

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)

- 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)

gegeben sei folgender Programmausschnitt (Hund ist eine Klasse):

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

Was veranlassen diese zwei Zeilen Programmcode im Arbeitsspeicher?

Bezeichnung Adresse Inhalt

myh1	0800	

3) (Alle Teilaufgaben müssen in **einem** einzigen Programm, kein UML realisiert werden)

- a) Erstellen Sie die Klasse Punkt, (mit den entsprechenden Attributen, Methoden und zwei Konstruktoren), die einen Punkt (Pixel) in einer grafischen zweidimensionalen Oberfläche repräsentieren soll.
- b) Erzeugen Sie einen Punkt mit der x-Koordinate 5 und der y-Koordinate 10.
- c) Verändern Sie mit Hilfe der entsprechenden set- und get-Methode die Koordinaten um den Wert +3 in x- und den Wert -7 in y-Richtung.
- d) Ermitteln Sie mit Hilfe der entsprechenden Methoden die neuen Koordianten des Punktes und geben diese auf dem Bildschirm aus.
- e) Erzeugen Sie einen anderen, neuen Punkt mit der x-Koordinate 6 und der y-Koordinate 11.

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) 4 Punkte

Bezeichnung Adresse Inhalt

myh1

0800	0900
...	
0900	Attribute
	des
	Objekts

3)

```
public class MainKlassen6 {
    public static void main(String[] args) throws Exception{
        Punkt p1, p2;
        // 3 P
        p1= new Punkt(5,10);
        // 3 P
        p1.setX(p1.getX()+3);
        p1.setY(p1.getY()-7);
        // 3 P
        System.out.println("x= "+p1.getX());
        System.out.println("y= "+p1.getY());
        // 3 P
        p2= new Punkt(6,11);
    }
}

class Punkt{
    private double x; // 1P
    private double y; // 1P

    public Punkt(){ // 3P
        x=0;
        y=0;
    }

    public Punkt(double px, double py){ // 3P
        setPunkt(px,py);
    }

    public void setX (double px){ // 3P
        x=px;
    }

    public void setY (double py){ // 3P
        y=py;
    }

    public double getX(){// 3P
        return(x);
    }

    public double getY(){// 3P
        return(y);
    }
}
```

Name, Vorname:

Hilfsmittel:
ausgeteilte Skript

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) 2P

- a) Was ist der Dateizeiger ?
- b) Schildern Sie einen Fall, wo ihn ein Programmierer benötigt.
- c) Bei welcher Anweisung wird er nicht benötigt (weil er automatisch verschoben wird) ?

2) 6P

Gegeben ist die Datei "c:\e3fi.txt", die aus einer Anzahl von Zeichen (als Bytes realisiert) besteht, wie zum Beispiel den 4 Bytes "aBBa".

Überall, wo das Zeichen 'a' in der Datei vorkommt, soll es durch das Zeichen 'A' ersetzt werden.

Erstellen Sie dazu ein Struktogramm, **kein** Programm!

Wichtige Bemerkung:

Das Programm soll so geschrieben werden, es nicht nur für eine bestimmte Menge von Zeichen (z.B. 4 Zeichen funktioniert), sondern für eine beliebig große Datei

"c:\e3fi.txt".

3) 12P

Schreiben Sie **ein** fehlerfreies Java-Programm, das folgendes **nacheinander** macht:

- a) Es legt die Dateien "c:\e3fiqueille.txt" und "c:\e3fiziel.txt" an.
- b) Es sollen alle Bytes von "c:\e3fiqueille.txt" nach "c:\e3fiziel.txt" kopiert werden.

Benutzen Sie dazu die entsprechenden Anweisungen aus der Java-Doku (siehe Rückseite).

Auszug aus der Java-Doku:

long getFilePointer()

Returns the current offset in this file.

long length()

Returns the length of this file.

int read()

Reads a byte of data from this file.

Bemerkung:

Das Byte muss in einer integer-Variablen gespeichert werden.

void write (int b)

Writes the specified byte to this file.

