

Name, Vorname:

Hilfsmittel:

selbstverfasste, handgeschriebene, schriftliche Unterlagen

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

1)

21P

a)

Man will mit Hilfe von XML die Noten einer Klassenarbeit darstellen.

Eine Klassenarbeit besteht aus mehreren Schülern, wobei jeder Schüler einen Namen und eine Note hat. Namen und Note sind nicht mehr weiter strukturiert (wie z.B. in Vorname und Nachname) , sondern bestehen aus reinem Text. Jeder Schüler hat einen Spitznamen, der durch das Attribut nickname beschrieben wird.

In einer Klasse gibt es genau die folgenden 2 Schüler, die in einer Klassenarbeit die folgenden Noten geschrieben haben:

Otto (Spitzname Fuchs) 3

Timo (Spitzname Wiesel) 4

Stellen Sie die Ergebnisse obiger Klassenarbeit durch die Verwendung entsprechender XML-Tags dar.

b)

Man will obigen XML-Code in einen HTML-Code übersetzen um ihn auf einen Webserver zu stellen. Was muß man machen, um die Übersetzung zu realisieren? (Stichwort XSL, XSLT)

c)

Wie kann man erreichen, daß eine Klassenarbeit, die das obige Schema verletzt (weil z.B. ein Tag falsch geschrieben wurde) als falsch erkannt wird ?

2)

18P

Man will in einer Tabelle allen geraden Tabellenzeilen die Hintergrundfarbe rot und allen ungeraden Tabellenzeilen die Hintergrundfarbe blau verleihen.

Dies soll mit CSS realisiert werden.

Die Style Sheet Angaben müssen im Kopf einer HTML-Datei angegeben werden.

a) Geben Sie die Style Sheet Angaben an.

b) Was muß in den ungeraden bzw. geraden Zeilen in <tr> geändert werden?

3)

62P

Erstellen Sie den HTML-Code (ohne css) , der den unten angegebenen Bildschirmausdruck ergibt. Die Tabelle hat 3 Zeilen und 4 Spalten und nimmt die ganze Breite des Anzeigefensters ein. Die Gitternetzlinien sollen nicht sichtbar sein. Spalte A hat 10%, Spalte B 40%, Spalte C 40%, Spalte D 10% Anteil an der Breite der Tabelle.

[1] Überschrift: zentriert, vom Typ 2

[2] Name, Datum, Klasse ist insgesamt ein Absatz.
Schriftart: Arial, Ersatz Helvetica
Größe: 4
fett

[3] Bild Donald.gif
Breite: 30
Höhe: 70
Lage: linksbündig
Ist die Grafik nicht da, soll das Wort Donald erscheinen

[4] über 2 Spalten
Lage: horizontal zentriert, vertikal oben
Größe: 2 größer als der Standard

[5] über 3 Zeilen
Lage des Textes: vertikal zentriert, horizontal zentriert

[6] Hintergrund: rot
Lage des Textes: vertikal oben

[7] Unsortierte Liste, rechtsbündig
Die 2 von qm ist hochgestellt
Die 2 vom H2O ist tiefgestellt

[8] Unsortierte Liste

[9] Hintergrund: grün

[10] Verweis auf eine HTML-Seite

[11] Verweis auf eine Homepage

[12] Text kursiv und unterschrieben

[13] Sortierte Liste, Farben:
Abele rot, Cäsar grün, Huber braun

[14] durchgezogener Balken
Größe: 50% des Anzeigebildschirms
Farbe: rot
Lage: zentriert

[15] Verweis auf Überschrift "Meine Website"

Meine Website [1]

Name: Mustermann [2]

Datum: 27.06.13

Klasse: E3FIF

Bild [3]	Produkte [4]		Unser [5] Leitbild: wir werden wollen müssen.
Artikel [6]	* Teppiche in m² [7] * Wasser (H ₂ O) * Textilien	* 15 Euro [8] * 10 Euro * 20 Euro	
Links [9]	zweiteSeite [10]	www.mms.de [11]	

Jetzt eine sortierte Liste [12]

1. Abele [13]

2. Cäsar

3. Huber

----- [14]

Zur Website [15]

Lösung:

1) 15P

a)

```
<klassenarbeit>
  <schueler>
    <name nickname="Fuchs">Otto</name>
    <note>3</note>
  </schueler>
  <schueler>
    <name>Timo nickname="Wiesel"</name>
    <note>4</note>
  </schueler>
</klassenarbeit>
```

b) 3P

In einer XSL-Datei werden "Regeln" erstellt, nach denen der XML-Code mit dem XSLT-Programm in den HTML-Code transformiert wird.

c) 3P

In einer DTD-Datei bzw. XSD-Datei werden die "grammatikalischen Regeln" erstellt, nach denen der XML-Code erzeugt werden muß.

2) 18P

a) 12P

```
<style type="text/css">
  .classFormat_tabelle_gerade{
    background-color: red;
  }

  .classFormat_tabelle_ungerade{
    background-color: blue;
  }
</style>
```

b) 6P

```
...
<tr class="classFormat_tabelle_ungerade">
...
...
<tr class="classFormat_tabelle_gerade">
...
```

3) 62P

```
<html>
  <head>
  </head>

  <body>
    <h2 align=center><a name="ueberschrift">Website</a></h2>
    <p>
      <font face="Arial, Helvetica" size=4>
        <b>
          Name: Mustermann
          <br>Datum: 07.01.13
          <br>Klasse: E3FI
        </b>
      </font>
    </p>
  </body>
</html>
```

</p>

<table width=70% border="1">

<tr>

<td width=10%>

</td>

<td align=center valign=bottom colspan=2>Produkte

</td>

<td width=10% align=center valign=middle rowspan=3>Unser Leitbild:
</td>

</tr>

<tr>

<td bgcolor=red valign=top>Artikel</td>

<td width=40%>

Teppich in m²

Wasser H₂O

Textilien

</td>

<td width=40%>

15 Euro

10 Euro

20 Euro

</td>

</tr>

<tr>

<td bgcolor=green>Links</td>

<td>ZweiteSeite</td>

<td>www.mms.de</td>

</tr>

</table>

<p>

<i><u>Jetzt eine sortierte Liste</u></i>

</p>

<p>

Abele

Cäsar

Huber

</p>

<hr width=50% size=4 color=red align=center>

<p>

Zur Überschrift (Kl.Nr.1)

</p>

</body>

</html>

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1)

100P

Für die Erstellung der folgenden Tabelle dürfen nur (maximal) die folgenden Tags, die entsprechenden schließenden Tags und Attribute (kein css) benutzt werden:

<table>, <tr>, <td>, colspan, rowspan,

Die Tabelle muss einen Rand (border = 1) haben.

Z1,R1			Z2,R1	Z3,R1
Z1,R2		Z2,R2		
Z1,R3	Z2,R3		Z3,R3	
Z1,R4	Z2,R4	Z3,R4	Z4,R4	Z5,R4

Lösung:

```
<table border="1">
```

```
<tr>
```

```
<td colspan="3">Z1, R1</td>
```

```
<td rowspan="2">Z2, R1</td>
```

```
<td rowspan="3">Z3, R1</td>
```

```
</tr>
```

```
<tr>
```

```
<td colspan="2">Z1, R2</td>
```

```
<td>Z2, R2</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Z1, R3</td>
```

```
<td colspan="2">Z2, R3</td>
```

```
<td>Z3, R3</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Z1, R4</td>
```

```
<td>Z2, R4</td>
```

```
<td>Z3, R4</td>
```

```
<td>Z4, R4</td>
```

```
<td>Z5, R4</td>
```

```
</tr>
```

```
</table>
```

*Name, Vorname:*Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

- 1) 12P
- a) Was ist MVC?
- b) Was ist ein Listener. Zu was wird er benötigt?
- c) Was bedeutet der Bezeichner "this" in Java ?
- d) Was ist ein Layout in einer GUI ?

