Name:_____Matrikelnummer: _____

Klausur zur Vorlesung

"Software-Entwicklung I und II" (WS 2003/04 und SS 2004)

Wichtig! Wichtig! Wichtig!

- Schreiben Sie Ihren Namen und Ihre Matrikelnummer auf jedes Blatt der Klausur!
- Blätter ohne Namen werden nicht gewertet.
- Lassen Sie die Klausur zusammengeheftet!
- Die Klausur dauert 3 Stunden.
- Die Klausur ist bestanden, wenn mindestens 50% der Punkte erreicht wurden.
- Abschreiben oder Abschreibenlassen führen zum Nichtbestehen der Klausur.
- Kennzeichnen Sie Ihre Lösung eindeutig! Es wird keine Lösung gewertet, wenn Sie zu einer Aufgabe mehr als eine Lösung abgeben.
- Als Hilfe: Rechnen Sie pro Punkt mit etwa 2,5 Minuten Bearbeitungszeit.

Aufkleber

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Σ**
7	8	7	12	9	10	12	7	72

Note:	

Bitte kreuzen Sie Ihren Studiengang an:

	Diplomstudiengang Informatik nach DPO 4
	Diplomstudiengang Informatik nach DPO 2 (Schein) oder 3
	Wirtschaftsinformatik
	Ingenieurinformatik Schwerpunkt Informatik
	Ingenieurinformatik Schwerpunkt Maschinenbau
	Ingenieurinformatik Schwerpunkt Elektrotechnik
	Lehramt SII
	Lehramt SII qualifizierender Studiennachweis (wird bei der Bewertung berücksichtigt)
	Medienwissenschaften
	Sonstige: _____

Name:_____Matrikelnummer: _____

1. Grundlegende Konzepte in Java (7 Punkte)

a) Literale und Type Casting (2 Pkt.)

Im folgenden Programmsegment stehen einige Wertzuweisungen von **Literalen** (Konstanten) an Variable, sowie **Type-Casting**-Operationen .

```
(1)  int i = "34";
(2)  String a_bis_f = "abc" + "def";
(3)  String s = 'false' + a_bis_f;
(4)  boolean b = false || (boolean)s;
(5)  int j = 4.e-1;
(6)  int k = (short)6.e2;
(7)  double d = (double)(i+j+k);
```

Geben Sie alle Zeilen an, in denen eine **fehlerhafte** Wertzuweisung eines Literals stattfindet:

1, 3, 4

Geben Sie alle Zeilen an, in denen eine Type-Casting-Operation **fehlerhaft** ist:

4

Name:_____Matrikelnummer: _____

b) Arithmetische Berechnungen (2 Pkt.)

Das folgende Programm berechnet alle Teiler einer positiven ganzen Zahl, die über die Kommandozeile eingegeben wird. Zum Beispiel gibt das Programm bei Eingabe von

```
java Aufgabe1b 45
```

alle Teiler der Zahl 45 aus, also `1 3 5 9 15 45`. Allerdings fehlt noch die aufgerufene Methode `istTeiler`. Diese Methode soll bei Aufruf mit den Parametern `t` und `n` feststellen, ob `t` ein Teiler von `n` ist.

```
(1)  class Aufgabe1b
(2)  {
(3)     static public void main (String[] args)
(4)     {
(5)        int n = Integer.parseInt(args[0]);
(6)        System.out.print("Teiler von "+n+": ");
(7)        for (int t=1; t<=n; t++)
(8)           if (istTeiler(t, n)) System.out.print(t+" ");
(9)        System.out.println();
(10)    }
(11) }
```

Hinter welcher Zeile soll die Methode `istTeiler` eingefügt werden? 2 oder 10

Schreiben Sie hier die Methode `istTeiler` auf:

```
static boolean istTeiler(int t, int n)

{ return n%t==0; }
```

Name:_____Matrikelnummer: _____

c) Arrays (3 Pkt.)

Schreiben Sie ein Java-Programmfragment, das

(1) ein `int`-Array `a` anlegt, das genau die ganzen Zahlen

`7, 93, 66, 34, -1, 101`

in dieser Reihenfolge enthält und

(2) anschließend in dem Array die Zahlen so aufaddiert, dass das Array-Element mit dem Index `i` am Schluß gerade die Summe der links davon stehenden Array-Elemente `0,1,2,…,i-1` und an der Stelle `0` eine `0` enthält.. In unserem Beispiel stünde also am Ende die Zahlenfolge

`0, 7, 100, 166, 200, 199`

im Array.

