

Name, Vorname:

Hilfsmittel:

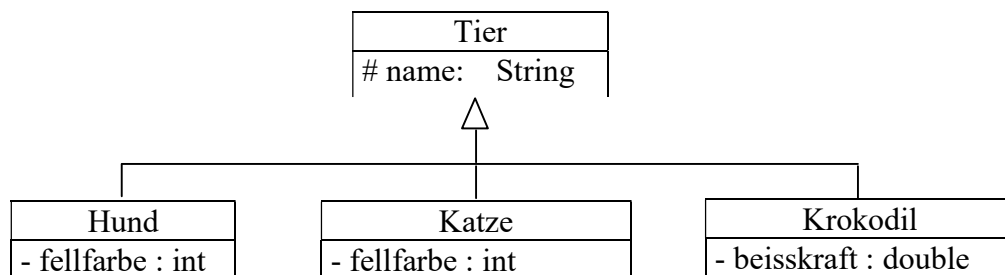
keine

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) Gegeben ist das folgende abgespeckte, **fest vorgegebene** UML-Diagramm.
Die Klassenhierarchie darf also **nicht** verändert werden.



a) 39P

Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen mit den jeweiligen Konstruktoren, set- und get-Methoden. Keine weiteren Attribute einfügen!
Die Konstruktoren müssen jeweils Parameter enthalten.

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

... -100: weiß -99:gelb,

Umgekehrt entspricht jedem Integer-Wert eine Farbe!

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

Hier ist eine Farbe ein Zahlenwert, mehr nicht!

b) 11P

Das Programm soll multipersonal entwickelt werden: Zu Testzwecken sollen die Entwickler der einzelnen Klassen gezwungen werden, die Methode

String getBeschreibung()

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

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()

b1) Erzeugen Sie einen Hund, eine Katze und ein Krokodil.

b2) Speichern Sie diese 3 Tiere in dem Feld.

b3) Geben Sie mit Hilfe einer Schleife und der Methode getBeschreibung() die Eigenschaften der jeweiligen Tiere auf dem Bildschirm aus.

Lösung:

```
public class MainKlassenarbeit {                                11P
    public static void main(String[] args) {
        int i;
        Katze katze;
        Hund hund;
        Krokodil krokodil;
        Tier tiere[];                                           1P
        tiere = new Tier[3];                                     1P
        katze = new Katze("Ute", 2);                             1P
        tiere[0] = katze;                                        1P
        hund = new Hund("Rex", 3);                               1P
        tiere[1] = hund;                                         1P
        krokodil = new Krokodil("Krok", 30);                    1P
        tiere[2] = krokodil;                                     1P
        System.out.println("Beschreibung der Tiere:");
        for(i=0;i< tiere.length; i++){
            3P
            System.out.println(tiere[i].getBeschreibung());
        }
    }
}

abstract class Tier{                                           1P
    protected String name;                                     1P

    public Tier(String pName){                                  2P
        name = pName;
    }

    public void setName(String pName){                          2P
        name=pName;
    }

    public String getName(){                                    2P
        return(name);
    }

    abstract public String getBeschreibung();                  3P
}

class Katze extends Tier {                                     10P
    private int fellfarbe;

    public Katze(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;
    }
}

```

```

class Hund extends Tier {
    private int fellfarbe;

```

10P

```

    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;
    }
}

```

```

class Krokodil extends Tier{
    private double beisskraft;

```

10P

```

    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;
    }
}

```

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

Bemerkungen:

B1) Ein Java-Programm muss nach dem Prinzip der OOP gestaltet werden.

Insbeondere müssen verschiedene Konzepte wie Klassen, Vererbung, Polymorphie, usw. verwendet werden.

Es wird auch bewertet, wie gut dieses Prinzip der OOP umgesetzt wurde.

B2) 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) 53P

Auf einem Fest wird eine Tombola (Lose ziehen) veranstaltet.

Ein Tombolagewinn ist eine Ware, kann hier also ein Auto oder eine Immobilie sein.

