

*Name, Vorname:*Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß auf das Aufgabenblatt 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.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN**I) 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

Beurteilen Sie den folgenden Java-Programmausschnitt bzgl. Syntax und Laufzeitverhalten:

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

Bem:

Es dürfen keine Klassen wie z.B. ArrayList, Vector, usw. der Java-Entwicklungsumgebung verwendet werden.

1) 50P

Erstellen Sie die Klasse Feldverwaltung mit genau den folgenden Attributen

```
private int[] feld;  
// Länge des Feldes (Menge des dafür reservierten Speichers)  
private int maxLen;  
// Stelle (Index) des letzten Elements im Feld.  
// Mit jedem angefügten Element wird dieser Wert um 1 erhöht und  
// letztesElement darf maximal den Wert maxLen-1 bekommen  
private int letztesElement;
```

Diese Klasse muß zusätzlich noch folgende Methoden enthalten:

(Diese Methoden müssen verhindern, daß eine Zahl in das Feld eingefügt wird, wenn das Feld voll ist, d.h. wenn `letztesElement = maxLen-1`)

1.1)

```
public Feldverwaltung(int maxLen)
```

Konstruktor, mit dem ein Feld einer bestimmten maximalen Länge erzeugt wird.

1.2)

```
public void anfüegen(int zahl)
```

Fügt an das Feldende das Element "zahl" an

1.3)

```
public void entferneLetztesElement()
```

Enfernt das letzte Element des Feldes

1.4)

```
public void einfügenAnPosition(int pos, int zahl)
```

Fügt rechts von der Stelle pos das Element zahl ein und verschiebt vorher die bis sich jetzt davon rechts befindlichen Elemente jeweils um eine Stelle nach rechts

1.5)

```
public void einfügenSortieren(int zahl)
```

Fügt das Element zahl rechts von der Stelle ein, wo zahl das 1. Mal vom Wert her größer ist als das dort sich befindliche Feldelement.

1.6)

```
public void löscheAnPosition(int pos)
```

Löscht das Feldelement an der Stelle pos.

Das Feld hat deshalb danach ein Feldelement weniger.

1.7)

```
public void printFeld()
```

Gibt alle Zahlen (mit der 0. Stelle beginnend) des Feldes auf dem Bildschirm aus.

2)

Folgendes soll in main realisiert werden:

2.1)

Erstellen Sie ein Objekt der Klasse Feldverwaltung, mit dem ein Feld der Länge 10 erzeugt wird.

2.2)

Mit Hilfe einer Schleife und der Methode anfüegen(...) sollen die Zahlen 100, 101, 102, ... , 119 in das Feld eingefügt werden.

2.3)

Danach sollen diese Zahlen mit Hilfe der entsprechenden Methode auf dem Bildschirm ausgegeben 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

I)

1)

22P

Entwickeln Sie ein Programm, das den Umfang und den Flächeninhalt von Kreisen berechnet. Das Programmdesign und die Lösung soll **objektorientiert** sein.

$$U = 2\pi r \quad \text{und} \quad A = \pi r^2$$

Bemerkung:

In der Klasse Kreis dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen.

a) 22P

Erzeugen Sie die Klasse Kreis mit genau dem Attribut (und nur dem Attribut) "radius" und genau (und nur) den folgenden Methoden:
den zu "radius" gehörigen get-und set-Methoden, 2 Konstruktoren und den Methoden berechneFlaeche und berechneUmfang.

2)

28P

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

a) 4P

Erzeugen Sie in main zwei Kreise:
k1 mit Radius 2 und k2 mit Radius 10.

b) 6P

Geben Sie die Daten (Radius, Umfang, Fläche) von k2 (durch Verwendung der entsprechende(n) Methoden) auf dem Bildschirm aus.

c) 6P

Zwischenzeitlich wurde der Radius von k1 auf einen unbekannten Wert verändert.

Dieser Wert des Objekts k1 soll nun ausgelesen werden und der ausgelesene Wert verdoppelt und in dem Objekt k2 gespeichert werden. Dies soll in genau **einer** Anweisung geschehen.

d) 6P

Sie bekommen die Erlaubnis zusätzlich zu den o.g. Methoden die Methode

...setUmfang(double pUmfang)

in der Klasse Kreis (aber nur mit Hilfe der o.g. Methoden) zu implementieren.

Implementieren Sie in der Klasse Kreis die Methode

...setUmfang(double pUmfang)

Tipp:

$U = 2\pi r$ nach r umgeformt ergibt: $r = U / (2\pi)$

e) 6P

Berechnen Sie (durch Verwendung der entsprechende(n) Methoden), um das Wievielfache sich dann der Flächeninhalt erhöht hat und geben Sie dieses Ergebnis auf dem Bildschirm aus.

