

*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

gegeben sei folgender Programmausschnitt (Hund ist eine Klasse):

```
...  
Hund myh1;  
myh1 = new Hund();  
Hund myh2;  
myh2 = new Hund();  
...
```

Was veranlassen diese obigen 4 Zeilen Programmcode im Arbeitsspeicher?

Bezeichnung	Adresse	Inhalt
myh1		
myh2		

Vervollständigen Sie den folgenden Satz:  
myh1 bezeichnet kein Objekt, sondern ....

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

Bemerkung:

Der neue Kontostand darf nicht selbst mit einem Taschenrechner berechnet werden, also  $5 + 1000$  nicht selbst mit dem Taschenrechner berechnen, sondern dies muß programmtechnisch realisiert werden.

Das gleiche gilt für den Zinssatz..

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

- 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 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

Bezeichnung	Adresse	Inhalt
myh1	0800	0900
	...	
	0900	Attribute
		des
		Objekts
myh2		
	0200	0300
	...	
	0300	Attribute
		des
		Objekts

Vervollständigen Sie den folgenden Satz:  
myh1 bezeichnet kein Objekt, sondern eine  
Referenz auf ein Objekt.

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

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 public class Startklasse {
{
    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) 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)

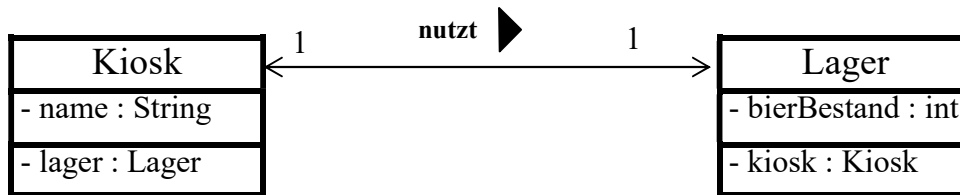
Lösungen:

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 Punkteabzug !!!!

## AUFGABEN

1)

20P

**Funktioniert das Programm unten? Begründen Sie!**

```
public class E2FI_8_1_2013_nach1 {
    public static void main(String[] args) {
        A myA = new A(3);
        B myB = new B("hallo");
        myA.setB(myB);
    }
}

class A{
    private int zahl;
    private B b;

    public A(int pZahl){
        zahl=pZahl;
    }

    B getB(){
        return b;
    }

    public void setB(B pB){
        b = pB;
        b.setA(this);
    }
}

class B{
    private String name;
    private A a;

    public B(String pName){
        name=pName;
    }

    A getA(){
        return a;
    }

    public void setA(A pA){
        a = pA;
        a.setB(this);
    }
}
```

2)

30P

Die Klasse "DynamischerSpeicher" hat u.a. das Attribut `feld`, in dem ganze Zahlen gespeichert werden und die folgenden Methoden:

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

Mit der Methode `void anfüegen(int zahl)` wird eine Zahl an das Ende des Attributs `feld` angefügt. Die Methode darf beliebig oft aufgerufen werden. Da ein Feld ein Array ist und ein Array eine feste Länge haben muss, ist dies kein einfaches Problem:

Deshalb muß bei jedem Aufruf zuerst das Array `feld` zwischengespeichert werden.

Danach muß das Array `feld` neu (mit einer) größeren Länge instanziiert werden.

Der Konstruktor `DynamischerSpeicher(int zahl)` erzeugt das Array `feld` der Länge 1 und speichert dort die Zahl `zahl` ab.

Die Methode `void printFeld()` gibt den Inhalt des Arrays `feld` auf dem Bildschirm aus.

a)

Erstellen Sie die Klasse "DynamischerSpeicher" u.a. mit dem Attribut `feld`, in dem ganze Zahlen gespeichert werden.

Erstellen Sie außerdem die folgenden Methoden:

b)

```
public DynamischerSpeicher(int zahl)
```

c)

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

d)

```
public void printFeld()
```

Lösung:

1)

Das Programm funktioniert nicht, weil sich die set-Methoden unendlich oft gegenseitig aufrufen.

