

*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 Variablenamen 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) 4P  
Was ist der Unterschied zwischen einer Klasse und einem Objekt?

2) 10P  
a)

Welche Werte haben die folgenden Variablen in der Methode main ? (Begründen Sie)

h1 , i , h2 , alter von h2

```
public class Startklasse{
    public static void main(String[] args) {
        Hund h1;                // <---
        int i;                   // <---
        Hund h2=new Hund ();     // <---
    }
}

class Hund{
    private int alter;
    public Hund(int pAlter){
    }
}
```

b)  
Würde eine Fehlermeldung erzeugt werden, wenn der Konstruktor in der Klasse Hund entfernt werden würde? Begründen Sie.

3) (Alle Teilaufgaben müssen in genau **einem** einzigen Programm realisiert werden) 37P  
**Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.**

a) 12P  
Erstellen Sie die Klasse Bruch, (mit genau den 2 Attributen zaehler und nenner, genau 2 set-Methoden, genau 2 get-Methoden und genau einem Konstruktor mit 2 Parametern), die eine mathematische Bruchzahl mit einem ganzzahligen Zähler und einem ganzzahligen Nenner repräsentieren soll.

Zusätzlich sollen genau noch folgende Methoden (und nur diese) erzeugt werden:

b) 3P  
... kuerzen(int teiler)  
kürzt den Bruch durch den wert teiler  
Beispiel: 8 / 40 --- Teiler 4 --> 2 / 10

c) 3P  
... alsKommazahl( ... )  
stellt den Bruch durch eine Kommazahl dar.  
Beispiel: 1/2 ---> 0.5

d) 3P  
... void quadrieren( )  
bildet das Quadrat eines Bruchs.  
Beispiel: 2 / 3 ---> 4 / 9

e) 5P  
... Bruch multiplizieren(Bruch bruch)  
multipliziert 2 Brüche miteinander  
Beispiel: 3 / 4 \* 5 / 6 ---> 15 / 24  
Tipp:  $\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$

f) 3P  
Legen Sie die Brüche b0 (Wert: 7/8) , b1 (Wert: 2/3) und b2 (Wert : 4/5) an.

g) 3P  
Speichern Sie die Kommazahl (mit der entsprechenden Methode) von b0 in der Variablen wert ab.

h) 2P  
Berechnen Sie (mit der entsprechenden Methode) das Quadrat von b0.  
Wo wird das Ergebnis gespeichert?

i) 3P  
Berechnen Sie (mit den entsprechenden Methoden) b1 \* b2 und speichern Sie das Ergebnis in der Variablen b4 (mit dem Datentyp Bruch) ab.

Lösungen:

1) 4P  
Eine Klasse ist ein Bauplan, nachdem ein Objekt gebastelt und im Arbeitsspeicher angelegt wird.

2) 10P  
a) 8P  
h1 , i sind undefiniert (dem Programmierer unbekannt)  
h2 zeigt auf ein Objekt im Arbeitsspeicher, alter von h2 hat den Wert 0

b) 2P  
Nein, da der Konstruktor automatisch vom Compiler angelegt wird.

3)

```
package bruch2;
public class Bruch2 {
    public static void main(String[] args) {
        Bruch b0=new Bruch(7,8); // 1P
        Bruch b1=new Bruch(2,3); // 1P
        Bruch b2=new Bruch(4,5); // 1P
        Bruch b4;
        double wert;

        wert=b0.alsKommazahl(); // 3P
        b0.quadrieren(); // 2P
        b4=b1.multiplizieren(b2); // 3P
    }
}

class Bruch{
    private int zaehler; // 1P
    private int nenner; // 1P

    public Bruch(int zaehler, int nenner) { // 2P
        this.zaehler = zaehler;
        this.nenner = nenner;
    }

    public int getZaehler() { // 2P
        return zaehler;
    }

    public void setZaehler(int zaehler) { // 2P
        this.zaehler = zaehler;
    }

    public int getNenner() { // 2P
        return nenner;
    }

    public void setNenner(int nenner) { // 2P
        this.nenner = nenner;
    }
}
```

```

public void kuerzen(int teiler) { // 3P
    nenner = nenner/teiler;
    zaehler = zaehler/teiler;
}

public double alsKommazahl() { // 3P
    return ((double) zaehler / (double) nenner);
}

public void quadrieren() { // 3P
    zaehler=zaehler*zaehler;
    nenner = nenner * nenner;
}

public Bruch multiplizieren(Bruch bruch) { // 5P
    int z, n;
    z=zaehler*bruch.zaehler;
    n=nenner*bruch.nenner;
    Bruch ergBruch=new Bruch(z,n);
    return ergBruch;
}

public void printBruch() {
    System.out.print(zaehler+"/"+nenner);
}
}