Bemerkung:

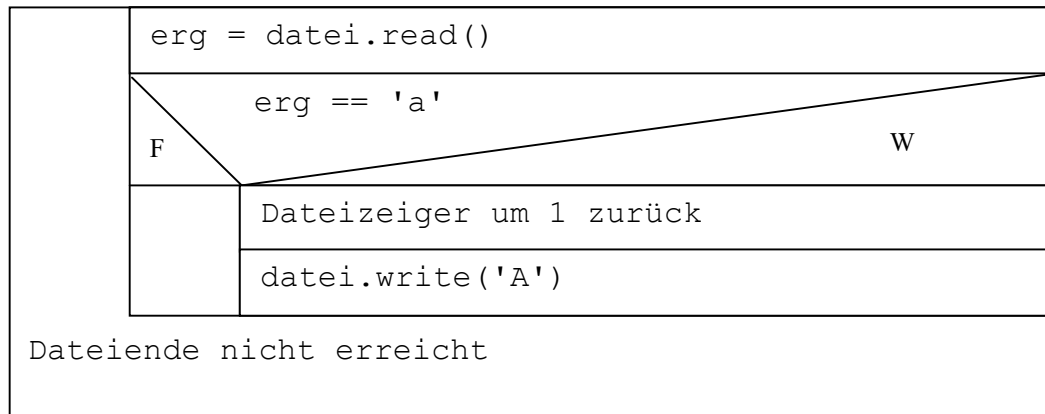
Der Parameter b ist zwar ein integer-Wert, doch werden von dem aus 4 Bytes bestehenden integer-Wert nur 1 Byte geschrieben (die höherwertigen Bytes werden nicht beachtet).

Lösungen

1)

- a) Ein Dateizeiger ist ein Zeiger, der auf die aktuelle Position in der Datei zeigt.
- b) Letzte Byte einer Datei lesen.
- c) Beim Lesen und Schreiben.

2)



3)

```
public class mainTest1{
    public static void main(String[] args) throws IOException {
        RandomAccessFile dateiZiel = null;
        RandomAccessFile dateiQuelle = null;
        int c;

        // Quelldatei anlegen
        dateiQuelle=new RandomAccessFile("C:\\E3FIquelle.txt","rw");

        // Zieldatei anlegen
        dateiZiel=new RandomAccessFile("C:\\E3FIziel.txt","rw");

        dateiQuelle.seek(0);
        dateiZiel.seek(0);

        try{
            while(dateiQuelle.getFilePointer()<=dateiQuelle.length()-1){
                c = dateiQuelle.read();
                dateiZiel.write(c);
            }
        }
        catch(FileNotFoundException e){
            System.out.println("Datei nicht gefunden"+e.toString());
        }
        catch(IOException e){
            System.out.println("Lese-Fehler"+e.toString());
        }
        }

        // Dateien schliessen
        try{
            dateiQuelle.close();
            dateiZiel.close();
        }
        catch(IOException e){
            System.out.println("Close-Fehler"+e.toString());
        }
    }
}
```

KLAUSUR 2 SAE E2FI Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

Bücher und Skripte eigener Wahl.

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 !!!!

1) 10P

Gegeben ist die Datei "c:\e3fi.txt", die aus einer Anzahl von Zeichen (als Bytes realisiert) besteht, wie zum Beispiel den 4 Bytes "abcd".

Die Reihenfolge der Bytes in dieser Datei soll umgekehrt werden (im Beispiel: "dcba").

Das letzte Byte wird zum ersten.

Erstellen Sie dazu ein Struktogramm, **kein** Programm!

Wichtige Bemerkung:

Das Programm soll so geschrieben werden, es nicht nur für eine bestimmte Menge von Zeichen (z.B. 4 Zeichen funktioniert), sondern für eine beliebig große Datei

"c:\e3fi.txt".

2) 10P

Schreiben Sie **ein** fehlerfreies Java-Programm, das folgendes macht:

Ermitteln Sie, ob die zwei Dateien "c:\e3fi1.txt" und "c:\e3fi2.txt" genau den gleichen Inhalt haben.

KLAUSUR 2 SAE E2FI Nachtermin 2 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

Bücher und Skripte eigener Wahl.

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 !!!!

1) 10P

Gegeben ist die Datei "c:\e3fi.txt", die aus einer Anzahl von Zeichen (als Bytes realisiert) besteht, wie zum Beispiel den 4 Bytes "abcd".

Der Inhalt dieser Datei soll ans Ende dieser Datei angefügt werden (im Beispiel: "abcdabcd").

Erstellen Sie dazu ein Struktogramm, **kein** Programm!

Wichtige Bemerkung:

Das Programm soll so geschrieben werden, es nicht nur für eine bestimmte Menge von Zeichen (z.B. 4 Zeichen funktioniert), sondern für eine beliebig große Datei

"c:\e2fi.txt".

