

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) 14P

- a) Was ist eine Klasse ?
- b) Was ist ein Objekt?
- c) Was enthält eine Klasse?
- d) Was ist der Sinn einer Methode?
- e) Was ist ein Attribut?
- f) Was bedeutet private?
- g) Was bedeutet public?

2) 10P

- a) 2P

Wann und wie wird ein Konstruktor aufgerufen?

Bitte Beispiel geben (nur Aufruf angeben, keine Klasse implementieren).

- b) 3P

Herr X behauptet : "Man braucht keine set-Methoden, da der Konstruktor genau das gleiche machen kann". Nehmen Sie dazu Stellung.

- c) 3P

Angenommen, ein Programmierer hat keinen Konstruktor erzeugt.

Wann gibt es eine Fehlermeldung beim Kompilieren?

- d) 2P

Was geschieht, wenn die Attribute einer Klasse nicht initialisiert werden und ein Objekt erzeugt wird? (Fehlermeldung?, Wert der Attribute ?, ...). Bitte genaue Beschreibung.

Auf der Rückseite geht es weiter !!!!!!!

3) (Alle Teilaufgaben müssen in **einem** Programm realisiert werden) 28P

a) Erstellen Sie die Klasse Konto, (mit genau 2 Attributen, genau 2 set-Methoden, genau 2 get-Methoden und genau 2 Konstruktoren), die ein Sparkonto mit einem Kontostand und einem Zinssatz repräsentieren soll.

b) Erzeugen Sie ein Konto mit dem Kontostand von 5 (Euro) und einem Zinssatz von 3%

c) Angenommen, man kennt nicht den aktuellen Kontostand und den Zinssatz .

Verändern Sie (nur mit Hilfe der get-bzw. set-Methoden) den Kontostand um den Wert +1000 (Euro) und den Zinssatz um -2 Prozentpunkte.

d) Ermitteln Sie mit Hilfe der entsprechenden Methoden den neuen Kontostand und den neuen Zinssatz und geben diese auf dem Bildschirm aus.

e) Erzeugen Sie ein anderes, neues Konto mit dem Kontostand von 10 (Euro) und einem Zinssatz von 5%.

Lösung:

1) 14 Punkte (jede Teilaufgabe 2 Punkte)

- a) Ein Bauplan, nach dem ein Objekt erstellt wird
- b) Ein nach einem Bauplan im Arbeitsspeicher angelegter, reservierter Speicher.
- c) Attribute und Methoden.
- d) Attribute zu lesen und zu beschreiben.
- e) Daten (Eigenschaften) in einer Klasse.
- f) Auf private Member darf nur innerhalb einer Klasse zugegriffen werden.
- g) Auf public Member darf innerhalb und außerhalb einer Klasse zugegriffen werden.

2) 10 Punkte

a) 2P

Ein Konstruktor wird aufgerufen, wenn ein Objekt mit new erzeugt wird.

Hund hund1 = new Hund("rex",45);

b) 3P

Diese Ansicht ist falsch:

Mit einem Konstruktor kann man nur einmal (beim Erzeugen des Objekts) die Attribute dieses Objekts festlegen. Will man diese später verändern, braucht man eine set-Methode.

c) 3P

Wenn der Programmierer einen Konstruktor mit mindestens einem Parameter erstellt und damit ein dazu entsprechendes Objekt erzeugt.

d) 2P

Es gibt keine Fehlermeldung, denn die Attribute des angelegten Objekts werden standardmäßig mit 0 vorgelegt.

3)

```
public class MainKonto1 {
    public static void main(String[] args) {
        Konto k1, k2;
        // 2 P
        k1= new Konto(5,3);
        // 6 P
        k1.setKontostand(k1.getKontostand()+1000);
        k1.setZinssatz(k1.getZinssatz()-2);
        // 4 P
        System.out.println("neuer Kontostand="
                           "+k1.getKontostand());
        System.out.println("neuer Zinssatz="
                           "+k1.getZinssatz());

        // 2 P
        k2= new Konto(10,5);
    }
}
```

```
class Konto{
    // 2 Punkte
    private double kontostand;
    private double zinssatz;

    // 2 Punkte
    public Konto(){
        kontostand = 0;
        zinssatz = 0;
    }

    // 2 Punkte
    public Konto(double pKontostand, double pZinssatz){
        kontostand = pKontostand;
        zinssatz = pZinssatz;
    }

    // 2 Punkte
    public double getKontostand() {
        return kontostand;
    }

    // 2 Punkte
    public double getZinssatz() {
        return zinssatz;
    }

    // 2 Punkte
    public void setKontostand(double kontostand) {
        this.kontostand = kontostand;
    }

    // 2 Punkte
    public void setZinssatz(double zinssatz) {
        this.zinssatz = zinssatz;
    }
}
```