Schreiben Sie die Lösung zu **(2)** so, dass sie für jedes `int`-Array – egal welcher Größe – unverändert funktioniert.

```
…
public static void main(String [] args){

/* Fügen Sie hier das Programmfragment ein. */

    int [] a = {7, 93, 66, 34, -1, 101};  // Teil a

    for (int i=a.length-1; i>=0; i--)   // Teil b
    {
      a[i] = 0;
      for (int j=0; j<i; j++) a[i] += a[j];
    }









}
```

Name:_____Matrikelnummer: _____

2. Iteration und Rekursion (8 Punkte)

a) **Iterative und rekursive Methode (5 Pkt)**

Eine Funktion `fun(i,n)` ist wie folgt für nicht-negative ganze Zahlen `i` und `n` rekursiv definiert:

```
fun(i,n) = n, falls i = 0

fun(i,n) = log(fun(i-1,n)), falls i > 0.
```

Es ergeben sich folgende Werte:

```
fun(0,n) = n,
fun(1,n) = log(n)
fun(2,n) = log(log(n))
fun(3,n) = log(log(log(n)))
```
...usw.

Hinweis: Die Methode für die Berechnung des Logarithmus liegt in der `Math`-Klasse vor als `public static double log(double a).` Diese liefert den natürlichen Logarithmus von `a`.

Schreiben Sie je eine rekursive und eine iterative Methode, die die Funktion `fun(i,n)` berechnet.

```
double funIterativ (int i,int n)
{

double ert = n;

for(int j = 0;j<i;j++){

if(j==1){ret = Math.log(n);}

else{ ret = Math.log(ret);}
}

double funRekursiv (int i ,int n)
{

    if(i<=0) return n;

    else return Math.log(funrekursiv(i+1,n));
```

Name:_____Matrikelnummer: _____

b) Verschiedene Formen der Iteration (3 Pkt)

Schreiben Sie ein Java-Methode, die die gleiche Wirkung erzielt wie die folgende mit zwei `for`-Schleifen, dafür aber `while`-Schleifen verwendet:

```
void iterationFor(int a)
{
   for(int aussen = a ; aussen >= 0 ; aussen --)
   {
      for(int innen = aussen ; innen <= 10 ; innen ++)
      {
         System.out.println("Ausgabe: " + innen * aussen);
      }
   }
}
```

```
void iterationWhile(int a){

int aussen = a;

        while(aussen>=0){

        int innen = aussen;

                while(innen <= 10){

                System.out.println(„Ausgabe:"+ innen*aussen);

                innen ++;

                }

                aussen--;

        }

}
```

Name:_____Matrikelnummer: _____

3. Suchen und Sortieren (7 Punkte)

a) Suchen (Häufigkeiten feststellen)(4 Pkt)

In einem Java-Programmsegment sei ein `int`-Array `zahlen` gegeben, das beliebige ganze Zahlen zwischen 0 und 99 enthält. Das folgende Programmsegment ermittelt diejenige Zahl `x` im Array `zahlen`, die unter den am häufigsten auftretenden Zahlen die größte ist:

```
(1)  int [] häufigkeit = new int[100];
(2)
(3)  int maximum = 0;
(4)  for (int i=0; i<zahlen.length; i++)
(5)  {
(6)     häufigkeit[zahlen[i]]++;
(7)     if (maximum < häufigkeit[zahlen[i]])
(8)        maximum = häufigkeit[zahlen[i]];
(9)  }
(10)
(11) int x=0;
(12) for (int i=0; i<100; i++)
(13)    if (häufigkeit[i]==maximum) x=i;
```

Würde diese Lösung auch funktionieren, wenn die im Array `zahlen` gespeicherten Zahlen `float`-Zahlen wären?

☐ Ja ☒ Nein, weil man mit dem Wertebereich der Zahlen in zahlen dann kein Array indizieren könnte.

Was ist zu ändern, wenn die **kleinste** der in `zahlen` am häufigsten auftretenden Zahlen ermittelt werden soll? (Geben Sie die neuen Zeilen samt Zeilennummer an!)