- 2) 10P
- Der Gültigkeitsbereich (Sichtbarkeit) einer Variable bezeichnet den Bereich, mit dem auf sie durch die Verwendung ihres Namens zugegriffen werden kann.
- Unter der Lebensdauer einer Variablen versteht man den Zeitraum, in dem für die Variable Speicherplatz reserviert ist
- a) Welche Eigenschaften treffen für die folgenden Variablen zu?
- Attribut, lokale Variable, formaler Parameter einer Methode.
- Machen Sie die Kreuze (Kreuz bedeutet: trifft zu, kein Kreuz bedeutet: trifft nicht zu) an die entsprechenden Stellen der Tabelle.

	Gültigkeit innerhalb aller Methoden	Gültigkeit nur innerhalb der Methode	Lebensdauer innerhalb des ganzen Objekts	Lebensdauer nur innerhalb der Methode
Attribut				
lokale Variable				
Formaler Parameter				

- b) Welchen Wert haben nicht initialisierte Attribute und lokale Variablen bei ihrer ersten Verwendung?

3)

28P

In einem Fenster (der GUI-Klasse "Fenster") befindet sich ein Button. Jedesmal wenn man einen Punkt in Flensburg bekommt, wird dieser Button vom Anwender angeklickt.

Dadurch werden die Punkte (durch die entsprechende Methode in der Wanze "ButtonActionListener") aufsummiert.

Diese Wanze (Klasse) "ButtonActionListener" wird nun ebenfalls (durch einen "Detektiv") durch die Klasse "FlensburgMelder" überwacht. Beim Überschreiten von insgesamt 19

Flensburgpunkten soll die entsprechende Methode in der Klasse "FlensburgMelder" die Meldung "Führerschein ist weg" auf Konsole ausgeben (also mit `System.out.println(...)`)

Implementieren Sie die Klassen "Fenster", "ButtonActionListener", "FlensburgMelder" und die Hauptklasse (Startklasse) "MainKlausur", in der sich die Methode `main()` befindet.

Orientieren Sie sich dazu an den vorgegebenen 2 Programmen im Anhang.

Bemerkung: Eine Klasse, die mit `implements` ein Interface implementiert, darf zusätzlich auch noch mit `extends` eine Klasse erben.

Hilfsmittel:

Programm1:

```
// packages usw....
public class MainListener1 {
    public static void main(String[] args) {
        Fenster fenster;
        fenster = new Fenster();
    }
}

class Fenster extends javax.swing.JFrame{
    public Fenster(){
        JButton button;
        Container mycont;
        ActionListener bal;
        button = new JButton("klick mich");
        bal = new ActionListener();
        mycont = getContentPane();
        add(button);
        button.addActionListener(bal);
        this.setSize(500, 500);
        this.setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class ActionListener implements ActionListener {
    public void actionPerformed(ActionEvent ae) {
        System.out.println("es wurde gefeuert ");
    }
}
```

Programm2:

```
// packages usw....
public class MainListener2 {
    public static void main(String[] args) {
        int i;
        Detektiv watson;
        Zielperson z;
        z = new Zielperson();
        watson = new Detektiv(z);
        // An Zielperson den Lauscher anbringen
        z.addObserver(watson);
        for(i=0;i<10;i++){
            z.change();
        }
    }
}

class Zielperson extends Observable{
    private int zahl;

    public Zielperson () {
        zahl=0;
    }

    public void change() {
        zahl=zahl+1;
        setChanged();
        notifyObservers();
    }

    public int getZahl() {
        return zahl;
    }
}

class Detektiv implements Observer{
    private Zielperson zielperson;

    public Detektiv(Zielperson ziel){
        zielperson = ziel;
    }

    public void update(Observable m, Object o) {
        if (m == zielperson) {
            System.out.print(zielperson.getZahl()+" ");
        }
    }
}
```

Lösungen:

1)

a) 3P

MVC (engl: Model-View-Controller, deutsch: Modell-Präsentation-Steuerung) ist ein Entwurfsmuster zur Strukturierung von Software, das eine spätere Änderung erleichtert und eine Wiederverwendbarkeit der einzelnen Komponenten ermöglicht

b) 3P

Ein Listener ist eine Wanze. Diese Wanze wird an ein Objekt angebracht. Wenn das Objekt feuert, wird automatisch ein bestimmtes Objekt erzeugt (geworfen). Dieses Objekt wird automatisch einer speziellen Methode der Wanze übergeben, die dann automatisch ausgeführt wird.

c) 3P

Mit this wird innerhalb einer Methode eines Objekts auf dieses Objekt zugegriffen.

d) 3P

Ein Layout gibt an, wie bzw. wo die mit add eingefügten sichtbaren Komponenten (z.B. Buttons, Textfelder, usw.) angeordnet werden sollen.

2)

a) 6P

	Gültigkeit innerhalb aller Methoden	Gültigkeit nur innerhalb der Methode	Lebensdauer innerhalb des ganzen Objekts	Lebensdauer nur innerhalb der Methode
Attribut	x		x	
lokale Variable		x		x
formaler Parameter		x		x

b) Welchen Wert haben nicht initialisierte Attribute und lokale Variablen?

Attribute: 0 bzw. null 2P

lokale Variablen: undefiniert 2P

3) 28P

```
package e3fi_3_12_12_nr1;

import java.awt.Container;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Observable;
import java.util.Observer;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class MainE3FI_3_12_12_nr1 {

    public static void main(String[] args) {
        Fenster fenster;
        fenster = new Fenster();

    }
}
```

```

class Fenster extends javax.swing.JFrame {
    public Fenster() {
        JButton button;
        //JTextField text;
        Container mycont;
        ActionListener bal;
        button = new JButton("setze Punkt in Flensburg");
        bal = new ButtonActionListener();
        FlensburgMelder fm = new FlensburgMelder(bal);
        bal.addObserver(fm);
        mycont = getContentPane();
        add(button);
        button.addActionListener(bal);
        this.setSize(500, 500);
        this.setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class ButtonActionListener extends Observable implements ActionListener {
    // Punkte in Flensburg

    private int punkte;

    ButtonActionListener() {
        punkte = 0;
    }

    public void actionPerformed(ActionEvent ae) {
        //System.out.println("es wurde gefeuert ");
        punkte++;
        setChanged();
        notifyObservers();
    }

    public int getPunkte(){
        return punkte;
    }
}

class FlensburgMelder implements Observer{
    private ButtonActionListener bal;

    FlensburgMelder(ButtonActionListener bal){
        this.bal=bal;
    }

    public void update(Observable m, Object o) {
        if (m == bal) {
            if(bal.getPunkte()>=5)
                System.out.println("Führerschein weg");
            else{
                System.out.println("Punkteanzahl="+bal.getPunkte());
            }
        }
    }
}

```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programnteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

I) 52P

Bemerkung:

Alle Teilaufgaben müssen in **einem** Java-Programm realisiert werden.

Um die Veranstaltungen eines Clubs besser auf seine Gäste zuschneiden zu können, wird das unten stehende UML-Diagramm entwickelt.

Die Klasse Club verwaltet die Gäste, Filme und Musik- bzw. Theatergruppen.

Gäste können sich Filme und Gruppen wünschen.

Bem: Im UML-Diagramm sind nicht alle Methoden angegeben, aber alle Attribute.

Die im UML-Diagramm angegebenen Methoden dürfen als implementiert vorausgesetzt werden.

1) 24P

Erstellen Sie in der Klasse Club folgende Methoden:

a)

Nichtstandardkonstruktor mit 1 Parameter.

b)

Anfügen eines Films an die Filmliste.

... addFilm(...)

c)

Anfügen einer Gruppe an die Gruppenliste.