```

## NACHTERMIN

x)

5P

gegeben sei folgender Programmausschnitt (Hund ist eine gegebene Klasse mit den entsprechenden Attributen und Methoden):

```
...  
Hund dopermann;  
dopermann.setGewicht(10);  
dopermann.bellen();  
...
```

- a) Warum gibt es während der Laufzeit eine Fehlermeldung?
- b) Welchen Wert hat dopermann?

X)

Signaturbeispiele machen  
UML erstellen lassen

*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) 5P  
gegeben sei folgender Programmausschnitt (Hund ist eine gegebene Klasse mit den entsprechenden Attributen und Methoden):

```
...
Hund dopermann;
dopermann.setGewicht(10);
dopermann.bellen();
...
```

Gibt es während der Laufzeit eine Fehlermeldung?  
Begründen Sie bitte.

2) 5P  
gegeben sei die Klasse Tiger.  
Diese besitzt u.a. die folgenden Methoden:

```
int fangen(int anzahl, double gewicht){
...
}

double fangen(int anzahl, double gewicht){
...
}
```

Gibt es während des Kompilierens eine Fehlermeldung?  
Begründen Sie bitte.

3) (Alle Teilaufgaben müssen in genau **einem** einzigen Programm realisiert werden) 41P  
**Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.**

Gegeben ist die Klasse Bruch mit genau den 2 Attributen zaehler und nenner, genau 2 set-Methoden, genau 2 get-Methoden und genau einem Konstruktor mit 2 Parametern. Die 2 Attribute repräsentieren eine mathematische Bruchzahl mit einem ganzzahligen Zähler und einem ganzzahligen Nenner.

Zusätzlich sollen genau noch folgende Methoden (und nur diese) erzeugt werden:

a) 10P

... kgV(int a, int b)

Berechnet das kleinste gemeinsame Vielfache von a und b, d.h. die kleinste Zahl, von der a und b Teiler ist.

Beispiel: 3, 8 ---> 24

3, 6 ---> 6

9, 6 ---> 18

b) 10P

... ggT(int a, int b)

Berechnet den größten gemeinsamen Teiler von a und b, d.h. die größte Zahl, die a und b teilt.

Beispiel: 3, 8 ---> 1

3, 6 ---> 3

18, 24 ---> 6

c) 7P

... kuerzen()

kürzt den Bruch so, daß er nicht mehr weiter gekürzt werden kann

Beispiel: 12 / 40 --> 3 / 10

Realisieren Sie dies mit Hilfe der Methode ggT(...) !

d) 7P

... gemeinsamenNennerBilden(Bruch bruch)

erweitert die Brüche so, daß sie einen gemeinsamen Nenner haben.

Realisieren Sie dies mit Hilfe der Methode kgV(...) !

e) 7P

... Bruch addieren (Bruch bruch)

addiert 2 Brüche miteinander

Realisieren Sie dies mit Hilfe der Methode gemeinsamenNennerBilden(Bruch bruch) !

*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) (Alle Teilaufgaben müssen in genau **einem** einzigen Programm realisiert werden)      50P  
**Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.**

a) 5P

Erstellen Sie die Klasse MyMath mit den unten geforderten Methoden.

Braucht diese Klasse Attribute?

Wenn ja, welche? Begründen Sie!

a) 5P

**... istGerade(...)**

Diese muß von einer ganzen Zahl feststellen, ob diese gerade ist.

b) 10P

Die Summe aller ganzen Zahlen zwischen zwei Zahlen soll berechnet werden

**... summeZwischen(...)**

Beispiel: Summe zwischen 3 und 8:  $3 + 4 + 5 + 6 + 7 + 8$

c) 10P

Eine natürliche Zahl größer oder gleich 2 heißt Primzahl, wenn sie nur durch 1 und sich selbst teilbar ist. Beispiele für Primzahlen: 2, 3, 5, 7, 11, 13, ....