Variante 1: Schleife in (12) von oben nach unten laufen lassen

(12) for (int i=99; i>=0; i--)

Variante 2: Von unten das erste Auftreten suchen

(12) gefunden = false;
(13) while (!gefunden)
(14) if (häufigkeit[x] == maximum) gefunden = true; else x++;

Name:_____Matrikelnummer: _____

b) Sortieren (3 Pkt)

Gegeben sei die folgende Methode `arraySort` zum aufsteigenden Sortieren eines `double`-Arrays.

```
(1)  static double[] arraySort(double[] a)
(2)  {
(3)     int i=0;
(4)     while (i<a.length-1)
(5)     {
(6)        if (a[i]>a[i+1])
(7)        {
(8)           double f = a[i]; a[i] = a[i+1]; a[i+1] = f;
(9)           i=0;
(10)       }
(11)       else i++;
(12)    }
(13)    return a;
(14) }
```

Darf die Methode in der Form

```
d_arr = arraySort(d_arr);
```

aufgerufen werden, vorausgesetzt `d_arr` ist als `double`-Array deklariert?

X Ja ☐ Nein, weil _____

Funktioniert das Sortierverfahren auch noch korrekt, wenn in Zeile `(6)` das „>“ durch „>=“ ersetzt wird?

☐ Ja X Nein, weil Endlosschleife durch unendliches Vertauschen gleicher Elemente

Funktioniert das Sortierverfahren auch noch korrekt, wenn in Zeile `(4) a.length-1` durch `a.length` ersetzt wird?

☐ Ja X Nein, weil Indexfehler in Zeile (6) resultiert

Funktioniert das Sortierverfahren auch noch korrekt, wenn in Zeile `(8)` die Vertauschung mit der Anweisungsfolge

```
double f = a[i+1]; a[i+1] = a[i]; a[i] = f;
```

durchgeführt wird?

X Ja ☐ Nein, weil _____

Name:_____Matrikelnummer: _____

4. Dynamische Datenstrukturen (12 Punkte)

a) **Lineare Listen (4 Pkt)**

In einem Java-Programm seien Objekte der Klasse `Knoten` verwendet worden, um eine einfach verkettete Liste mit `int`-Zahlen zu speichern. Die Klasse `Knoten` sei wie folgt gegeben und darf nicht verändert werden:

```
class Knoten
{
   private int wert;
   private Knoten nächster;

   Knoten(int w, Knoten n)
   {
      setWert(w);
      setNächster(n);
   }

   public int getWert() { return wert; }
   public void setWert(int w) { wert = w; }
   public Knoten getNächster() { return nächster; }
   public void setNächster(Knoten n) { nächster = n; }
}
```

Gegeben seien drei Methoden namens `fügeVorneEin`, `fügeHintenEin` und `verdoppeleLetzten`, die jeweils die Liste verändern und einen Verweis auf den Anfang der veränderten, korrekt gebildeten Liste zurückgeben sollen.

```
(1)  static Knoten fügeVorneEin(Knoten liste, int x)
(2)  { return new Knoten(x, liste); }
(3)
(4)  static Knoten fügeHintenEin(Knoten liste, int x)
(5)  {
(6)     Knoten p = liste;
(7)     while (p.getNächster() != null)
(8)        p = p.getNächster();
(9)     p.setNächster(new Knoten(x, liste));
(10)    return liste;
(11) }
(12)
(13) static Knoten verdoppeleLetzten(Knoten liste)
(14) {
(15)    Knoten p = liste;
(16)    while (p.getNächster() != null)
(17)       p = p.getNächster();
(18)    p.setNächster(new Knoten(p.getWert(),p.getNächster()));
(19)    return p;
(20) }
```

Name:_____Matrikelnummer: _____

Fügt die Methode `fügeVorneEin` wirklich einen neuen Knoten mit der als Parameter übergebenen Zahl am Anfang der Liste ein und gibt eine korrekt gebildete Liste zurück?

X Ja ☐ Nein, weil _____

Fügt die Methode `fügeHintenEin` wirklich einen neuen Knoten mit der als Parameter übergebenen Zahl ans Ende der Liste an und gibt eine korrekt gebildete Liste zurück?

