

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

1) 12P

a)

Was versteht man unter dem Begriff "Vererbung" ?

Welchen Vorteil hat man beim Vererben ?

b)

Was versteht man unter einem verdeckten Member ?

c)

Was geschieht, wenn beim Aufruf einer Methode ein Wert benutzt wird, der nicht dem erwarteten Datentyp entspricht ?

d)

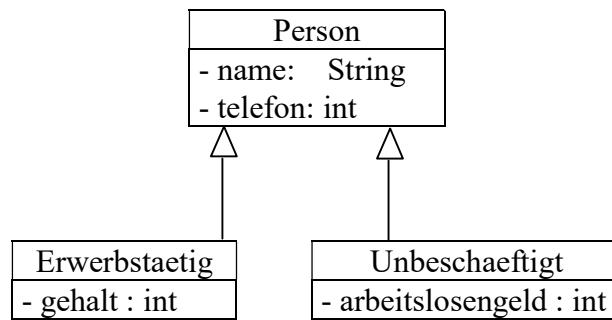
Wann unterscheiden sich die Spezifizierer public und protected ?

Kein Programm angeben, sondern eine verbale Beschreibung geben.

2)

39P

Gegeben ist das folgende abgespeckte UML-Diagramm:



a)

Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen mit den jeweiligen Konstruktoren, set- und get-Methoden.

Die Konstruktoren müssen jeweils Parameter enthalten.

b)

Erzeugen Sie in der Methode main (jeweils durch eine Anweisung) den Erwerbstätigen "Schaffer" mit der Telefonnummer 4711, dem Gehalt von 1500 Euro und den Unbeschäftigten "Schröder" mit der Telefonnummer 4712 und Arbeitslosengeld von 400 Euro.

c)

Herr Schaffer bekommt 100 Euro mehr Gehalt.

Realisieren Sie dies durch genau eine einzige Anweisung, die zum alten Gehalt die 100 Euro dazu addiert.

In der Anweisung muss also der alte Gehalt ermittelt werden.

Lösung:

1)

12P

a) 3P

Alle Member einer Klasse (außer dem Konstruktor) werden an die erbende Klasse weitervererbt, d.h. können dort benutzt werden.

b) 2P

Damit können Softwareteile wiederverwendet werden und müssen nicht nochmals implementiert werden (Redundanzverringern).

b) 3P

Eine Methode gleichen Namens existiert in der Ober- und Unterklasse.

Die Methode der Oberklasse wird durch die Methode der Unterklasse verdeckt, d.h. beim Aufruf in der Unterklasse wird die Methode der Unterklasse verwendet.

c) 3P

Bei der Vererbung: Der Datentyp der Unterklasse wird in den der Oberklasse konvertiert.
Sonst: Fehlermeldung des Compilers.

d) 1P

Wenn eine Methode weder in der gleichen Klasse, noch einer Unterklasse, noch im gleichen Package sind, dann kann auf diese Methode (im Gegensatz zu public) bei protected nicht zugegriffen werden.

2)

39P

```
public class Main_E2FI_8_4_08_nr1 {
    public static void main(String[] args){
        Erwerbstaetig e = new Erwerbstaetig("Schaffer", 4711, 1500);    // 2P
        Unbeschaeftigt u = new Unbeschaeftigt("Schroeder", 4712, 400); // 2P
        e.setGehalt(e.getGehalt()+100); // 3P
    }
}

class Person{ // 14P
    private String name;
    private int telefon;

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

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

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

    public int getTelefon(){
        return(telefon);
    }

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

```

class Unbeschaeftigt extends Person{ // 9P
    private int arbeitlosengeld ;

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

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

    public int getArbeitslosengeld(){
        return(arbeitslosengeld);
    }
}

class Erwerbstaetig extends Person{ // 9P
    private int gehalt ;

