

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) 51P

1) 20P

a) 13P

Erzeugen Sie die Klasse Henne mit genau den 2 Attributen (und nur diesen Attributen) "name" und "legeleistung" und den zu diesen Attributen gehörigen get-und set-Methoden. Erzeugen Sie genau einen Konstruktor mit genau 2 Parametern.

b) 3P

Erzeugen Sie die Methode berechneTierwert(...), die den Tierwert einer Henne berechnet (der Tierwert berechnet sich aus dem 10-fachen seiner Legeleistung).

c) 4P

Erzeugen Sie die Methode print (...), die den Wert aller Attribute auf dem Bildschirm ausgibt.

Bemerkungen:

B1)

Außer den o.g. Methoden dürfen keinen weiteren Methoden in der Klasse Henne implementiert werden.

B2)

Ausgaben auf Bildschirm NIE in "normalen" Methoden machen, sondern nur in eigens dafür erstellten Ausgabe-Methoden realisieren!

In "normalen" Methoden müssen statt Bildschirmausgaben diese Werte über return zurückgeliefert werden oder in entsprechenden Attributen gespeichert werden.

2)

31P

Realisieren Sie programmtechnisch (mit Hilfe der entsprechenden Methode der Klasse Henne) folgende Aufgaben in der Methode main(..) der Startklasse:

Bemerkung:

Wenn in den folgenden Aufgaben sich der Wert eines Attributs ändert (z.B. Legeleistung) bzw. dann später verwendet wird, dann darf dieser neue Wert nicht selbst mit einem Taschenrechner (oder im Kopf) berechnet werden, sondern dies muß programmtechnisch (mit den entsprechenden Methoden) realisiert werden. Dies gilt natürlich auch für Attribute, deren Wert sich nicht ändern:

Es müssen immer die entsprechenden Methoden verwendet werden.

a) 2P

Erzeugen Sie eine Henne mit dem Namen Bert und der Legeleistung 10.

b) 2P

Erzeugen Sie eine Henne mit dem Namen Uta und der Legeleistung 5.

c) 2P

Geben Sie die Werte der Attribute der Henne namens Bert auf dem Bildschirm aus.

d) 2P

Geben Sie die Werte der Attribute der Henne namens Uta auf dem Bildschirm aus.

e) 2P

Geben Sie nur die Legeleistung der mit dem Namen Uta bezeichneten Henne auf dem Bildschirm aus.

f) 2P

Es wurde festgestellt, dass es sich um die namens Bert bezeichnete Henne um ein weibliches Tier handelt. Benennen Sie es in Berta um.

g) 2P

Berechnen Sie den Tierwert der mit dem Namen Uta bezeichneten Henne und speichern diesen in einer Variablen namens erg ab.

h) 3P

Die Legeleistung der Henna namens Berta hat sich um den Wert 1 erhöht.

i) 3P

Die Legeleistung der Henna namens Uta hat sich auf das Doppelte erhöht.

j) 3P

Der Besitzer der 2 Hennen will die gesamte Legeleistung der 2 Hennen berechnen.

Berechnen Sie die gesamte Legeleistung der 2 Hennen und geben diese dann auf dem Bildschirm aus.

k) 6P

Die Hennen mit dem Namen Berta und Uta wurden versehentlich vertauscht.

Bringen Sie das nachträglich programmtechnisch in Ordnung. D.h. die Namen und Legeleistungen der 2 Hennen müssen vertauscht werden.

l) Die Henne namens Uta wurde von einem Fuchs gefressen.

Lösungen:

}

```

public class Startklasse {
    public static void main(String[] args) {
        int summe;
        int erg;
        String name1;
        String name2;
        int legeleistung1;
        int legeleistung2;

        // a) 2P
        Henne h1 = new Henne("Berta", 10);
        // b) 2P
        Henne h2 = new Henne("Uta", 5);
        // c) 2P
        h1.print();
        // d) 2P
        h2.print();
        // e) 2P
        System.out.println("Legeleistung
                           Uta="+h2.getLegeleistung());

        // f) 2P
        h1.setName("Berta");
        // g) 2P
        erg= h2.berechneTierwert();
        // h) 3P
        h1.setLegeleistung(h1.getLegeleistung()+1);
        // i) 3P
        h2.setLegeleistung(h2.getLegeleistung()*2);
        // j) 3P
        summe=h1.getLegeleistung()+h2.getLegeleistung();
        System.out.println("Summe="+summe);
        // k) 6P
        name1=h1.getName();
        name2=h1.getName();
        legeleistung1=h1.getLegeleistung();
        legeleistung2=h2.getLegeleistung();
        h1.setName(name2);
        h1.setLegeleistung(legeleistung2);
        h2.setName(name1);
        h2.setLegeleistung(legeleistung1);
        // l) 2P
        h2 = null;
    }
}

```

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
Alle Teilaufgaben müssen in genau **einem** einzigen Programm realisiert werden.