☐ Ja X Nein, weil der neue Knoten einen falschen "nächster"-Zeiger hat

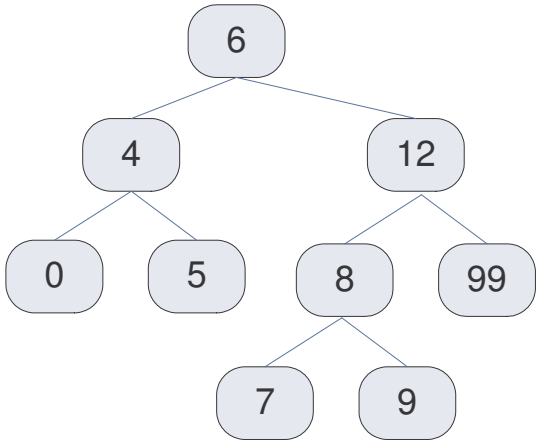
Fügt die Methode `verdoppeleLetzten` wirklich einen neuen Knoten mit der gleichen Zahl ans Ende der Liste an wie bei Aufruf am Ende der Liste stand und gibt eine korrekt gebildete Liste zurück?

☐ Ja X Nein, weil zwar die richtige Liste gebildet, aber dann der falsche Zeiger zurückgegeben wird (statt auf den Anfang auf den alten letzten Knoten)

Name:_____Matrikelnummer: _____

b) **Baumdurchläufe (2 Pkt.)**

Gegeben sei der binäre Baum wie rechts gezeigt. Geben Sie die Reihenfolge der Knoten bei den drei Arten von Baumdurchläufen an:



Präfixdurchlauf:

6, 4, 0, 5, 12, 8, 7, 9, 99

Infixdurchlauf:

0, 4, 5, 6, 7, 8, 9, 12, 99

Postfixdurchlauf:

0, 5, 4, 7, 9, 8, 99, 12, 6

Name:_____Matrikelnummer: _____

c) **Baumstrukturen (6 Pkt.)**

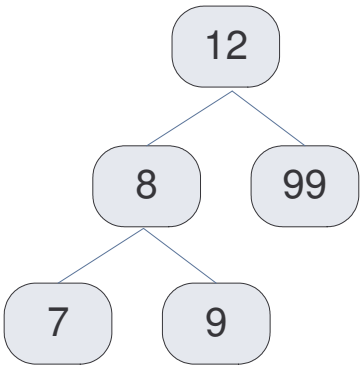
In einem Java-Programm wurde mit Objekten der Klasse `TreeNode` ein binärer Baum aufgebaut, in dessen Knoten jeweils eine Zahl gespeichert wird (ein Beispiel für so einen Baum sehen Sie in Aufgabe 4b).

```
class TreeNode
{
   private int v;
   private TreeNode l, r;

   TreeNode(TreeNode l, int v, TreeNode r)
   {
      this.l = l; this.v = v; this.r = r;
   }

   int getValue() { return v; }
   TreeNode getLeft() { return l; }
   TreeNode getRight() { return r; }
}
```

Schreiben Sie eine Methode `getSubTree`, die eine Zahl `v` im Baum sucht, und dann einen Verweis (vom Typ `TreeNode`) auf die Wurzel des darunterhängenden Teilbaums zurückgibt.



Sei zum Beispiel die Wurzel des in Aufgabe 4b verwendeten Beispielbaums in der Variable `TreeNode root` gegeben, dann würde der Aufruf `getSubTree(root, 12)` einen Verweis auf die Wurzel des rechts gezeigten Teilbaums zurückgeben.

```
static BaumKnoten getSubTree(BaumKnoten b, int w)
{
  if (b.getWert()==w) return b;
  else
  {
    if (b.getLinks()!=null)
    {
      BaumKnoten h = getSubTree(b.getLinks(), w);
      if (h != null) return h;
      else
        if (b.getRechts()!=null)
        {
          h = getSubTree(b.getRechts(), w);
          if (h != null) return h;
          else return null;
        }
        else return null;
    }
    else return null;
  }
}
```

Name:_____Matrikelnummer: _____

5. Klassen, Interfaces, Vererbung (9 Punkte)

a) Interfaces (5 Pkt)

Gegeben seien die folgenden Definitionen eines Interface und dreier Klassen, die dieses Interface implementieren:

```
interface F
{
   double at(double x);
   String toString();
}

class Square implements F
{
   public double at(double arg) { return arg*arg; }
   public String toString() { return "Square"; }
}

class Root implements F
{
   public double at(double arg) { return Math.sqrt(arg); }
   public String toString() { return "Root"; }
}

class Linear implements F
{
   private double a, b;
   public double at(double arg) { return a*arg+b; }
   public String toString() { return "Linear:"+a+"*X+"+b; }

   Linear(double a, double b) {this.a=a; this.b=b; }
}
```

Jede Instanz einer Klasse repräsentiert eine mathematische Funktion, die als Argument eine `double`-Zahl erhält und eine `double`-Zahl berechnet.

a.1) Ist es möglich, ein Array `f` zu deklarieren und anzulegen, in dem Sie beliebige Objekte all dieser Klassen ablegen können?