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

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

    public int getGehalt(){
        return(gehalt);
    }
}

```

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 folgenden Teilaufgaben müssen (soweit sie die Programmierung betreffen) in genau **einem** einzigen Programm realisiert werden.

Zur Information:

Eine zweispurige Strasse wurde auf einer Teilstrecke von einem Kilometer Länge durch einen Bergrutsch so beeinträchtigt, daß nur noch eine Spur zu befahren ist. Um den Verkehr zu steuern wurden 2 Ampeln aufgestellt, so daß die Teilstrecke abwechselnd immer wieder in eine Richtung und dann in die andere Richtung befahren werden kann.

Wenn die Ampelanlage gestartet wird, ist jedem Zeitpunkt auf jeder Ampel genau eine der 3 Farben rot, gelb oder grün zu sehen

Jede Ampel hat die 4 Zustände 1 (rot) , 2 (grün) , 12 (rot-gelb) 21 (grün-gelb), die bekanntermaßen nach folgender Vorschrift auf einander folgen:

1 (rot) --> 12 (rot-gelb) --> 2 (grün) --> 21 (grün-gelb) --> 1 (rot) --> usw.

Außerdem hat jede Ampel 3 Lampen, von denen jede die 4 Farben rot, gelb, grün oder schwarz annehmen kann, wobei schwarz eine ausgeschaltete Lampe bedeutet.

Der Zustand 1 bedeutet: rote Lampe an, restliche Lampen schwarz.

Der Zustand 2 bedeutet: grüne Lampe an, restliche Lampen schwarz.

Der Zustand 12 bedeutet: gelbe Lampe an, restliche Lampen schwarz.

Der Zustand 21 bedeutet: gelbe Lampe an, restliche Lampen schwarz.

Es gibt die 3 Klassen "Lampe", "Ampel" und "Steuerung".

1) 9P
Implementieren Sie die Klasse "Lampe" mit genau (und nur) dem Attribut farbe und dem Datentyp String. Zusätzlich müssen noch folgende Methoden erzeugt werden:

- a) ... getFarbe(...) :
gibt die Farbe einer Lampe zurück.
- b) ... einschalten(...) :
setzt eine Lampe auf eine bestimmte Farbe.
- c) ... ausschalten(...) :
schaltet eine Lampe aus
- d) einen Konstruktor der Klasse

2) 32P
Die Klasse Ampel hat als Attribute genau 3 Lampen und das Attribut "zustand". Implementieren Sie die Klasse "Ampel" mit genau (und nur) diesen Attributen. Zusätzlich müssen noch folgende Methoden erzeugt werden:

- a) ...setZustand(...) :
Beachten Sie, daß beim Setzen eines Zustand zusätzlich noch die zugehörige Farbe der entsprechenden Lampe eingeschaltet werden muß.
- b) ... getZustand(...) :
gibt den Zustand der Ampel zurück
- c) ...rotSchalten(...) :
schaltet die rote Lampe auf rot und die anderen aus.
- d) ... gelbSchalten(...) :
entsprechendes für die gelbe Lampe.
- e) ... grünSchalten(...) :
entsprechendes für die grüne Lampe.
- f) ... weiterSchalten(...) :
schaltet eine Ampel in den nächsten Zustand.
- g) einen Konstruktor der Klasse

3) 10P
Die Klasse Steuerung hat als Attribute genau 2 Ampeln. Implementieren Sie die Klasse "Steuerung" mit genau (und nur) diesen Attributen. Zusätzlich müssen noch folgende Methoden erzeugt werden:

- a) einen Konstruktor der Klasse
- b) ...start(...) :
beginnt den Steuerungsvorgang:
Ampel1 rot, Ampel2 grün --> Ampel1 gelb, Ampel2 gelb --> Ampel1 grün, Ampel2 rot --> Ampel1 gelb, Ampel2 gelb ---> usw.

Lösung:

```
class Lampe{ // 9P
    private String farbe; // 1P

    public String getFarbe(){ // 2P
        return farbe;
    }

    public void einschalten(String pFarbe){ // 2P
        farbe=pFarbe;
    }

    public void ausschalten(){ // 2P
        farbe="schwarz";
    }

    public Lampe(){ // 2P
        farbe="schwarz";
    }
}

class Ampel{ // 32P
    private Lampe grünLampe; // 1P
    private Lampe gelbLampe; // 1P
    private Lampe rotLampe; // 1P
    private int zustand; // 1P

    public Ampel(){ // 3P
        grünLampe = new Lampe();
        gelbLampe = new Lampe();
        rotLampe = new Lampe();
    }

    public void setZustand(int pZustand){ // 8P
        if(pZustand == 1){ // rot
            rotSchalten();
        }
        else if(pZustand == 12){ // rot-gelb
            gelbSchalten();
        }
        else if(pZustand == 2){ // grün
            grünSchalten();
        }
        else if(pZustand == 21){
            gelbSchalten();
        }
        else { // Programmierfehler
            zustand=-1;
            return;
        }
        zustand=pZustand;
    }
}
```

```

public int getZustand(){                                // 2P
    return zustand;
}

private void rotSchalten(){                             // 3P
    grünLampe.ausschalten();
    gelbLampe.ausschalten();
    rotLampe.einschalten("rot");
}

private void gelbSchalten(){                           // 3P
    grünLampe.ausschalten();
    gelbLampe.einschalten("gelb");
    rotLampe.ausschalten();
}

private void grünSchalten(){                          // 3P
    grünLampe.einschalten("grün");
    gelbLampe.ausschalten();
    rotLampe.ausschalten();
}

public void weiterSchalten(){                         // 6P
    if(zustand == 1){
        setZustand(12);
    }
    else if(zustand == 12){
        setZustand(2);
    }
    else if(zustand == 2){
        setZustand(21);
    }
    else if(zustand == 21){
        setZustand(1);
    }
    else { // Programmierfehler
        zustand=-1;
    }
}
}

```



