

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

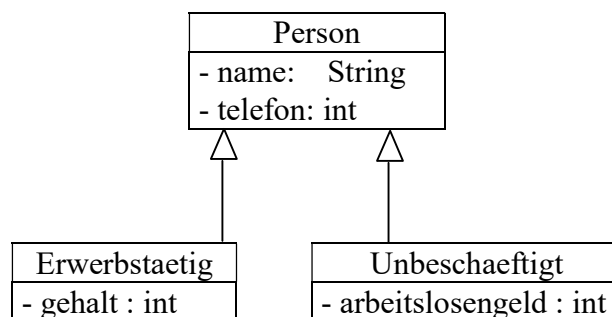
**ACHTUNG:** Die Klassenarbeit besteht aus genau der Aufgabe I (die aus Teilaufgaben besteht).

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

I) 50P

Gegeben ist das folgende UML-Diagramm, in dem nur die Klassennamen und alle Attribute vorkommen (und die nicht durch andere ergänzt werden dürfen, außer bei Aufgabe 3)).

Die zugehörigen Methoden werden hier nicht dargestellt.



1) 32P

Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen und genau den (und NUR den) folgenden Methoden (es dürfen also keine weiteren Methoden erzeugt werden): jeweilige Konstruktoren (keine Standardkonstruktoren), set- und get-Methoden.

2) 9P

Erzeugen Sie in der Methode main (jeweils durch eine Anweisung)

a) den Erwerbstätigen "Schaffer" mit der Telefonnummer 4711, dem Gehalt von 1500 Euro

b) den Unbeschäftigten "Schröder" mit der Telefonnummer 4712 und Arbeitslosengeld von 400 Euro

c) die Person "Schulze" mit der Telefonnummer 12345

d)

Herr Schaffer bekommt 100 Euro mehr Gehalt.

Realisieren Sie dies durch genau eine einzige Anweisung (keine neue Methode implementieren), die zum alten Gehalt die 100 Euro dazu addiert.

In der Anweisung muss also der alte Gehalt ermittelt werden.

3)

9P

Bemerkung:

Es gibt die Klasse Hund mit genau den 2 Attributen "name", "alter" und genau nur einen Konstruktor (dieser besteht genau aus 2 Parametern).

Diese Klasse wurde schon implementiert und darf deshalb hier nicht nochmals implementiert werden. Sie wird hier als bekannt vorausgesetzt.

a) 2P

Zu einem Erwerbstätigen soll genau ein Hund gehören.

Realisieren Sie dies durch genau eine einzige Zeile Quellcode.

b) 5P

Implementieren Sie die Methode setHund(...), die einen Hund (mit Namen und Alter) erzeugt

c) 2P

Der Erwerbstätige "Schaffer" holt sich einen 10-jährigen Hund mit Namen "Rex" vom Tierheim.

Realisieren Sie dies in main(...) nur durch Verwendung der Methode setHund(...) durch genau eine Anweisung.

## Lösung:

1)

```
public class Startklasse {
    public static void main(String[] args){
        Erwerbstaetig e = new Erwerbstaetig("Schaffer", 4711, 1500);    // 2P
        Unbeschaeftigt u = new Unbeschaeftigt("Schroeder", 4712, 400); // 2P
        Person p = new Person("Schulze",12345);                        // 2P
        e.setGehalt(e.getGehalt()+100);                                // 3P
        e.setHund("Rex", 10);                                          // 2P
    }
}

class Person{                                                         // 12P
    private String name;      // 1P
    private int telefon;      // 1P

    public Person (String pName, int pTelefon){ // 2P
        name = pName;
        telefon = pTelefon;
    }

    public void setTelefon(int pTelefon){ //2P
        telefon = pTelefon;
    }

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

    public int getTelefon(){ // 2P
        return(telefon);
    }

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

class Unbeschaeftigt extends Person{ //1P                               // 10P
    private int arbeitlosengeld; // 1P

    public Unbeschaeftigt(String pName, int pTelefon, // 4P
        int pArbeitslosengeld){
        super(pName, pTelefon);
        arbeitlosengeld = pArbeitslosengeld;
    }

    public void setArbeitslosengeld(int pArbeitslosengeld){ // 2P
        arbeitlosengeld = pArbeitslosengeld;
    }

    public int getArbeitslosengeld(){ // 2P
        return(arbeitslosengeld);
    }
}
```

```
class Erwerbstaetig extends Person{ // 1P
    private int gehalt ; // 1P
    private Hund hund; // 2P

    public Erwerbstaetig(String pName, int pTelefon, int pGehalt){ // 4P
        super(pName, pTelefon);
        gehalt = pGehalt;
    }

    public void setGehalt(int pGehalt){ // 2P
        gehalt = pGehalt;
    }

    public int getGehalt(){ // 2P
        return(gehalt);
    }

    public void setHund(String pName, int pAlter){ // 5P
        hund=new Hund(pName,pAlter);
    }
}
```