Außer der Startklasse müssen genau die folgenden 3 Klassen erzeugt werden:

(d.h. es dürfen keine weiteren Klassen erzeugt werden)

Auto, Immobilie und TombolaGewinn.

1) 25P

a)

Erzeugen Sie die Klasse Auto mit genau den 2 Attributen (und nur diesen Attributen) "preis" und "alter" und den zu diesen Attributen gehörigen get-und set-Methoden.

Erzeugen Sie genau einen Konstruktor mit genau 2 Parametern.

b)

Erzeugen Sie die Methode ... berechneWert(...), die den Wert eines Autos berechnet (der Wert berechnet sich aus dem Preis des Autos dividiert durch das Alter).

c)

Erzeugen Sie die Klasse Immobilie mit genau dem einen Attribut (und nur diesem Attribut) "miete" und die zu diesem Attribut zugehörige get-und set-Methode.

Erzeugen Sie genau einen Konstruktor mit genau 1 Parameter.

d)

Erzeugen Sie die Methode ... berechneWert(...), die den Wert einer Immobilie berechnet (der Wert berechnet sich aus dem 1000-fachen der Miete).

Ausser diesen Methoden dürfen in der Klasse Auto und Immobile keinen anderen Methoden erstellt werden.

2) 8P

a) 1P

Erzeugen Sie in main(...) ein Auto, das 2 Jahre alt ist und zum Preis von 2000 gekauft wurde und speichern es in der Variable "auto1" ab.

b) 1P

Erzeugen Sie in main(...) eine Immobilie, die eine Miete von 500 Euro abwirft und speichern es in der Variable "immo1" ab.

c) 1P

Erzeugen Sie in main(...) ein Auto, das 4 Jahre alt ist und zum Preis von 40000 gekauft wurde und speichern es in der Variable "auto2" ab.

d) 5P

Speichern Sie in main(...) diese 3 Tombolagewinne (Waren) hintereinander in dem Feld "tombolaGewinne" der Länge 3.

3) 4P

Der Nettogewinn eines Tombolagewinns ist das 0,8 fache des mit der Methode berechneWert(...) berechneten Werts eines Tombolagewinns.

Erstellen Sie die Methode nettoGewinn(...) in (genau einer) der entsprechenden Klasse. (siehe Aufgabe 4)

4) 8P

a) 5P

In der Klasse TombolaGewinn muss die static-Methode

..... berechneMaximalenNettoGewinn(...) implementiert werden, die von einem Feld, in dem die gewonnen Waren (Autos und Immobilien) gespeichert sind, von der Ware den Nettogewinn zurückliefert, der am höchsten ist.

b) 3P

In main() muss ein Aufruf von berechneMaximalenNettoGewinn(...) realisiert werden, der den Wert der Ware (Tombolagewinn) zurückgibt, die den höchsten Nettogewinn hat und sich im Feld "tombolaGewinne" befindet.

Die Rückgabe muß in der noch zu erstellenden Variablen mit Namen "wareMax1" gespeichert werden.

5) 8P

a) 5P

Von 2 Tombolagewinnen (Waren), soll die Ware mit dem grösseren Wert (berechneten Wert) von 2 Waren zurückgeliefert werden.

Erstellen Sie dazu in der Klasse Tombolagewinn die folgende nicht static-Methode:

.... vergleiche(...)

b) 3P

In main() muss ein Aufruf von vergleiche(...) realisiert werden, der die Tombolagewinne (Waren) der zwei Autos vergleicht, die oben erzeugt wurden und das mit dem grösseren Warenwert zurückgibt.

Die Rückgabe muß in der noch zu erstellenden Variablen mit Namen "wareMax2" speichert.