☐ Nein, weil _____

X Ja, die Deklaration für ein solches Array `f` mit 5 Elementen sieht wie folgt aus:

```
F[] f = new F[5];
```

Name:_____Matrikelnummer: _____

a.2) Ist es möglich, eine Methode `max_f` zu schreiben, der ein solches Array gemäß Aufgabe a.1 und ein `double`-Wert `x` übergeben werden und welche dann diejenige Instanz aus dem Array zurückgibt, welche für das Argument `x` den maximalen Funktionswert hat?

☐ Nein, weil schon das Array nach Aufgabe a.1 gar nicht existieren kann.

☐ Nein, weil _____

X Ja, die Methode sieht wie folgt aus:

```
static F maxF(F[] f, double x)
{
  F max_f = f[0];
  for (int i=1; i<f.length; i++)
    if (f[i].at(x) > max_f.at(x))
      max_f = f[i];
  return max_f;
}
```

Name:_____Matrikelnummer: _____

b) **Klassen und Objekte (4 Pkt.)**

Ein Java-Programm enthalte folgende Klassenhierarchie:

```
class A
{
   char c = 'a';
   void x() { System.out.print("A"+c); }
   void y() { System.out.print("Ay"); }
   void z() { y(); x(); }
}

class B extends A
{
   void x() { System.out.print("Bx"); }
   void y() { System.out.print("B"+c); }
}

class C extends B
{
   void x() { System.out.print("C"+c); }
   C() { c = 'c'; }
}
```

Bitte geben Sie für jeden der folgenden Methodenaufrufe an, ob er erlaubt ist und wenn ja, welche Ausgabe er erzeugt. Voraussetzung ist, dass vorher die Variablen

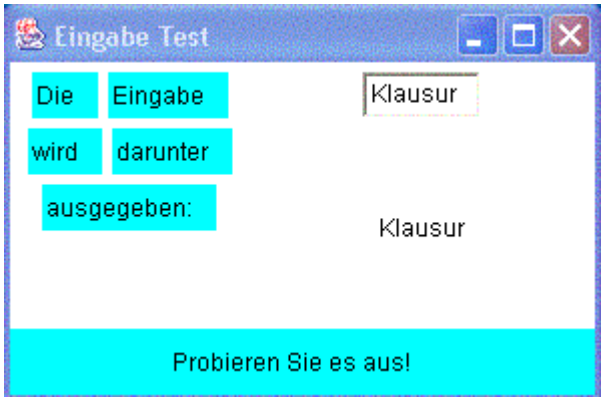
```
A a = new A(); B b = new B(); C c = new C();
```

deklariert und definiert wurden.

Methodenaufruf	Zulässig?	Falls zulässig, welche Ausgabe wird erzeugt?
`a.x();`	X Ja ☐ Nein	Aa
`a.y();`	X Ja ☐ Nein	Ay
`a.z();`	X Ja ☐ Nein	Ay Aa
`b.x();`	X Ja ☐ Nein	Bx
`b.y();`	X Ja ☐ Nein	Ba
`b.z();`	X Ja ☐ Nein	Ba Bx
`c.x();`	X Ja ☐ Nein	Cc
`c.y();`	X Ja ☐ Nein	Bc
`c.z();`	X Ja ☐ Nein	Bc Cc

Name:_____Matrikelnummer: _____

6. Bedienoberfläche (10 Punkte)



a) Objektbaum zeichnen (4 Pkt)

Zeichnen Sie einen **Objektbaum**, der die Hierarchie der verwendeten GUI-Komponenten darstellt:

- Jeder **Knoten** repräsentiert eine Instanz einer GUI-Komponente.
- Die **Kanten** des Baumes geben an, welche Komponenten die Container enthalten.
- Schreiben Sie den Namen der Klasse der Komponente an den Knoten.