... addGruppe(...)

d)

Anfügen eines Gastes (falls dieser nicht schon vorkommt) an die Gästeliste.

... addGast(...)

e)

Zurückliefern eines Gastes aus der Gästeliste

... getGast(int index)

f)

Zurückliefern eines Films aus der Filmliste

... Film getFilm(String name)

Falls er gefunden wird, wird ein Verweis auf diesen Film zurückgegeben, ansonsten wird der Film (mit dem Name des Regisseurs: "?") an die Filmliste angehängt.

2)

28P

Realisieren Sie Folgendes in der Methode main(..) der Startklasse:

Folgendes soll (muss) implementiert werden:

a)

einen Club mit dem Namen "V3-Club" bilden.

b)

In diesen Club den Gast "Reinhold" aufnehmen.

c)

In diesen Club genau (und nur) die folgenden Filme in die Filmliste hintereinander aufnehmen:

"Mutter Theresa" von Regisseur "Theresa Lowski "

"Die Geschichte des V3" von Regisseur "Vau Drei"

"V3 für alle" von Regisseur "Vrau Drei"

d)

In diesen Club die Gruppe "Rolling Stoneds" mit dem Genre "Rockmusik" aufnehmen.

e)

Der Gast "Reinhold" des Clubs nimmt hintereinander die folgenden in c) angelegten Filme der Filmliste des Clubs in seine Wunschliste auf:

"V3 für alle" von Regissuer "Vrau Drei"

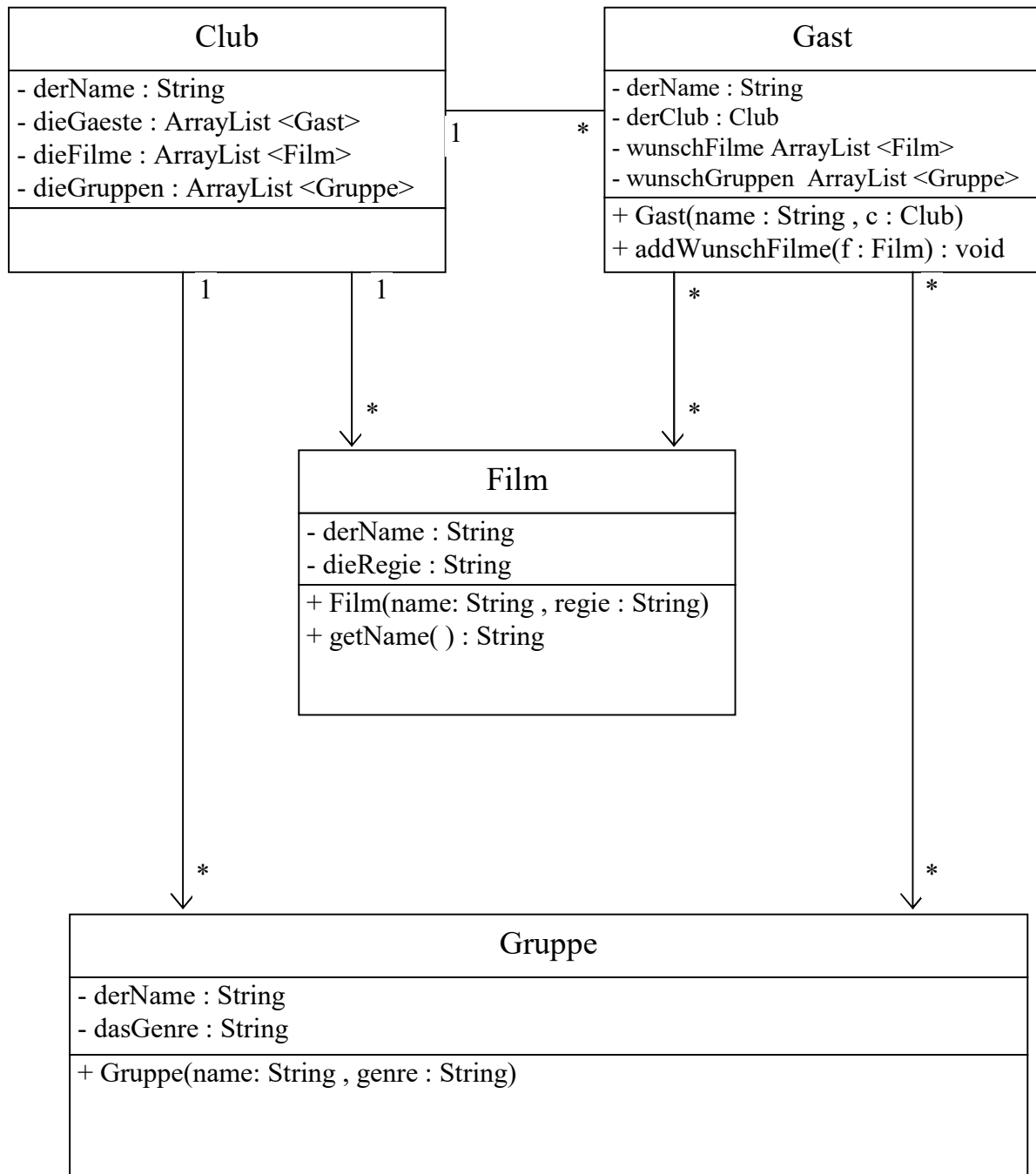
"Die Geschichte des V3" von Regissuer "Vau Drei"

f)

Suchen Sie in der Filmliste des V3-Clubs einen Film (konkret: den Film "Der V3 zerstört dein täglich Einerlei").

Falls er gefunden wird, wird ein Verweis auf ihn zurückgeliefert, ansonsten wird dieser Film an die Filmliste angehängt.

UML-Diagramm



Einige Methoden der Klasse ArrayList

1) Anfügen eines Elements (an das Ende von ArrayList)

```
public boolean add(Object e)
```

2) Entfernen eines Elements an der Stelle (Index) i

```
public Object remove (int index)
```

3) Testet ob das Element in ArrayList vorkommt (und liefert dann true zurück)

```
public boolean contains(Object e)
```

4) Testet ob ArrayList keine Elemente enthält (und liefert dann true zurück)

```
public boolean isEmpty()
```

5) Liefert den Index des Objekts zurück (falls das Objekt in ArrayList vorkommt, oder falls nicht -1)

```
public int indexOf (Object e)
```

6) Liefert die Anzahl der sich aktuell in ArrayList befindlichen Elemente zurück.

```
public int size()
```

7) Liefert das Objekt zurück, das sich an dieser Stelle (Index) von ArrayList befindet.

```
public Object get(int index)
```

Beispiel:

```
...
ArrayList <Hund> hundeListe;
hundeListe = new ArrayList <Hund>();
Hund h;
Hund h1 = new Hund("Bello", 12);
...
```

Eine Methode der Klasse String:

Methode der Klasse String:

Liefert genau dann true, wenn 2 Zeichenfolgen gleich sind, sonst false

```
boolean equals(String zeichenfolge)
```

Beispiel:

```
if(s1.equals(s2)) {
    ...
}
```


Lösungen:

1)

```
class Club{
```

```
// a)
```

5P

```
public Club(String pDerName) {
    derName = pDerName;
    dieGaeste = new ArrayList <Gast>();
    dieFilme = new ArrayList <Film>();
    dieGruppen = new ArrayList <Gruppe>();
}
```

```
// b)
```

2P

```
public void addFilm(Film pFilm) {
    dieFilme.add(pFilm);
}
```

```
// c)
```

2P

```
public void addGruppe(Gruppe pGruppe) {
    dieGruppen.add(pGruppe);
}
```

```
// d)
```

4P

```
public void addGast(Gast g) {
    if(!dieGaeste.contains(g)) {
        dieGaeste.add(g);
    }
}
```

```
// e)
```