Lösungen:

1)

25P

```
class Auto extends TombolaGewinn{           // 1P
    private double preis;                   // 1P
    private int alter;                     // 1P

    public double getPreis() {              // 2P
        return preis;
    }

    public void setPreis(double ePreis) {   // 2P
        this.preis = ePreis;
    }

    public int getAlter() {                 // 2P
        return alter;
    }

    public void setAlter(int alter) {       // 2P
        this.alter = alter;
    }

    public Auto(double preis, int alter) {  // 2P
        this.preis = preis;
        this.alter = alter;
    }

    public double berechneWert(){           // 2P
        return preis/alter;
    }
}

class Immobilie extends TombolaGewinn{     // 1P
    private double miete;                  // 1P

    public Immobilie(double miete) {       // 2P
        this.miete = miete;
    }

    public double getMiete() {              // 2P
        return miete;
    }

    public void setMiete(double ertrag) {   // 2P
        this.miete = miete;
    }

    public double berechneWert(){           // 2P
        return miete*1000;
    }
}
```

2)

8P

```
public static void main(String[] args) {
    // 2a) bis 2d)
    Auto auto1 = new Auto(2000,2);         // 1P
    Immobilie immol = new Immobilie(500);  // 1P
    Auto auto2 = new Auto(40000,4);        // 1P
}
```

```

// 2d)
TombolaGewinn[] tombolaGewinne;           // 1P
tombolaGewinne = new TombolaGewinn[3];     // 1P
tombolaGewinne[0]=auto1;                   // 1P
tombolaGewinne[1]=immo1;                   // 1P
tombolaGewinne[2]=auto2;                   // 1P
}

```

3) 4P

```

abstract class TombolaGewinn{
    ...
    public double nettoGewinn(){
        return 0.8 * berechneWert();
    }
    ....
}

```

4) 8P

a) 5P

```

public static double berechneMaximalenNettoGewinn(
    TombolaGewinn[] tombolaGewinne){
    double max=0;
    for(int i=0;i<tombolaGewinne.length;i++){
        if (tombolaGewinne[i].nettoGewinn()>max){
            max=tombolaGewinne[i].nettoGewinn();
        }
    }
    return max;
}

```

b) 3P

```

double maxWare1; // 1P
maxWare1=        // 2P
TombolaGewinn.berechneMaximalenNettoGewinn(tombolaGewinne);

```

5) 8P

a) 5P

```

public TombolaGewinn vergleiche(TombolaGewinn tg){
    TombolaGewinn temp;
    if(this.berechneWert()<=tg.berechneWert()){
        temp=tg;
    }
    else{
        temp=this;
    }
    return temp;
}

```

b) 3P

```

TombolaGewinn maxWare2; //1P
maxWare2 =               //2P
tombolaGewinne[0].vergleiche(tombolaGewinne[2]);
// alternativ:
// maxWare2 = auto1.vergleiche(auto2);

```

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

Bemerkungen:

Ein Java-Programm muss nach dem Prinzip der OOP gestaltet werden.

Insbeondere müssen dazu - falls möglich - verschiedene Konzepte wie Klassen, Vererbung, Polymorphie, Interface, usw. verwendet werden.

Es wird auch bewertet, wie gut dieses Prinzip der OOP umgesetzt wurde.

Hier speziell: Kein instanceof benutzen

1)

104P

Zur Information:

Frauen, Hündinnen und Männer sind Lebewesen. Hündinnen und Frauen können schwanger werden und müssen deshalb eine Fruchtwasseruntersuchung machen.

Die Klasse Hündin hat genau (d.h. keine weiteren) das Attribut "gewicht".

Die Klasse Frau hat genau das Attribut "alter".

Die Klasse Mann hat genau das Attribut "iq" (bedeutet Intelligenzquotient).

Die Klasse Lebewesen hat genau das Attribut "name".

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

`int getAttraktivität()`

zu entwickeln, die die "Attraktivität" eines Lebewesens berechnet.