2)

```
package dynamischerspeicher1;
public class MainDynamischerSpeicher1 {

    public static void main(String[] args) {
        int j;
        DynamischerSpeicher ds;
        ds=new DynamischerSpeicher(3);
        for(j=0;j<10;j++){
            ds.anfuegen(j+5);
        }
        ds.printFeld();
    }
}

class DynamischerSpeicher{
    private int[] feld;
    private int[] feldTemp;
    private int len;

    public DynamischerSpeicher(int zahl){
        feld = new int[1];
        feld[0]=zahl;
        len=1;
    }

    public void anfuegen(int zahl){
        int i;
        feldTemp=new int[len];
        for(i=0;i<len;i++){
            feldTemp[i]=feld[i];
        }
        feld = new int[len+1];
        for(i=0;i<len;i++){
            feld[i]=feldTemp[i];
        }
        feld[len]=zahl;
        len=len+1;
    }

    void printFeld(){
        int i;
        for(i=0;i<len;i++){
            System.out.print(feld[i]+" ");
        }
    }
}
```

*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)

Ein Bauer will mit möglichst wenig Elektrozaun eine bestimmte Fläche (die kreisförmig oder rechteckig gestaltet sein kann) abstecken, auf der seine Schafe weiden.

Dazu werden die Klassen Kreis und Rechteck erstellt.

Bemerkungen:

Berechnung des Umfangs bzw. der Fläche eines Rechtecks wird als bekannt vorausgesetzt.

$\text{UmfangKreis} = 2\pi r$

$\text{FlächeKreis} = \pi r^2$

a)

14P

Implementieren Sie die Klassen Kreis (mit Attribut "radius") und Rechteck (mit den Attributen "a" und "b" für die Seitenlängen). Implementieren Sie dazu jeweils einen Konstruktor und die set- und get-Methoden.

Keine weiteren Attribute einfügen!

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

b)

Das Programm soll multipersonal entwickelt werden:

In einem Feld sollen verschiedene Objekte (Kreise, Rechteck) abgespeichert werden.

Danach soll jeweils das Verhältnis Umfang/Fläche der einzelnen Objekte auf dem Bildschirm ausgegeben werden.

b1)

14P

Erstellen Sie ein UML-Diagramm (keine Angabe der Attribute, aber alle Methoden, außer den Konstruktoren und den get- bzw. set-Methoden für die Manipulation der Attribute)



b2) 3P

Was muß programmtechnisch gemacht werden, damit man die Entwickler der einzelnen Klassen zwingen kann, unbedingt die obigen Methoden "... getUmfang(...)" und "...getFlaeche(...)" zu implementieren? Verbale Beschreibung !

b3) 3P

Wo muß die Methode "...getVerhaeltnis(...)" am Sinnvollsten implementiert werden, die das Verhältnis von Umfang zu Fläche berechnet (nicht auf dem Bildschirm ausgibt)? Begründen Sie!

b4) 10P

Ergänzen Sie dazu die bisherige Implementierung (Unbedingt sinnvolle Namen geben!!).

Bitte keine Attribute der bereits bestehenden Klassen verändern oder neue einfügen!

b5) 7P

Machen Sie folgendes in der Methode main()

b51) Erzeugen Sie den Kreis "kreis1" mit Radius 1 und das Rechteck "rechteck1" mit den Seitenlängen 2 und 3.

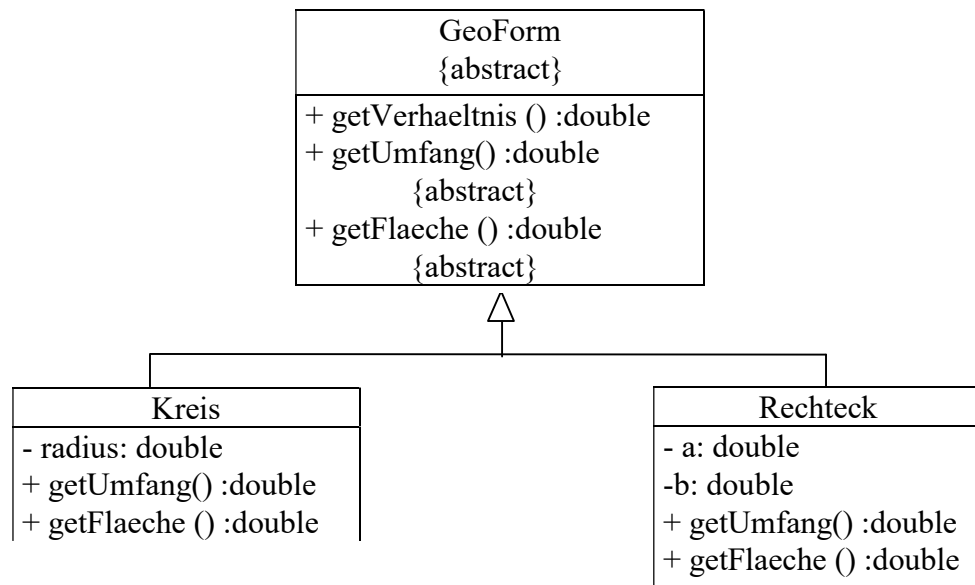
b52) Speichern Sie diese 2 Objekte in dem Feld.

b53) Geben Sie mit Hilfe einer Schleife und der Methode "... getVerhaeltnis(...)" und der Ausgabemethode System.out.println(...) das jeweilige Verhältnis von Umfang zu Fläche auf dem Bildschirm aus.