3P

```
public Gast getGast(int index) {
    return dieGaeste.get(index);
}
```

```
// f)
```

8P

```
public Film getFilm(String name) {
    for(int i=0; i<dieFilme.size(); i++) {
        if(dieFilme.get(i) != null) {
            if(dieFilme.get(i).getName().equals(name)) {
                return dieFilme.get(i);
            }
        }
    }
    Film f = new Film(name, "?");
    dieFilme.add(f);
    return f;
}
```

2)	
// a)	2P
Club club;	
club=new Club("V3-Club");	
// b)	4P
Gast myGast = new Gast("Reinhold", club);	
club.addGast(myGast);	
// c)	9P
club.addFilm(new Film("Mutter Theresa", "Thersa Lowski"));	
club.addFilm(new Film("Die Geschichte des V3" , "Vau Drei"));	
club.addFilm(new Film(V3 für alle", "Vrau Drei"));	
// d)	3P
club.addGruppe(new Gruppe("Rolling Stoneds", "Rockmusik"));	
// e)	8P
club.getGast(0).addWunschFilme(club.getFilm("V3 für alle"));	
club.getGast(0).addWunschFilme(club.getFilm("Die Geschichte des V3"));	
// f)	2P
club.getFilm("Der V3 zerstört dein täglich Einerlei");	

KLAUSUR 2 SAE E3FI 13.1.2014 Zeit: 90 Minuten
Systemintegratoren

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

Bemerkung:

Alle Implementierungen mit Hilfe der Programmiersprache Java erstellen.

I) 40P

Für die Geräteverwaltung der Firma Machheim soll ein Programm entwickelt werden.
Zunächst werden nur Recher verwaltet, später sollen auch andere Geräte (z.B. Kopierer, usw.) integriert werden.

Eigenschaft	Datentyp	Beschreibung
raumId	String	Raum-Nummer des Gerätes
bKosten	double	Beschaffungskosten in Euro

Geraet
raumID : String # bKosten : double
+ Geraet(raumID: String , bKosten : double) + getBKosten() : double + setBKosten(bKosten : double) : void + getRaumID() : String + setRaumID(raumID : String) : void

1) 12P
Implementieren Sie die Klasse Geraet (vgl. UML-Klassendiagramm) mit den entsprechenden Attributen und Methoden des UML-Klassendiagramms.

2)

10P

Implementieren Sie eine Methode `aktuelleGeraete()`, die den aktuellen Wert eines Gerätes in Abhängigkeit vom Alter in Jahren ermittelt.

Eingabeparameter der Methode ist das Alter in Jahren und der Rückgabewert ist der aktuelle Geldwert.

Von folgendem Werteverlust wird ausgegangen:

Wert eines Rechners in Abhängigkeit vom Alter				
1. Jahr	2. Jahr	3. Jahr	4. Jahr	danach
0 %	50 %	75 %	87,5 %	90 %

Bemerkung:

Diese Aufgabe muß mit Hilfe eines Feldes (lokale Variable) gelöst werden.

In diesem Feld müssen die Werteverluste gespeichert werden.

3)

18P

Für die Verwaltung der Rechner wird die Klasse `Geraet` erweitert und eine neue Klasse `Rechner` entworfen.

In der Klasse `Rechner` werden die zusätzlichen Eigenschaften, wie der Rechnername und der Domänenname sowie die entsprechenden Setter - und Getter-Methoden implementiert.

Erweitern Sie das obige UML-Klassendiagramm.

Eigenschaft	Datentyp	Beschreibung
<code>rName</code>	String	Rechnername (Bsp: pizza)
<code>rDomain</code>	String	Domänenname (Bsp: machheim.com)

II)

13P

Die Methode `checkIP(char[] str)` überprüft den input-Parameter auf eine gültige IP-Adresse.

Das Feld `str` ist immer mit dem Zeichen 'x' beendet, wobei vorausgesetzt wird, daß 'x' nicht noch einmal in der Zeichenfolge vorkommt.

Implementieren Sie die zugehörige Methode.

Beispiele für den Input-Parameter:

- a) 123.4.5.56x
- b) 6789.23.5.e.6x
- c) 6..5.23.y.e.1x
- d) 3.5.1x

Bei gültiger Eingabe liefert die Methode den Wert `true` zurück, sonst `false`.

Lösungen

I)

1)+2)

12P + 10P

```
class Geraet{
    protected String raumID;
    protected double bKosten;

    public Geraet(String raumID, double bKosten) {
        this.raumID = raumID;
        this.bKosten = bKosten;
    }

    public String getRaumID() {
        return raumID;
    }

    public void setRaumID(String raumID) {
        this.raumID = raumID;
    }

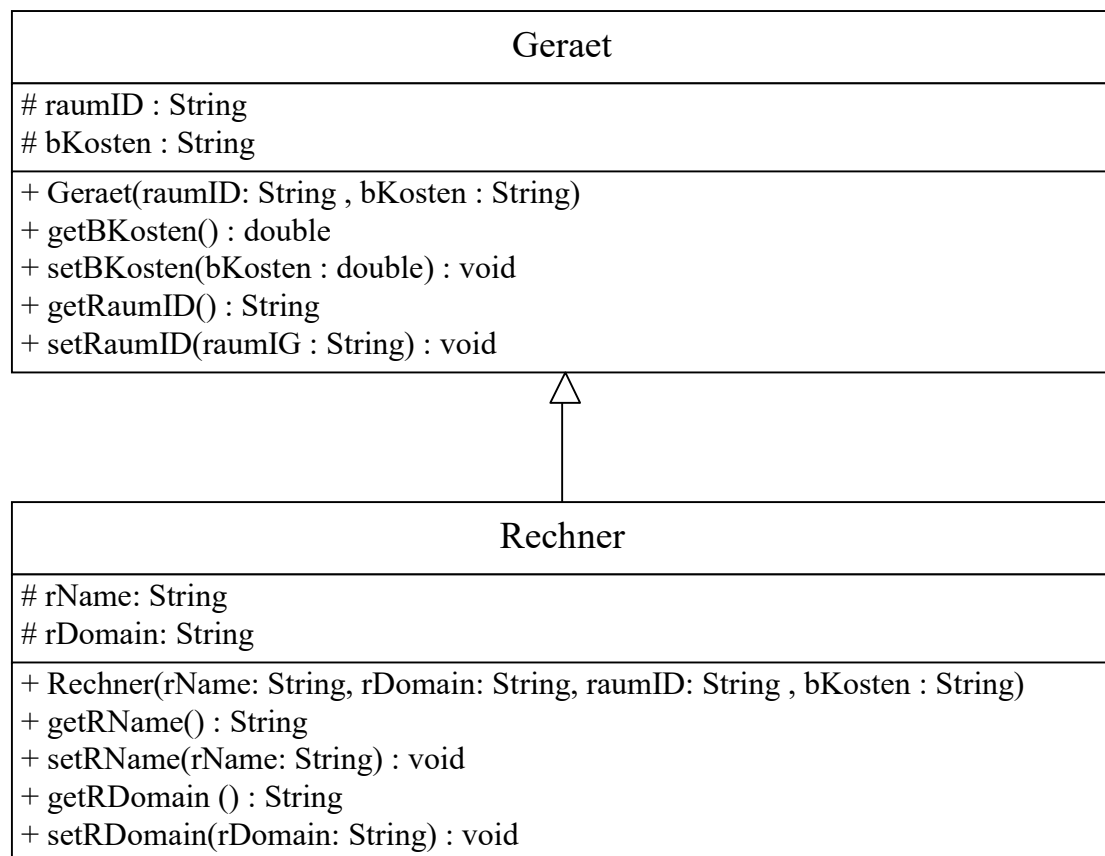
    public double getbKosten() {
        return bKosten;
    }

    public void setbKosten(double bKosten) {
        this.bKosten = bKosten;
    }

    public double aktuellerGeraeteWert(int jahr){
        double[] werte = {0 , 50, 75, 87.5 , 90};
        if(jahr>0 && jahr<5){
            return (100-werte[jahr-1])/100;
        }
        else{
            return (100-werte[4])/100;
        }
    }
}
```

3)
14P für UML-Klassendiagramm und 4P für Vererbung

18P



II)
Programm fehlt noch

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

I) 50P
1)

Ein Spielsüchtiger will auf Anraten seines Freundes seine Wetten verwalten. Dazu wird ein Entwickler beauftragt die Klasse "**Verwaltung**" zu implementieren.