Erstellen Sie die Methode

**... istPrimzahl(...)**

Diese muß von einer ganzen Zahl größer oder gleich 2 feststellen, ob diese eine Primzahl ist.

d) 10P

Die Quersumme einer ganzen Zahl ist die Summe seiner Ziffern.

Beispiele:

Die Quersumme von 367 ist:  $3 + 6 + 7 = 16$

Die Quersumme von 1996 ist:  $1 + 9 + 9 + 6 = 25$

Erstellen Sie die Methode

**... quersumme (...)**



e)

10P

Erstellen Sie die Methode

**... anzahl (...)**

Diese muß von einem Feld von ganzen Zahlen feststellen, wie oft dort eine bestimmte ganze Zahl vorkommt.

*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) 9P****1) 3P**

Ein Java-Programmierer bekommt die Aufgabe, ein Feld (kein dynamisches Feld) der Länge 5 zu erzeugen, in dem folgende Daten abgespeichert werden sollen:

2.0	'x'	3	"abcd"	3.14
-----	-----	---	--------	------

Beurteilen Sie diese Aufgabe bzgl. einer Implementierung (Realisierung) in Java.

**2) 3P**

Beurteilen Sie den folgenden Java-Programmausschnitt bzgl. Fehler bei der Syntax bzw. dem Laufzeitverhalten:

```
...
double i=0;
double j=9;
int[] v;
v = new int[10];
j = (j + i)/2;
v[j] = -3;
...
```

**3) 3P**

Beurteilen Sie den folgenden Java-Programmausschnitt bzgl. Fehler bei der Syntax bzw. dem Laufzeitverhalten:

```
...
int[] v;
int[] w;
v = new int[123];
w = new int[11];
v[122] = 11;
w[v[122]] = -4;
...
```

II)

43P

Bemerkung:

In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe-Methoden.

Alle Teilaufgaben müssen in **einem** Programm realisiert werden

1)

16P

a) Erzeugen Sie nur die Klasse Rechteck (und keine andere) mit genau den Attributen (und nur diesen Attributen) "laenge" und "breite", den dazugehörigen get-und set-Methoden und einem Konstruktor (kein Standardkonstruktor).

Außerdem müssen die Methoden "... getUmfang(...)" und "...getFlaeche(...)" implementiert werden, die den Umfang bzw. die Fläche berechnen.

2)

27P

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

a)

2P

Erstellen Sie das Feld "rechtecke" der Länge 100 mit dem Datentyp "Rechteck".

b)

2P

Erstellen Sie in dem Feld "rechtecke" in den ersten 50 Zellen Rechtecke mit gleichen Längen und Breiten (also Quadrate) von 0, 1, 2, ...49 Länge.

c)

5P

Verlängern Sie (nur mit Hilfe der get-bzw. set-Methoden) die Längen der obigen Rechtecke um den Wert +10 (man erzeugt also "echte" Rechtecke)

Bemerkung:

Die neue Länge darf nicht selbst mit einem Taschenrechner (oder im Kopf) berechnet werden, also, sondern dies muß programmtechnisch (mit den entsprechenden Methoden) realisiert werden.

d)

6P

Durch weitere Anweisungen wird eine weitere (Ihnen unbekannte Zahl) an Rechtecken in das feld "rechtecke" eingefügt.

Geben Sie die Umfänge aller Rechtecke - und deren Index (Stelle) im Feld - auf dem Bildschirm aus.

e)

6P

Kopieren Sie die ersten 50 Rechtecke hintereinander in das restliche Feld (also in die Zellen 50 bis 99)

f)

6P

Kopieren Sie die Längen aller Rechtecke in das zu erstellende double feld laengen  
Kopieren Sie die Breiten aller Rechtecke in das zu erstellende double feld breiten

Lösungen:

I)

1)

3P

Syntaxfehler:

In einem Feld dürfen nur Werte des gleichen Datentyps abgespeichert werden.

2)

3P

Beurteilen Sie den folgenden Java-Programmausschnitt bzgl. Syntx und Laufzeitverhalten:

Der Wert von j darf keine Fleißkommazahl sein.

