

*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 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 !!!!

1) 50P

In einem Fenster einer grafischen Oberfläche befindet sich ein Button mit der Beschriftung "Flensburg-Punkt". Jedesmal wenn man einen Punkt in Flensburg bekommt, wird dieser Button vom Anwender angeklickt. Beim Überschreiten von insgesamt 19 Flensburgpunkten soll die Meldung "Führerschein ist weg" ausgegeben werden.

Zur Vereinfachung soll diese Meldung nicht in einem Textfenster der grafischen Oberfläche, sondern auf Konsole (also mit `System.out.println(...)`) ausgegeben werden.

Das entsprechende Programm soll mit Hilfe des Entwurfsmusters MVC realisiert werden.

Es müssen als die folgenden Klassen implementiert werden:

Modell, View , Control und die Hauptklasse (Startklasse) "MainKlausur"

Orientieren Sie sich dazu an den vorgegebenen 2 Programmen (siehe unten).

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;
        ButtonActionListener bal;
        button = new JButton("klick mich");
        bal = new ButtonActionListener();
        mycont = getContentPane();
        add(button);
        button.addActionListener(bal);
        this.setSize(500, 500);
        this.setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class ButtonActionListener 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:

```
package e3fi_nachtermin1_12_12;

import java.awt.Container;
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.JTextField;

public class MainE3FI_nachtermin1_12_12 {

    public static void main(String[] args) {
        Controller controller = new Controller();
    }

    class Modell extends Observable {

        private int punkte;

        void vermehrePunkte() {
            punkte++;
            setChanged();
            notifyObservers();
        }

        int getPunkte() {
            return punkte;
        }
    }

    class Controller {
        private View view;
        private Modell modell;

        public Controller() {
            modell = new Modell();
            view = new View(modell);
            /* erstelle einen Listener und hefte diesen an den
               Button in der View. Der Controller enthält also
               (in der lokalen Variablen bal) eine einfangende
               Methode für die Eingabe (also Klicken des Buttons).
            */
            ButtonActionListener bal = new
                ButtonActionListener(modell);
            view.setButtonActionListener(bal);
        }
    }
}
```

```

class View extends javax.swing.JFrame implements Observer {
    private Modell modell;
    private JButton button;

    public View(Modell modell) {
        Container mycont;
        ActionListener bal;
        this.modell = modell;
        modell.addObserver(this);
        button = new JButton("erzeuge Punkt in Flensburg");
        mycont = getContentPane();
        add(button);
        this.setSize(500, 500);
        this.setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void update(Observable m, Object o) {
        if (m == modell) {
            System.out.println("Punkte in Flensburg=" +
                               modell.getPunkte());

            if (modell.getPunkte() > 5) {
                System.out.println("Führerschein weg");
            }
        }
    }

    void setButtonActionListener(ActionListener bal) {
        button.addActionListener(bal);
    }
}

class ButtonActionListener implements ActionListener {
    private Modell modell;

    ButtonActionListener(Modell modell) {
        this.modell = modell;
    }

    public void actionPerformed(ActionEvent ae) {
        modell.vermehrePunkte();
        //System.out.println("Punkte="+modell.getPunkte());
    }
}

```

Name, Vorname:

Hilfsmittel:

2 Seiten selbstgeschriebene 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

Bemerkung:

Das erste Tag in einer HTML-Datei ist hier immer <html>

1) 5P

Geben Sie ein konkretes Beispiel an (kein Quellcode), was man mit CSS und HTML im Gegensatz zu reinem HTML machen kann.

2) 15P

Welche 3 Möglichkeiten gibt es Style-Sheets in einer HTML-Datei zu verwenden ("einbinden") ? Geben Sie dazu 3 konkrete Beispiele an.

3) 10P

Erstellen Sie den HTML-Quellcode, der die folgende, mit einem Rahmen versehene Tabelle auf dem Bildschirm ausgibt.