In der Klasse Verwaltung gibt es das Attribut (integer-Feld) "**wetten**", in dem der Gewinn (bzw. Verlust) bei einer Wette abgespeichert wird. Dieses Feld ist kein dynamisches Feld.

Die Länge des Feldes wird in dem Attribut "**gesamtLaenge**" abgespeichert und beim Anlegen eines Objekts der Klasse Verwaltung festgelegt.

Das Ende des Feldes wetten wird immer größer, je mehr Wetten abgespeichert werden. Dieses Ende wird in dem integer-Attribut "**ende**" gespeichert und gibt den Index an, wo der nächste neue Wettgewinn abgespeichert wird.

Bemerkung:

Der Gewinn bzw. Verlust bei einer Wette wird mit Wettgewinn bezeichnet. Dieser Begriff wird auch benutzt, wenn die Wette mit einen Verlust endet (dann ist der Wettgewinn eben negativ).

Beispiel:

10	-20	30			
----	-----	----	--	--	--



gesamtLaenge = 6

ende = 3

Gesamtgewinn = $10 - 20 + 30 = 20$

Wenn nun ein neuer Wettgewinn (z.B. 100 Euro verloren) angefügt wird, bedeutet dies, daß die der mit dem Wert 30 folgenden Zelle mit dem -100 belegt wird. Insbesondere ändert sich dadurch der Wert des Attributs ende.

Erstellen Sie in der Klasse Verwaltung genau (und nur) folgende Methoden:

a) 6P

... Verwaltung(...) :

Konstruktor, der die Gesamtlänge des Feldes festlegt und Speicher für das Feld wetten im Arbeitsspeicher reserviert.

b) 6P

... anfüegen(...) :

fügt einen Wettgewinn an das Ende der bisherigen Wettgewinne an.

Wenn das Feld voll ist, werden keine neuen Wettgewinne angefügt.

c) 6P

... gibErgebnis (...) :

gibt den Wettgewinn einer Wette zurück, der in einer bestimmten Zelle abgespeichert wurde.

Dazu muß auch überprüft werden, ob sich die Zelle (deren Inhalt zurückgegeben werden soll) innerhalb der bis jetzt angelegten Wettgewinne befindet.

d) 6P

... aendern(...)

Falls ein falscher Wettgewinn in eine Zelle eingetragen wird, kann man diesen nachträglich abändern. Dazu muß auch überprüft werden, ob sich die Zelle (deren Inhalt geändert werden soll) innerhalb der bis jetzt angelegten Wettgewinne befindet.

e) 6P

... loeschen(...)

Löscht den letzten Wettgewinn aus dem Feld wetten. Das bedeutet nicht, daß der letzte Wettgewinn 0 wird, sondern daß der letzte Eintrag getilgt wird.

f) 7P

... bilanzieren(...)

Bildet die Summe aller Wettgewinne und zeigt damit dem Spieler effektiv an, was er in Wirklichkeit insgesamt gewonnen bzw. verloren hat.

g) 6P

... ausgabe(...)

Gibt alle Wettgewinne auf dem Bildschirm aus.

2)

Realisieren Sie (mit Hilfe der entsprechenden Methode der Klasse Verwaltung) Folgendes in der Methode main(..) der Startklasse:

a) 1P

Erzeugen Sie das Objekt "**verwalten**" der Klasse Verwaltung, so daß das Feld (Attribut) "wetten" die Länge 10 hat.

b) 4P

Fügen Sie im Feld "wetten" die Wettgewinne -100 , -200, +30, +1000 hintereinander ein.

c) 1P

Löschen Sie den letzten Wettgewinn aus der Wettgewinnliste.

d) 1P

Bilden Sie die Bilanz ihrer Wettgewinne.

Lösungen:

```
class Verwaltung{
    private int ende;
    private int gesamtLaenge;
    private int[] wetten;
    // 6P
    public Verwaltung(int gesamtLaenge){
        this.gesamtLaenge=gesamtLaenge;
        ende=0;
        wetten = new int[gesamtLaenge];
    }
    // 6P
    public int gibErgebnis(int index){
        if(index >=0 && index < ende){
            return wetten[index];
        }
        else{
            return -1;
        }
    }
    // 6P
    public void anfüegen(int wert){
        if(ende<=gesamtLaenge-1){
            wetten[ende]=wert;
            ende++;
        }
    }
    // 6P
    public void ändern(int index, int wert){
        if(index>=0 && index < ende){
            wetten[index]=wert;
        }
    }
    // 6P
    public void löschen(){
        if(ende>0){
            ende--;
        }
    }
    // 7P
    public int bilanzieren(){
        int i;
        int summe=0;
        for(i=0;i<ende;i++){
            summe=summe+wetten[i];
        }
        return summe;
    }
    // 6P
    public void ausgabe(){
        int i;
        for(i=0;i<ende;i++){
            System.out.println("Wette["+i+"]="+wetten[i]);
        }
    }
    // 7P
}

public class Startklasse {
    public static void main(String[] args) {
        int i;
        Verwaltung verwaltung = new Verwaltung(10);
        verwaltung.anfüegen(-100);
        verwaltung.anfüegen(-200);
        verwaltung.anfüegen(+30);
        verwaltung.anfüegen(+1000);
        verwaltung.löschen();
        verwaltung.bilanzieren();
    }
}
```

KLAUSUR 3 SAE E3FI 17.3.2014 Zeit: 90 Minuten
Anwendungsentwickler

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

I) 51P

Bemerkung:

Alle Programmier-Teilaufgaben müssen in **einem** Java-Programm realisiert werden.

Die Gehaltsabrechnung einer Firma soll mit Hilfe der OOP realisiert werden.

In der Firma sind die Arbeiter, Manager und Angestellten, also die Mitarbeiter (mit jeweils einer Personalnummer) beschäftigt. Die Bruttomonatslöhne der Mitarbeiter werden wie folgt berechnet:

Arbeiter : stundenlohn * anzahlstunden

Angestellter : grundgehalt + ortszuschlag;

Manager : fixgehalt + umsatz * provision / 100

Das Programm soll multipersonal entwickelt werden.

U.a. sollen die Entwickler der entsprechenden Klassen gezwungen werden, die Methode ... monatsBruttolohn(...) zu entwickeln.

1) 28P

a) 11P

Erstellen Sie ein vereinfachtes UML-Diagramm ohne Attribute, aber mit Methode monatsBruttolohn(), das die Beziehungen zwischen den Klassen "Arbeiter", "Manager" "Angestellten" und "Mitarbeiter" darstellt.

Evtl. vorhandene abstrakte Klassen sind in geeigneter Weise zu kennzeichnen.

b) 8P

Implementieren Sie die Klasse Mitarbeiter mit einem Konstruktor.

c) 9P

Implementieren Sie die Klasse Arbeiter mit einem Konstruktor und der Methode ... monatsBruttolohn(...)

2) 23P

In der Klasse Gehaltsberechnung werden die Mitarbeiter in einer dynamischen Liste (ArrayList) verwaltet. Hinweise zu ArrayList siehe unten.