3)

3P

v[122] hat den Wert 11.

Deshalb ist w[11] außerhalb ein Zugriff außerhalb der Feldgrenzen.

II)

16P

```
class Rechteck {  
    private double laenge;           // 1P  
    private double breite;           // 1P  
  
    public Rechteck(double plaenge, double pbreite){ // 2P  
        setLaenge(plaenge);  
        setBreite(pbreite);  
    }  
  
    public void setLaenge(double plaenge){           // 2P  
        laenge = plaenge;  
    }  
  
    public void setBreite(double pbreite){           // 2P  
        breite = pbreite;  
    }  
  
    public double getLaenge(){                       // 2P  
        return (laenge);  
    }  
  
    public double getBreite(){                       // 2P  
        return (breite);  
    }  
  
    public double getFlaeche(){                      // 2P  
        return(laenge*breite);  
    }  
  
    public double getUmfang(){                      // 2P  
        return(2*(laenge+breite));  
    }  
}
```

```

public static void main(String[] args) {
    int i;
    // a) // 2P
    Rechteck[] rechtecke;
    rechtecke=new Rechteck[100];
    // b) // 2P
    for(i=0;i<50;i++){
        rechtecke[i]=new Rechteck(i,i);
    }
    // c) // 5P
    for(i=0;i<50;i++){
        double tempLaenge;
        tempLaenge=rechtecke[i].getLaenge();
        rechtecke[i].setLaenge(tempLaenge+10);
        //oder alternativ;
        //rechtecke[i].setLaenge(rechtecke[i].getLaenge()+10);
    }
    // d) // 6P
    for(i=0;i<100;i++){
        double umfang;
        if(rechtecke[i]!=null){
            umfang=rechtecke[i].getUmfang();
            System.out.println("Umfang Rechteck["+i+"]="+umfang);
        }
    }
    // e) // 6P
    for(i=0;i<50;i++){
        rechtecke[i+50]=rechtecke[i];
    }
    // f) //6P
    double[] laengen=new double[100];
    double[] breiten=new double[100];
    for(i=0;i<100;i++){
        laengen[i]=rechtecke[i].getLaenge();
        breiten[i]=rechtecke[i].getBreite();
    }
}

```

Name, Vorname:

Hilfsmittel:  
keine

Hinweise (unbedingt beachten):

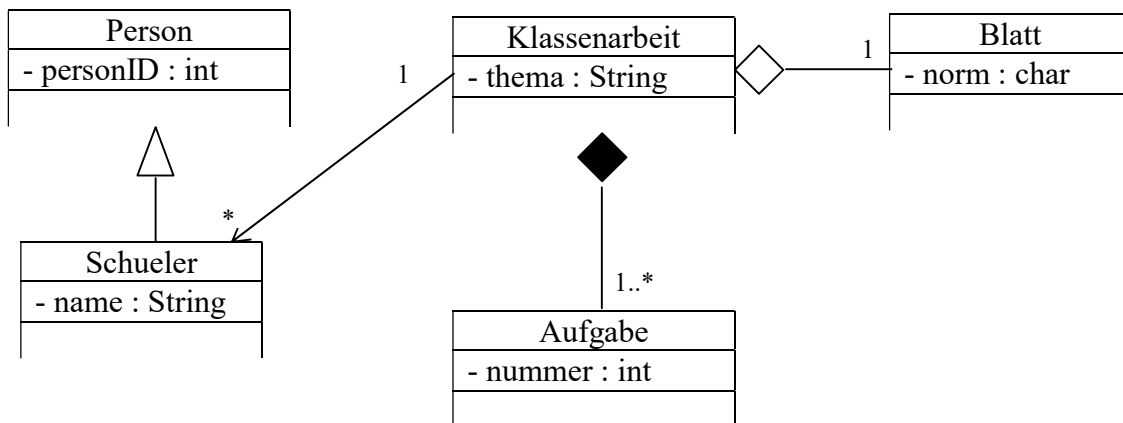
- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote 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

I)

50P

Gegeben ist das folgende unvollständige UML-Diagramm (nur die Attribute der Klasse Klassenarbeit sind unvollständig). In allen Klassen fehlen die entsprechenden Methoden.



Bemerkung:

In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe-Methoden.