Lösung analog zu der letzten Klausur

Name:_____Matrikelnummer: _____

b) **LayoutManager** **(2 Pkt)**

Geben Sie zu jedem der vorkommenden **Container-Objekte** an, welcher LayoutManager verwendet wird. Begründen Sie, woran Sie ihn erkannt haben.

Im Frame: BorderLayout, da unten ein textfield.mit voller Breite

Container links: Flow, da Buttons unterschiedlich lang, und zentriert in Zeilen angeordnet

Container rechts: Grid, da Tabelle erkennbar: 1 Spalte und 2 Zeilen.

c) Eigenschaften des FlowLayout Managers **(2 Pkt)**

Geben Sie für den FlowLayout-Manager an, welche Eigenschaften unverändert bleiben, wenn die Form des **Containers** verändert wird.

Größe der Objekte

Farbe der Objekte

Reihenfolge der Objekte

Ausrichtung der objekte (links, rechts, mitte)

Name:_____Matrikelnummer: _____

d) **Programmieraufgabe (2 Pkt)**

Ergänzen Sie das folgende Programmstück so, dass jede Eingabe in das `TextField` in dem `Label` out angezeigt wird:

```
out = new Label ("nichts");

TextField input = new TextField ("", 5);

...input.addActionListener(

        new ActionListener(){

                public void actionPerformed(ActionEvent e){

                out.setText(e.getActionCommand());}

        });
```

Name:_____Matrikelnummer: _____

7. Synchronisation der Prozesse (12 Punkte)

Eine Firma für Büro-Service bietet Ihren Kunden die Nutzung eines Großraumbüros an. Ein Kunde kann dort für eine Weile einen Arbeitsplatz belegen. Bei Bedarf kann er auch ein Telefon und/oder einen Laptop ausleihen. Für das Büro stehen 10 Arbeitsplätze, 8 Telefone und 5 Laptops zur Verfügung. Die Firma hat in ihren drei Filialen jeweils ein so ausgestattetes Büro.

Hinweis !

Bevor Sie in **(d)** eine Monitorklasse zur Simulation der Abläufe implementieren, beantworten Sie die Fragen **(a)** bis **(c):**

a) Welche Rollen spielen die Monitore und die Prozesse in dieser Aufgabe? (2 Pkt)

Das Büro ist ein Monitor, der die Reccourcen verwaltet.

Die Kunden sind Prozesse, die etwas ausleihen und wieder zurück geben.

b) Welche Monitor-Operationen mit welchen Parametern benötigen Sie hier? **(2 Pkt)**

leihe Arbeitsplatz(int anzahl);

gibArbeitsPlatzZurück()int anzahl;

leiheTelefon(int anzahl);

gibTelefonZurück(int anzahl);

leiheLaptop(int anzahl);

gibLaptopZurück(int anzahl);

c) Welche der Operationen benötigen Wartebedingungen? Geben Sie die Wartebedingungen zunächst in Sätzen an. Durch Ausführen welcher Operationen können sie erfüllt werden? **(2 Pkt)**

Die „belegeArbeitsplatz"-Methode benötigt Wartebedingung, falls dies nicht verfügbar ist.

Durch das Ausführen von den „gibZurück"-Methoden, werden die Wartebedingungen wieder erfüllt.

Name:_____Matrikelnummer: _____

d) Implementieren Sie eine Monitor-Klasse für diese Aufgabe. Sie soll nur die synchronisierte Ressourcevergabe leisten. **(4 Pkt)**

```
cass Filiale{

int a= 10, t =8, l=5;

public void synchronised belegArbeitsPlatz(int kl, int lap){

        if(a> 0 ; tel<=t ; lap<=l){

                a--;t-=tel;l-=lap;

                try{sleep(1000);} catch(interruptedException e){}

        }

        else try{wait();} catch(InterruptedException e){}

        }

public void synchronised gibFreiArbeitsplatz(int tel, int lap){

        a++ ; t += tel ; l += lap ;

        notifyAll();

}

}
```

e) In der Firma hat man solch eine Software eine Weile erprobt. Dabei stellte man fest, dass das System manchmal blockiert. Was vermuten Sie, ist die Ursache? Was schlagen Sie als Abhilfe vor? **(2 Pkt)**