Lösungen:

a) 14P

b1) 14P



b2) 3P

Die Klasse GeoForm und die Methoden

`berechneFlaeche()` und `berechneUmfang()` jeweils `abstract` machen.

Bei den Unterklassen jeweils `extends` hinzufügen.

b3) 3P

In der Oberklasse GeoForm. Da das Verhältnis in jeder Unterklasse gleich berechnet wird (Umfang / Fläche) ist es nicht sinnvoll dies in den einzelnen Unterklassen zu machen.

Implementierung siehe unten.

b4) 7P

Das Programm:

```
package Maine2fi_7_5_13_nr1;
public class E2FI_7_5_13_nr1 {
    public static void main(String[] args) {
        int i;
        GeoForm[] feld;           // 1P
        feld = new GeoForm[2];    // 1P
        feld[0]=new Rechteck(2,3); // 1P
        feld[1]=new Kreis(1);     // 1P
        for (i = 0; i < 2; i++) { // 3P
            System.out.println("Flaeche des "+i+"-ten Objekts=
                               "+feld[i].berechneFlaeche());
            System.out.println("Umfang des "+i+"-ten Objekts=
                               "+feld[i].berechneUmfang());
            System.out.println("Umfang/Flaeche des "+i+"-ten Objekts=
                               "+feld[i].getVerhaeltnis());
        }
    }
}
```

```

abstract class GeoForm {                                // 1P
    public double getVerhaeltnis() {                    // 3P
        double v;
        v=berechneUmfang()/berechneFlaeche();
        return v;
    }
    abstract public double berechneFlaeche();          // 2P
    abstract public double berechneUmfang();           // 2P
}

class Rechteck extends GeoForm {                        // 9P
    private double laenge;
    private double breite;

    public Rechteck(double plaenge, double pbreite) {
        setLaenge(plaenge);
        setBreite(pbreite);
    }

    public void setLaenge(double plaenge) {
        laenge = plaenge;
    }

    public void setBreite(double pbreite) {
        breite = pbreite;
    }

    public double getLaenge() {
        return (laenge);
    }

    public double getBreite() {
        return (breite);
    }

    public double berechneFlaeche(){
        return(laenge*breite);
    }

    public double berechneUmfang(){
        return(2*(laenge+breite));
    }
}

class Kreis extends GeoForm {                          // 7P
    private double radius;

    public Kreis(double pradius) {
        setRadius(pradius);
    }

    public void setRadius(double pradius) {
        radius = pradius;
    }

    public double getRadius() {
        return (radius);
    }

    public double berechneFlaeche(){
        return(Math.PI*radius*radius);
    }

    public double berechneUmfang(){
        return(2*Math.PI*radius);
    }
}

```

# KLAUSUR 3      SAE      E2FI      Nachtermin 1      Zeit: 45 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

**ACHTUNG:** Die Klassenarbeit besteht aus einer Aufgabe.

Die gesamte Aufgabe muß als genau ein Programm in einem Projekt implementiert werden.

I)

1)

12P

Ein Fach (in der Schule) besteht aus dem Namen und der Note (ganzzahlig).

(keine weiteren Attribute einfügen!)

a) Implementieren Sie die Klasse Fach mit den 2 entsprechenden Attributen und den entsprechenden Methoden.

Der Konstruktor setzt den Namen gleich "-----" und die Note auf 0.

2)

24P

Ein Zeugnis besteht aus 4 Fächern (bitte als Array implementieren) und dem Schülernamen (keine weiteren Attribute einfügen!)

Implementieren Sie die Klasse Zeugnis mit den 2 entsprechenden Attributen und den entsprechenden Methoden:

a) Der Konstruktor setzt den Namen des Schülers auf "Max Muster", setzt die Namen der einzelnen Fächer auf "-----" und setzt die Noten der einzelnen Fächer auf 0.

b) set- bzw. get-Methode, die zum Schülernamen gehören.

c) Die Methode setFach(...) bekommt als Parameter eine Fachnummer zwischen 0 und 3 sowie einen Fachnamen. Zusätzlich die dazugehörige get-Methode implementieren.

d) Die Methode setNote(...) bekommt als Parameter eine Fachnummer zwischen 0 und 3 sowie eine Zeugnisnote. Zusätzlich die dazugehörige get-Methode implementieren.

e) Die Methode showNoten() gibt den Schülernamen und alle 4 Fächer mit Fachnummer und Noten auf dem Bildschirm aus.

