

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 !!!!

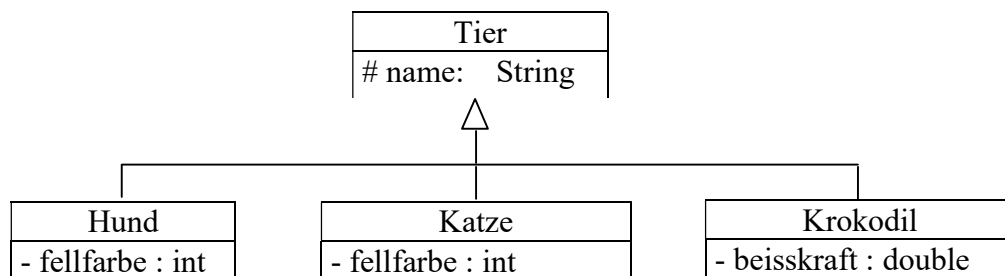
AUFGABEN

ACHTUNG: Die Klassenarbeit besteht aus genau der Aufgabe I (die aus Teilaufgaben besteht).

Die gesamte Aufgabe muß als genau ein Programm in einem Projekt implementiert werden.

I)

Gegeben ist das folgende UML-Diagramm, in dem nur die Klassennamen und alle Attribute vorkommen (und die nicht durch andere ergänzt werden dürfen). Die zugehörigen Methoden werden hier nicht dargestellt.



1)

31P

Implementieren Sie (unter Zuhilfenahme des obigen folgenden UML-Diagramms) alle im UML-Diagramm vorkommenden Klassen außer der Klasse Katze (von der man annehmen darf, dass sie schon implementiert wurde) mit den zugehörigen set- und get-Methoden und den Konstruktoren (bei Klasse Tier mit genau einem Parameter, bei allen anderen mit genau 2 Parametern).

Aus dem obigen UML-Diagrammen soll nicht hervorgehen, ob es sich um abstrakte oder "normale" Klassen handelt.

Wichtig: Jede Farbe wird durch einen Integer-Wert dargestellt, wie z.B:

-100: weiß 1:gelb, usw.

Umgekehrt entspricht jedem Integer-Wert eine Farbe!

Bitte keine "Umrechnungstabelle" der Zahlenwerte in tatsächliche Farben erstellen!

Hier ist eine Farbe ein Zahlenwert, mehr nicht!

2)

8P

Das Programm soll multipersonal entwickelt werden: Zu Testzwecken sollen die Entwickler (also diejenigen, die heute diese Klassenarbeit schreiben!) der einzelnen Klassen gezwungen werden, die Methode

`String getBeschreibung()`

zu entwickeln, die die Namen der Attribute mit den zugehörigen Werten auf dem Bildschirm ausgibt.

a) Implementieren Sie dazu den nötigen Quellcode in der bzw. den entsprechenden Klassen. In einem Feld sollen verschiedene Tiere (Hunde, Katzen, Krokodile) abgespeichert werden. Danach sollen die Werte der Attribute mit der Methode `getBeschreibung()` ausgegeben werden. Machen Sie folgendes in der Methode `main()`

b) Erzeugen Sie einen Hund, eine Katze und ein Krokodil.

c) Speichern Sie diese 3 Tiere in dem Feld.

d) Geben Sie mit Hilfe einer Schleife und der Methode `getBeschreibung()` die Eigenschaften der jeweiligen Tiere auf dem Bildschirm aus.

3)

12P

Beachten Sie:

Ein Krokodil hat - nach heutigem biologischen Wissenstand - kein Fell.

Dagegen besitzt eine Katze bzw. ein Hund ein Fell.

In dem Feld "felltier" sollen die im vorigen Programmteil erstellte Katze und Hund abgespeichert werden und dann mit Hilfe einer Schleife und der Methode `getFellfarbe()` die Farbe des jeweiligen Tiers auf dem Bildschirm ausgegeben werden.

a) Was wäre der Nachteil der Lösung, in der man die Methode `getFellfarbe()` in der Klasse `Tier` implementiert?

b) Beurteilen Sie die folgende Lösung:

Die Methode `getFellfarbe()` wird nur (und sonst nirgends) in die Klasse `Hund` und `Katze` implementiert.

c) Was ist die beste Lösung ? (keine Implementierung, nur verbale Beschreibung und Begründung).

In dem UML dürfen nur Ergänzungen, aber keine Löschungen vorgenommen werden.