*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)

50P

Zur Information:

Das Spiel "Black Jacket" ("17 + 4") funktioniert wie folgt beschrieben:

Spieler 1 "würfelt" so lange mit einem Spezialwürfel (mit dem genau eine der folgenden Zahlen 2, 3, 4, 7, 8, 9, 10, 11 erwürfelt werden kann), mit dem Ziel als Summe nicht über 21 zu kommen. Die einzelnen Würfe sind verdeckt und vom Gegenspieler nicht zu sehen.

Dann macht der Gegenspieler die gleiche Prozedur.

Wer die größere Summe hat, hat gewonnen.

Wer eine Summe  $> 21$  hat, hat verloren (wenn die Summe des Gegeners  $< 22$ ).

Bei gleicher Summe ist der Spielausgang unentschieden.

Hat jeder Spieler eine Summe  $> 21$ , ist der Spielausgang ebenfalls unentschieden.

Wenn ein Spieler nicht mehr würfeln will oder eine Summe  $> 21$  hat, wird das Spiel sofort beendet und auf dem Bildschirm die entsprechenden Infos ausgegeben (Gewinner und Verlierer mit jeweiliger Punktzahl).

Bemerkung:

Die Methode `würfeln()` darf benutzt werden. Diese liefert genau eine der Zahlen

2, 3, 4, 7, 8, 9, 10, 11

Die Methode `eingabe()` darf benutzt werden. Mit ihr kann man Werte über Tastatur eingeben.

1)

Implementieren Sie die Klasse "Spiel" mit den entsprechenden Attributen und Methoden (mit Hilfe der OOP).

2)

2 Spieler sollen gegeneinander Spielen.

Implementieren Sie das zugehörige Programm in `main()`.

Geben Sie jeweils den Gewinner bzw. Verlierer (mit der jeweiligen Summe) auf dem Bildschirm aus.

Lösung:

```
package blackjack1;  
import java.util.Scanner;  
import java.io.*;
```

```
public class Startklasse {  
    public static void main(String[] args) {  
        int spielerNr=1;  
        int erg;  
        int beenden=0;  
        Spiel spiel = new Spiel();  
  
        while(spiel.istSpielBeendet()==false){  
            erg=spiel.eingabe(spielerNr);  
            if(erg==1){  
                spiel.ziehenUndWurfAusgeben(spielerNr);  
            }  
            else{  
                spiel.spielBeenden();  
            }  
  
            if(spielerNr==1)  
                spielerNr=2;  
            else  
                spielerNr=1;  
        }  
        spiel.ausgabeSpielErgebnis();  
    }  
}
```

```
class Spiel{  
    private int summeSpieler1;  
    private int summeSpieler2;  
    private int spielZustand;  
  
    public Spiel(){  
        summeSpieler1=0;  
        summeSpieler2=0;  
        spielZustand = 0; // Spiel nicht beenden  
    }  
  
    public void spielBeenden(){  
        spielZustand = 1;  
    }  
  
    public int getPunktezahl(int spielerNr){  
        int erg;  
        if(spielerNr==1)  
            erg=summeSpieler1;  
        else  
            erg=summeSpieler2;  
        return erg;  
    }  
}
```

```

public int ziehen(int spielerNr){
    int wert;
    wert = würfeln();
    if(spielerNr==1)
        summeSpieler1 = summeSpieler1 + wert;
    else
        summeSpieler2 = summeSpieler2 + wert;
    return wert;
}

public Boolean istSpielBeendet(){
    Boolean erg;
    if(summeSpieler1>=21 || summeSpieler2>=21 || spielZustand==1)
        erg = true;
    else
        erg = false;
    return erg;
}

public int gewinnErmittlung(){
    int erg;
    if(summeSpieler1>summeSpieler2)
        erg=1;
    else if(summeSpieler1<summeSpieler2)
        erg=2;
    else
        erg=0;
    if(summeSpieler1>21)
        erg=2;
    if(summeSpieler2>21)
        erg=1;
    return erg;
}

public void ausgabeSpielErgebnis(){
    int erg;
    erg=gewinnErmittlung();
    if(erg==1)
        System.out.println("Spieler 1 hat gewonnen");
    else if(erg==2)
        System.out.println("Spieler 2 hat gewonnen");
    else
        System.out.println("das Spiel ist unentschieden");
    System.out.println("summeSpieler1= " +summeSpieler1);
    System.out.println("summeSpieler2= " +summeSpieler2);
}

