

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

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

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 vorbelegt.

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;
    }
}
```

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.

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

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", den den zu "radius" gehörigen get-und set-Methoden und 2 Konstruktoren. Erzeugen Sie die Methode berechneFlaeche und berechneUmfang.

2)

28P

Realisieren Sie (mit Hilfe der entsprechenden Methode der Klasse Rechteck) 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

Aus Testgründen wird der über Tastatur eingegebene Radius des Objekts k1 verdoppelt und in dem Objekt k2 gespeichert. Dies soll in **einem** Ausdruck geschehen.

d) 6P

Geben Sie die Daten (Radius, Umfang, Fläche) von k_2 (durch Verwendung der entsprechende(n) Methoden) nach der Verdopplung aus.

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)
        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 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

1) 51P
Ein Autor schreibt Bücher.

1.1) 7P

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

1.2)

a) 7P

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

b) 4P

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

1.3)

Zu einem Autor soll es genau ein Buch geben.

a) 8P

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

Der Konstruktor von Autor darf nicht verändert werden.

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

b) 8P

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

c) 4P

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

1.4) 3P

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

Verbale Beschreibung, kein Java-Anweisungen.

1.5)

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

Modellieren Sie diesen Fall wie folgt:

a) 4P

Ändern Sie dazu den Konstruktor der Klasse Autor.

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

b) 2P

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

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

c) 4P

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

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

Lösungen:

1.1)

7P

```
class Autor{
    private String name;                // 1P

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

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

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

1.2)

11P

```
class Buch{
    private String titel;               // 1P

    public Buch(String pTitel){         // 2P
        titel=pTitel;
    }

    public String getTitel() {           // 2P
        return titel;
    }

    public void setTitel(String titel) { // 2P
        this.titel = titel;
    }
}
```

1.2 b)

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

1.3)

20P

a)

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

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

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

b)

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

c)

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

1.4)

3P

Z.B. durch ein Feld

1.5)

10P

a)

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

b)

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

c)

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

Name, Vorname:

Hilfsmittel:

keine

Hinweise (unbedingt beachten):

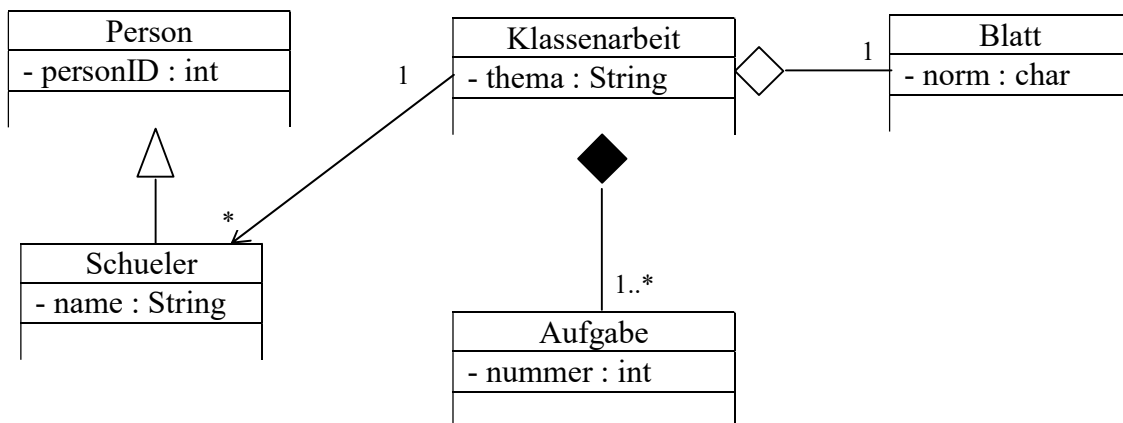
- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote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

AUFGABEN

I)

50P

Gegeben ist das folgende unvollständige UML-Diagramm (nur die Attribute der Klasse Klassenarbeit sind unvollständig). In allen Klassen fehlen die entsprechenden Methoden.