2) 10P

Schreiben Sie **ein** fehlerfreies Java-Programm, das folgendes macht:

Ermitteln Sie, an welcher Stelle der Datei (in Richtung Dateiende) "c:\e3fi.txt" das Zeichen x das letzte Mal vorkommt und wie oft es vorgekommen ist.

Name, Vorname:

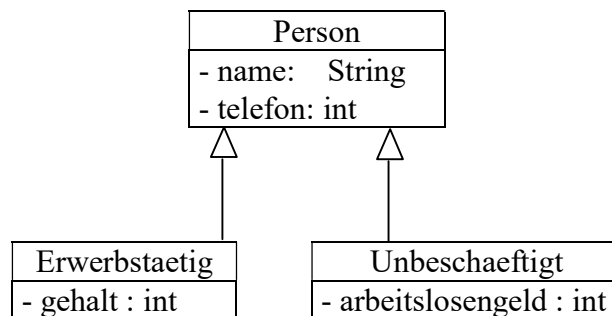
Hilfsmittel:
keine

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) Gegeben ist das folgende abgespeckte UML-Diagramm:



a) Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen mit den jeweiligen Konstruktoren, set- und get-Methoden. Entnehmen Sie insbesondere dem UML-Diagramm, ob es sich um eine "normale" oder eine abstrakte Klasse bzw. um ein Interface handelt. Die Konstruktoren müssen jeweils Parameter enthalten. 25P

b) Erzeugen Sie in der Methode main (jweils durch eine Anweisung) den Erwerbstätigen "Schaffer" mit der Telefonnummer 4711, dem Gehalt von 1500 Euro und den Unbeschäftigten "Schröder" mit der Telefonnummer 4712 und Arbeitslosengeld von 400 Euro. 4P

c) Herr Schaffer bekommt 100 Euro mehr Gehalt.
Realisieren Sie dies durch eine Anweisung, die zum alten Gehalt die 100 Euro dazuaddiert.
In der Anweisung muss also der alte Gehalt ermittelt werden. 3P

2)

a) Geben Sie ein sinnvolles Beispiel für eine abstrakte Klasse, das nicht im Unterricht benutzt wurde oder in den "Folien" bzw. hier als Aufgabe vorkam. Es reicht eine kurze verbale Beschreibung (keine Implementierung). 2P

b)

4P

Bewerten Sie folgende Aussagen (ohne Begründung) mit wahr oder falsch:

A1)

Wenn eine abstrakte Methode in einer Klasse verwendet wird, dann muss diese Klasse eine abstrakte Klasse sein.

A2)

Wenn eine Klasse abstrakt ist, dann muss mindestens eine sich darin befindlichen Methode abstrakt sein.

c)

2P

Aus der Klasse Person in Aufgabe 1 soll eine abstrakte Klasse gemacht werden.

Was muss in an der Klasse Person geändert werden? (kurze verbale Beschreibung).

d)

10P

In der Klasse Person soll die abstrakte Methode `bestimmeMaximalenKredit()` eingeführt werden.

Was muss im Programm bei Aufgabe 1 geändert werden?

Geben Sie bitte den Quellcode (nur die Änderungen) an.

Überlegen Sie sich eine sinnvolle Implementierung für `bestimmeMaximalenKredit()` !

Lösung:

```
1)
public class Main_E2FI_8_4_08_nr1 {
    public static void main(String[] args){
        Erwerbstaetig e = new Erwerbstaetig("Schaffer", 4711, 1500); // 2P
        Unbeschaeftigt u = new Unbeschaeftigt("Schroeder", 4712, 400); //2P
        e.setGehalt(e.getGehalt()+100); // 3P
    }
}

class Person{ // 11P
    public String name;
    public int telefon;

    public Person (String pName, int pTelefon){
        name = pName;
        telefon = pTelefon;
    }

    public void setTelefon(int pTelefon){
        telefon = pTelefon;
    }

    public void setName(String pName){
        name = pName;
    }

    public int getTelefon(){
        return(telefon);
    }

    public String getName(){
        return(name);
    }
}

class Unbeschaeftigt extends Person{ // 7P
    private int arbeitlosengeld ;

    public Unbeschaeftigt(String pName, int pTelefon,
                           int pArbeitslosengeld){
        super(pName, pTelefon);
        arbeitlosengeld = pArbeitslosengeld;
    }

    public void setArbeitslosengeld(int pArbeitslosengeld){
        arbeitlosengeld = pArbeitslosengeld;
    }

    public int getArbeitslosengeld(){
        return(arbeitslosengeld);
    }
}
```

```

class Erwerbstaetig extends Person{ // 7P
    private int gehalt ;

    public Erwerbstaetig(String pName, int pTelefon, int pGehalt){
        super(pName, pTelefon);
        gehalt = pGehalt;
    }

    public void setGehalt(int pGehalt){
        gehalt = pGehalt;
    }

    public int getGehalt(){
        return(gehalt);
    }
}

