

Name, Vorname:

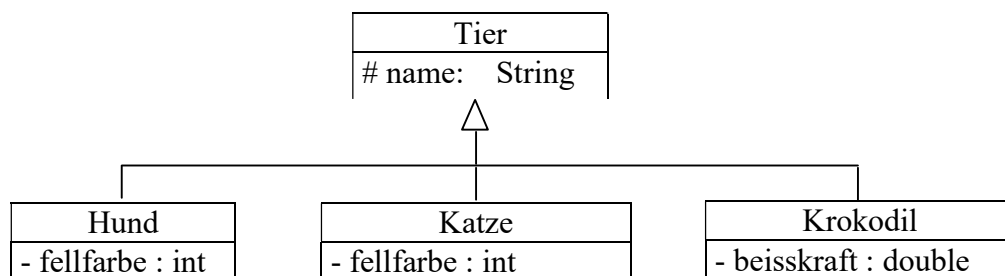
Hilfsmittel:
keine

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 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.



a) 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!

b)

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.

b0) 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()`

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.

c)

12P

Beachten Sie:

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

Dagegen besitzt eine Katze bzw. ein Hund ein Fell.

In dem Feld "felltiere" 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.

c1) Was wäre der Nachteil der Lösung, in der man die Methode `getFellfarbe()` in der Klasse `Tier` implementiert?

c2) Beurteilen Sie die folgende Lösung:

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

c3) Was ist die beste Lösung ? (keine Implementierung, nur verbale Beschreibung und Begründung).

Lösung:

public class MainKlassenarbeit {		8P
public static void main(String[] args) {		
int i;		
Tier tiere[];	1P	
tiere = new Tier[3];	1P	
tiere[0] = new Katze("Ute", 2);	1P	
tiere[1] = new Hund("Rex", 3);	1P	
tiere[2] = new Krokodil("Krok", 30);	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	11P
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 Hund extends Tier {	1P	10P
private int fellfarbe;	1P	
 public Hund(String pName, int pFellfarbe){	2P	
super(pName);		
fellfarbe = pFellfarbe;		
}		
 public String getBeschreibung(){	2P	
String s;		
s="Name="+name+" Fellfarbe="+fellfarbe;		
return s;		
}		
 public void setFellfarbe(int pFellfarbe){	2P	
fellfarbe = pFellfarbe;		
}		
 public int getFellfarbe(){	2P	
return fellfarbe;		
}		
}		

<pre> 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; } } </pre>	1P 1P 2P 2P 2P 	10P
--	--	---

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
 Da in einer Schleife z.B. mit felldiere. 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.
 Man kann also keine Katzen und Tiere gemeinsam in einem Feld abspeichern und deren Fellfarbe mit getFellfarbe() abfragen.

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:
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 immo1 = 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:

B1) Ein Java-Programm muss nach dem Prinzip der OOP gestaltet 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.
Es dürfen nur Felder (konstante Länge) verwendet werden, keine ArrayLists.

B3) Der Ausgang einer Oder-Schaltung ist genau dann 1, wenn mindestens einer der Eingänge den Wert 1 hat.

Der Ausgang einer Und-Schaltung ist genau dann 0, wenn mindestens einer der Eingänge den Wert 0 hat.

I) 59P

Es sollen verschiedene Digitalschaltungen (mit jeweils festzulegender Anzahl Eingänge und genau einem Ausgang) am Rechner simuliert werden.

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

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

"OderSchaltung", "UndSchaltung" und einer Basisklasse "Schaltung".

Digitalschaltungen arbeiten nur mit den Werten 0 und 1. Falls ein Parameterwert einen integer-Wert ungleich 0 und ungleich 1 hat, wird er innerhalb der Methode in 0 bzw. 1 umgewandelt (negative Werte werden in positive, gerade Werte in 0 und ungerade Werte in 1 umgewandelt).

Beispiel:

-19 --> 1 20 --> 0 7 --> 1 -14 --> 0

Tipp: Benutzen Sie u.a. den Modulo-Operator %

Beispiele:

-15 % 2 = -1

16 % 2 = 0

1)

25P

a)

Erzeugen Sie die Klasse "Schaltung" mit genau den 2 Attributen (und nur diesen Attributen) "anzahlEingänge" und "eingänge" und den zu diesen Attributen gehörigen get-und set-Methoden.

Erzeugen Sie genau einen Konstruktor mit genau 1 Parameter.

Im Konstruktor müssen die Anzahl der Eingänge festgelegt und alle Eingänge mit 1 vorbelegt werden.

Zusätzlich müssen noch die folgenden Methoden (und keine weiteren!) implementiert werden:

... setEingang(...)

ein (vom Programmierer) bestimmter Eingang wird auf einen (vom Programmierer) bestimmten Wert gesetzt. Beachten Sie die Umwandlung der Zahlen von der vorherigen Seite.