Diese muß (bitte nur als Modell auffassen, hat nichts mit der Realität zu tun) wie folgt berechnet werden:

Bei einer Hündin: $2 * \text{gewicht} + 50$

Bei einer Frau: $2 * \text{alter}$

Beim Mann: $100 - \text{iq}$

Die Dringlichkeit einer Fruchtwasseruntersuchung muß in der Methode

`int getfwUntersuchung()`

wie folgt berechnet werden:

Bei einer Hündin: $\text{gewicht} - 10$

Bei einer Frau: $\text{alter} - 35$

- a) 12P
Erstellen Sie dazu ein UML-Diagramm, in dem nur die Klassennamen und alle Attribute vorkommen (und die nicht durch andere ergänzt werden dürfen).
Alle zugehörigen Methoden müssen dargestellt werden (Ausnahme: get- und set-Methoden).
Bitte vor Erstellung des UML-Diagramms die Klassenarbeit vollständig durchlesen.
- b) 49P
Implementieren Sie dazu den nötigen Quellcode (mit korrekter Syntax, so daß das Programm lauffähig ist). Die entsprechenden Konstruktoren müssen erstellt werden.
Es dürfen keine weiteren Attribute eingefügt werden.
Die Konstruktoren müssen jeweils Parameter enthalten.
Keine set- und get-Methoden implementieren. Diese werden als implementiert angenommen.
- c) 15P
Erstellen Sie die Klasse "Familie" mit genau den Attributen "name" und "mitglieder"
(Ein Feld der Länge 10 in dem Familienmitglieder abgespeichert werden können).
c1) Erzeugen Sie einen Konstruktor mit genau 2 Parametern.
c2) Erzeugen Sie die Methode add(...) mit der man an die i-te Stelle des Feldes ein Lebewesen abspeichern kann.
c3) Implementieren Sie dort zusätzlich die Methode:
... attraktivVergleich(...)
Diese berechnet von 2 Lebewesen dasjenige mit der höheren Attraktivität und gibt dieses zurück.
- d) 28P
Machen Sie folgendes in der Methode main()
d1) Erzeugen Sie folgende Objekte:
Hündin h1: (Name: Luna, Gewicht: 10),
Frau f1: (Name: berta, Alter: 50),
Mann m1: (Name: Hohlger, iq: 60),
d2) Speichern Sie diese Lebewesen in dem Feld "lebewesen".
d3) Berechnen Sie mit Hilfe einer Schleife und der Methode getAttraktivität()
die Attraktivität der jeweiligen Lebewesen und geben diese auf dem Bildschirm aus.
d4) In dem Feld "schwangereLW" (schwangere Lebewesen) müssen die im vorigen Programmteil erstellte Hündin und Frau abgespeichert werden und dann mit Hilfe einer Schleife und der Methode getfwUntersuchung() die Dringlichkeit einer Fruchtwasseruntersuchung des jeweiligen Lebewesens berechnet und auf dem Bildschirm ausgegeben werden.
d5) Erzeugen Sie eine Familie (Variablenname: familie) mit dem Namen "lowinzig"
Berechnen Sie, wer attraktiver ist: m1 oder h1.
Speichern Sie das Ergebnis in der Variablen tempLebewesen ab.
d6) h1 muß Mitglied der Familie "lowinzig" werden.

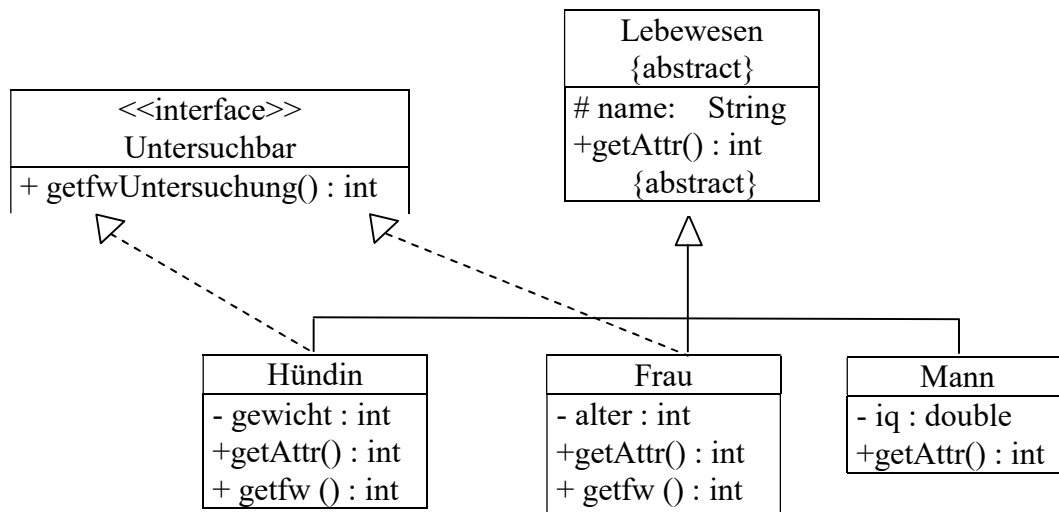
Lösung:

a)

12P

Bemerkung: Um Platz zu sparen wurde abgekürzt:

getfwUntersuchung durch getfw und getAttraktivität durch getAttr



b)

49P

```
// Interface erstellen 5P
public interface Untersuchbar {
    int getfwUntersuchung();
}

// 9P
abstract class Lebewesen{
    private String name;
    public Lebewesen(String name) {
        this.name = name;
    }

    public abstract int getAttraktivität();
}

// 13P
class Hündin extends Lebewesen implements Untersuchbar{
    private int gewicht;

    public int getGewicht() {
        return gewicht;
    }

    public void setGewicht(int gewicht) {
        this.gewicht = gewicht;
    }

    public Hündin(int gewicht, String name) {
        super(name);
        this.gewicht = gewicht;
    }

    public int getfwUntersuchung(){
        return gewicht-10;
    }

    public int getAttraktivität(){
        return (2*gewicht+50);
    }
}
```

```
// 13P
class Frau extends Lebewesen implements Untersuchbar{
    private int alter;

    public Frau(int alter, String name) {
        super(name);
        this.alter = alter;
    }

    public int getfwUntersuchung(){
        return alter-35;
    }

    public int getAttraktivität(){
        return (2*alter);
    }
}
```

```
// 9P
class Mann extends Lebewesen{
    private int iq;

    public Mann(int iq, String name) {
        super(name);
        this.iq = iq;
    }

    public int getAttraktivität(){
        return (100-iq);
    }
}
```

c)

15P

```
class Familie{
    private String name;
    private Lebewesen [] mitglieder;

    public Familie (String name){
        mitglieder = new Lebewesen[10];
        this.name=name;
    }

    public void add(Lebewesen l, int i){
        mitglieder[i]=l;
    }

    public Lebewesen attraktivVergleich(Lebewesen l1, Lebewesen l2){
        if(l1.getAttraktivität()<l2.getAttraktivität()){
            return l2;
        }
        else{
            return l1;
        }
    }
}
```

d)

28P

```
public class Startklasse {
    public static void main(String[] args) {
        int i;
        Hündin h1=new Hündin(10,"Luna");           // 2P
        Frau f1=new Frau(50,"Berta");              // 2P
        Mann m1=new Mann(60,"Hohlger");            // 2P
        Lebewesen [] lebewesen = new Lebewesen[3]; // 2P
        lebewesen[0]=h1;                           // 1P
        lebewesen[1]=f1;                           // 1P
        lebewesen[2]=m1;                           // 1P
        Untersuchbar [] schwangereLW = new Untersuchbar [2]; // 2P
        schwangereLW[0]=h1;                        // 1P
        schwangereLW[1]=f1;                        // 1P

        for(i=0;i<3;i++){                          // 3P
            System.out.println("Attraktionswert="
                               +lebewesen[i].getAttraktivität());
        }

        for(i=0;i<2;i++){                          // 3P
            System.out.println("Dringlichkeit="
                               +schwangereLW[i].getfwUntersuchung());
        }

        Familie familie = new Familie("lowinzig"); // 2P
        Lebewesen tempLebewesen;                  // 1P
        familie.add(h1, 0);                        // 2P
        tempLebewesen =familie.attraktivVergleich(m1, h1); // 2P
    }
}
```

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

Bemerkungen:

B1) Ein Java-Programm muss nach dem Prinzip der OOP gestaltet werden.

