninvariante darf nicht von den Attributen der Klasse abhängen.
	2.	☐	☒	Wenn eine Vorbedingung nicht erfüllt ist, ist auch die Nachbedingung nicht erfüllt.
	3.	☒	☐	Ein Teil einer Komposition ist ohne diese nicht sinnvoll.
	4.	☐	☒	Jede Assoziation ist auch eine Aggregation.
b)				**Java**
	1.	☒	☐	Der Java-Typ `byte` ist vorzeichenbehaftet.
	2.	☒	☐	checked exceptions muss man explizit abfangen.
	3.	☐	☒	int i = 5.0f; ist korrekt.
	4.	☒	☐	double d = 5.0f; ist korrekt.
c)				**Programmieren**
	1.	☐	☒	Die Signatur einer Methode umfasst den Namen und den Ergebnistyp.
	2.	☐	☒	Rechtsassoziativität bindet stärker als Linksassoziativität.
	3.	☐	☒	Mehrfachzuweisungen wie x=y=z=1 erfordern Rechtsassoziativität des =-Operators.
	4.	☐	☒	protected-Methoden dürfen in Oberklassen benutzt werden.

Aufgabe 05 (12 Punkte)

UML



Ein Lehrerkollegium besteht aus anzahlMitglieder Lehrern, jeder Lehrer unterrichtet eines oder mehrere Unterrichtsfächer. Die Lehrer sind durch die Felder name und vorname gekennzeichnet, ein Fach durch seine Bezeichnung fachName.

a) Tragen Sie die Beziehungen „besteht aus“ und „unterrichtet“ in das angedeutete UML-Diagramm ein und geben Sie an, ob es sich bei den Beziehungen ggfs. um Assoziationen oder Aggregationen handelt.
b) Bestimmen Sie die Multiplizitäten.
Es ist auch möglich, „unterrichtet“ als binäre *:*-Assoziation zu implementieren. hier die einfachere Interpretation.
c) Skizzieren Sie Java-klassen, der das UML-Klassendiagramm implementiert.

```
import java.util.Vector;

public class Lehrer {
	String name;
	String vorname;
	Vector<Unterrichtsfach> facultas;
}

public class Lehrerkollegium {
	Vector<Lehrer> mitglieder;
	int anzahlMitglieder=0;
}

public class Unterrichtsfach {
	String fachBezeichnung;

}
```

Aufgabe 06 (12 Punkte)
Einfaches Objekt

Der Datentyp V3 soll 3-Dimensionale Vektoren darstellen.

```
public class V3 {
   double x,y,z;

   public V3(double x, double y, double z) {
      this.x = x; this.y = y; this.z = z;
   }


   double betrag() { // bitte hier programmieren

     return Math.sqrt(x*x+y*y+z*z);



   }

   void add(V3 v) { // bitte ausprogrammieren

     x += v.x;
     y += v.y;
     z += v.z;









   }
}
```

Implementieren sie die Methoden betrag und add. Hinweis: der Betrag eines Vektors (x,y,z) ist $\sqrt{x^2+y^2+z^2}$. Die Summe zweiter Vektoren ist der Vektor der Summen der einzelnen Koordinaten.

Aufgabe 07 (12 Punkte)

EBNF

Bei Kaufleuten ist es üblich, große Währungsbeträge übersichtlicher darzustellen, indem man den Vorkommaanteil in Tausendergruppen durch Punkte unterteilt, etwa 1.238.999,47. Hinter dem Komma werden immer 2 Dezimalstellen angegeben.

Geben Sie eine EBNF-Syntax an, die solche kaufmännischen Zahlen darstellt. Wählen Sie als Startsymbol <Geldbetrag>.

<Geldbetrag> ::= <Hunderter>{<Tausender>}<Cent>

<Hunderter> ::= <Dezimale> | <Dezimale><Dezimale> | <Dezimale><Dezimale><Dezimale>

<Tausender> ::= “.“ <Dezimale><Dezimale><Dezimale>

<Cent> ::= “,“ <Dezimale><Dezimale>

<Dezimale> ::= „“0“ | “1“ | “2“ | “3“ | “4“ | „“5“ | “6“ | “7“ | “8“ | “9“

Aufgabe 08 (12 Punkte)

DP

Aus der Vorlesung kennen Sie folgende Implementation von Integer-Listen:

```
public interface List {
     void accept(Visitor v);
}

public class Cons implements List {
     int head;
     List tail;

     public void accept(Visitor v) {
          v.visitCons(this);
     }

     public Cons(int h, List t) {
          head = h;
     }
}

public class Nil implements List {

     public void accept(Visitor v) {
          v.visitNil(this);
     }

}
```

Diese Klassen sind für das Visitor-Entwurfsmuster vorgerüstet. Implementieren Sie einen Visitor, der eine solche Liste wie folgt ausdrucken soll:

Die Liste
`List l=new Cons(1,Cons(2, Cons(3,new NIL())))` wird ausgegeben als
`“( 1 2 3 )”`
ohne die Doppelapostrophe!

Leiten Sie die konkrete Visitorklasse PrintVisitor aus dem Interface

```
public interface Visitor {
      void visitNil(Nil n);
      void visitCons(Cons c);
}
```

ab. Tip: die öffnende Klammer sollte man im Konstruktor ausgeben!

siehe nächste Seite!

```
public class PrintVisitor implements Visitor { // hier programmieren

      public PrintVisitor() {
           System.out.println("( ");
      }

      public void visitNil(Nil n) {
           System.out.println(")");
      }

      public void visitCons(Cons c) {
           System.out.print(c.head+" ");
           c.tail.accept(this).

      }


}
```