Implementieren Sie die Klasse Gehaltsberechnung mit einer dynamischen Liste als Attribut und den folgenden Methoden:

a)

Konstruktor

b)

...addMitarbeiter(...)

fügt einen Mitarbeiter in die Liste ein.

c)

...bestVerdiener(...)

gibt den Mitarbeiter mit dem höchsten monatlichen Bruttolohn zurück.

d)

... istSchlechterMonatBruttolohn(...)

gibt true zurück, falls der Bruttomonatslohn eines Mitarbeiters schlechter ist als der Mittelwert (Durchschnitt) aller Bruttomonatslöhne aller Mitarbeiter der Liste.

Einige Methoden der Klasse ArrayList

1) Anfügen eines Elements (an das Ende von ArrayList)

```
public boolean add(Object e)
```

2) Entfernen eines Elements an der Stelle (Index) i

```
public Object remove (int index)
```

3) Testet ob das Element in ArrayList vorkommt (und liefert dann true zurück)

```
public boolean contains(Object e)
```

4) Testet ob ArrayList keine Elemente enthält (und liefert dann true zurück)

```
public boolean isEmpty()
```

5) Liefert den Index des Objekts zurück (falls das Objekt in ArrayList vorkommt, oder falls nicht -1)

```
public int indexOf (Object e)
```

6) Liefert die Anzahl der sich aktuell in ArrayList befindlichen Elemente zurück.

```
public int size()
```

7) Liefert das Objekt zurück, das sich an dieser Stelle (Index) von ArrayList befindet.

```
public Object get(int index)
```

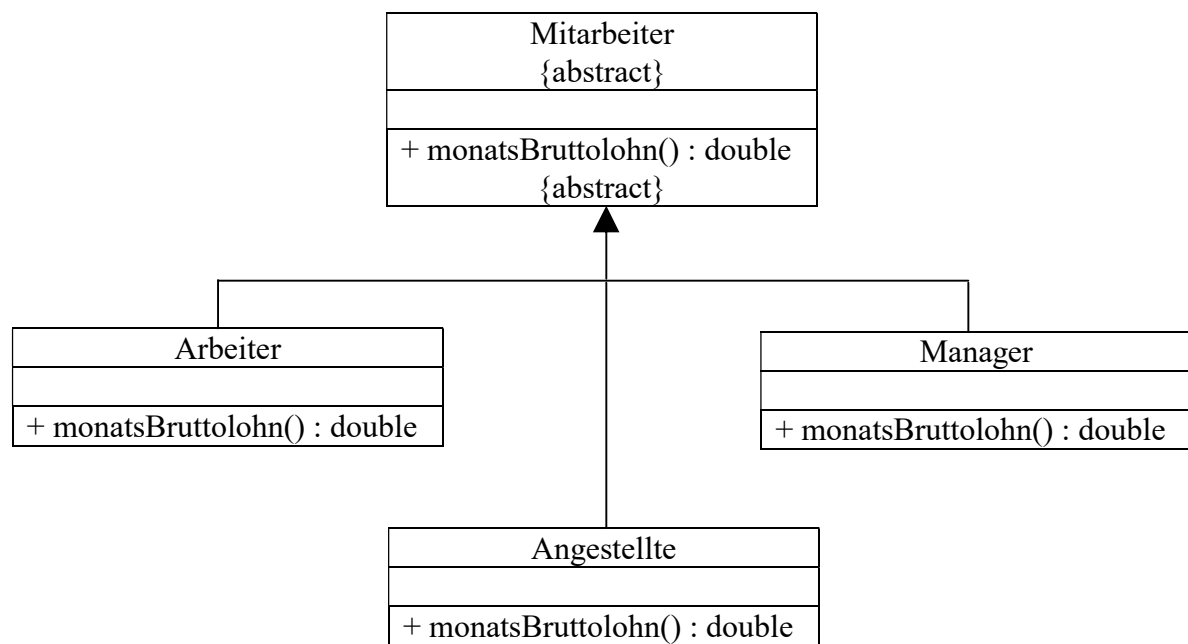
Beispiel:

```
...
ArrayList <Hund> hundeListe;
hundeListe = new ArrayList <Hund>();
Hund h;
Hund h1 = new Hund("Bello", 12);
...
```

Lösungen:

1)a)

11P



b)

8P

```

abstract class Mitarbeiter{
    private int persnr;

    public Mitarbeiter(int persnr){
        this.persnr=persnr;
    }

    public int getPersnr(){
        return persnr;
    }

    public abstract double monatsBrutto();
}
    
```

c)

9P

```

class Arbeiter extends Mitarbeiter{
    private double stundenlohn;
    private anzahlstunden;

    public Arbeiter(int persnr,double stundenlohn, double anzahlstunden){
        super(persnr);
        this.stundenlohn=stundenlohn;
        this.anzahlstunden=anzahlstunden;
    }

    public double monatsBrutto(){
        return stundenlohn*anzahlstunden;
    }
}
    
```

2)

23P

```
class Gehaltsberechnung{
    private ArrayList <Mitarbeiter> mitarbeiterListe; // 1P

    public Gehaltsberechnung(){ // 2P
        mitarbeiterListe = new ArrayList <Mitarbeiter>();
    }

    public void addMitarbeiter(Mitarbeiter m){ // 2P
        mitarbeiterListe.add(m);
    }

    // polymorphe Rückgabe
    public Mitarbeiter bestVerdiener(){ // 8P
        double maximum=0;
        int index=0;
        int i;
        for(i=0;i<mitarbeiterListe.size();i++){
            if(mitarbeiterListe.get(i).monatsBrutto()>maximum){
                index=i;
                maximum=mitarbeiterListe.get(i).monatsBrutto();
            }
        }
        return mitarbeiterListe.get(index);
    }

    // polymorpher Parameter
    public boolean istSchlechtesBrutto(Mitarbeiter m){ // 10P
        double summe=0;
        double mittelwert=0;
        int index=0;
        int i;
        for(i=0;i<mitarbeiterListe.size();i++){
            summe=summe+mitarbeiterListe.get(i).monatsBrutto();
        }
        mittelwert=summe/mitarbeiterListe.size();

        if(m.monatsBrutto()<mittelwert)
            return true;
        else
            return false;
    }
}
```

KLAUSUR 3 SAE E3FI 17.3.2014 Zeit: 90 Minuten
Systemintegratoren

Name, Vorname:

Hilfsmittel:

zwei DIN A4-Seiten handschriftliche Aufschriebe (keine Kopien).

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programnteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

I) 50P

Die Verwaltung der Hard - und Software Komponenten des Intranets werden zentral von der EDV-Abteilung erfasst. Dazu wurde ein Programm entwickelt. Ein Teil dieses Programms ist in folgendem UML-Diagramm beschrieben:

ArbeitsplatzPC
- macAdresse : String - netz : char
+ ArbeitsplatzPC (pMacAdresse : String , pNetz : char) + setNetz(pNetz:char) : boolean + getNetz() : char + setMAC(pMacAdresse : String) : boolean + getMAC() : String - checkNetz (pNetz : char) : boolean - checkMAC(pMacAdresse : String) : boolean - checkHEX(pNetz : char) : boolean

Für das Attribut netz sind entsprechende der Netztopologie nur die Werte A und B gültig. Kleinbuchstaben sind auch erlaubt und werden in einer Methode in Großbuchstaben umgewandelt. Die Methode ArbeitsplatzPC(pMacAdresse : String , pNetz : char) setzt die Attribute der macAdresse und netz.

1)

2P

Die Methode ArbeitsplatzPC (pMacAdresse : String , pNetz : char) hat keinen Rückgabewert. Ist dies ein Fehler? Begründen Sie Ihre Antwort.

2)

16P

Erstellen Sie in Java den Quellcode für die Klasse ArbeitsplatzPC.

Implementieren Sie nur die öffentlichen Methoden.

Die privaten Methoden testen nur die Zeichenketten bzw. Zeichen auf ihre Korrektheit.