KLAUSUR 1 SAE E2FI Nachtermin Zeit: 70 Minuten

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) 4P
Was ist der Unterschied zwischen einer Klasse und einem Objekt?

2) 10P

a)

Welche Werte haben die folgenden Variablen in der Methode main ? (Begründen Sie)
h1 , i , h2 , alter von h2

```
public class Startklasse{
    public static void main(String[] args) {
        Hund h1;                // <---
        int i;                   // <---
        Hund h2=new Hund ();    // <---
    }
}

class Hund{
    private int alter;
    public Hund(int pAlter){
    }
}
```

b)

Würde eine Fehlermeldung erzeugt werden, wenn der Konstruktor in der Klasse Hund entfernt werden würde? Begründen Sie.

3) 37P

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

Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.

a) 12P

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) 3P

... kuerzen(int teiler)

kürzt den Bruch durch den wert teiler

Beispiel: 8 / 40 --- Teiler 4 --> 2 / 10

c) 3P

... alsKommazahl(...)

stellt den Bruch durch eine Kommazahl dar.

Beispiel: 1/2 ---> 0.5

d) 3P

... void quadrieren()

bildet das Quadrat eines Bruchs.

Beispiel: 2 / 3 ---> 4 / 9

e) 5P

... Bruch multiplizieren(Bruch bruch)

multipliziert 2 Brüche miteinander

Beispiel: 3 / 4 * 5 / 6 ---> 15 / 24

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

f) 3P

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

g) 3P

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

h) 2P

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

Wo wird das Ergebnis gespeichert?

i) 3P

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

Lösungen:

1) 4P
Eine Klasse ist ein Bauplan, nachdem ein Objekt gebastelt und im Arbeitsspeicher angelegt wird.

2) 10P
a) 8P
h1 , i sind undefiniert (dem Programmierer unbekannt)
h2 zeigt auf ein Objekt im Arbeitsspeicher, alter von h2 hat den Wert 0

b) 2P
Nein, da der Konstruktor automatisch vom Compiler angelegt wird.

3)

```
package bruch2;
public class Bruch2 {
    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;

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

class Bruch{
    private int zaehler;                   // 1P
    private int nenner;                    // 1P

    public Bruch(int zaehler, int nenner) { // 2P
        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 kuerzen(int teiler) { // 3P
    nenner = nenner/teiler;
    zaehler = zaehler/teiler;
}

public double alsKommazahl() { // 3P
    return ((double) zaehler / (double) nenner);
}

public void quadrieren() { // 3P
    zaehler=zaehler*zaehler;
    nenner = nenner * nenner;
}

public Bruch multiplizieren(Bruch bruch) { // 5P
    int z, n;
    z=zaehler*bruch.zaehler;
    n=nenner*bruch.nenner;
    Bruch ergBruch=new Bruch(z,n);
    return ergBruch;
}

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

```


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

1a) 2P

Was ist falsch an folgendem Java-Programmausschnitt? (mit Begründung!)

```
...  
int[] v;  
v = new int[3];  
v[3] = -3;  
...
```

b) 2P

Was ist falsch an folgendem Java-Programmausschnitt? (mit Begründung!)

```
...  
double[] w, zahlen;  
zahlen[0] = 12;  
...
```

c) 2P

Ist der folgende Java-Programmausschnitt syntaktisch korrekt?

Wenn ja, welche Werte haben die Elemente der Variablen w?

```
...  
int[] w;  
w = new int[2];  
w[0] = -123;  
...
```

2) 4 P

Die Klasse Schaf soll schon existieren (mit genau einem zweiparametrischen Konstruktor aus integer Parametern, die das Gewicht und das Alter festlegen).