```

```

private int würfeln(){
    double zufall;
    int wert;
    zufall=Math.random();
    if(zufall>=0 && zufall<1.0/8.0)
        wert = 2;
    else if(zufall>=1.0/8.0 && zufall<2.0/8.0)
        wert = 3;
    else if(zufall>=2.0/8.0 && zufall<3.0/8.0)
        wert = 4;
    else if(zufall>=3.0/8.0 && zufall<4.0/8.0)
        wert = 7;
    else if(zufall>=4.0/8.0 && zufall<5.0/8.0)
        wert = 8;
    else if(zufall>=5.0/8.0 && zufall<6.0/8.0)
        wert = 9;
    else if(zufall>=6.0/8.0 && zufall<7.0/8.0)
        wert = 10;
    else
        wert = 11;
    return wert;
}

public void ausgabeGewürfelteZahl(int punkte){
    System.out.println(punkte);
}

public void ziehenUndWurfAusgeben(int spielerNr){
    int erg;
    erg=ziehen(spielerNr);
    System.out.print("Spieler " +spielerNr + " hat gezogen: ");
    ausgabeGewürfelteZahl(erg);
    System.out.println("Gesamtpunktzahl von Spieler "+spielerNr+ " ist: " +
getPunktezahl(spielerNr) +" Punkte");
}

public int eingabe(int spielerNr){
    int erg;
    Scanner scanner = new Scanner(System.in);
    System.out.println("Wollen Sie (Spieler "+ spielerNr +" würfeln 0:nein, 1:ja");
    erg = scanner.nextInt();
    return erg;
}
}

```



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

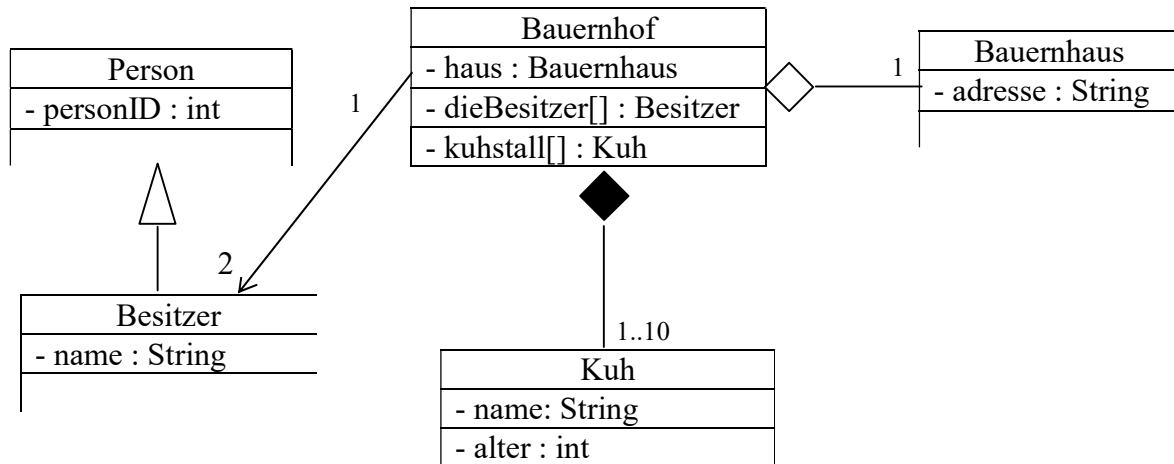
Bemerkung:

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)



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 anfuegenKuh(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());
            }
        }
    }
}
```

*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) 51P  
Ein Autor schreibt Bücher.

1.1) 7P

Erzeugen Sie die Klasse Autor mit genau dem Attribut (und nur dem Attribut) "name", den zu "name" gehörigen get-und set-Methoden und genau einem Nicht-Standardkonstruktor.

1.2)

a) 7P

Erzeugen Sie die Klasse Buch mit genau dem Attribut (und nur dem Attribut) "titel", den zu "titel" gehörigen get-und set-Methoden und genau einem Nicht-Standardkonstruktor.

b) 4P

Erstellen Sie in main() den Autor "Amann" und das Buch "Abuch"

1.3)

Zu einem Autor soll es genau ein Buch geben.

a) 8P

Programmieren Sie diesen Fall, indem Sie die Klasse Autor um die entsprechenden Methoden und genau ein Attribut ergänzen.

Der Konstruktor von Autor darf nicht verändert werden.

Schreiben Sie das Attribut und die Methoden unter die Überschrift: "Ergänzungen 1 zu Autor"

b) 8P

Der Autor "Bmann" schreibt das Buch "Bbuch". Schreiben Sie dazu in main() die entsprechenden Anweisungen.

c) 4P

Danach soll mit genau einer einzigen Anweisung vom Autor "Bmann" ausgehend (d.h. mit Hilfe der entsprechenden Objektvariablen, in der "Bmann" gespeichert ist) der Name des Autors und der Titel seines Buches auf dem Bildschirm ausgegeben werden.

1.4) 3P

Da ein Autor im Laufe seines Lebens mehrere Bücher schreibt, soll dies modelliert werden.  
Wie kann man das programmtechnisch machen?

Verbale Beschreibung, kein Java-Anweisungen.

1.5)

Der Vorgesetzte des Entwicklers des Programms will nicht, daß wie oben geschehen, das Buch in der Hauptmethode main erzeugt wird. Das Buch soll beim Anlegen des Autors mit den Infos "Name des Autors" und "Titel des Buches" nicht in main erstellt werden.

Modellieren Sie diesen Fall wie folgt:

a) 4P

Ändern Sie dazu den Konstruktor der Klasse Autor.

Schreiben Sie den Konstruktor unter die Überschrift: "Ergänzungen 2 zu Autor"

b) 2P

Der Autor "Cmann" schreibt das Buch "Cbuch".

Schreiben Sie dazu in main() genau eine einzige Anweisung, die dies realisiert.

c) 4P

Da der Autor nur nachts gearbeitet hat, hat sich ein Fehler in seinem Buchtitel eingeschlichen.  
Das Buch heißt nicht "Cbuch", sondern "Tsebuch"

Schreiben Sie dazu in main() genau eine einzige Anweisung, die dies realisiert.

## Lösungen:

1.1)

7P

```
class Autor{  
    private String name;                // 1P  
  
    public Autor(String pName){        // 2P  
        name=pName;  
    }  
  
    public String getName() {          // 2P  
        return name;  
    }  
  
    public void setName(String name) { // 2P  
        this.name = name;  
    }  
  
}
```

1.2)

11P

```
class Buch{  
    private String titel;              // 1P  
  
    public Buch(String pTitel){       // 2P  
        titel=pTitel;  
    }  
  
    public String getTitel() {        // 2P  
        return titel;  
    }  
  
    public void setTitel(String titel) { // 2P  
        this.titel = titel;  
    }  
  
}
```

1.2 b)

```
Autor a1 = new Autor("Amann");      // 2P  
Buch b1 = new Buch("Abuch");        // 2P
```

1.3)

20P

a)

```
class Autor
    private Buch buch; // 2P

    public void setBuch(Buch pBuch) { // 3P
        buch = pBuch;
    }

    public Buch getBuch() { // 3P
        return buch;
    }
}
```

b)

```
Autor a2 = new Autor("Bmann"); // 2P
Buch b2 = new Buch("Bbuch"); // 3P
a2.setBuch(b2); // 3P
```

c)

```
// 4P
System.out.println("Der Autor "+a2.getName()+
    " und sein Buch " +a2.getBuch().getTitel());
```

1.4)

3P

Z.B. durch ein Feld

1.5)

10P

a)