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) 4P
Was bedeuten die 2 verschiedenen Pfeile (offene bzw. geschlossene Pfeilspitze) und die Rauten (weiß und schwarz). Jeweils nur ein Stichwort angeben.
- 2) 7P
Erzeugen Sie die Klasse Blatt mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).
- 3) 7P
Erzeugen Sie die Klasse "Person" mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).
- 4) 9P
Erzeugen Sie die Klasse "Schueler" mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau zwei Parametern).
- 5) 7P
Erzeugen Sie die Klasse "Aufgabe" mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).
- 6) 4P
a) Erzeugen Sie die Klasse "Klassenarbeit" mit allen nötigen (genau) 4 Attributen.
- b) 4P
Erzeugen Sie genau einen Konstruktor (mit genau zwei Parametern). Falls ein Array benötigt wird, soll dieses nicht dynamisch sein und im Konstruktor das Feld die Länge 10 erhalten).
- c) 4P
Erzeugen Sie die folgende Methode:
... insertAufgabe(...) :
fügt eine Aufgabe an eine bestimmte Stelle des Feldes ein.
Beachten Sie dazu, daß eine Komposition und eine Aggregation anders implementiert werden.
- d) 4P
Erzeugen Sie die folgende Methode:
... insertBlatt (...) :
fügt ein Blatt ein.
Beachten Sie dazu, daß eine Komposition und eine Aggregation anders implementiert werden.

Lösungen:

1) 4P

geschlossene Pfeilspitze: Vererbung, geöffnete Pfeilspitze: Assoziation,
schwarze Raute: Komposition, weiße Raute: Aggregation

2) 7P

```
class Blatt {
    private char norm;

    public Blatt(char pNorm) {
        norm = pNorm;
    }

    public char getNorm() {
        return norm;
    }

    public void setNorm(char norm) {
        this.norm = norm;
    }
}
```

3) 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;
    }
}
```

4) 9P

```
class Schueler extends Person {
    private String name;

    public Schueler(int personID, String pName){
        super(personID);
        name=pName;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

5) 7P

```
class Aufgabe{
    private int nummer;

    public int getNummer() {
        return nummer;
    }

    public void setNummer(int nummer) {
        this.nummer = nummer;
    }

    public Aufgabe(int pNummer){
        nummer=pNummer;
    }
}
```

6)

```
class Klassenarbeit{
    private String thema;
    private Aufgabe[] dieAufgaben;
    private Schueler[] dieSchueler;
    private Blatt blatt;

    public Klassenarbeit(String pThema, Blatt pBlatt){
        thema=pThema;
        blatt=pBlatt;
        dieAufgaben=new Aufgabe[10];
        dieSchueler=new Schueler[10];
    }

    public void insertAufgabe(int pNummer, int index){
        dieAufgaben[index]=new Aufgabe(pNummer);
    }

    public void insertSchuler(Schueler pSchueler, int index){
        dieSchueler[index]=pSchueler;
    }

    public void insertBlatt(Blatt pBlatt ){
        blatt=pBlatt;
    }
    ...
}
```

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) 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, die Sie implementieren müssen:

1) `public void printFeld()` :
gibt alle Elemente des Feldes auf dem Bildschirm aus.

2) `public void insert(int zahl)` :
fügt eine Zahl an das Ende des Feldes an.
Dies soll dadurch realisiert werden, daß ein neues Feld erstellt (Längenanpassung!) 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.

3) `public void delete (int zahl)` :
wenn die Zahl `zahl` das erste Mal im Feld vorkommt, wird sie entfernt. Falls sie nicht vorkommt, passiert nichts.
Dies soll dadurch realisiert werden (Längenanpassung), daß ein neues Feld erstellt wird (mit einer um 1 geringeren Länge als das alte Feld), wobei dann die Zahlen des alten Feldes in das neue kopiert werden.