1)
a) 35P
Erstellen Sie die Klasse Bruch, (mit genau den 2 Attributen zaehler und nenner, genau 2 set-Methoden, genau 2 get-Methoden und genau einem Konstruktor mit 2 Parametern), die eine mathematische Bruchzahl mit einem ganzzahligen Zähler und einem ganzzahligen Nenner repräsentieren soll.

Zusätzlich sollen genau noch folgende Methoden (und nur diese) erzeugt werden:

b) 2P
... printBruch (...)
gibt Zähler und Nenner in Bruchschreibweise auf den Bildschirm aus.
8 / 40

c) 4P
... kuerzen(int teiler)
kürzt den Bruch durch den wert teiler
Beispiel: 8 / 40 --- Teiler 4 --> 2 / 10

d) 4P
... alsKommazahl(...)
stellt den Bruch durch eine Kommazahl dar.
Beispiel: 1/2 ---> 0.5

e) 4P
... void quadrieren()
bildet das Quadrat eines Bruchs.
Beispiel: 2 / 3 ---> 4 / 9

f) 4P

... Bruch multiplizieren (Bruch bruch)

multipliziert 2 Brüche miteinander

Beispiel: $3/4 * 5/6 \rightarrow 15/24$

Tipp: $\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$

g) 4P

... Bruch kleiner Vergleich (Bruch bruch)

vergleicht 2 Brüche bzgl ihrer Größe miteinander und gibt den kleineren Bruch zurück.

Beispiel: $3/4, 2/3 \rightarrow 2/3$

2)

17P

Realisieren Sie programmtechnisch (mit Hilfe der entsprechenden Methode der Klasse Bruch) folgende Aufgaben in der Methode main(..) der Startklasse:

a) 3P

Legen Sie die Brüche b0 (Wert: 7/8), b1 (Wert: 2/3) und b2 (Wert : 4/5) an.

b) 2P

Geben Sie den Bruch b0 auf dem Bildschirm aus.

c) 3P

Speichern Sie die Kommazahl (mit der entsprechenden Methode) von b0 in der Variablen wert ab.

d) 3P

Berechnen Sie (mit der entsprechenden Methode) das Quadrat von b0.

Wo wird das Ergebnis gespeichert?

e) 3P

Berechnen Sie (mit den entsprechenden Methoden) $b1 * b2$ und speichern Sie das Ergebnis in der Variablen b4 (mit dem Datentyp Bruch) ab.

f) 3P

Berechnen Sie (mit den entsprechenden Methoden), ob b1 kleiner als b2 ist und speichern Sie den kleineren Bruch in das Ergebnis in der Variablen b5 (mit dem Datentyp Bruch) ab.

Lösungen:

1)

```
package bruch2;
public class Bruch2 {                                // gesamt: 17P
    public static void main(String[] args) {
        Bruch b0=new Bruch(7,8);                    // 1P
        Bruch b1=new Bruch(2,3);                    // 1P
        Bruch b2=new Bruch(4,5);                    // 1P
        Bruch b4;
        double wert;

        b0. printBruch();                            // 2P
        wert=b0.alsKommazahl();                      // 3P
        b0.quadrieren();                             // 3P
        b4=b1.multiplizieren(b2);                   // 3P
        b5=b1.kleinerVergleich (b2);                // 3P
    }
}

class Bruch{                                         // gesamt: 35P
    private int zaehler;                             // 1P
    private int nenner;                              // 1P

    public Bruch(int zaehler, int nenner) {          // 3P
        this.zaehler = zaehler;
        this.nenner = nenner;
    }

    public int getZaehler() {                        // 2P
        return zaehler;
    }

    public void setZaehler(int zaehler) {            // 2P
        this.zaehler = zaehler;
    }

    public int getNenner() {                         // 2P
        return nenner;
    }

    public void setNenner(int nenner) {              // 2P
        this.nenner = nenner;
    }

    public void printBruch(){                        // 2P
        System.out.print(zaehler+"/"+nenner);
    }

    public void kuerzen(int teiler) {                // 4P
        nenner = nenner/teiler;
        zaehler = zaehler/teiler;
    }

    public double alsKommazahl() {                   // 4P
        return ((double) zaehler/(double) nenner);
    }
}
```

```

public void quadrieren(){                                     // 4P
    zaehler=zaehler*zaehler;
    nenner = nenner * nenner;
}

public Bruch multiplizieren(Bruch bruch){                    // 4P
    int z, n;
    z=zaehler*bruch.zaehler;
    n=nenner*bruch.nenner;
    Bruch ergBruch=new Bruch(z,n);
    return ergBruch;
}

public Bruch kleinervergleich (Bruch bruch){                // 4P
    Bruch temp;
    if(this. alsKommazahl()<bruch. alsKommazahl()){
        temp=this;
    }
    else{
        temp=bruch;
    }
    return temp;
}
}

```


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 implementiert werden. Außer den u.g. Methoden dürfen **keine** weiteren Methoden in der Klasse Quader implementiert werden.

I) 54P

1) 40P

a) 22P

Erzeugen Sie die Klasse Quader mit genau den 3 Attributen (und nur diesen Attributen) "laenge" und "breite" und "hoehe" und den zu diesen Attributen gehörigen get-und set-Methoden.

Erzeugen Sie genau einen Konstruktor mit genau 3 Parametern.

Die entsprechenden Methoden müssen garantieren, dass keine negativen Werte für Länge, Breite und Höhe gesetzt werden. Sollte dies versucht werden, muss der entsprechende Wert auf 0 gesetzt werden.

b) 3P

Erzeugen Sie die Methode berechneVolumen (...), die das Volumen des Quaders berechnet.

c) 3P

Erzeugen Sie die Methode berechneOberflaeche(...), die die Oberfläche des Quaders berechnet.

d) 4P

Erzeugen Sie die Methode istWuerfel(...), die berechnet, ob der Quader ein Würfel ist oder nicht. Entsprechenden Rückgabebetyp wählen!

e) 3P

Erzeugen Sie die Methode verdoppeln (...), die jede Kante eines Quaders verdoppelt.

f) 5P

Erzeugen Sie die Methode vergleicheVolumen(...), die das Volumen zweier Quader vergleicht und true zurückgibt, wenn es gleich groß ist, sonst false.

2)

Realisieren Sie programmtechnisch (mit Hilfe der entsprechenden Methode der Klasse Quader) folgende Aufgaben in der Methode main(..) der Startklasse:

Bemerkung:

Wenn in den folgenden Aufgaben sich der Wert eines Attributs ändert (z.B. laenge) bzw. dann später verwendet wird, dann darf dieser neue Wert nicht selbst mit einem Taschenrechner (oder im Kopf) berechnet werden, sondern dies muß programmtechnisch (mit den entsprechenden Methoden) realisiert werden. Dies gilt natürlich auch für Attribute, deren Werte sich nicht ändern:

Es müssen immer die entsprechenden Methoden verwendet werden.

a) 1P

Erzeugen Sie einen Quader mit Variablennamen "quad1" der Länge=1,5 und der Breite=2,5 und der Höhe=3,5.

b) 1P

Erzeugen Sie einen anderen Quader mit Variablennamen "quad2" der Länge=4,5 und der Breite=5,5 und der Höhe=6,5.

c) 2P

Berechnen Sie das Volumen von "quad1" und speichern es in einer Variablen namens erg1 ab.

d) 2P

Berechnen Sie die Oberfläche von "quad2" und speichern es in einer Variablen namens erg2 ab.

e) 2P

Ist "quad1" ein Würfel ?

Speichern Sie das Ergebnis in einer Variablen namens erg3 ab.

f) 4P

Jede Kantenlänge des Quaders "quad1" wird mit Hilfe genau einer einzigen Methode verdoppelt.

Dann muss der Wert jedes Attributs von "quad1" in einer neuen Objektvariablen namens "quad3" abgespeichert werden (kurz "quad1" muss in "quad3" kopiert werden).