3)

12P

Es sollen Zeugnisse für Berufskollegs und für Berufsschulen erstellt werden. Die Zeugnisse unterscheiden sich nur durch den Zeugnis-Kopf:

- ZEUGNIS DES BERUFSKOLLEGS

- ZEUGNIS DER BERUFSSCHULE

Die Darstellung des Schülernamens und der Fächer mit Noten bleibt dieselbe.

Eine neue Methode `printZeugnis()` druckt den jeweiligen Zeugniskopf, den Schülernamen und die Fächer mit Noten. Zum Drucken des Schülernamens und der Fächer mit Noten wird innerhalb von `printZeugnis()` die Methode `showNoten()` aufgerufen.

a) Erstellen Sie zwei weitere Klassen `BKZeugnis` (Berufskolleg) und `BSZeugnis` (Berufsschule) als Unterklassen von `Zeugnis` mit jeweils der Methode `printZeugnis()`.

Das Druckergebnis könnte (muß aber nicht) z.B. so aussehen:

<div>ZEUGNIS DES BERUFSKOLLEGS</div> <div>Uwe Maier</div> <div>Deutsch 4</div> <div>Mathe 3</div> <div>Englisch 5</div> <div>EDV 2</div>	<div>ZEUGNIS DER BERUFSSCHULE</div> <div>Uta Wind</div> <div>Deutsch 5</div> <div>Mathe 1</div> <div>Englisch 3</div> <div>EDV 2</div>
------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------

b) Testen Sie die neuen Klassen mit einem Berufsschulzeugnis und einem Berufskollegzeugnis.

4)

4P

Erstellen Sie ein UML-Diagramm, in dem alle Klassen vorkommen.

Zur Vereinfachung müssen in den Klassen nicht die Attribute und Methoden angegeben werden.

## Lösung:

```
public class E2FI_17_6_13_nr1 { // 4P
    public static void main(String[] args) {
        BSZeugnis zs1 = new BSZeugnis();
        zs1.setName("Dettlef");
        zs1.setFach(0,"Mathe");
        zs1.setFach(1,"Englisch");
        zs1.setFach(2,"Religion");
        zs1.setFach(3,"Deutsch");
        zs1.setNote(0, 1);
        zs1.setNote(1, 3);
        zs1.setNote(2, 4);
        zs1.setNote(3, 5);
        zs1.printZeugnis();

        BKZeugnis zs2 = new BKZeugnis();
        zs2.setName("Dettlef");
        zs2.setFach(0,"Mathe");
        zs2.setFach(1,"Englisch");
        zs2.setFach(2,"Religion");
        zs2.setFach(3,"Deutsch");
        zs2.setNote(0, 6);
        zs2.setNote(1, 1);
        zs2.setNote(2, 3);
        zs2.setNote(3, 5);
        zs2.printZeugnis();
    }
}

class Fach{ // Jede Methode 2P
    private String name;
    private int note;

    public Fach(){
        name="-----";
        note=0;
    }
    public String getName() {
        return name;
    }

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

    public int getNote() {
        return note;
    }

    public void setNote(int note) {
        this.note = note;
    }
}

class Zeugnis{ // Jede Methode 3P, ausser 2 Methoden
    private String name;
    private Fach[] faeher;

    public Zeugnis(){
        int i;
        name="Max Muster";
        faeher=new Fach[4];
        for(i=0;i<5;i++){
            faeher[i]=new Fach();
            faeher[i].setName("-----");
            faeher[i].setNote(0);
        }
    }
}
```

```

public String getName() {
    return name;
}

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

public void setNote(int i, int note){
    faecher[i].setNote(note);
}

public int getNote(int i) {
    return faecher[i].getNote();
}

public void setFach(int i, String name){
    faecher[i].setName(name);
}

public String getFach(int i) {
    return faecher[i].getName();
}

public void showNoten(){
    int i;
    System.out.println("Schueler="+name);
    for(i=0;i<4;i++){
        System.out.println("Fach "+faecher[i].getName() + " :
                               "+faecher[i].getNote());
    }
}
}

class BKZeugnis extends Zeugnis{ // 4 P
    public void printZeugnis(){
        System.out.println("ZEUGNIS DES BERUFSSKOLLEGS");
        showNoten();
    }
}

class BSZeugnis extends Zeugnis{ // 4P
    public void printZeugnis(){
        System.out.println("ZEUGNIS DER BERUFSSCHULE");
        showNoten();
    }
}

```