Lösung:

```
1)
class Kreis{
    private double radius;                // 2P
    public static final double pi = 3.14;

    public Kreis(double pRadius){        // 2P
        radius = pRadius;
    }

    public Kreis(){                      // 2P
    }

    public void setRadius(double pRadius){ // 4P
        radius = pRadius;
    }

    public double getRadius(){           // 4P
        return(radius);
    }

    public double berechneUmfang(){      // 4P
        return(2*pi*radius);
    }

    public double berechneFlaeche(){     // 4P
        return(pi*radius*radius);
    }
}

2)
public class MainTest4 {
    public static void main(String[] args){
        double verhaeltnis;
        // Teilaufgabe a)
        Kreis k1 = new Kreis(2);        // 2P
        Kreis k2 = new Kreis(10);       // 2P

        // Teilaufgabe b)
        System.out.println("k2.getRadius()= "+k2.getRadius()); // 2P
        System.out.println("k2.berechneUmfang()="+k2.berechneUmfang()); //2P
        System.out.println("k2.berechneFlaeche()="+k2.berechneFlaeche()); //2P

        // Teilaufgabe c)
        k2.setRadius(k1.getRadius()*2); // 6P

        // Teilaufgabe d)
        public void setUmfang(double pUmfang){ // 6P
            setRadius(pUmfang/(2*pi));
        }

        // Teilaufgabe e)
        verhaeltnis = k2.berechneFlaeche()/k1.berechneFlaeche(); // 4P
        System.out.println("verhaeltnis= "+verhaeltnis); // 2P
    }
}
```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
- In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punktabzug !!!!

AUFGABEN

Bem:

Es dürfen keine Klassen wie z.B. ArrayList, Vector, usw. der Java-Entwicklungsumgebung verwendet werden.

1) 50P

Erstellen Sie die Klasse Feldverwaltung mit genau den folgenden Attributen

```
private int[] feld;  
// Länge des Feldes (Menge des dafür reservierten Speichers)  
private int maxLen;  
// Stelle (Index) des letzten Elements im Feld.  
// Mit jedem angefügten Element wird dieser Wert um 1 erhöht und  
// letztesElement darf maximal den Wert maxLen-1 bekommen  
private int letztesElement;
```

Diese Klasse muß zusätzlich noch folgende Methoden enthalten:

(Diese Methoden müssen verhindern, daß eine Zahl in das Feld eingefügt wird, wenn das Feld voll ist, d.h. wenn `letztesElement = maxLen-1`)

1.1)

```
public Feldverwaltung(int maxLen)
```

Konstruktor, mit dem ein Feld einer bestimmten maximalen Länge erzeugt wird.

1.2)

```
public void anfüegen(int zahl)
```

Fügt an das Feldende das Element "zahl" an

1.3)

```
public void entferneLetztesElement()
```

Enfernt das letzte Element des Feldes

1.4)

```
public void einfügenAnPosition(int pos, int zahl)
```

Fügt rechts von der Stelle pos das Element zahl ein und verschiebt vorher die bis sich jetzt davon rechts befindlichen Elemente jeweils um eine Stelle nach rechts

1.5)

```
public void einfügenSortieren(int zahl)
```

Fügt das Element zahl rechts von der Stelle ein, wo zahl das 1. Mal vom Wert her größer ist als das dort sich befindliche Feldelement.

1.6)

```
public void löscheAnPosition(int pos)
```

Löscht das Feldelement an der Stelle pos.

Das Feld hat deshalb danach ein Feldelement weniger.

1.7)

```
public void printFeld()
```

Gibt alle Zahlen (mit der 0. Stelle beginnend) des Feldes auf dem Bildschirm aus.

2)

Folgendes soll in main realisiert werden:

2.1)

Erstellen Sie ein Objekt der Klasse Feldverwaltung, mit dem ein Feld der Länge 10 erzeugt wird.

2.2)

Mit Hilfe einer Schleife und der Methode anfüegen(...) sollen die Zahlen 100, 101, 102, ... , 119 in das Feld eingefügt werden.

2.3)

Danach sollen diese Zahlen mit Hilfe der entsprechenden Methode auf dem Bildschirm ausgegeben werden.