Für die Tabelle dürfen **nur** die Tags <table>, <tr>, <td> bzw. die entsprechenden schließenden Tags verwendet werden.

	Mathe	Physik	EDV
Note 1	1	2	3
Note 2	3	4	5
Mittel	2	3	4

4) 20P

Die Überschriften Mathe, Physik, EDV der obigen Tabelle müssen jeweils auf die 3 Zellen bezogen zentriert werden. Die Rahmen dieser 3 einzelnen Zellen müssen jeweils rot sein. Außerdem ist die Hintergrundfarbe genau dieser 3 Zellen gelb und der Rahmen der gesamten Tabelle grün. Dies muss mit Hilfe von css realisiert werden.

a) Erstellen Sie den Quellcode für die entsprechende Tabelle.

Die Style-Sheets müssen sich im head-Teil des Quellcodes der HTML-Datei befinden.

Hilfestellung:

In css gibt es die Selektoren: border-color, background-color und text-align.

Lösungen:

1)

Der Überschrift <h1> ein anderes Erscheinungsbild geben, als dies standardmäßig vorgesehen ist.

2)

a)

Mit <link ...> wird ganz allgemein auf eine fremde Ressource verwiesen.

Mit dem folgenden Befehl wird eine Datei mit Stylesheet-Angaben eingebunden.

```
<link rel=stylesheet type="text/css" href="myCssStyle.css">
```

b)

```
<style ...>
```

...

```
</style>
```

im Kopfe einer HTML-Datei definiert man einen Bereich für Style Sheet Angaben.

Damit kann man einzelnen Tags, wie z.B. <h1> seine eigene Darstellung (Erscheinung) auf dem Bildschirm aufzwingen.

c)

Man kann einzelne HTML-Elemente innerhalb eines Tags formatieren, wie z.B:

```
<h1 style="font-size:48px;">Das style-Attribut</h1>
```

3)

```
<html>
  <head>
  </head>
  <body>
    <table border=1>
      <tr>
        <td>&nbsp;</td>
        <td>Mathe</td>
        <td>Physik</td>
        <td>EDV</td>
      </tr>
      <tr>
        <td>Note1</td>
        <td>1</td>
        <td>2</td>
        <td>3</td>
      </tr>
      <tr>
        <td>Note2</td>
        <td>3</td>
        <td>4</td>
        <td>5</td>
      </tr>
      <tr>
        <td>Note3</td>
        <td>2</td>
        <td>3</td>
        <td>4</td>
      </tr>
    </table>
  </body>
</html>
```

4)

```
<html>
<head>
  <style type="text/css">
    .classFormat_table{
      border-color:green;
      border-style:solid;
    }
    .classFormat_td{
      border-color:red;
      background-color:yellow;
      text-align:center;
    }
  </style>
</head>
<body>
  <table class="classFormat_table" border=1>
    <tr>
      <td>&nbsp;</td>
      <td class="classFormat_td">Mathe</td>
      <td class="classFormat_td">Physik</td>
      <td class="classFormat_td">EDV</td>
    </tr>
    <tr>
      <td>Notel</td>
      <td>1</td>
      <td>2</td>
      <td>3</td>
    </tr>
    <tr>
      <td>Note2</td>
      <td>3</td>
      <td>4</td>
      <td>5</td>
    </tr>
    <tr>
      <td>Mittel</td>
      <td>2</td>
      <td>3</td>
      <td>4</td>
    </tr>
  </table>
  <br><br><br>
</body>
</html>
```

KLAUSUR 2 SAE E3FI Nachtermin Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

2 Seiten selbstgeschriebene 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 !!!!

Bemerkung:

Das erste Tag in einer HTML-Datei ist hier immer <html>

1) 20P

Erstellen Sie den HTML-Quellcode für das folgende Formular (nur entsprechendes HTML-Formular, kein Servlet) mit dem Button "Date versende" und "Rücksetzen", bei dem die jeweils Noten der Fächer Mathe, Physik und EDV eingegeben werden. Diese Daten werden dann an die folgende Adresse gesendet:

`http://localhost:8080/test1.html`

Für die Tabelle müssen genau die Tags <form> <input> <table>, <tr>, <td> bzw. die entsprechenden schließenden Tags verwendet werden (kein css).