Erstellen Sie das Feld "schafstall" mit 2 Schafen.

Das erste Schaf wiegt 20 Kilo und ist 2 Jahre alt,

das zweite Schaf wiegt 30 Kilo und ist 3 Jahre alt.

3)

a)

18P

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

Erzeugen Sie die Methode berechneFlaeche und berechneUmfang.

b)

22P

Machen Sie in main() hintereinander Folgendes

b1) Erstellen Sie in dem Feld "kreise" 100 Kreise mit den Radien 0, 1, 2, ...99

b2) Geben Sie die Flächeninhalte dieser Kreise auf dem Bildschirm aus.

b3) Verdoppeln Sie die Radien dieser Kreise

b4) Speichern Sie diese Kreise (mit doppeltem Radius) zur Sicherheit in einem neuen Feld mit dem Namen "kreisSicherung".

b5) Angenommen, jemand fügt an das Ende in main()die 2 folgenden Programmierzeilen:

```
kreisSicherung[10].setRadius(13);  
System.out.println("Radius =" + kreise[10].getRadius());
```

Was wird auf dem Bildschirm ausgegeben?

Begründen Sie!

Bemerkung:

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.

Lösungen:

1a) 2P

`v[3] = -3;` überschreibt nicht reservierten Speicher

1b) 2P

Für das Feld `zahlen` wurde kein Speicher reserviert.

`zahlen` ist eine lokale Variable, deren Wert undefiniert ist.

c) 2P

Die Werte des Felds `w` sind nach dem Erstellen mit 0 vorbelegt, `w[1]` wurde verändert, also:

`w[0] = -123`

`w[1] = 0`

2) 4P

`Schaf [] schafstall = {new Schaf (20, 2), new Schaf (30, 3)};`

3) a)

```
class Kreis{
    private double radius;                // 2P
    // pi=3,14.. gilt auch außerhalb der Klasse Kreis,
    // deshalb keine schöne Lösung !!
    public static final double pi = 3.14;

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

    public Kreis(){                       // 2P
    }

    public void setRadius(double pRadius){ // 3P
        radius = pRadius;
    }

    public double getRadius(){            // 3P
        return(radius);
    }

    public double berechneUmfang(){       // 3P
        return(2*pi*radius);
    }

    public double berechneFlaeche(){      // 3P
        return(pi*radius*radius);
    }
}
```

```

public class MainBK11_13_11_10_Nr3 {
    public static void main(String[] args) {
        int i;
        // b1
        Kreis[] kreise; //1P
        kreise = new Kreis[100]; //1P
        for(i=0;i<100;i++){ //4P
            kreise[i]=new Kreis(i);
        }

        // b2
        for(i=0;i<100;i++){ //4P
            System.out.println("Flaeche="+kreise[i].berechneFlaeche());
        }

        // b3
        for(i=0;i<100;i++){ //4P
            kreise[i].setRadius(kreise[i].getRadius()*2);
        }

        // b4
        Kreis[] kreisSicherung = new Kreis[100]; //2P
        kreisSicherung=kreise;

        //alternativ
        for(i=0;i<100;i++){ //4P
            kreisSicherung[i]=kreise[i];
        }

        // b5)
        // Radius=13 //1P
        // kreisSicherung[10] und kreise [10] zeigen auf das //1P
        // gleiche Objekt.
    }
}

```

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

Bem:

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

1) 50P

Erstellen Sie die Klasse DynamischerSpeicher mit genau den folgenden Attributen (und den entsprechenden get und set-Methoden)

```
private int[] feld;  
private int len;
```

Diese Klasse muß zusätzlich noch folgende Methoden enthalten:

1.1) public void anfüegen(int zahl) :

fügt eine Zahl an das Ende des Feldes an.

Dies soll dadurch realisiert werden, daß ein neues Feld erstellt wird (mit einer um 1 größeren Länge als das alte Feld). Die Zahl muß an das Ende des neuen Feldes angefügt werden.

Der Inhalt des alten Feldes muß dann noch in das neue Feld kopiert werden. Dann wird das neue Feld dem alten Feld zugewiesen.

1.2) public void printFeld() :

gibt alle Elemente des Feldes auf dem Bildschirm aus.