...getEingang(...)

gibt den Wert eines (vom Programmierer) bestimmten Eingangs zurück.

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

... printAusgang(...)

zu entwickeln, die den Wert des Ausgangs auf dem Bildschirm ausgibt.

Ausser diesen o.g. Methoden dürfen in dieser Klasse keine anderen Methoden erstellt werden.

b)

10P

Erzeugen Sie die Klasse OderSchaltung mit 0 Attributen, genau einem Konstruktor mit genau einem Parameter und den Methoden ...berechneAusgang(...) und ... printAusgang(...), die den Wert des Ausgangs auf dem Bildschirm ausgibt.

Ausser diesen o.g. Methoden dürfen in dieser Klasse keine anderen Methoden erstellt werden.

c)

10P

Erzeugen Sie die Klasse UndSchaltung mit 0 Attributen, genau einem Konstruktor mit genau einem Parameter und den Methoden ... berechneAusgang(...) und ... printAusgang(...), die den Wert des Ausgangs auf dem Bildschirm ausgibt.

Ausser diesen o.g. Methoden dürfen in dieser Klasse keine anderen Methoden erstellt werden.

2)

14P

a) 4P

Erzeugen Sie in main(...) eine Oder-Schaltung "oder1" mit den Eingängen (0,0,1).

Geben Sie den Wert des Ausgangs mit der entsprechenden Methode auf dem Bildschirm aus.

b) 4P

Erzeugen Sie in main(...) eine Und-Schaltung "und1" mit den Eingängen (1,0,1).

Geben Sie den Wert des Ausgangs mit der entsprechenden Methode auf dem Bildschirm aus.

c) 6P

Erzeugen Sie in main(...) ein Feld der Länge 2.

Speichern Sie dort die bei a) und b) erzeugte Und-Schaltung bzw. Oder-Schaltung ab und geben die entsprechenden Werte der Ausgänge dieser Schaltungen in einer Schleife auf dem Bildschirm aus.

Lösungen:

```
public class Startklasse {
    public static void main(String[] args) {
        int i;
        // 4P
        OderSchaltung oder1 = new OderSchaltung(3);
        int[] v1 = {0,0,1};
        oder1.setEingänge(v1);
        oder1.printAusgang();
        // 4P
        UndSchaltung und1 = new UndSchaltung(3);
        int[] v2 = {1,0,1};
        und1.setEingänge(v1);
        und1.printAusgang();
        // 6P
        Schaltung[] schaltungen;
        schaltungen = new Schaltung[2];
        schaltungen[0]=oder1;
        schaltungen[1]=und1;
        for(i=0;i<2;i++){
            schaltungen[i].printAusgang();
        }
    }
}

abstract class Schaltung{
    private int anzahlEingänge;
    private int[] eingänge;

    public Schaltung(int anzahlEingänge){
        int i;
        this.anzahlEingänge=anzahlEingänge;
        eingänge = new int[anzahlEingänge];
        for(i=0;i<anzahlEingänge;i++){
            eingänge[i]=1;
        }
    }

    public int getAnzahlEingänge() {
        return anzahlEingänge;
    }

    public void setAnzahlEingänge(int anzahlEingänge) {
        this.anzahlEingänge = anzahlEingänge;
    }

    public int[] getEingänge() {
        return eingänge;
    }

    public void setEingänge(int[] eingänge) {
        this.eingänge = eingänge;
    }

    public int getEingang(int pos) {
        return eingänge[pos];
    }
}
```

```

    public void setEingang(int pos, int wert) { // 5P
        if(wert<0){
            wert = -wert;
        }
        wert = wert%2;
        this.eingänge[pos] = wert;
    }

    public abstract void printAusgang(); // 2P
}

class OderSchaltung extends Schaltung{ // 1P
    public OderSchaltung(int anzahlEingänge){ // 2P
        super(anzahlEingänge);
    }

    int berechneAusgang() { // 5P
        int i;
        int erg=0;
        for(i=0;i<getAnzahlEingänge();i++){
            if(getEingang(i)==1){
                erg=1;
                break;
            }
        }
        return erg;
    }

    public void printAusgang(){ // 2P
        System.out.println("Ausgangswert Oder-
                           Schaltung="+berechneAusgang());
    }
}

class UndSchaltung extends Schaltung{ // 1P
    public UndSchaltung(int anzahlEingänge){ // 2P
        super(anzahlEingänge);
    }

    int berechneAusgang() { // 5P
        int i;
        int erg=1;
        for(i=0;i<getAnzahlEingänge();i++){
            if(getEingang(i)==0){
                erg=0;
                break;
            }
        }
        return erg;
    }

    public void printAusgang(){ // 2P
        System.out.println("Ausgangswert Und-
                           Schaltung="+berechneAusgang());
    }
}

```