Insbeondere müssen verschiedene Konzepte wie Klassen, Vererbung, Polymorphie, usw. verwendet werden.

Es wird auch bewertet, wie gut dieses Prinzip der OOP umgesetzt wurde.

B2) 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) 53P

Auf einem Fest wird eine Tombola (Lose ziehen) veranstaltet.

Ein Tombolagewinn ist eine Ware, kann hier also ein Auto oder eine Immobilie sein.

Außer der Startklasse müssen genau die folgenden 3 Klassen erzeugt werden:

(d.h. es dürfen keine weiteren Klassen erzeugt werden)

Auto, Immobilie und TombolaGewinn.

1) 25P

a)

Erzeugen Sie die Klasse Auto mit genau den 2 Attributen (und nur diesen Attributen) "preis" und "alter" und den zu diesen Attributen gehörigen get-und set-Methoden.

Erzeugen Sie genau einen Konstruktor mit genau 2 Parametern.

b)

Erzeugen Sie die Methode ... berechneWert(...), die den Wert eines Autos berechnet (der Wert berechnet sich aus dem Preis des Autos dividiert durch das Alter).

c)

Erzeugen Sie die Klasse Immobilie mit genau dem einen Attribut (und nur diesem Attribut) "miete" und die zu diesem Attribut zugehörige get-und set-Methode.

Erzeugen Sie genau einen Konstruktor mit genau 1 Parameter.

d)

Erzeugen Sie die Methode ... berechneWert(...), die den Wert einer Immobilie berechnet (der Wert berechnet sich aus dem 1000-fachen der Miete).

Ausser diesen Methoden dürfen in der Klasse Auto und Immobile keinen anderen Methoden erstellt werden.

2) 8P

a) 1P

Erzeugen Sie in main(...) ein Auto, das 2 Jahre alt ist und zum Preis von 2000 gekauft wurde und speichern es in der Variable "auto1" ab.

b) 1P

Erzeugen Sie in main(...) eine Immobilie, die eine Miete von 500 Euro abwirft und speichern es in der Variable "immo1" ab.

c) 1P

Erzeugen Sie in main(...) ein Auto, das 4 Jahre alt ist und zum Preis von 40000 gekauft wurde und speichern es in der Variable "auto2" ab.

d) 5P

Speichern Sie in main(...) diese 3 Tombolagewinne (Waren) hintereinander in dem Feld "tombolaGewinne" der Länge 3.

3) 4P

Der Nettogewinn eines Tombolagewinns ist das 0,8 fache des mit der Methode berechneWert(...) berechneten Werts eines Tombolagewinns.

Erstellen Sie die Methode nettoGewinn(...) in (genau einer) der entsprechenden Klasse. (siehe Aufgabe 4)

4) 8P

a) 5P

In der Klasse TombolaGewinn muss die static-Methode

..... berechneMaximalenNettoGewinn(...) implementiert werden, die von einem Feld, in dem die gewonnen Waren (Autos und Immobilien) gespeichert sind, von der Ware den Nettogewinn zurückliefert, der am höchsten ist.

b) 3P

In main() muss ein Aufruf von berechneMaximalenNettoGewinn(...) realisiert werden, der die Ware (Tombolagewinn) zurückgibt, die den höchsten Nettogewinn hat und sich im Feld "tombolaGewinne" befindet.

Die Rückgabe muß in der noch zu erstellenden Variablen mit Namen "wareMax1" speichern.

5) 8P

a) 5P

Von 2 Tombolagewinnen (Waren), soll die Ware mit dem grösseren Wert (berechneten Wert) von 2 Waren zurückgeliefert werden.