```

2)

a) Fahrzeug // 2P

b)

A1) wahr // 2P

A2) falsch // 2P

c)

abstract class Person // 2P

d)

```

abstract class Person{
    public String name;
    public int telefon;

    public abstract int bestimmeMaximalenKredit(); // 2P
    // ...
}

```

```

class Erwerbstaetig extends Person{
    private int gehalt ;
    // ...

    public int bestimmeMaximalenKredit(){ // 4P
        return(2*gehalt);
    }
}

```

```

class Unbeschaeftigt extends Person{
    private int arbeitlosengeld ;
    // ...

    public int bestimmeMaximalenKredit(){ // 4P
        return(3*arbeitlosengeld);
    }
}

```

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

1a) Was ist falsch an folgendem Java-Programmausschnitt? (mit Begründung!) 2P

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

b) Was ist falsch an folgendem Java-Programmausschnitt? (mit Begründung!) 2P

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

c) Ist der folgende Java-Programmausschnitt syntaktisch korrekt? 2P

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

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

2a) gegeben sei folgender Programmausschnitt (Hund ist eine Klasse): 3P

```
...  
Hund myh1;  
...
```

b) gegeben sei folgender Programmausschnitt (Hund ist eine Klasse): 6P

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

Was veranlasst jeweils diesen Programmausschnitt im Arbeitsspeicher?

Bitte jeweils eine Zeichnung der folgenden Form ausfüllen:

Bezeichnung Adresse Inhalt

3) Gegeben ist das folgende abgespeckte UML-Diagramm:

Fliegenpilz
- gewicht : double
- beschreibung: String

Himbeerhecke
- volumen : double
- beschreibung: String

Die Klassen Fliegenpilz und Himbeerhecke sind sich ähnlich und sollen eine gemeinsame Oberklasse bekommen, wobei die abstrakte Methode `getBeschriftung(...)` die Zeichenkette zurückgibt, die aus den Werten der Attributen besteht und nachher auf einem Aufkleber steht, der an die Pflanze angeheftet wird.

Beispiel:

Für einen Fliegenpilz kann z.B. die Zeichenkette "10-giftig" zurückgegeben werden

a) Geben Sie das vollständige UML-Diagramm an.

8,5P

b) Erstellen Sie den Quellcode für die Methode `getBeschriftung(...)` in der Klasse Fliegenpilz und in der Klasse Himbeerhecke

6P

Bemerkung:

Mit der Methode

`Double.toString(...)`

wird eine Doublezahl in einen String umgewandelt.

4) In einem Feld sollen verschiedene Tiere (Hennen oder Kühe) abgespeichert werden.

Danach sollen die Tiere mit der Methode `sichVorstellen()` vorgestellt werden (d.h. mit ihrem Name und ihrem Tierlaut auf dem Bildschirm ausgegeben werden).

Da der junge, hochgebildete Mensch heutzutage von seinen Eltern und Lehrern gezwungen wird, den ganzen Tag vor dem Computer zu sitzen um in der virtuellen Spielwelt besser zurecht zu kommen und sich damit für das spätere Leben vorzubereiten, hat er natürlich keine Ahnung von der Landwirtschaft, Euter, Hörnern, Eiern, Tierlauten, usw.

Deswegen ist die Entwicklungsabteilung hocherfreut, dass dieses Projekt multipersonal entwickelt wird, d.h. ein dem Bäuerlichen nahestehender Entwickler schreibt die Klasse **Henne** (die u.a. die Methode `getTierlaut()` und nur das Attribut **gewicht** enthalten muss), ein Landwirten abstammender Programmierer entwirft die Klasse **Kuh** (die u.a. die Methode `getTierlaut()` und nur das Attribut **alter** enthalten muss).

a) Da in einem Feld nur Objekte des gleichen Typs abgespeichert werden können, muss man sich einen "Trick" einfallen lassen. Welcher "Trick" ist das?