Dies muss mit genau einer einzigen Anweisung gemacht werden (die natürlich aus Teilanweisungen bestehen kann)

g) 2P

Vergleichen Sie das Volumen von "quad1" und "quad3" mit Hilfe genau einer einzigen Methode und speichern das Ergebnis in einer Variablen erg4 ab.

Lösungen:

```
public class Startklasse {
    public static void main(String[] args) {
        Quader quad1;
        Quader quad2;
        quad1=new Quader(1.5,2.5,3.5);           // a)    1P

        quad2=new Quader(4.5,5.5,6.5);           // b)    1P

        double erg1;                             // c)    2P
        erg1=quad1.berechneVolumen();

        double erg2;                             // d)    2P
        erg1=quad2.berechneOberflaeche();

        boolean erg3;                             // e)    2P
        erg3 = quad1.istWuerfel();

        quad1.verdoppeln();                       // f)    4P
        Quader quad3;
        quad3=new Quader(quad1.getLaenge(),
                        quad1.getBreite(), quad1.getHoehe());

        double erg4;                             // g)    2P
        erg4=quad1.vergleicheVolumen(quad3);
    }
}

class Quader {
    double laenge;                               // 1P
    double breite;                               // 1P
    double hoehe;                               // 1P

    public Quader(double laenge, double breite, double hoehe) { // 4P
        this.laenge = laenge;
        this.breite = breite;
        this.hoehe = hoehe;
        if(laenge<0){
            this.laenge=0;
        }
        if(breite<0){
            this.breite=0;
        }
        if(hoehe<0){
            this.hoehe=0;
        }
    }

    public double getLaenge() {                  // 2P
        return laenge;
    }

    public void setLaenge(double laenge) {       // 3P
        this.laenge = laenge;
        if(laenge<0){
            this.laenge=0;
        }
    }

    public double getBreite() {                  // 2P
        return breite;
    }

    public void setBreite(double breite) {       // 3P
        this.breite = breite;
        if(breite<0){
            this.breite=0;
        }
    }
}
```

```

public double getHoehe() { // 2P
    return hoehe;
}

public void setHoehe(double hoehe) { // 3P
    this.hoehe = hoehe;
    if(hoehe<0){
        this.hoehe=0;
    }
}

public double berechneVolumen() { // 3P
    return laenge * breite * hoehe;
}

public double berechneOberflaeche() { // 3P
    return 2 * (laenge * breite + laenge * hoehe + breite * hoehe);
}

public boolean istWuerfel() { // 4P
    boolean b;
    if (laenge == breite && laenge == hoehe && breite == hoehe) {
        b = true;
    } else {
        b = false;
    }
    return b;
}

public void verdoppeln(){ // 3P
    laenge=2*laenge;
    breite=2*breite;
    hoehe=2*hoehe;
}

public boolean vergleicheVolumen(Quader quad) { // 5P
    boolean erg;

    if(quad.berechneVolumen()==this.berechneVolumen()){
        erg=true;
    }
    else{
        erg=false;
    }
    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 Punkteabzug !!!!

AUFGABEN

I) 50P

Alle Teilaufgaben müssen in genau **einem** einzigen Programm realisiert werden.

1)

Zur Information über das Nimm-Spiel:

Es gibt zwei Spieler. Im Folgenden Spieler-Schwarz und Spieler-Weiss genannt.

Auf dem Tisch liegt ein Haufen von z.B. 50 Streichhölzern. (Anzahl kann vom Spieler-Weiss festgelegt werden).

Jeder Spieler muss eine Anzahl zwischen 1 und 3 Streichhölzern vom Tisch nehmen. Wer nicht mehr ziehen kann (weil kein Streichholz mehr da ist) hat verloren.

Spieler-Weiss beginnt zu ziehen.

a)

Erstellen Sie die Klasse Nimm mit den entsprechenden Attributen und Methoden.
(mit Hilfe der OOP).

b)

Spieler-Schwarz und Spieler-Weiss sollen gegeneinander spielen.

Implementieren Sie das zugehörige Programm in main() unter Verwendung der Methoden der Klasse Nimm.

Geben Sie jeweils den Gewinner bzw. Verlierer auf dem Bildschirm aus.

Bem:

Für Eingaben über Tastatur braucht nicht der Scanner verwendet werden. Es genügt eingabe(...) zu schreiben.

Beispiel:

```
int zahl;
```

```
eingabe(zahl);
```

Lösung:

```
package nimm_spiel1;
import java.util.Scanner;

public class Startklasse {
    public static void main(String[] args) {
        Nimm spiel=new Nimm(0);
        int anzahl;
        do{
            System.out.println("Spieler Weiss, gibt die Anzahl der Hoelzer
>0 vor");
            anzahl=spiel.eingabe();
        }
        while(anzahl<=0);
        spiel.setAnzahlHoelzer(anzahl);
        System.out.println("Es sind "+anzahl+" Hoelzer auf dem Tisch");

        do {
            spiel.macheNaechstenZug();
        } while(spiel.getAnzahlHoelzer()!=0);

        if(spiel.werIstAmZug()==0){
            System.out.println("Weiss hat verloren");
        }
        else{
            System.out.println("Schwarz hat verloren");
        }
    }
}

class Nimm{
    private int anzahlHoelzer;
    private int werIstAmZug;

    public Nimm(int anz){
        if(anz>0){
            anzahlHoelzer=anz;
        }
        else{
            anz=20;
        }
        werIstAmZug=0;
    }

    public void macheNaechstenZug(){
        int zahl;
        boolean erg;
        Scanner scanner=new Scanner(System.in);

        if(werIstAmZug==0){
            do{
                System.out.println("Spieler weiss, wieviel Hölzer nimmst du
weg?");
                System.out.println("Bitte Anzahl zwischen 1 und 3
eingeben");
                zahl = scanner.nextInt();
                erg=ziehen(zahl);
            }while (erg==false);
        }
        else{
            do{
                System.out.println("Spieler schwarz, wieviel Hölzer nimmst
du weg?");
                System.out.println("Bitte Anzahl zwischen 1 und 3
eingeben");
                zahl = scanner.nextInt();
                erg=ziehen(zahl);
            }while (erg==false);
        }
        System.out.println("Es sind noch "+anzahlHoelzer +" Hoelzer auf dem
Tisch");
    }
}
```

```

        bestimmeNaechstenSpieler();
    }

    public boolean ziehen(int anz){
        boolean erg=false;
        if(anzahlHoelzer-anz>=0 && anz>=1 && anz<=3){
            anzahlHoelzer=anzahlHoelzer-anz;
            erg=true;
        }
        return erg;
    }

    public void bestimmeNaechstenSpieler(){
        werIstAmZug=werIstAmZug+1;
        werIstAmZug=werIstAmZug%2;
    }

    public void setAnzahlHoelzer(int anz){
        if(anz>0){
            anzahlHoelzer=anz;
        }
    }

    public int getAnzahlHoelzer(){
        return anzahlHoelzer;
    }

    public int werIstAmZug(){
        return werIstAmZug;
    }

    public int eingabe(){
        int zahl;
        Scanner scanner=new Scanner(System.in);
        zahl = scanner.nextInt();
        return zahl;
    }
}

```

*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) 9P****1) 3P**

Ein Java-Programmierer bekommt die Aufgabe, ein Feld (kein dynamisches Feld) der Länge 5 zu erzeugen, in dem folgende Daten abgespeichert werden sollen:

2.0	'x'	3	"abcd"	3.14
-----	-----	---	--------	------

Beurteilen Sie diese Aufgabe bzgl. einer Implementierung (Realisierung) in Java.

2) 3P

Beurteilen Sie den folgenden Java-Programmausschnitt bzgl. Fehler bei der Syntax bzw. dem Laufzeitverhalten:

```
...
double i=0;
double j=9;
int[] v;
v = new int[10];
j = (j + i)/2;
v[j] = -3;
...
```

3) 3P

Beurteilen Sie den folgenden Java-Programmausschnitt bzgl. Fehler bei der Syntax bzw. dem Laufzeitverhalten:

```
...
int[] v;
int[] w;
v = new int[123];
w = new int[11];
v[122] = 11;
w[v[122]] = -4;
...
```


II)

43P

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 müssen in **einem** Programm realisiert werden

1)

16P

a) Erzeugen Sie nur die Klasse Rechteck (und keine andere) mit genau den Attributen (und nur diesen Attributen) "laenge" und "breite", den dazugehörigen get-und set-Methoden und einem Konstruktor (kein Standardkonstruktor).

Außerdem müssen die Methoden "... getUmfang(...)" und "...getFlaeche(...)" implementiert werden, die den Umfang bzw. die Fläche berechnen.

2)

27P

Realisieren Sie (mit Hilfe der entsprechenden Methode der Klasse Rechteck) Folgendes in der Methode main(..) der Startklasse:

a)

2P

Erstellen Sie das Feld "rechtecke" der Länge 100 mit dem Datentyp "Rechteck".

b)

2P

Erstellen Sie in dem Feld "rechtecke" in den ersten 50 Zellen Rechtecke mit gleichen Längen und Breiten (also Quadrate) von 0, 1, 2, ...49 Länge.

c)

5P

Verlängern Sie (nur mit Hilfe der get-bzw. set-Methoden) die Längen der obigen Rechtecke um den Wert +10 (man erzeugt also "echte" Rechtecke)

Bemerkung:

Die neue Länge darf nicht selbst mit einem Taschenrechner (oder im Kopf) berechnet werden, also, sondern dies muß programmtechnisch (mit den entsprechenden Methoden) realisiert werden.

d)

6P

Durch weitere Anweisungen wird eine weitere (Ihnen unbekannte Zahl) an Rechtecken in das feld "rechtecke" eingefügt.

Geben Sie die Umfänge aller Rechtecke - und deren Index (Stelle) im Feld - auf dem Bildschirm aus.

e)

6P

Kopieren Sie die ersten 50 Rechtecke hintereinander in das restliche Feld (also in die Zellen 50 bis 99)

f)

6P

Kopieren Sie die Längen aller Rechtecke in das zu erstellende double feld laengen
Kopieren Sie die Breiten aller Rechtecke in das zu erstellende double feld breiten

Lösungen:

I)

1)

3P

Syntaxfehler:

In einem Feld dürfen nur Werte des gleichen Datentyps abgespeichert werden.

2)

3P

Der Wert von j darf keine Fleißkommazahl sein.

3)

3P

v[122] hat den Wert 11.

Deshalb ist w[11] außerhalb ein Zugriff außerhalb der Feldgrenzen.

II)

16P

```
class Rechteck {  
    private double laenge;           // 1P  
    private double breite;           // 1P  
  
    public Rechteck(double plaenge, double pbreite){ // 2P  
        setLaenge(plaenge);  
        setBreite(pbreite);  
    }  
  
    public void setLaenge(double plaenge){ // 2P  
        laenge = plaenge;  
    }  
  
    public void setBreite(double pbreite){ // 2P  
        breite = pbreite;  
    }  
  
    public double getLaenge(){ // 2P  
        return (laenge);  
    }  
  
    public double getBreite(){ // 2P  
        return (breite);  
    }  
  
    public double getFlaeche(){ // 2P  
        return(laenge*breite);  
    }  
  
    public double getUmfang(){ // 2P  
        return(2*(laenge+breite));  
    }  
}
```