3)

Zur Erfassung der Daten für die einzelnen Geräte werden die privaten Methoden eingesetzt um die Parameter zu überprüfen. Die MAC-Adresse wird als 6 Byte Wert in hexadezimaler Form dargestellt, konkret im Format xx-xx-xx-xx-xx-xx, wobei x einen ganzzahligen Wert zwischen 0-9 oder A-F oder a-f annimmt. Beispiel: 7a-4B-f8-78-23-3F

3.1)

12P

Die Methode boolean checkHex(char z) überprüft den input-Parameter z.

Sie liefert den Wert true zurück, wenn das Zeichen die Ziffern 0, ..., 9, a - f oder A - F annimmt (gültige HEX-Zeichen).

Implementieren Sie diese Methode in der Programmiersprache Java.

Methode checkHex(char z)

z						
'0'	'1'	...	'a', 'A'	...	'f', 'F'	sonst
Rückgabe true	Rückgabe true	Rückgabe true	Rückgabe true	Rückgabe true	Rückgabe true	Rückgabe false

4)

20P

Die Methode checkMac(...) überprüft den input-String auf eine gültige MAC-Adresse, die wie folgt dargestellt wird: xx-xx-xx-xx-xx-xx

wobei ein x ein Zeichen ist mit 0, ..., 9, a - f oder A - F.

Beispiel: 7A-4B-F8-78-23-3F

Bei gültiger Eingabe liefert die Methode den Wert true zurück.

Zunächst wird die Länge des Strings kontrolliert, dann die Eingabefolgen der HEX-Ziffern und des Trennzeichens (d.h. -).

Verwenden Sie die Methode checkHex(...) aus der vorigen Aufgabe.

Implementieren Sie die Methode checkMac(...) in der Programmiersprache Java.

Bemerkungen (Einige Methoden der Klasse String):

int length() : gibt Länge des Strings zurück

char charAt(int i) : gibt das Zeichen am Index i zurück (Zählung beginnt bei 0)

Lösungen:

1) 2P
Es ist kein Fehler. Die Methode ArbeitsplatzPC (...) ist ein Konstruktor.

2) 16P

```
class ArbeitsplatzPC{  
    private String macAdresse;           // 1P  
    private char netz;                   // 1P  
  
    public ArbeitsplatzPC (String pMacAdresse, char pNetz){ // 2P  
        macAdresse= pMacAdresse;  
        netz= pNetz;  
    }  
  
    public boolean setNetz(char pNetz){ // 4P  
        if(checkNetz (pNetz)== true)){  
            netz= pNetz;  
            return true;  
        }  
        else{  
            return false;  
        }  
    }  
  
    public char getNetz(){ // 2P  
        return netz;  
    }  
  
    public boolean setMAC(String pMacAdresse){ // 4P  
        if(checkNetz (pMacAdresse)== true)){  
            macAdresse = pMacAdresse;  
            return true;  
        }  
        else{  
            return false;  
        }  
    }  
  
    public String getMAC(){ // 2P  
        return macAdresse;  
    }  
}
```

3) 12P

```
boolean checkHex(char z){  
    switch(z){  
        case '0':  
        case '1':  
        ...  
        case 'F':  
            return true;  
        default: return false;  
    }  
}
```


4)

20P

```
boolean checkMac(String pMacAdresse){
    int i;
    boolean back=true;
    if(pMacAdresse.length()==17)
        back=false;
    else{
        for(i=0;i<17;i++){
            if(i%3==2){
                if(pMacAdresse.charAt(i)!='-'){
                    back=false;
                    break;
                }
            }
            else{
                if(checkHex(z)==false{
                    back=false;
                    break;
                }
            }
        }
    }
    return back;
}
```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

Bem:

Es dürfen keine Klassen (wie z.B. ArrayList, Vector, usw.) der Java-Entwicklungsumgebung verwendet werden.

1) 50P

Erstellen Sie die Klasse DynamischerSpeicher mit genau den folgenden Attributen (und den entsprechenden get und set-Methoden)

```
private int[] feld;  
private int len;
```

Diese Klasse muß zusätzlich noch folgende Methoden enthalten:

1.1) public void anfüegen(int zahl) :

fügt eine Zahl an das Ende des Feldes an.

Dies soll dadurch realisiert werden, daß ein neues Feld erstellt wird (mit einer um 1 größeren Länge als das alte Feld). Die Zahl muß an das Ende des neuen Feldes angefügt werden.

Der Inhalt des alten Feldes muß dann noch in das neue Feld kopiert werden. Dann wird das neue Feld dem alten Feld zugewiesen.

1.2) public void printFeld() :

gibt alle Elemente des Feldes auf dem Bildschirm aus.

1.3) In main() soll mit Hilfe einer Schleife und der Methode anfüegen(...) die Zahlen 100, 101, 102, ... , 120 in das Feld eingefügt werden.

Danach sollen diese Zahlen auf dem Bildschirm ausgegeben werden.

1.4) schwierigere Zusatzaufgaben:

Erstellen Sie die Methoden, die folgendes leisten:

1.4.1) public void entferneLetztesElement() :

entfernt das letzte Element des Feldes, wobei die Länge des Feldes angepaßt (um 1 verringert) wird.

1.4.2) public void einfuegenAnPosition (int pos) :

einfügen eines Elements an der Stelle pos i des Feldes (mit Längenangepaßung).

1.4.3) `public void entferneAnPosition (int pos) :`
entfernen eines Elements an der Stelle pos des Feldes (mit Längenangepassung).

1.4.4) `public void einfuegen(int zahl) :`
die Zahl zahl wird so eingefügt (mit Längenangepassung), daß nach jedem Einfügen das Feld in aufsteigender Reihenfolge sortiert ist (dabei keine von der Java-Entwicklungsumgebung vorgegebene Sortiermethode verwenden).

1.4.5) `public void entferneZahl (int zahl) :`
wenn die Zahl zahl das erste Mal im Feld vorkommt, wird sie entfernt (mit Längenangepassung).
Falls sie nicht vorkommt, passiert nichts.

KLAUSUR 3 SAE E3FI NACHTERMIN Zeit: 45 Minuten
Systemintegratoren

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programnteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

1) 50P

Entwickeln Sie ein Programm, das einen Kilometerzähler simuliert.

Der Kilometerzähler soll hochgezählt werden und der aktuelle Kilometerstand auf dem Bildschirm ausgegeben werden (z.B. von 00...00 bis 99..99 hochzählen)

Das Programm muß mit Hilfe eines Integer-Feldes der Länge LEN (z.B. LEN = 4) realisiert werden.

Die Anzahl der verschachtelten Schleifen darf nicht von LEN abhängen.

Es dürfen auch keine Klassen der Java-Entwicklungsumgebung (wie z.B. String) oder deren Methoden (wie z.B. charAt()) verwendet werden.

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

1)

Eine einfache Geschwindigkeits-Messanlage besteht aus 2 Lichtschranken, die im Abstand d (in Meter) voneinander am Straßenrand aufgestellt sind. Die Geschwindigkeit v eines vorbeifahrenden KFZ wird berechnet aus den Zeitpunkten t_1 und t_2 (jeweils in Sekunden), in denen die Lichtschranken unterbrochen wurden, also: $v = 3,6 * d / (t_2 - t_1)$ wobei v die Dimension km/h hat.

Diese Berechnung erfolgt in der Basisklasse "Messanlage".

Am Ortseingang gibt es die eine Warntafel, die dem Autofahrer anzeigt, mit welcher Geschwindigkeit er in den Ort einfährt ("Sie fahren 55km/h"). Diese wird durch die Klasse "Warntafel" modelliert.

Im Ort steht ein Blitzer. Er blitzt zusätzlich bei Überschreitung einer Geschwindigkeit von 60 km/h und speichert Geschwindigkeit und Blitzsituation ab. Diese wird durch die Klasse "Blitzer" modelliert.