4)
`public void einfuegenAnPosition (int index, int zahl)` :
einfügen eines Elements an der Stelle `index` des Feldes, wobei die Elemente rechts von `index` nach rechts verschoben werden und damit das (neu zu erstellende Feld) wie bei 2) um 1 größer wird.

5) In `main()` soll mit Hilfe einer Schleife und der Methode `anfuegen(...)` die Zahlen 100, 101, 102, ... , 120 in das Feld eingefügt werden.
Danach sollen diese Zahlen auf dem Bildschirm ausgegeben werden.

Bem:

Es dürfen keine Methoden der Java-API verwendet werden, wie z.B. der Klasse `ArrayList` (o.ä.)

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) 10P

a) 4P

In der Java-Methode `truncDivision(int zähler, int nenner)` soll der ganzzahlige Anteil bei der Division zweier ganzzahliger Zahlen berechnet werden.

Beispiel: $7/4 = 1$

Da eine Division durch 0 nicht erlaubt ist, muß dies programmtechnisch abgefangen werden.

Realisieren Sie die Methode `truncDivision(int zähler, int nenner)` in Java mit Hilfe des Exception-Handlings und einer Bildschirmausgabe (im Fall einer Division durch 0).

b) 4P

Realisieren Sie die Methode `truncDivision(int zähler, int nenner)` wie bei a), aber ohne eine Ausgabe auf dem Bildschirm zu verwenden (weiterleiten der Exception an die umgebende Methode).

c) 2P

Die Division durch 0 könnte zwar bei b) auch mit Hilfe von `if` bzw. `if-else` ermittelt werden. Aber warum ist es nicht so einfach diesen Fehler aus der Methode zurückzugeben ?

2) 20P

In der 10 KB großen Datei `D:\\test1.txt` befinden sich lauter Integer-Zahlen.

Um diese Datei zu verschlüsseln, sollen nur die ersten 100 Integer-Zahlen in umgekehrter Reihenfolge angeordnet werden.

d.h. aus

11	27	3		13	14	55	Rest
----	----	---	----	----	--	----	----	----	------

wird:

55	14	13		3	27	11	Rest
----	----	----	----	----	--	---	----	----	------

Tipp:

Nehmen Sie als Vorlage das funktionierende Programm unten!

```

public class MainDatei3 {
    public static void main(String[] args) {
        RandomAccessFile myDatei = null;
        long myPosition;
        int zahl=5;
        int wert;

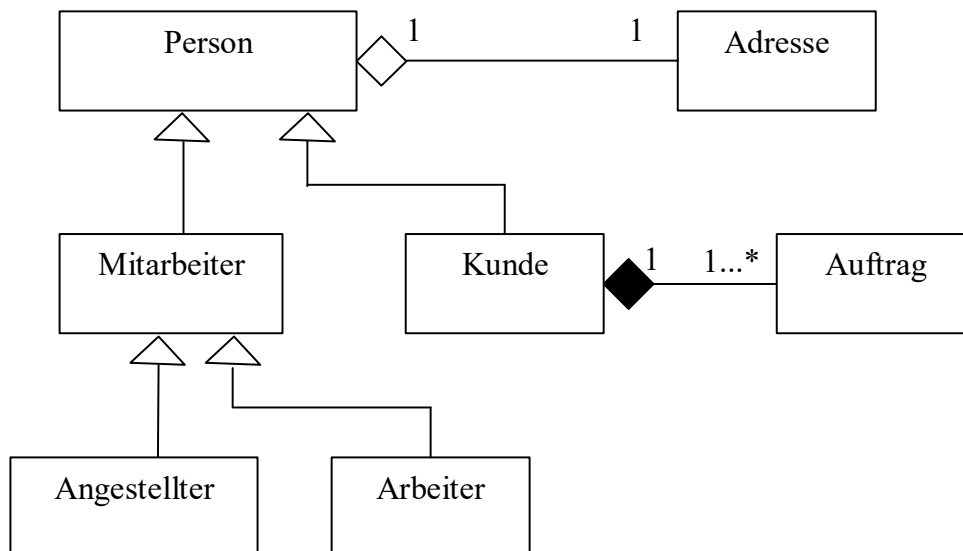
        try{
            // Lese- und Schreibzugriff. Nur wenn die Datei nicht existiert, wird sie angelegt.
            myDatei = new RandomAccessFile("C:\\\\test1.txt", "rw");
            // Schreibe Integer-Zahl in Datei an Position des Dateizeigers
            myDatei.writeInt(zahl);
            // Verschiebe Dateizeiger an Dateianfang
            myDatei.seek(0);
            // Lese Zahl aus
            wert=myDatei.readInt();
            // Gib Zahl auf Bildschirm aus
            System.out.println("Zahl="+wert);
            myDatei.close();
        }
        catch(FileNotFoundException e){
            System.out.println("Datei nicht da: "+e.toString());
        }
        catch(IOException e){
            System.out.println("Dateizugriff: "+e.toString());
        }
    }
}