Lösungen:

```
package e2fi_nachtermin_jan_2018;
public class Startklasse {
    public static void main(String[] args) {
        int j;
        int zahl=100;
        Feldverwaltung fv = new Feldverwaltung(10);
        for(j=0;j<=9;j++){
            fv.einfügenSortieren(100+j);
        }
        fv.printFeld();
    }
}

class Feldverwaltung{
    private int[] feld;
    private int letztesElement;
    private int maxLen;

    public Feldverwaltung(int maxLen){
        this.maxLen = maxLen;
        letztesElement = -1;
        feld = new int[maxLen];
    }

    public void anfüegen(int zahl){
        if(letztesElement<=maxLen-2){
            letztesElement++;
            feld[letztesElement]=zahl;
        }
    }

    public void entferneLetztesElement(){
        if(letztesElement>=0){
            letztesElement--;
        }
    }

    public void einfügenAnPosition(int pos, int zahl){
        if(letztesElement<=maxLen-2){
            // Am Feldeende
            if(letztesElement==pos){
                anfüegen(zahl);
            }
            else{
                for(int i=letztesElement;i>=pos+1;i--){
                    feld[i+1]=feld[i];
                }
                letztesElement++;
                feld[pos+1]=zahl;
            }
        }
    }
}
```

```

public void einfuegenSortieren(int zahl){
    int pos=-1;
    int i=0;
    // Feld ist leer
    if(letztesElement==-1){
        anfuegen(zahl);
    }
    else{
        // pos bestimmen
        while(i<=letztesElement){
            if(feld[i]>zahl){
                pos=i;
            }
            i++;
        }
        if(pos>=0){
            einfuegenAnPosition(pos, zahl);
        }
    }
}

public void löscheAnPosition(int pos){
    if(pos>=0 && pos<=letztesElement){
        for(int i=pos; i<letztesElement; i++){
            feld[i]=feld[i+1];
        }
        letztesElement--;
    }
}

public void printFeld(){
    int i;
    System.out.print("Feld=");
    for(i=0;i<=letztesElement;i++){
        System.out.print(feld[i]+" ");
    }
    System.out.println("");
}
}

```

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:

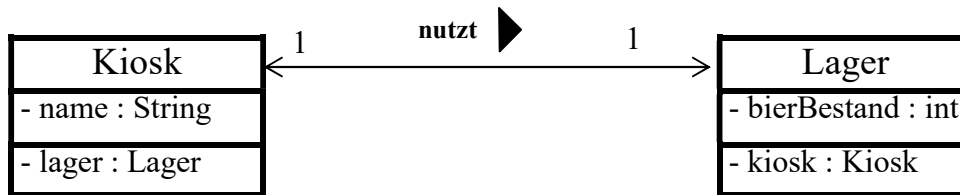
1)

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;
    }
}
```

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 nötigen Attributen und Methoden.

3) 9P
Implementieren Sie die Klasse "Besitzer" mit den nötigen 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:

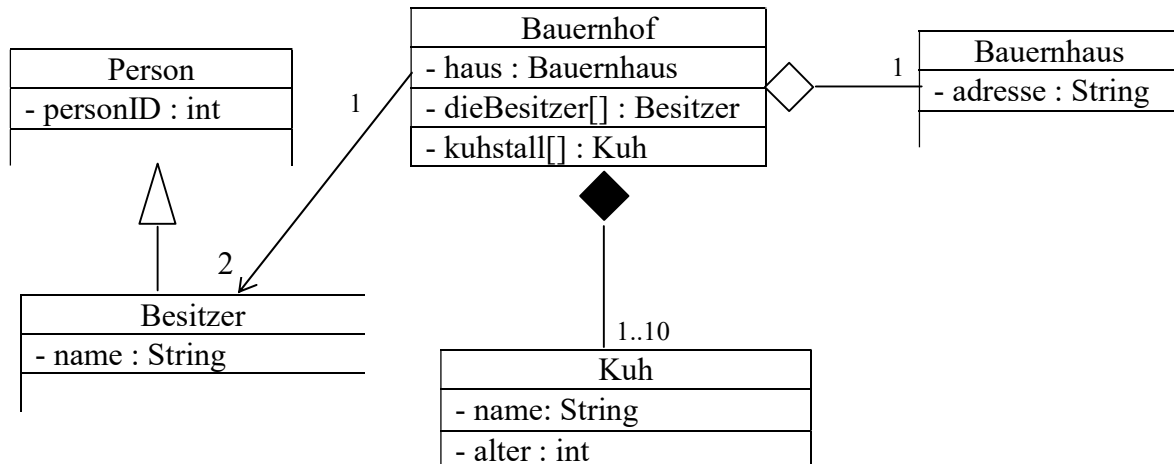
In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe-Methoden.

Alle Teilaufgaben - sofern es sich um Programmieraufgaben handelt - müssen in **einem** Programm realisiert werden

Lösung:

1)

16P



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

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)

19P

```

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 insertKuh(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());
            }
        }
    }
}

```