Die Tabelle muss einen Rand haben.

	Mathe	Physik	EDV
Note 1	1	2	3
Note 2	3	4	5
Mittel	2	3	4

2) 30P

Es soll folgende "Tabelle" erzeugt werden:

a11	a12		
a21	a22	a23	
a31	a32	a33	a34

wobei jede Zelle 50 Pixel breit und 20 Pixel hoch ist.

Die Zellen der 1. Zeile werden folgt "formatiert":

Text zentriert, Rand: 2Pixel, Farbe des Randes: blau, Hintergrundfarbe: gelb, Innenabstand: 0
Außenabstand: 0

Die Zellen der 2. Zeile und 3. Zeile werden folgt "formatiert":

Text zentriert, Rand: 2Pixel, Farbe des Randes: rot, Hintergrundfarbe: pink, Innenabstand: 0
Außenabstand: 0

Benutzen Sie für die Erstellung dieser "Tabelle" nur den <div> Tag und u.U. die css-Elemente:

background-color, width, height, padding, border:2px, margin, float, text-align:center, clear.

Lösung:

1)

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

  <body>
    <form action="http://localhost:8080//test1.html" method="post">
      <table border=1>
        <tr>
          <td>&nbsp;</td>
          <td>Mathe</td>
          <td>Physik</td>
          <td>EDV</td>
        </tr>
        <tr>
          <td>Note1</td>
          <td><input type=text name="zahl11" size=10 maxlength=10 value=""></td>
          <td><input type=text name="zahl12" size=10 maxlength=10 value=""></td>
          <td><input type=text name="zahl13" size=10 maxlength=10 value=""></td>
        </tr>
        <tr>
          <td>Note2</td>
          <td><input type=text name="zahl21" size=10 maxlength=10 value=""></td>
          <td><input type=text name="zahl22" size=10 maxlength=10 value=""></td>
          <td><input type=text name="zahl23" size=10 maxlength=10 value=""></td>
        </tr>
        <tr>
          <td>Note2</td>
          <td><input type=text name="zahl31" size=10 maxlength=10 value=""></td>
          <td><input type=text name="zahl32" size=10 maxlength=10 value=""></td>
          <td><input type=text name="zahl33" size=10 maxlength=10 value=""></td>
        </tr>
        <tr>
          <td>Mittel</td>
          <td><input type=text name="mittel1" size=10 maxlength=10 value=""></td>
          <td><input type=text name="mittel2" size=10 maxlength=10 value=""></td>
          <td><input type=text name="mittel3" size=10 maxlength=10 value=""></td>
        </tr>
      </table>
      <input type=submit value="Daten versenden">
      <input type=reset value="Rücksetzen">
    </form>
  </body>
</html>
```

2)

```
<html>
<head>
  <style type="text/css">

    .classFormat_zelle1{
      background-color: yellow;
      width:50px;
      height:20px;
      padding:1px;
      border:2px solid blue;
      margin:0px;
      float:left;
      text-align:center;
    }

    .classFormat_zelle2{
      background-color:pink;
      width:50px;
      height:20px;
      padding:1px;
      border:2px solid red;
      margin:0px;
      float:left;
      text-align:center;
    }

    .classFormat_clear{
      clear:both;
    }
  </style>
</head>

<body>
  <div>
    Hier wird eine Tabelle nur mit css erstellt.
  </div>

  <div class=classFormat_zelle1>
    a11
  </div>

  <div class=classFormat_zelle1>
    a12
  </div>

  <div class=classFormat_clear>
  </div>

  <div class=classFormat_zelle2>
    a21
  </div>

  <div class=classFormat_zelle2>
    a22
  </div>

  <div class=classFormat_zelle2>
    a23
  </div>

  <div class=classFormat_clear>
  </div>

  <div class=classFormat_clear>
  </div>

  <div class=classFormat_zelle2>
    a31
  </div>

  <div class=classFormat_zelle2>
    a32
  </div>

  <div class=classFormat_zelle2>
    a33
  </div>

  <div class=classFormat_zelle2>
    a34
  </div>

  <div class=classFormat_clear>
  </div>
</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 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.