Alle Teilaufgaben müssen in **einem** Programm realisiert werden

- 1) 4P  
Was bedeuten die 2 verschiedenen Pfeile (offene bzw. geschlossene Pfeilspitze) und die Rauten (weiß und schwarz). Jeweils nur ein Stichwort angeben.
- 2) 7P  
Erzeugen Sie die Klasse Blatt mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).
- 3) 7P  
Erzeugen Sie die Klasse "Person" mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).
- 4) 9P  
Erzeugen Sie die Klasse "Schueler" mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau zwei Parametern).
- 5) 7P  
Erzeugen Sie die Klasse "Aufgabe" mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).
- 6) 4P  
a) Erzeugen Sie die Klasse "Klassenarbeit" mit allen nötigen (genau) 4 Attributen.
- b) 4P  
Erzeugen Sie genau einen Konstruktor (mit genau zwei Parametern). Falls ein Array benötigt wird, soll dieses nicht dynamisch sein und im Konstruktor das Feld die Länge 10 erhalten).
- c) 4P  
Erzeugen Sie die folgende Methode:  
... insertAufgabe( ... ) :  
fügt eine Aufgabe an eine bestimmte Stelle des Feldes ein.  
Beachten Sie dazu, daß eine Komposition und eine Aggregation anders implementiert werden.
- d) 4P  
Erzeugen Sie die folgende Methode:  
... insertBlatt ( ... ) :  
fügt ein Blatt ein.  
Beachten Sie dazu, daß eine Komposition und eine Aggregation anders implementiert werden.

## Lösungen:

1) 4P  
geschlossene Pfeilspitze: Vererbung, geöffnete Pfeilspitze: Assoziation,  
schwarze Raute: Komposition, weiße Raute: Aggregation

2) 7P

```
class Blatt {
    private char norm;

    public Blatt(char pNorm) {
        norm = pNorm;
    }

    public char getNorm() {
        return norm;
    }

    public void setNorm(char norm) {
        this.norm = norm;
    }
}
```

3) 7P

```
class Person{
    private int personID;

    public Person(int personID) {
        this.personID = personID;
    }

    public int getPersonID() {
        return personID;
    }

    public void setPersonID(int personID) {
        this.personID = personID;
    }
}
```

4) 9P

```
class Schueler extends Person {
    private String name;

    public Schueler(int personID, String pName){
        super(personID);
        name=pName;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

5) 7P

```
class Aufgabe{
    private int nummer;

    public int getNummer() {
        return nummer;
    }

    public void setNummer(int nummer) {
        this.nummer = nummer;
    }

    public Aufgabe(int pNummer){
        nummer=pNummer;
    }
}
```



6)

```
class Klassenarbeit{
    private String thema;
    private Aufgabe[] dieAufgaben;
    private Schueler[] dieSchueler;
    private Blatt blatt;

    public Klassenarbeit(String pThema, Blatt pBlatt){
        thema=pThema;
        blatt=pBlatt;
        dieAufgaben=new Aufgabe[10];
        dieSchueler=new Schueler[10];
    }

    public void insertAufgabe(int pNummer, int index){
        dieAufgaben[index]=new Aufgabe(pNummer);
    }

    public void insertSchuler(Schueler pSchueler, int index){
        dieSchueler[index]=pSchueler;
    }