Erstellen Sie dazu in der Klasse Tombolagewinn die folgende nicht static-Methode:

.... vergleiche(...)

b) 3P

In main() muss ein Aufruf von vergleiche(...) realisiert werden, der die Tombolagewinne (Waren) der zwei Autos vergleicht, die oben erzeugt wurden und das mit dem grösseren Warenwert zurückgibt.

Die Rückgabe muß in der noch zu erstellenden Variablen mit Namen "wareMax2" speichern.

Lösungen:

25P

```
1)
class Auto extends TombolaGewinn{           // 1P
    private double preis;                   // 1P
    private int alter;                     // 1P

    public double getPreis() {              // 2P
        return preis;
    }

    public void setPreis(double ePreis) {   // 2P
        this.preis = ePreis;
    }

    public int getAlter() {                 // 2P
        return alter;
    }

    public void setAlter(int alter) {       // 2P
        this.alter = alter;
    }

    public Auto(double preis, int alter) {  // 2P
        this.preis = preis;
        this.alter = alter;
    }

    public double berechneWert(){          // 2P
        return preis/alter;
    }
}

class Immobilie extends TombolaGewinn{     // 1P
    private double miete;                  // 1P

    public Immobilie(double miete) {       // 2P
        this.miete = miete;
    }

    public double getMiete() {              // 2P
        return miete;
    }

    public void setMiete(double ertrag) {   // 2P
        this.miete = miete;
    }

    public double berechneWert(){          // 2P
        return miete*1000;
    }
}
```

```
2)
public static void main(String[] args) {
    // 2a) bis 2d)
    Auto auto1 = new Auto(2000,2);         // 1P
    Immobilie immol = new Immobilie(500);  // 1P
    Auto auto2 = new Auto(40000,4);        // 1P
}
```

8P

```

// 2d)
TombolaGewinn[] tombolaGewinne;           // 1P
tombolaGewinne = new TombolaGewinn[3];     // 1P
tombolaGewinne[0]=auto1;                   // 1P
tombolaGewinne[1]=immo1;                   // 1P
tombolaGewinne[2]=auto2;                   // 1P
}

```

3) 4P

```

abstract class TombolaGewinn{
    ...
    public double nettoGewinn(){
        return 0.8 * berechneWert();
    }
    ....
}

```

4) 8P

a) 5P

```

public static double berechneMaximalenNettoGewinn(
    TombolaGewinn[] tombolaGewinne){
    double max=0;
    for(int i=0;i<tombolaGewinne.length;i++){
        if (tombolaGewinne[i].nettoGewinn()>max){
            max=tombolaGewinne[i].nettoGewinn();
        }
    }
    return max;
}

```

b) 3P

```

double maxWare1; // 1P
maxWare1=        // 2P
TombolaGewinn.berechneMaximalenNettoGewinn(tombolaGewinne);

```

5) 8P

a) 5P

```

public TombolaGewinn vergleiche(TombolaGewinn tg){
    TombolaGewinn temp;
    if(this.berechneWert()<=tg.berechneWert()){
        temp=tg;
    }
    else{
        temp=this;
    }
    return temp;
}

```

b) 3P

```

TombolaGewinn maxWare2; //1P
maxWare2 =              //2P
tombolaGewinne[0].vergleiche(tombolaGewinne[2]);
// alternativ:
// maxWare2 = auto1.vergleiche(auto2);

```


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

Um die Veranstaltungen besser auf die Gäste zuschneiden zu können, wird ein intelligentes Planungsmodul entwickelt.