Lösung:

```
public class MainKlassenarbeit {
    public static void main(String[] args) {
        int i;
        Tier tiere[];
        tiere = new Tier[3];
        tiere[0] = new Katze("Ute", 2);
        tiere[1] = new Hund("Rex", 3);
        tiere[2] = new Krokodil("Krok", 30);
        System.out.println("Beschreibung der Tiere:");
        for(i=0; i<tiere.length; i++){
            System.out.println(tiere[i].getBeschreibung());
        }
    }
}

abstract class Tier{
    protected String name;

    public Tier(String pName){
        name = pName;

    }

    public void setName(String pName){
        name=pName;
    }

    public String getName(){
        return(name);
    }

    abstract public String getBeschreibung();
}

class Hund extends Tier {
    private int fellfarbe;

    public Hund(String pName, int pFellfarbe){
        super(pName);
        fellfarbe = pFellfarbe;
    }

    public String getBeschreibung(){
        String s;
        s="Name="+name+" Fellfarbe="+fellfarbe;
        return s;
    }

    public void setFellfarbe(int pFellfarbe){
        fellfarbe = pFellfarbe;
    }

    public int getFellfarbe(){
        return fellfarbe;
    }
}
```

8P

1P

1P

1P

1P

1P

3P

1P

11P

1P

2P

2P

2P

3P

10P

```

class Krokodil extends Tier{
    private double beisskraft;

    Krokodil(String pName, double pBeisskreaft){
        super(pName);
        beisskraft = pBeisskreaft;
    }

    public String getBeschreibung(){
        String s;
        s="Name="+name+" Beisskraft="+beisskraft;
        return s;
    }

    public void setBeisskraft(double pBeisskraft){
        beisskraft = pBeisskraft;
    }

    public double getBeisskraft(){
        return beisskraft;
    }
}

```

c1) 4P
 Dadurch erbt die Klasse Krokodil u.a. die Methode getFellfarbe().
 Dadurch kann man die Fellfarbe eines Krokodils herausfinden, obwohl ein Krokodil keine Fellfarbe besitzt.

c2) 4P
 Die Methode getFellfarbe() werden nur (und sonst nirgends) in die Klasse Hund und Katze implementiert.
 Da in einer Schleife z.B. mit felltiere. getFellfarbe() die Farbe abgefragt wird, muss die Methode auch in der Klasse Tier existieren. Da dies nicht der Fall ist, meldet der Compiler einen Fehler.

c3) 4P
 Katze und Hund implementieren das Interface FelltierIF, wo die Methodenköpfe getFellfarbe() und setFellfarbe(...) angegeben werden.
 Nur in den Klassen Hund und Katze müssen diese Methoden dann ausprogrammiert werden.

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 Punkteabzug !!!!

AUFGABEN

I)

Bemerkung:

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

Die (verkäuflichen) Waren (z.B. Wein, Brot, usw.) und die (nicht verkäufliche) Betriebsausstattung (z.B. Stühle, Tische, Geschäftsautos, usw.) bilden zusammen das Betriebsvermögen.

1) 9P

Zeichnen Sie ein UML-Diagramm, in dem die Klassen "Wein", "Brot", "Stuhl", "Tisch", "Betriebsvermögen" mit ihren Beziehungen untereinander vorkommen.

Von den Klassen muß im UML-Diagramm jeweils nur der Name dargestellt werden (ohne Attribute und Methoden).

2) 2P

Die Klasse Brot besitzt genau das Attribut Gewicht.

Der Preis des Brots berechnet sich aus dem doppelten seines Gewichts.

Die Klasse Wein besitzt genau die 2 Attribute Volumen und Alter.

Der Preis des Weins berechnet sich aus der Summe seines Volumens und seines Alters.

Was muß programmtechnisch gemacht werden, damit man genau die Entwickler der Klassen Wein und Brot zwingen kann, unbedingt die Methoden "... getPreis(...)" zu implementieren?
Verbale Beschreibung und Ergänzung des UML-Diagramms oben.