Es kommt zu einer Verklemmung.siehe: http://ag-kastens.uni-paderborn.de/lehre/material/sweii/folien/Folie151.html

mögliche Abhilfe: alle Ressourcen atomar auf einmal belegen:

http://ag-kastens.uni-paderborn.de/lehre/material/sweii/folien/Folie153.html

oder:Symmetrie der Wartebedingugen aufbrechen und damit den Zyklus vermeiden.

http://ag-kastens.uni-paderborn.de/lehre/material/sweii/folien/Folie156.html

Name:_____Matrikelnummer: _____

8. Verständnisfragen (7 Punkte)

Betrachten Sie das korrekte Java-Programm auf der nächsten Seite und beantworten Sie dazu die folgenden Fragen:

Wichtiger Hinweis: Sie müssen **nicht** verstehen, was das Programm inhaltlich macht – Sie müssen hier nur die **Programmkonstrukte** betrachten.

1. Geben Sie eine **Objektvariable** des Programms an: data, link, id oder s

2. Geben Sie eine **Objektmethode** des Programms an: printMe (5) oder (17)

3. Geben Sie eine **Klassenvariable** des Programms an: noOfNodes

4. Geben Sie eine **Klassenmethode** des Programms an: average oder main

5. Geben Sie eine Programmzeile an, in der ein **Type Cast** stattfindet: (39)

   Welche Datentypen sind beteiligt? float und int

6. Geben Sie eine Programmzeile an, in der eine **implizite Typkonversion** vorgenommen wird:

   5 oder 18

   Welche Datentypen sind beteiligt? int und String

7. In Java gibt es das Konzept der **dynamischen Methodenbindung**. Geben Sie eine Programmzeile an, in der ein Methodenaufruf vorkommt, bei dem dieses Konzept zum Tragen kommt:

   29

   Welcher Methodenaufruf ist betroffen? p.printMe()

8. Könnte die Methode `printMe(…)` der Klasse `NodeSpecial` (Zeile 17) auf die Variable `id`, die in der Klasse `Node` deklariert ist (Zeile 8), zugreifen?

   ☐ Ja ☒ Nein, weil id als private deklariert ist

Name:_____Matrikelnummer: _____

```
(1)   class Node
(2)   {
(3)      Node (int v, Node n) { data=v; link=n; id=noOfNodes++; }
(4)
(5)      void printMe() { System.out.print(id+":"+data+" "); }
(6)
(7)      int data; Node link;
(8)      private int id;
(9)      static int noOfNodes = 0;
(10)  }
(11)
(12)  class NodeSpecial extends Node
(13)  {
(14)     NodeSpecial(String s, int v, Node n)
(15)     { super(v, n); this.s = s; }
(16)
(17)     void printMe()
(18)     { System.out.print(s+data+" "); }
(19)
(20)     String s = "";
(21)  }
(22)
(23)  class Aufgabe8
(24)  {
(25)     static void printList(Node p)
(26)     {
(27)        while (p != null)
(28)        {
(29)           p.printMe(); p = p.link;
(30)        }
(31)     }
(32)
(33)     static float average(Node start)
(34)     {
(35)        float sum = 0.0f; int count = 0;
(36)        while (start != null)
(37)        {
(38)           count++;
(39)           sum += (float)start.data;
(40)           start = start.link;
(41)        }
(42)        return sum/count;
(43)     }
(44)
(45)     public static void main (String [] args)
(46)     {
(47)        Node start = null;
(48)
(49)        for (int i=0; i<10; i++) start = new Node(i, start);
(50)
(51)        start = new NodeSpecial("***", 56, start);
(52)
(53)        printList(start);
(54)        System.out.println("Average is: "+average(start));
(55)     }
(56)  }
```

Name:_____Matrikelnummer: _____

9. Nennen Sie drei verschiedene Ereignisse, die an Komponenten von grafischen Benutzungsoberflächen ausgelöst werden können.

ActinEvent, MouseEvent, KeyEvent

10. Was bedeutet "Bedingungssynchronisation"?

Ein Prozess wartet bis eine Bedingung erfüllt ist verursacht durch einen anderen Prozess z.B. erst dann in einen Puffer schreiben dwenn er nicht mehr voll ist.

11. An welchen Stellen eines Monitor-Programms sollen Aufrufe `notifyAll()` stehen?

…