```

3)

24P

Gegeben ist das folgende UML-Diagramm. In allen Klassen fehlen die entsprechenden Methoden und Attribute.



Eine Adresse hat genau (nur) die 2 Attribute "ort" und "plz".

Ein Auftrag hat genau (nur) das Attribut Auftragsnummer ("anr").

Eine Person hat genau (nur) die 2 Attribute "name" und "adresse".

Ein Mitarbeiter hat genau (nur) das Attribut Mitarbeiternummer ("mnr").

Ein Kunde hat genau (nur) ein Attribut. Welches?

Ein Angestellter hat genau (nur) das Attribut Gehalt ("gehalt").

Ein Arbeiter hat genau (nur) das Attribut Lohn ("lohn").

- a) Erstellen Sie die Klasse Auftrag mit den o.g. Attributen, genau einem Konstruktor (mit genau einem Parameter) und den entsprechenden get und set-Methoden.
- b) Erstellen Sie die Klasse Adresse mit den o.g. Attributen, genau einem Konstruktor (mit genau 2 Parametern) und den entsprechenden get und set-Methoden.
- c) Erstellen Sie die Klasse Mitarbeiter mit den o.g. Attributen, genau einem Konstruktor (mit genau 3 Parametern) und ohne get und set-Methoden.
- d) Erstellen Sie die Klasse Kunde mit den o.g. Attributen, genau einem Konstruktor (mit genau 3 Parametern) und ohne get und set-Methoden.
- e) Erstellen Sie in main() Folgendes:
- die Adressen Uhingen mit PLZ 72435 und Riedlingen mit PLZ 12345
 - den Mitarbeiter M_Uli aus Uhingen
 - den Kunden K_Rolf aus Riedlingen, der 2 Aufträge mit den Auftragsnummern 10 und 20 hat.

Lösung:

1)

a)

4P

```
public static int truncDivision1(int zähler, int nenner){
    int quotient;
    try{
        quotient=zähler/nenner;
    } catch (Exception e) {
        System.out.println("Division durch 0 verboten: " + e.toString());
    }
    return quotient;
}
```

b)

4P

```
public static int truncDivision2(int zähler, int nenner) throws Exception{
    int quotient;
    quotient=zähler/nenner;
    return quotient;
}
```

c)

2P

output kann nicht über einen Parameter mit elementarem Datentyp zurückgegeben werden.

2)

....

```
public class Startklasse {
    public static void main(String[] args) throws Throwable {
        // Für Deklarationen
        int i;
        int zahl;
        int wert;
        RandomAccessFile myDatei = null;
        long myPosition;
        int feld[];

        try {
            // Datei erzeugen
            myDatei = new RandomAccessFile("D:\\test1.txt", "rw");

            // Datei mit Zahlen belegen
            for(i=0;i<1000;i++) {
                myDatei.writeInt(i);
            }
            feld=new int[100];
            // erste 100 Zahlen auslesen
            myDatei.seek(0);

            for(i=0;i < 100;i++) {
                feld[i]=myDatei.readInt();
            }
            myDatei.seek(0);
            for(i=99;i>=0;i--) {
                myDatei.writeInt(feld[i]);
            }
            myDatei.seek(0);
            for(i=0;i < 100;i++) {
                System.out.print(myDatei.readInt()+" ");
            }
            myDatei.close();
        } catch (FileNotFoundException e) {
            System.out.println("Datei nicht da: " + e.toString());
        } catch (EOFException e) {
            System.out.println("Dateiende erreicht: " + e.toString());
        }
    }
}
```