Die folgenden Aufgaben müssen alle in einem Programm erstellt werden

3) 8P

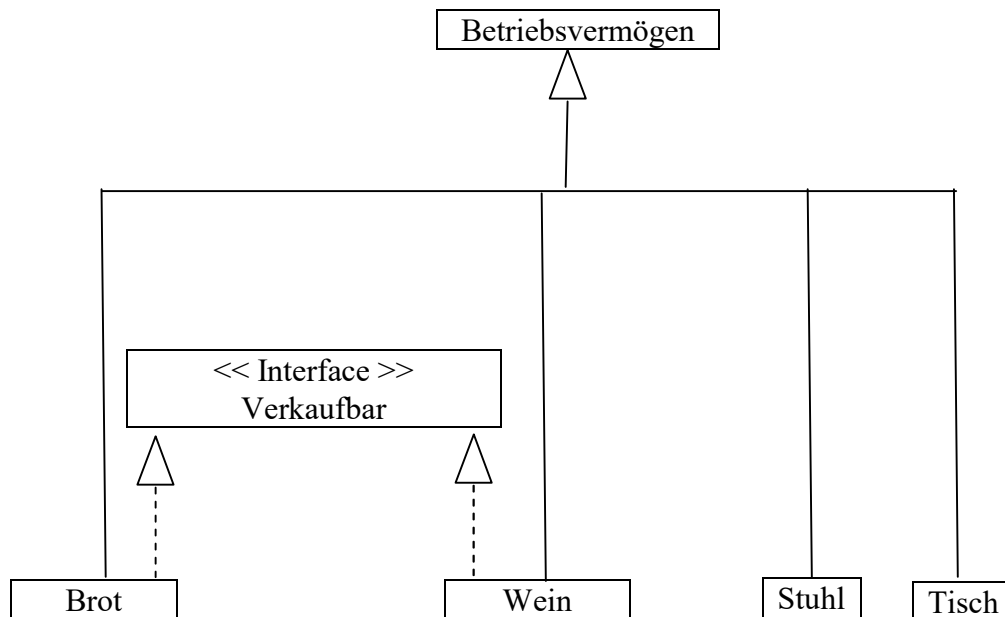
Erzeugen Sie die Klasse "Betriebsvermögen" mit genau dem Attribut "name" (und nur diesem Attribut), den dazugehörigen get- und set-Methoden und einem Konstruktor (mit genau einem Parameter)

- 4) 7P
Erzeugen Sie die Klasse "Brot" mit genau dem Attribut "gewicht" (und nur diesem Attribut), einem Konstruktor (mit genau zwei Parametern) und die Methode "...getPreis(...)", aber ohne die dazugehörigen get-und set-Methoden.
- 5) 7P
Erzeugen Sie die Klasse "Wein" mit genau den Attributen "volumen" und "alter" (und nur diesen Attributen), genau einem Konstruktor (mit genau drei Parametern) und die Methode "...getPreis(...)", aber ohne die dazugehörigen get-und set-Methoden.
- 6) 3P
Erzeugen Sie das Interface "Verkaufbar"
- 7) 2P
Erzeugen Sie in der Klasse "Betriebsvermögen" die statische Methode ...verkaufspreis(...).
- 8) 14P
Machen Sie folgendes in der Methode main()
a) Erstellen Sie ein Brot (mit Variable b bezeichnen) mit Gewicht 10
b) Erstellen Sie einen Wein (mit Variable w bezeichnen) mit Volumen 3 und Alter 4.
c) Speichern Sie diese 2 Objekte in dem Feld "waren" der Länge 2.
d) Mit einer Schleife muß der Preis dieser 2 Waren auf dem Bildschirm ausgegeben werden.
e) Alternativ soll zur Sicherheit zusätzlich noch der Preis des Brotes b mit Hilfe der statischen Methode ...verkaufspreis(...) auf dem Bildschirm ausgegeben werden.

Lösung:

1)

9P



2)

2P

Interface "Verkaufbar" erzeugen und dort den Methodenkopf `getPreis()` angeben.

3-8)

// 14P

```
public class Startklasse {
    public static void main(String[] args) {
        int i;
        Brot b1=new Brot("Dinkel",10);           //2P
        Wein w1=new Wein("Hirnhammer",3,4);       //2P
        Verkaufbar waren[]=new Verkaufbar[2];     //2P
        waren[0]=b1;                             //2P
        waren[1]=w1;                             //2P
        for(i=0;i<2;i++){                       //2P
            System.out.println(waren[i].getPreis());
        }
        System.out.println("Preis="+Betriebsvermögen.verkaufspreis(b1) ); //2P
    }
}
```