Methoden der Klasse Messanlage:

Messanlage(double abstand) :

Der Konstruktor hat als Parameter den Abstand der Lichtschranken (in Meter). Der Konstruktor kopiert den Parameter abstand in das Attribut d .

void messen(double t_1 , double t_2)

t_1 ist der Zeitpunkt der Unterbrechung der ersten Lichtschranke, t_2 der zweiten. Die Geschwindigkeit des durchfahrenden Fahrzeuges wird berechnet und im Attribut v gespeichert.

void ausgeben(void)

Diese Methode wird nur deklariert und nur in den Unterklassen ausprogrammiert.

Methoden der Klasse Warntafel:

Warntafel(double abstand)

Dieser Konstruktor ruft den Basiskonstruktor von Messanlage auf und übergibt ihm den Parameter abstand.

void ausgeben(void)

gibt die gemessene Geschwindigkeit aus (z.B: "Sie fahren 55km/h!").

Methoden der Klasse Blitzer:

Blitzer(String stdort, double abstand)

Dieser Konstruktor speichert einen Standort-Name im Attribut standort, ruft den Basiskonstruktor von Messanlage auf und übergibt ihm den Parameter abstand.

void messen(double t1, double t2)

überschreibt die gleichnamige Methode der Basisklasse Messanlage!

Die Berechnung der Geschwindigkeit erfolgt, indem die Methode der Oberklasse verwendet wird. Zusätzlich wird das Attribut blitz gleich 1 gesetzt, wenn die Geschwindigkeit v größer 60km/h beträgt, sonst ist blitz gleich 0.

void ausgeben(void)

gibt die gemessene Geschwindigkeit aus, zusätzlich den Standort des Blitzers und das Wort "GEBLITZT!", wenn geblitzt wurde (bei blitz = 1).

Beispiel: Der folgende Quelltext erzeugt die Bildschirm-Ausgabe:

```
public static void main(String[] args) {
    Warntafel w = new Warntafel(20);
    w.messen(170.1, 172.1);
    w.ausgeben();

    Blitzer b = new Blitzer("Parkstrasse", 20);
    b.messen(201.2, 203.2);
    b.ausgeben();
    b.messen(321.9, 322.9);
    b.ausgeben();
}
```

Bildschirmausgaben

// 1. Bildschirmausgabe

Sie fahren 36.0 km/h

// 2. Bildschirmausgabe

* Standort: Parkstrasse

* Geschwindigkeit: 36.0 km/h

// 3. Bildschirmausgabe

* Standort: Parkstrasse

* Geschwindigkeit: 72.0 km/h GEBLITZT!

AUFGABEN:

a) Erstellen Sie das zugehörige UML-Diagramm.

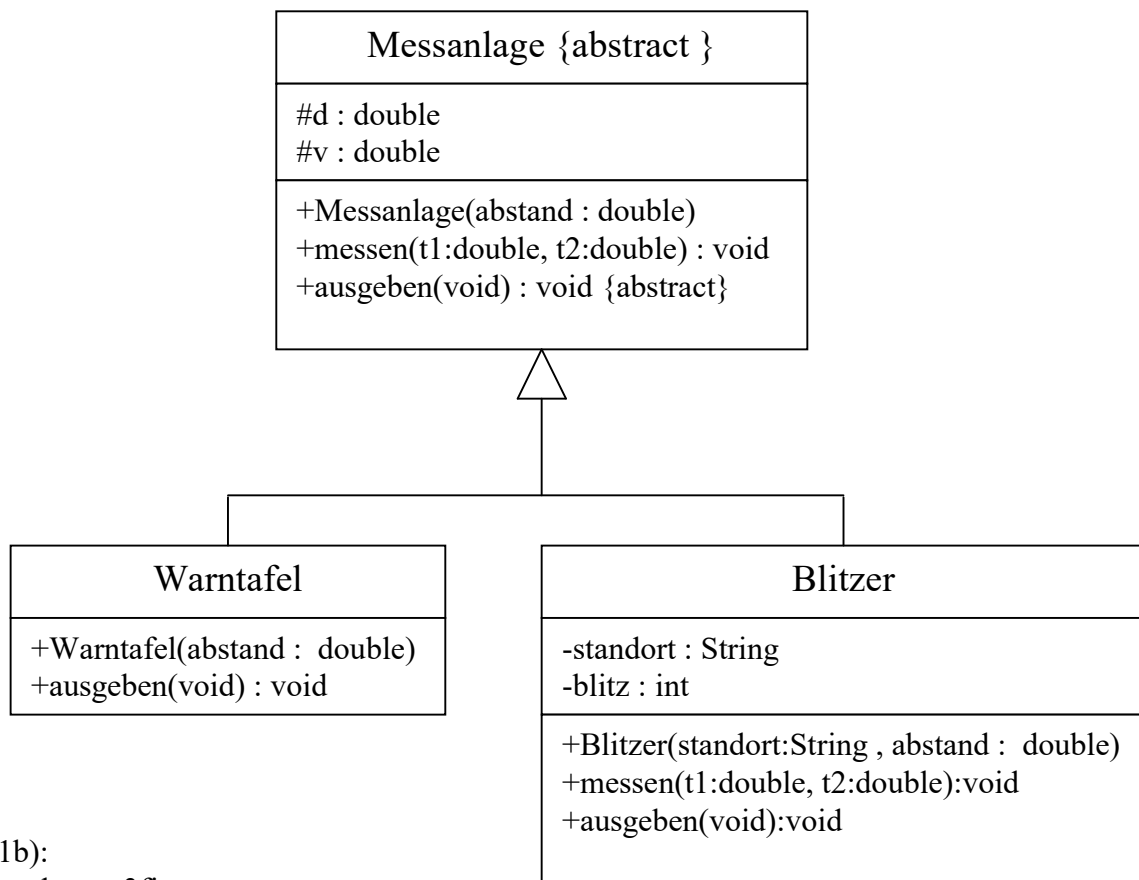
16P

b) Implementieren Sie die 3 Klassen mit den oben beschriebenen Methoden..

34P

Lösungen:

1a)



1b):

```
package e3fi;
```

```
public class MainE3FI {
```

```
    public static void main(String[] args) {
        Warntafel w = new Warntafel(20);
        w.messen(170.1, 172.1);
        w.ausgeben();
```

```
        Blitzer b = new Blitzer("Parkstrasse", 20);
        b.messen(201.2, 203.2);
        b.ausgeben();
        b.messen(321.9, 322.9);
        b.ausgeben();
```

```
    }
}
```

```
abstract class Messanlage{
    protected double d;
    protected double v;
```

```
    public Messanlage(double abstand){
        d=abstand;
    }
}
```

```

    public void messen(double t1, double t2) {
        v = (d/(t2 - t1))*3.6;
    }

    public abstract void ausgeben();
}

class Warntafel extends Messanlage{
    public Warntafel(double abstand){
        super(abstand);
    }

    public void ausgeben() {
        System.out.println("\nSie fahren " + v + " km/h");
    }
}

class Blitzer extends Messanlage{
    private String standort;
    private int blitz;

    public Blitzer(String ort, float abstand){
        super(abstand);
        standort = ort;
    }

    public void messen(double t1, double t2) {
        //v = (d/(t2 - t1))*3.6;
        super.messen(t1, t2);
        blitz = 0;
        if(v > 60.0) {
            blitz = 1;
        }
    }

    public void ausgeben(){
        System.out.println("\n*****");
        System.out.println("* Standort: " + standort);
        if(blitz==0)
            System.out.println("* Geschwindigkeit: "+v+ " km/h");
        else
            System.out.println("* Geschwindigkeit: "+v+ " km/h GEBLITZT! ");
        System.out.println("*****");
    }
}

```