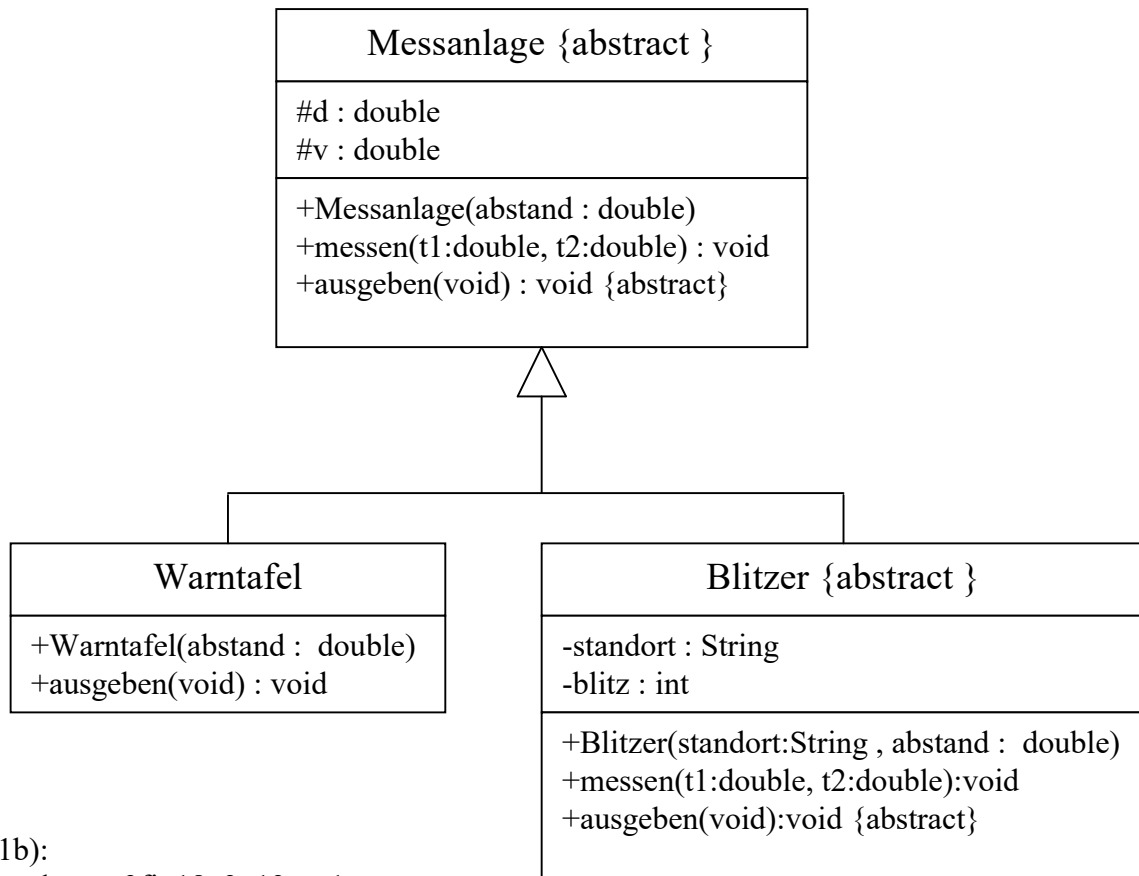
16P

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

34P

Lösungen:

1a)



1b):

```
package e3fi_18_3_13_nr1;
```

```
public class MainE3FI_18_3_13_nr1 {
```

```
    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("*****");
    }
}

```