// 7P

```
class Brot extends Betriebsvermögen implements Verkaufbar{
    private double gewicht;

    public Brot(String name, double gewicht){
        super(name);
        this.gewicht=gewicht;
    }

    public double getPreis(){
        return 2*gewicht;
    }
}
```

```

// 7P
class Wein extends Betriebsvermögen implements Verkaufbar{
    private double alter;
    private double volumen;

    public Wein(String name, double alter, double gewicht){
        super(name);
        this.alter=alter;
        this.volumen=volumen;
    }

    public double getPreis(){
        return volumen+alter;
    }
}

// 10P
abstract class Betriebsvermögen{
    private String name;

    public Betriebsvermögen(String name){
        this.name=name;
    }

    public static double verkaufspreis(Verkaufbar v){
        return v.getPreis();
    }

    public String getName() {
        return name;
    }

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

// 3P
public interface Verkaufbar {
    double getPreis();
}

```


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)

Bemerkung:

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

Die (verkäuflichen) Waren (z.B. Wein, Brot, usw.) und die (nicht verkäufliche) Betriebsausstattung (z.B. Stühle, Tische, Geschäftsautos, usw.) bilden zusammen das Betriebsvermögen.

1) 9P

Zeichnen Sie ein UML-Diagramm, in dem die Klassen "Wein", "Brot", "Stuhl", "Tisch", "Betriebsvermögen" mit ihren Beziehungen untereinander vorkommen.

Von den Klassen muß im UML-Diagramm jeweils nur der Name dargestellt werden (ohne Attribute und Methoden).

2) 2P

Die Klasse Brot besitzt genau das Attribut Gewicht.

Der Preis des Brots berechnet sich aus dem doppelten seines Gewichts.

Die Klasse Wein besitzt genau die 2 Attribute Volumen und Alter.

Der Preis des Weins berechnet sich aus der Summe seines Volumens und seines Alters.

Was muß programmtechnisch gemacht werden, damit man genau die Entwickler der Klassen Wein und Brot zwingen kann, unbedingt die Methoden "... getPreis(...)" zu implementieren?
Verbale Beschreibung und Ergänzung des UML-Diagramms oben.

Die folgenden Aufgaben müssen alle in einem Programm erstellt werden

3) 8P

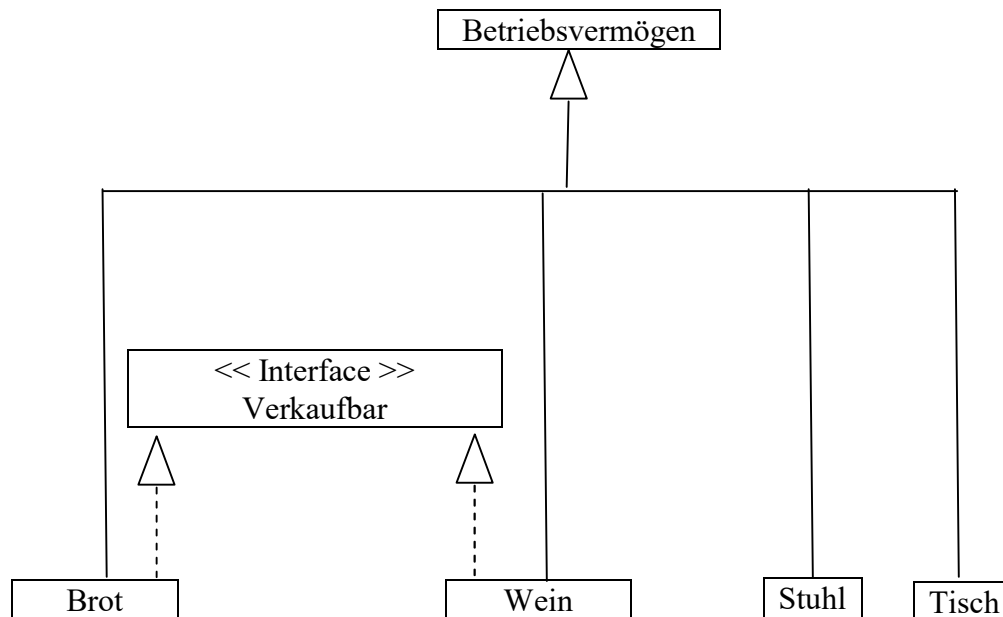
Erzeugen Sie die Klasse "Betriebsvermögen" mit genau dem Attribut "name" (und nur diesem Attribut), den dazugehörigen get- und set-Methoden und einem Konstruktor (mit genau einem Parameter)