```
class Steuerung{ // 10P
    private Ampel ampel1; // 1P
    private Ampel ampel2; // 1P

    public Steuerung(){ // 2P
        ampel1 = new Ampel();
        ampel2 = new Ampel();
    }

    public void start(){ // 6P
        ampel1.setZustand(1);
        ampel2.setZustand(2);

        while (true){
            ampel1.weiterSchalten();
            ampel2.weiterSchalten();
        }
    }

    public void stopp(){
        ampel1.setZustand(1);
        ampel2.setZustand(1);
    }
}
```

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)

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

1.4)

3P

Z.B. durch ein Feld

1.5)

10P

a)

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

b)

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

c)

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

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());
    }
}

```

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 einer Aufgabe (die aus Teilaufgaben besteht). Die gesamte Aufgabe muß als genau ein Programm in einem Projekt implementiert werden.

I) 51P

Zur Information:

Eine Teilstruktur einer Firma soll modelliert werden:

Eine Abteilung hat einen Namen (z.B. Elektroabteilung), besitzt einen Abteilungsleiter (z.B. die Person mit Namen Dofski), und besteht aus mehreren Personen.

Eine Person hat eine Identitätsnummer idNr und einen Namen.

1) 15P

Erstellen Sie die Klasse Person (mit genau den 2 Attributen idNr und name, genau 2 set-Methoden, genau 2 get-Methoden und genau einem Konstruktor mit 2 Parametern)
Zusätzlich muß noch genau die Methode printAllAttributes(...) erzeugt werden, die die Werte aller Attribute dieser Klasse auf dem Bildschirm ausgibt.

2) 24P

a) Erstellen Sie die Klasse Abteilung (mit genau den 3 Attributen "name", der Person "leiter" und dem Feld "diePersonen" und die zu name und leiter zugehörigen get - und set-Methoden und genau einem Konstruktor (mit genau 3 Parametern, wobei ein Parameter die Anzahl der Personen in der Abteilung festlegt).

b) Zusätzlich sollen genau noch folgende Methoden erstellt werden:

public void setPerson(Person pPerson, int index)
speichert eine Person an einer bestimmten Stelle des Feldes.

public Person getPerson(int index)
gibt die Person an einer bestimmten Stelle des Feldes zurück.

public void printAllAttributes()
Gibt die Attribute name, leiter und alle Personen der Abteilung auf dem Bildschirm aus.

3) 12P

Erstellen sie in `main()` eine Abteilung (der Variablennamen soll dieAbteilung heißen) mit dem Namen "Elektroabteilung", deren Leiter "Dofski" (Identitätsnummer 10) und den 2 Personen "Maier" mit Identitätsnummer 20 und "Maurer" mit Identitätsnummer 30.

Geben Sie mit genau einem einzigen Aufruf (einer bestimmten Methode) die Personen (mit Idenditätsnummer und Name) auf dem Bildschirm aus.

Lösung:

```
public class Startklasse {
    public static void main(String[] args) {
        Abteilung dieAbteilung = new Abteilung("Elektro", // 4P
            new Person(10,"Dofski"), 2);
        dieAbteilung.setPerson(new Person(20,"Maier"),0); // 3P
        dieAbteilung.setPerson(new Person(30,"Maurer"),1); // 3P
        dieAbteilung.printAllAttributes(); // 2P
    }
}

class Person {
    private int idNr; // 1P
    private String name; // 1P

    public Person(int idNr, String name) { // 2P
        this.idNr = idNr;
        this.name = name;
    }

    public int getIdNr() { // 2P
        return idNr;
    }

    public void setIdNr(int idNr) { // 2P
        this.idNr = idNr;
    }

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

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

    public void printAllAttributes(){ // 3P
        System.out.println("Personen-Id= " + idNr);
        System.out.println("Personen-Id= " + name);
    }
}
```

```

class Abteilung {
    private String name; // 1P
    private Person leiter; // 1P
    private Person[] ihrePersonen; // 1P

    public Abteilung(String abteilung, Person leiter, int anzahl){ //3P
        this.name=abteilung;
        this.leiter = leiter;
        ihrePersonen = new Person[anzahl];
    }

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

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

    public void setLeiter(Person pName) { // 2P
        leiter = pName;
    }

    public Person getLeiter() { // 2P
        return (leiter);
    }

    public void setPerson(Person pName, int index) { // 3P
        ihrePersonen[index] = pName;
    }

    public Person getPerson(int index) { // 3P
        return (ihrePersonen[index]);
    }

    public void printAllAttributes(){ // 4P
        System.out.println("Abteilungsname= " + name);
        System.out.println("Leiter= " + leiter);
        for(int i=0; i<ihrePersonen.length;i++) {
            ihrePersonen[i].printAllAttributes();
        }
    }
}

```