```

public static void main(String[] args) {
    int i;
    // a) // 2P
    Rechteck[] rechtecke;
    rechtecke=new Rechteck[100];
    // b) // 2P
    for(i=0;i<50;i++){
        rechtecke[i]=new Rechteck(i,i);
    }
    // c) // 5P
    for(i=0;i<50;i++){
        double tempLaenge;
        tempLaenge=rechtecke[i].getLaenge();
        rechtecke[i].setLaenge(tempLaenge+10);
        //oder alternativ;
        //rechtecke[i].setLaenge(rechtecke[i].getLaenge()+10);
    }
    // d) // 6P
    for(i=0;i<100;i++){
        double umfang;
        if(rechtecke[i]!=null){
            umfang=rechtecke[i].getUmfang();
            System.out.println("Umfang Rechteck["+i+"]="+umfang);
        }
    }
    // e) // 6P
    for(i=0;i<50;i++){
        rechtecke[i+50]=rechtecke[i];
    }
    // f) //6P
    double[] laengen=new double[100];
    double[] breiten=new double[100];
    for(i=0;i<100;i++){
        laengen[i]=rechtecke[i].getLaenge();
        breiten[i]=rechtecke[i].getBreite();
    }
}

```

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) 3 P
Wie werden programmtechnisch Beziehungen zwischen verschiedenen Objekten hergestellt?
Bitte kurze verbale Beschreibung.

2) 49P

In einem Planungsbüro hat ein Lehrling folgende Gesprächsschnipsel gehört:
Es soll ein Kiosk gebaut werden. Dieses soll einen Namen haben.
Zu diesem Kiosk gibt es genau ein Lager. In dem Lager sollen eine bestimmte Anzahl Bierdosen gelagert werden, die nur für dieses Kiosk reserviert sind. D.h:
Zu einem Kiosk gibt es genau ein Lager und zu einem Lager gibt es genau ein Kiosk.
Dieser Sachverhalt soll objektorientiert modelliert werden, wobei die Navigation in alle 2 Richtungen geht.
Machen Sie dazu Folgendes:

a) 10P

Zeichnen Sie dazu das passende UML mit den entsprechenden Attributen und ohne die Methoden (einschließlich Name der Assoziation mit Leserichtung und Kardinalitäten).

(Alle folgenden Teilaufgaben müssen in **einem** Programm realisiert werden)

b) 24P

Erstellen Sie die Klasse Kiosk und Lager (mit jeweils genau 2 Attributen, genau 2 set-Methoden, genau 2 get-Methoden und genau 1 Konstruktor).

c) 8P

Erzeugen Sie einen Kiosk mit dem Namen "standerl" und ein Lager, das einen Anfangsbestand von 1000 Bierdosen hat (einschließlich "Verlinkung").

d) 5P

Bei einer heftigen Zecherei hatten sich die dabei anwesenden Zeher 100 Bierdosen einverleibt.

Verändern Sie (**vom Kiosk ausgehend**) den entsprechenden Bierbestand des Lagers.

Bemerkung:

Dabei darf der neue Bierbestand nicht mit einem Taschenrechner (oder im Kopf) berechnet werden, also $1000-100$, sondern dies muß programmtechnisch mit Hilfe der Benutzung der gegebenen Methoden realisiert werden.

e) 2P

Geben Sie (**vom Lager ausgehend**) den neuen Bierbestand auf dem Bildschirm aus.

Siehe Bemerkung bei d)

Lösungen:

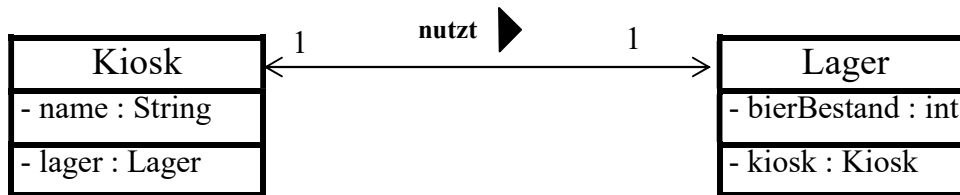
1)

3P

Ein Attribut der einen Klasse ist ein Objekt vom Typ der anderen Klasse

2a)

10P



```

package e2fi_7_1_2013_nr1;
public class MainE2FI_7_1_2013_nr1 {
    public static void main(String[] args) {
        Lager myLager = new Lager(1000); // 2P
        Kiosk myKiosk = new Kiosk("standerl"); // 2P
        myLager.setKiosk(myKiosk);
        myKiosk.setLager(myLager); // 4P

        int anzahl; // 1P
        anzahl = myKiosk.getLager().getBierBestand();
        myKiosk.getLager().setBierBestand(anzahl - 100); // 4P
        System.out.println("neuer Bierbestand=" +
                           myLager.getBierBestand()); // 2P
    }
}

class Kiosk {
    private String name; // 1P
    private Lager lager; // 1P

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

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

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

    public void setLager(Lager lager) { // 2P
        this.lager = lager;
    }

    public Lager getLager() { // 2P
        return lager;
    }
}
  
```

```
class Lager {
    private int bierBestand;           // 1P
    private Kiosk kiosk;               // 1P

    public Lager(int bierBestand) {    // 2P
        this.bierBestand = bierBestand;
    }

    public int getBierBestand() {      // 2P
        return bierBestand;
    }

    public void setBierBestand(int bierBestand) { // 2P
        this.bierBestand = bierBestand;
    }

    public void setKiosk(Kiosk kiosk) { // 2P
        this.kiosk = kiosk;
    }

    public Kiosk getKiosk() {          // 2P
        return kiosk;
    }
}
```