- 4) 7P
Erzeugen Sie die Klasse "Brot" mit genau dem Attribut "gewicht" (und nur diesem Attribut), einem Konstruktor (mit genau zwei Parametern) und die Methode "...getPreis(...)", aber ohne die dazugehörigen get-und set-Methoden.
- 5) 7P
Erzeugen Sie die Klasse "Wein" mit genau den Attributen "volumen" und "alter" (und nur diesen Attributen), genau einem Konstruktor (mit genau drei Parametern) und die Methode "...getPreis(...)", aber ohne die dazugehörigen get-und set-Methoden.
- 6) 3P
Erzeugen Sie das Interface "Verkaufbar"
- 7) 2P
Erzeugen Sie in der Klasse "Betriebsvermögen" die statische Methode ...verkaufspreis(...).
- 8) 14P
Machen Sie folgendes in der Methode main()
a) Erstellen Sie ein Brot (mit Variable b bezeichnen) mit Gewicht 10
b) Erstellen Sie einen Wein (mit Variable w bezeichnen) mit Volumen 3 und Alter 4.
c) Speichern Sie diese 2 Objekte in dem Feld "waren" der Länge 2.
d) Mit einer Schleife muß der Preis dieser 2 Waren auf dem Bildschirm ausgegeben werden.
e) Alternativ soll zur Sicherheit zusätzlich noch der Preis des Brotes b mit Hilfe der statischen Methode ...verkaufspreis(...) auf dem Bildschirm ausgegeben werden.

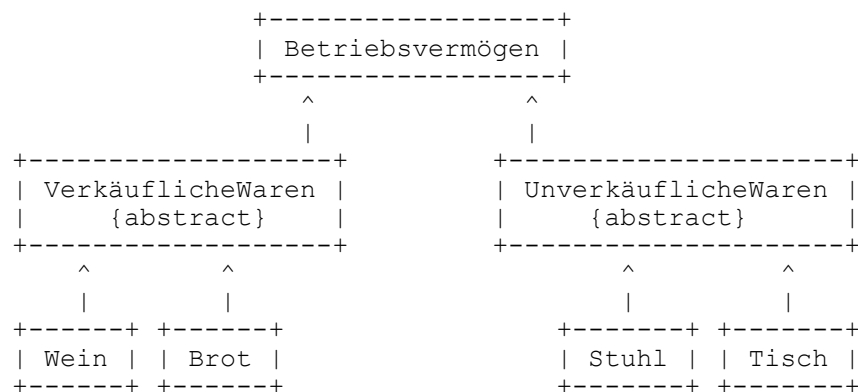
Lösung:

1)

9P



Alternnative Lösung:



Da die Klassen "VerkäuflicheWaren" und "UnverkäuflicheWaren" abstrakt sind und nur abstrakte Methoden (nämlich genau die eine Methode `getPreis()` enthält) hat sie im Prinzip genau die gleiche Funktion wie ein Interface, das auch nur abstrakte Methoden enthält. Also könnte man auch gleich ein Interface nehmen wie in der ersten Lösung!

2)

2P

Interface "Verkaufbar" erzeugen und dort den Methodenkopf `getPreis()` angeben.

3-8)

// 14P

```
public class Startklasse {
    public static void main(String[] args) {
        int i;
        Brot b1=new Brot("Dinkel",10); //2P
        Wein w1=new Wein("Hirnhammer",3,4); //2P
        Verkaufbar waren[]=new Verkaufbar[2]; //2P
        waren[0]=b1; //2P
        waren[1]=w1; //2P
        for(i=0;i<2;i++){ //2P
            System.out.println(waren[i].getPreis());
        }
        System.out.println("Preis="+Betriebsvermögen.verkaufspreis(b1) ); //2P
    }
}
```

```
// 7P
class Brot extends Betriebsvermögen implements Verkaufbar{
    private double gewicht;

    public Brot(String name, double gewicht){
        super(name);
        this.gewicht=gewicht;
    }

    public double getPreis(){
        return 2*gewicht;
    }
}
```

```
// 7P
class Wein extends Betriebsvermögen implements Verkaufbar{
    private double alter;
    private double volumen;

    public Wein(String name, double alter, double volumen){
        super(name);
        this.alter=alter;
        this.volumen=volumen;
    }

    public double getPreis(){
        return volumen+alter;
    }
}
```

```
// 10P
abstract class Betriebsvermögen{
    private String name;

    public Betriebsvermögen(String name){
        this.name=name;
    }

    public static double verkaufspreis(Verkaufbar v){
        return v.getPreis();
    }

    public String getName() {
        return name;
    }

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