1P

b) Ein dem Hochadel Angehöriger (very educated, absolutely developed and specially cultivated) und deswegen naturgemäß allem Bäuerlichen sehr distanziert gegenüber stehender Entwickler programmiert die gemeinsame Oberklasse **Nutztier**. Diese muss u.a. die Methode `sichVorstellen()` (siehe oben) enthalten

Erstellen Sie ein UML-Diagramm mit den Klassen Nutztier, Henne, Kuh und den entsprechenden Methoden.

8P

c) Welche Eigenschaft sollte die Methode `getTierlaut` (und damit auch die Klasse Nutztier) haben (Begründung)?

3P

d) Implementieren Sie die Methoden `sichVorstellen()`, `getTierlaut()` in der Klasse Kuh, `getTierlaut()` in der Klasse Henne

9P

Lösung:

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

2P

`w[0] = -123`

`w[1] = 0`

2a) 3P

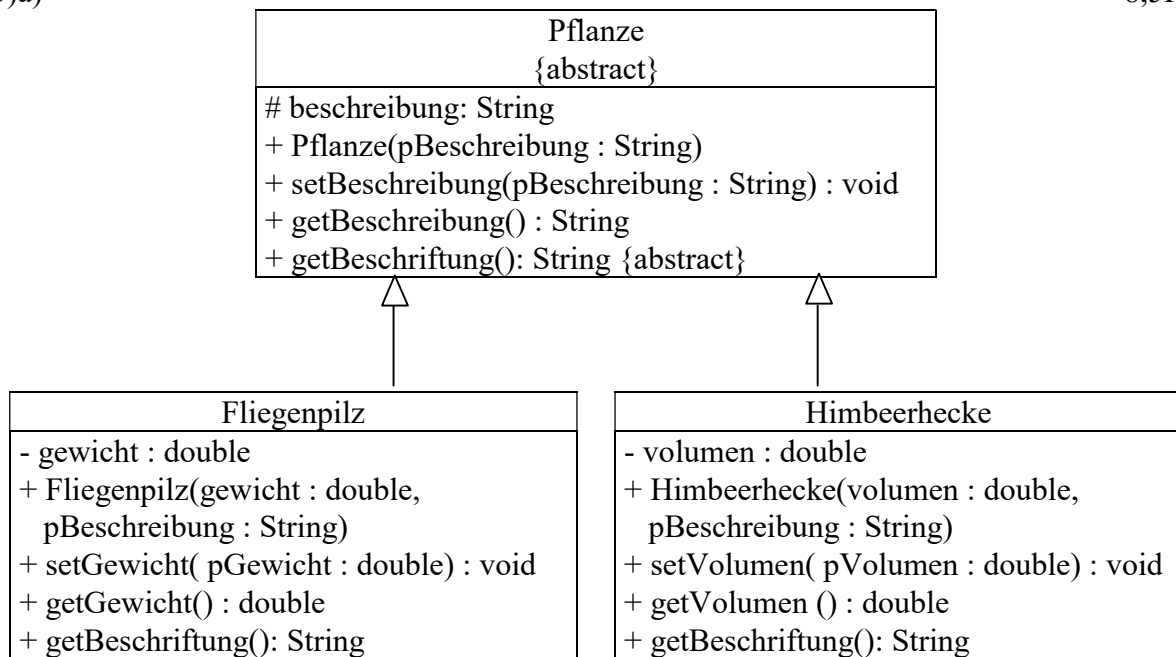
Bezeichnung	Adresse	Inhalt
myh1	0800	0900

2b) 6P

Bezeichnung	Adresse	Inhalt
myh1	0800	0900
	...	
	0900	Attribute
		des
		Objekts

3)a)

8,5P



b)

6P

```

class Himbeerhecke extends Pflanze{
    // ...

    String getBeschriftung(){
        return(Double.toString(volumen)+"-"+beschreibung);
    }
}

class Fliegenpilz extends Pflanze{
    // ...

    String getBeschriftung(){
        return(Double.toString(gewicht)+"-"+beschreibung);
    }
}