3)

a)

4P

```
class Auftrag{
    private int anr;

    public Auftrag(int anr){
        this.anr=anr;
    }

    public int getAnr() {
        return anr;
    }

    public void setAnr(int anr) {
        this.anr = anr;
    }
}
```

b)

6P

```
class Adresse{
    private String ort;
    private int plz;

    public Adresse(String ort, int plz){
        this.ort = ort;
        this.plz = plz;
    }

    public String getOrt() {
        return ort;
    }

    public void setOrt(String ort) {
        this.ort = ort;
    }

    public int getPlz() {
        return plz;
    }

    public void setPlz(int plz) {
        this.plz = plz;
    }
}
```

c)

5P

```
class Mitarbeiter extends Person{
    private int mnr;

    public Mitarbeiter(int mnr, String name, Adresse adresse){
        super(name, adresse);
        this.mnr=mnr;
    }
}
```

d)

5P

```
class Kunde extends Person{
    private Auftrag dieAufträge[];

    public Kunde(int[] nrN, String name, Adresse adresse){
        super(name, adresse);
        int i;
        int len;
        len = nrN.length;
        dieAufträge=new Auftrag[len];
        for(i=0;i<len;i++){
            dieAufträge[i]=new Auftrag(nrN[i]);
        }
    }
}
```

e)

4P

```
Adresse a1= new Adresse("Uhingen", 72435);
Adresse a2= new Adresse("Riedlingen", 12345);
Mitarbeiter m1 = new Mitarbeiter(1, "M_Uli", a1);
int nummern[] = {10,20};
Kunde k1 = new Kunde(nummern, "K_Rolf", a2);
```

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.

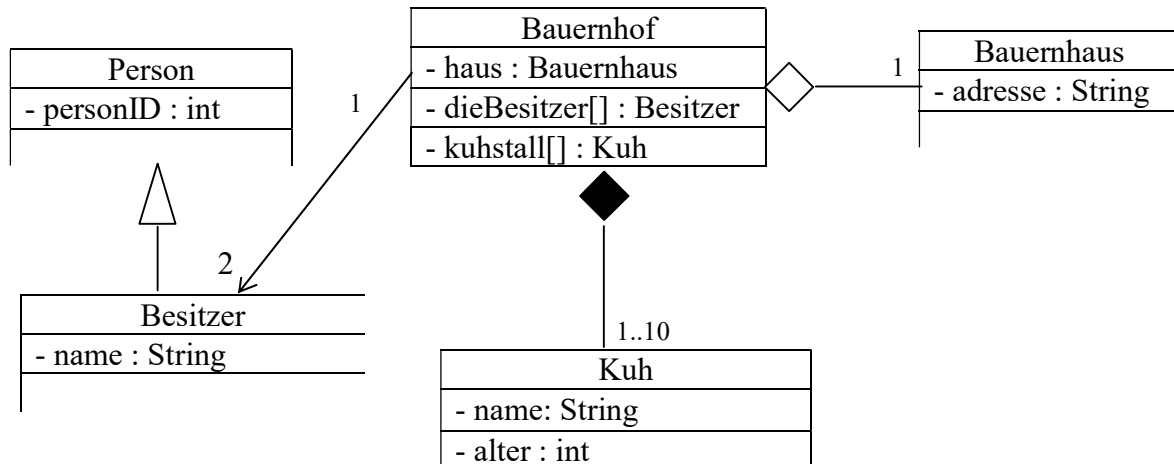
Bemerkung:

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)



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)

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