```
// 3P
public interface Verkaufbar {
    double getPreis();
}
```

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) 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"

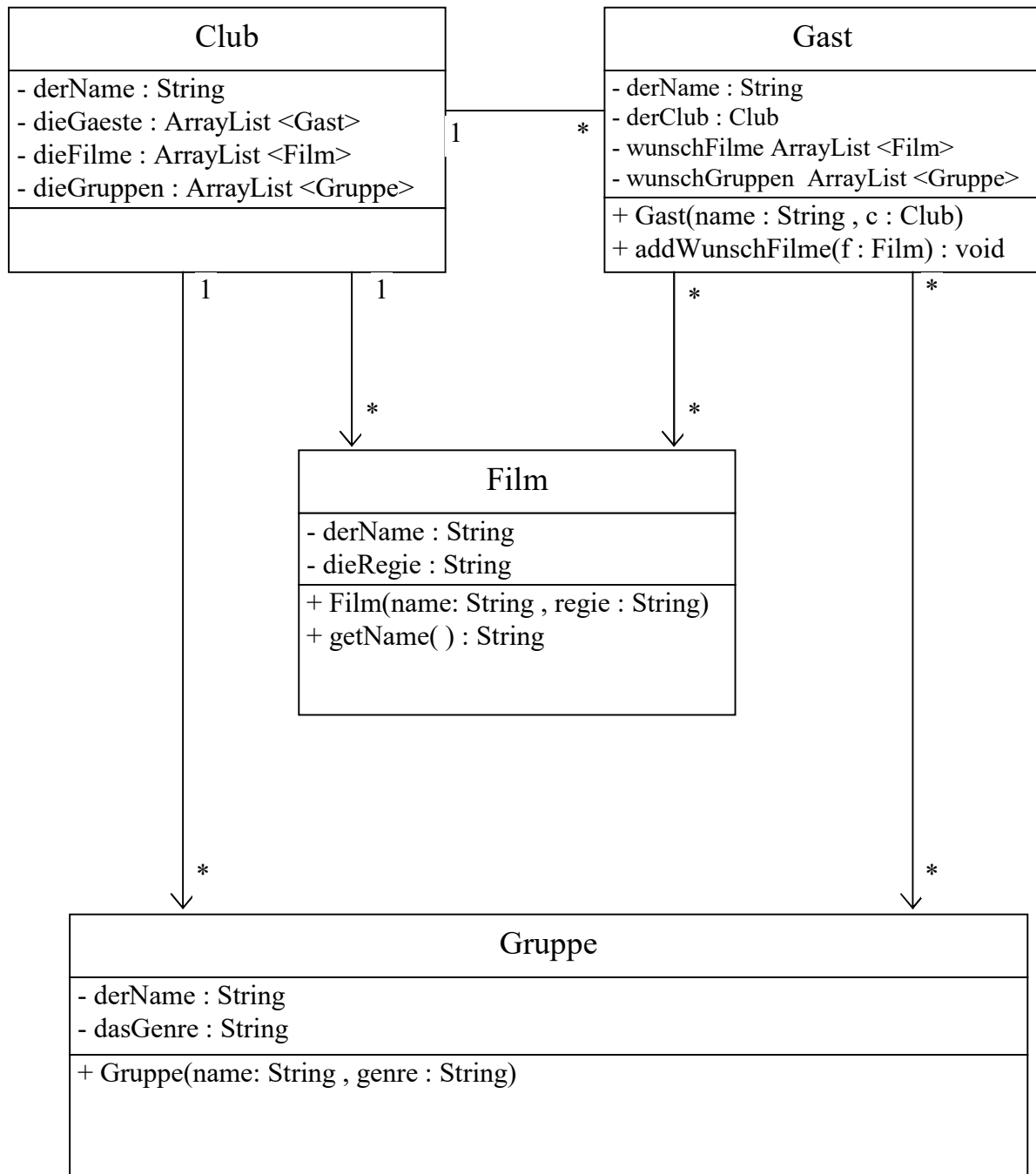
"Die Geschichte des V3" von Regissuer "Vau Drei"

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");	