```
// 4P
public Autor(String pName, String title){
    name=pName;
    buch = new Buch(title);
}
```

b)

```
// 2P
Autor a3 = new Autor("Cmann", "Cbuch");
```

c)

```
// 4P
a3.getBuch().setTitel("Tsebuch");
```



*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.

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) 55P

Es sollen verschiedene Digitalerschaltungen (mit jeweils 2 Eingängen 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 der zugehörigen Basisklasse "Schaltung".

Digitalerschaltungen arbeiten nur mit den Werten 0 und 1.

1) 25P

a)

Erzeugen Sie die Klasse "Schaltung" mit genau den 2 Attributen (und nur diesen Attributen) "e1" und "e2" und den zu diesen Attributen gehörigen get-und set-Methoden.

Erzeugen Sie genau einen Konstruktor mit genau 1 Parameter.

Im Konstruktor müssen alle Eingänge mit einem bestimmten (dem gleichen) Wert vorbelegt werden.

(Diese Werte können vom Programmierer, der diese Methode verwendet, bestimmt werden).

Zusätzlich müssen noch die folgenden Methoden implementiert werden:

... setEingänge(...)

setzt alle Eingänge, also e1 und e2, jeweils auf einen (nicht notwendig den gleichen) Wert.  
(Diese Werte können vom Programmierer, der diese Methode verwendet, bestimmt werden).

... getEingänge(...)

liest die Werte aller Eingänge aus und gibt sie als in einem Feld zurück.

... setEingängeLuxus(...)

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

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 der 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 der 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) 10P

a) 4P

Erzeugen Sie in main(...) eine Oder-Schaltung "oder1" mit lauter Nullen an den Eingängen.

Setzen Sie dann die Eingänge auf (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 lauter Einsen an den Eingängen.

Setzen Sie dann die Eingänge auf (1,0).

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

c) 6P

Erzeugen Sie in main(...) eine Und-Schaltung "und2" mit den Eingängen der Ausgänge von den Schaltungen "und1" und "oder1".

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

### Lösungen:

```
public class Startklasse {                                     //10P
    public static void main(String[] args) {
        OderSchaltung oder1 = new OderSchaltung(0);
        oder1.setEingänge(0,1);
        oder1.printAusgang();

        UndSchaltung und1 = new UndSchaltung(1);
        und1.setEingänge(1,0);
        und1.printAusgang();

        UndSchaltung und2 = new UndSchaltung(1);
        und2.setE1(und1.berechneAusgang());
        und2.setE2(oder1.berechneAusgang());
        und2.printAusgang();
    }
}

class Schaltung{                                             //25P
    private int e1;                                         // 1P
    private int e2;                                         // 1P

    public Schaltung(int wert){ // 2P
        e1=wert;
        e2=wert;
    }

    public int getE1(){ // 2P
        return e1;
    }

    public void setE1(int pE1){ // 2P
        this.e1=pE1;
    }

    public int getE2(){ // 2P
        return e2;
    }

    public void setE2(int pE2){ // 2P
        this.e2=pE2;
    }

    public void setEingänge(int pE1, int pE2){ // 3P
        setE1(pE1);
        setE1(pE2);
    }

    public int[] getEingänge(){ // 4P
        int[] temp=new int[2];
        temp[0]=e1;
        temp[1]=e2;
        return temp;
    }

    public void setEingängeLuxus(int pE1,int pE2){ // 6P
        if(e1<0){
            e1=-e1;
        }
    }
}
```

```

        }
        e1 = e1%2;

        if(e2<0){
            e2=-e1;
        }
        e1 = e2%2;
    }
}

//10P
class OderSchaltung extends Schaltung{ // 1P
    public OderSchaltung(int wert){ // 3P
        super(wert);
    }

    int berechneAusgang(){ // 4P
        int erg;
        if(getE1()==0 && getE2()==0){
            erg=0;
        }
        else{
            erg=1;
        }
        return erg;
    }

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

//10P
class UndSchaltung extends Schaltung{ // 1P
    public UndSchaltung(int wert){ // 3P
        super(wert);
    }

    int berechneAusgang(){ // 4P
        int erg;
        if(getE1()==1 && getE1()==1){
            erg=1;
        }
        else{
            erg=1;
        }
        return erg;
    }

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

```