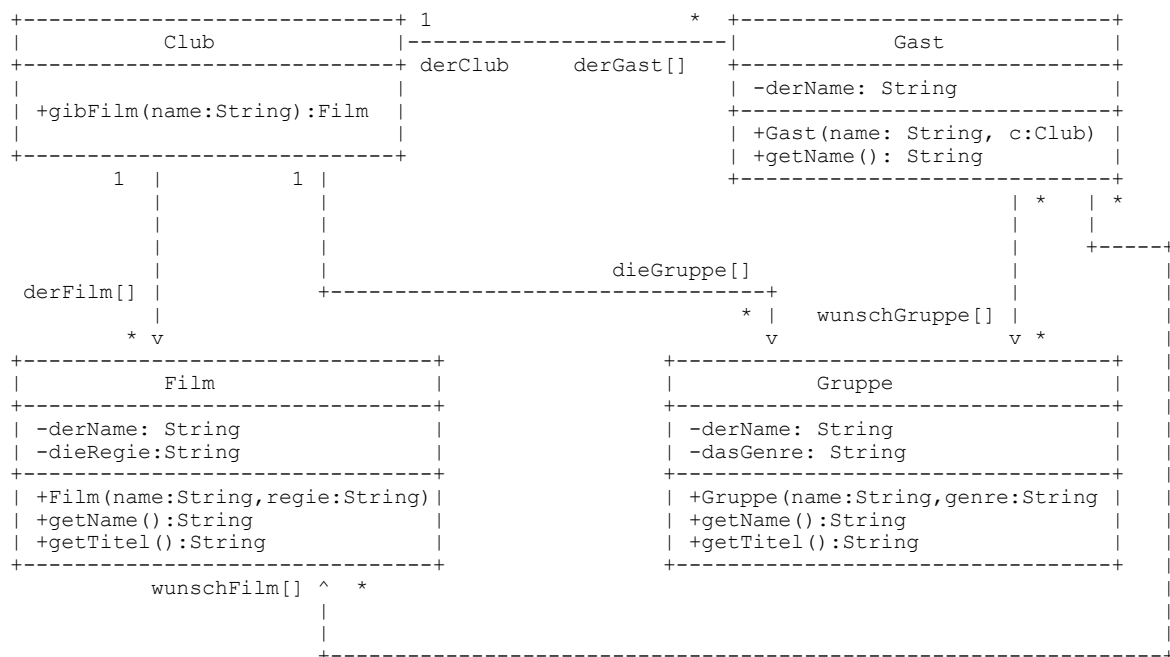
Eine Klasse Club (Steuerungsschicht) verwaltet die Gäste, Filme, Musik- bzw.

Theatergruppen. Gäste können sich Filme und Gruppen wünschen.

Hier ein Vorentwurf des Klassendiagramms (Wichtig: Es sind nicht alle Attribute und Methoden angegeben). Die Klasse Club greift über die Attribute derGast[], der Film[] und dieGruppe[] auf die Objekte der entsprechenden Klasse zu.

Beispiel:

derGast[0] verweist auf das 1. Gastobjekt. Die Anzahl der verwalteten Gastobjekt kann durch derGast.length ermittelt werden. Jeder Gast hat eine Film- und Gruppenwunschliste.



1)

Entwickeln Sie den Quellcode für den Konstruktor:

Gast(name: String, c: Club)

2)

Die Methode:

Club.gibFilm(name: String) : Film

gibt den Verweis auf ein Filmobjekt zurück, dessen Attribut derName mit dem Parameter name übereinstimmt. Wenn noch kein entsprechendes Objekt existiert, erzeugt sie ein neues Filmobjekt.

Implementieren Sie die Methode.

3)

Die Klassen Film und Gruppe sind sich sehr ähnlich und sollen mit einer abstrakten Klasse Veranstaltung generalisiert werden.

Zeichnen Sie ein neues Klassendiagramm mit den Klassen Club, Gast, Veranstaltung, Gruppe und Film.

Die Methode Film.getTitel(): String gibt "<derName>, Regie: <dieRegie>" zurück, z.B: "Der kalte Winter, Regie: Leo Lego".

Die Methode Gruppe.getTitel(): String gibt "<derName>, <dasGenre>" zurück, z.B: "Sex Pistols, Punk".

Lösung:

1)

```
public Gast(String name, Club c){
    derName = name;
    derClub = c;
}
```

```
public Film gibFilm(String name){
    int i;
    for(i=0;i<derFilm.lengt;i++){
        if(derFilm[i].getname().equals(),name){
            return name;
        }
    }
    derFilm.add(new Film(name,""));
}
```

