

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 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 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 einem Nicht-Standardkonstruktor.

b) 4P

Erstellen Sie in main() den Autor "Lowinzki" und das Buch mit dem Titel "Dorfmann ohne r"

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 soll nicht verändert werden.

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

b) 8P

Der Autor "Dofinzki" schreibt das Buch "V3 höher als K2". Schreiben Sie dazu in main() die entsprechenden Anweisungen.

c) 4P

Danach soll mit einer einzigen Anweisung vom Autor "Dofinzki" ausgehend (d.h. mit Hilfe der entsprechenden Objektvariablen, in der "Dofinzki" 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 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.

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 "Mary Jane" schreibt das Buch "Bedman". Schreiben Sie dazu die entsprechende Anweisung im main().

c) 4P

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

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;
    }
}

Autor a1 = new Autor("Lowinzki");      // 2P
Buch b1 = new Buch("Dorfmann ohne r"); // 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("Dofinzki"); // 2P
Buch b2 = new Buch("V3 höher als K2"); // 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("Mary Jane", "Bedman ");
```

c) 4P

```
a3.getBuch().setTitel("Badman ");
```


Name, Vorname:

Hilfsmittel:
keine

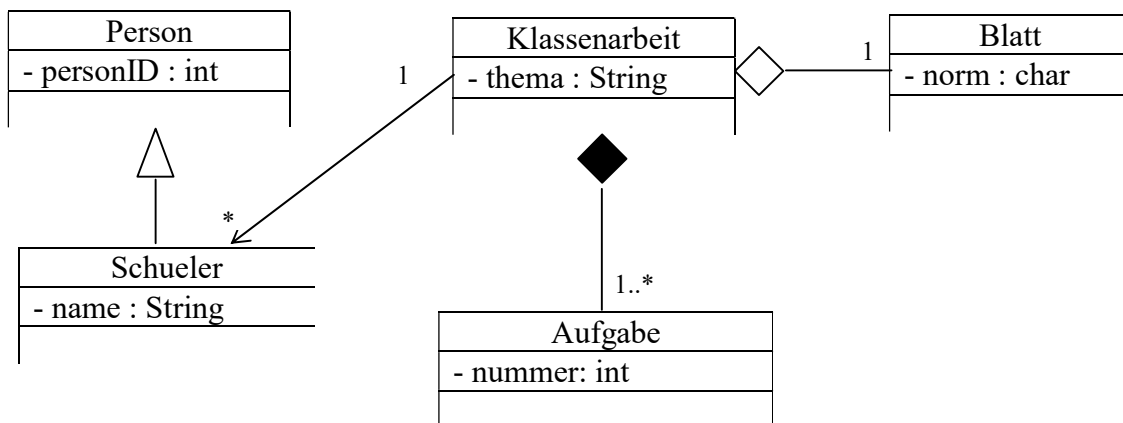
Hinweise (unbedingt beachten):

- 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

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



Alle Teilaufgaben müssen in **einem** Programm realisiert werden und die einzelnen Aufgabenteile als Kommentar eingefügt werden, wie z.B.

// 1)

bzw.

// 1a)

- 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;
    }
    ...
}
```