1.3) In main() soll mit Hilfe einer Schleife und der Methode anfüegen(...) die Zahlen 100, 101, 102, ... , 120 in das Feld eingefügt werden.

Danach sollen diese Zahlen auf dem Bildschirm ausgegeben werden.

1.4) schwierigere Zusatzaufgaben:

Erstellen Sie die Methoden, die folgendes leisten:

1.4.1) public void entferneLetztesElement() :

entfernt das letzte Element des Feldes, wobei die Länge des Feldes angepaßt (um 1 verringert) wird.

1.4.2) public void einfuegenAnPosition (int pos) :

einfügen eines Elements an der Stelle pos i des Feldes (mit Längenangepaßung).

1.4.3) `public void entferneAnPosition (int pos) :`
entfernen eines Elements an der Stelle pos des Feldes (mit Längenanpaßung).

1.4.4) `public void einfuegen(int zahl) :`
die Zahl zahl wird so eingefügt (mit Längenanpaßung), daß nach jedem Einfügen das Feld in aufsteigender Reihenfolge sortiert ist (dabei keine von der Java-Entwicklungsumgebung vorgegebene Sortiermethode verwenden).

1.4.5) `public void entferneZahl (int zahl) :`
wenn die Zahl zahl das erste Mal im Feld vorkommt, wird sie entfernt (mit Längenanpaßung).
Falls sie nicht vorkommt, passiert nichts.

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 letztesElement;  
// 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 maxLen;
```

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

1) 3 P

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

2) 47P

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) 8P

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 Zecher 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)

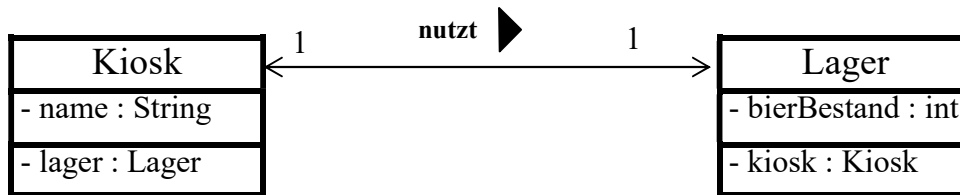
Lösungen:

1)
Ein Attribut der einen Klasse ist ein Objekt vom Typ der anderen Klasse

3P

2a)

8P



```

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 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.

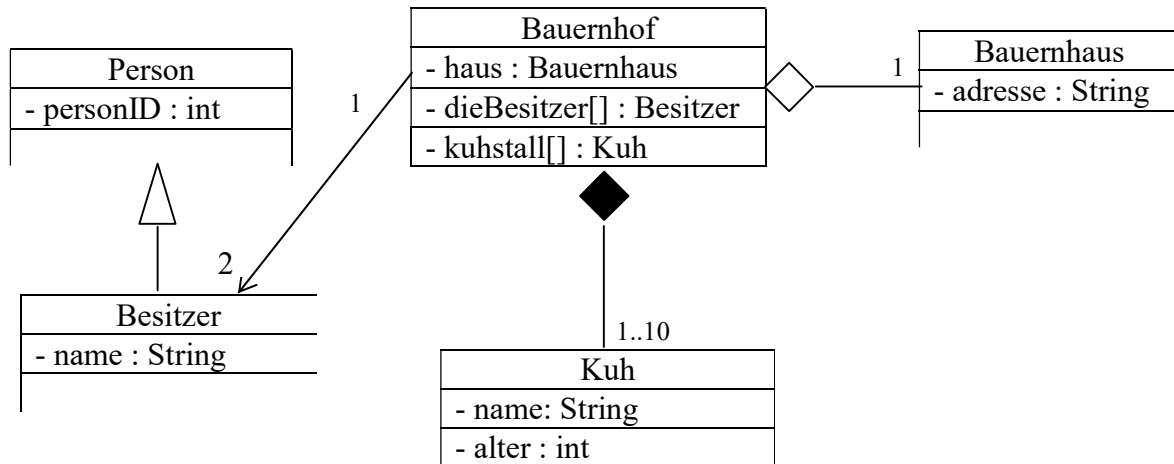
Bemerkung:

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());
            }
        }
    }
}
```