    public void insertBlatt(Blatt pBlatt ){
        blatt=pBlatt;
    }
    ...
}
```

*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

Die Verwaltung eines Bauernhofs soll mit Hilfe der OOP auf EDV umgestellt werden.  
In einem Gespräch zwischen der EDV-Firma „Makall“ und den Besitzern des Bauernhofs wurde folgendes festgehalten:

Der Bauernhof gehört genau 2 Besitzern d.h. Personen und besteht aus genau einem Bauernhaus und einem Kuhstall, in dem maximal 10 Kühe untergebracht werden können..  
Die Klasse Kuh wurde schon implementiert (mit den Attributen name und alter und den entsprechenden Methoden).

Aus Gründen der Vereinfachung besitzen die Klassen "Besitzer", "Person" und "Bauernhaus" jeweils genau ein Attribut

1) 16P  
Erstellen Sie ein UML-Diagramm (mit den entsprechenden Attributen, aber ohne Methoden), das diesen Fall modelliert.

2) 7P  
Implementieren Sie die Klasse "Person" mit den nötigen Attributen und Methoden.

3) 9P  
Implementieren Sie die Klasse "Besitzer" mit den nötigen Attributen und Methoden.

4) 4P  
a)  
Erzeugen Sie die Klasse "Bauernhof" mit allen nötigen Attributen.

b) 5P  
Erzeugen Sie genau einen Konstruktor (mit genau 4 Parametern). Falls ein Array benötigt wird, soll dieses nicht dynamisch sein und die Länge 10 haben.

c)

10P

Erstellen Sie in der Klasse Bauernhof eine Methode

... anfüegenKuh (Kuh pKuh) :

fügt eine Kuh an das Ende der bisherigen gespeicherten Kühe an.

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

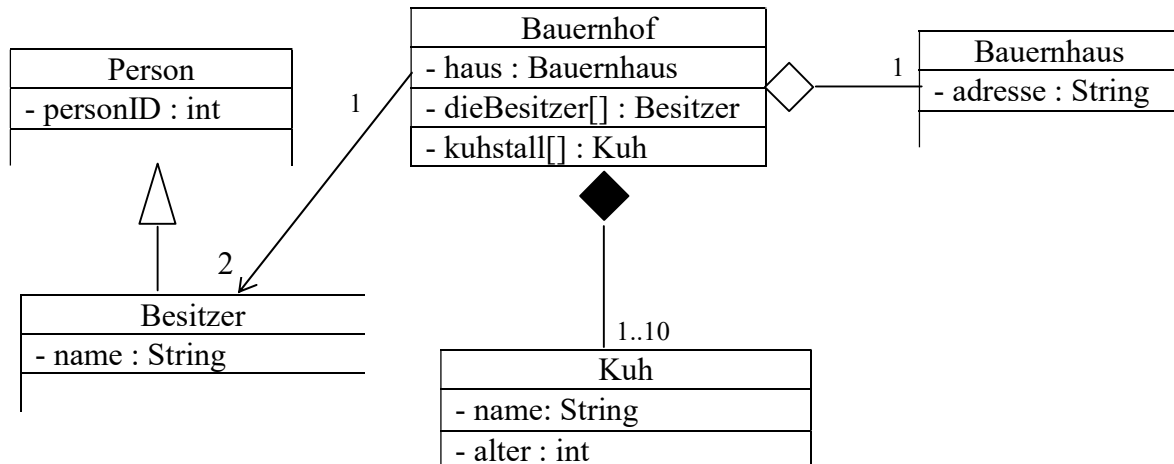
Bemerkung:

In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe-Methoden.

Alle Teilaufgaben - sofern es sich um Programmieraufgaben handelt - müssen in **einem** Programm realisiert werden

Lösung:

1)



Bemerkung:

In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe-Methoden.

Alle Teilaufgaben müssen in **einem** Programm realisiert werden

2)

7P

```
class Person{
    private int personID;

    public Person(int personID) {
        this.personID = personID;
    }

    public int getPersonID() {
        return personID;
    }

    public void setPersonID(int personID) {
        this.personID = personID;
    }
}
```

3)

9P

```
class Besitzer extends Person {
    private String name;

    public Besitzer(int personID, String pName){
        super(personID);
        name=pName;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
```

```
        this.name = name;
    }
}
```

4)

```
class Bauernhof{
    private String adresse;
    private Kuh[] kuhstall;
    private Besitzer[] dieBesitzer;
    private Bauernhaus haus;

    public Bauernhof(String adresse, Besitzer b1, Besitzer b2,
        Bauernhaus haus){
        this.adresse = adresse;
        dieBesitzer=new Besitzer[2];
        dieBesitzer[0]=b1;
        dieBesitzer[1]=b2;
        this.haus=haus;
        kuhstall = new Kuh[10];
    }

    public void insertKuh(Kuh k){
        for(int i=0;i<10;i++){
            if(kuhstall[i]==null){
                kuhstall[i]=k;
                return;
            }
        }
    }

    public void printKuhstall(){
        for(int i=0;i<10;i++){
            if(kuhstall[i]!=null){
                sout ("Kuhname="+kuhstall[i].getName());
                sout ("Kuhgewicht="+kuhstall[i].getGewicht());
            }
        }
    }
}
```