```

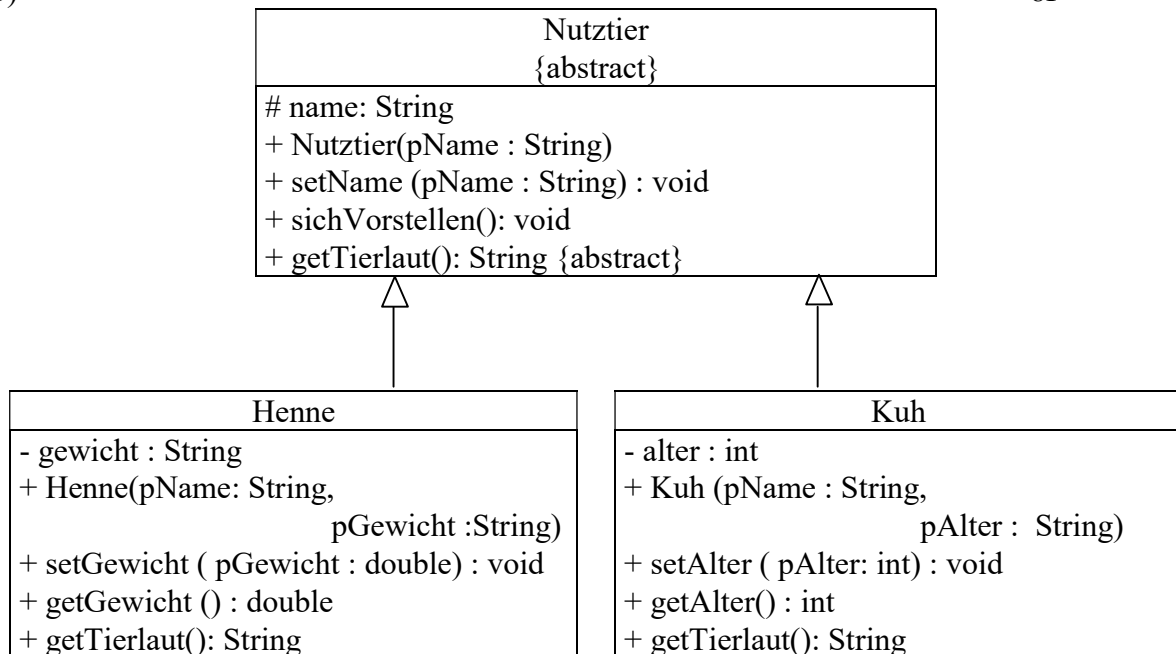
4)

a) Eine gemeinsame Oberklasse z.B. Nutztier bilden.

1P

b)

8P



c) `getTierlaut()` soll eine abstrakte Methode sein, damit in den Unterklassen diese Methode ausprogrammiert werden muss, weil sie in der Methode `sichVorstellen()` benutzt wird. 3P

d)

3P

```

public void sichVorstellen(){
    System.out.println("ich heiße "+getName()+" und mache "+getTierlaut());
}

```

```

public String getTierlaut(){
    return("muuuh");
}

```

3P

```

public String getTierlaut(){
    return("gaag");
}

```

3P

Aufgabe 4 (Gesamtprogramm)

```
public class MainE2FI_3_6_08_nr2 {
    public static void main(String[] args) {
        double zufall;
        int i;
        String str;
        Nutztier nutztiere[];
        nutztiere = new Nutztier[2];

        for(i=0;i<2;i++){
            zufall = Math.random();
            if (zufall<0.5){
                nutztiere[i] = new Henne("ute", 3.5);
            }
            else{
                nutztiere[i] = new Kuh("Elsa", 12);
            }
        }

        for(i=0;i<2;i++){
            nutztiere[i].sichVorstellen();
        }
    }
}

abstract class Nutztier{
    private String name;

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

    public String getName(){
        return(name);
    }

    public void sichVorstellen(){
        System.out.println("ich heie "+getName()+" und mache "+getTierlaut());
    }

    abstract public String getTierlaut();
}

class Kuh extends Nutztier{
    private int  alter;

    public Kuh(String pName, int  pAlter){
        super(pName);
        alter = pAlter;
    }

    public String getTierlaut(){
        return("muuuh");
    }
}

class Henne extends Nutztier{
    private double gewicht ;
    public Henne(String pName, double pGewicht){
        super(pName);
        gewicht = pGewicht;
    }

    public String getTierlaut(){
        return("gaag");
    }
}
```

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 Variablenamen 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) 4P

gegeben sei folgender Programmausschnitt (Hund ist eine Klasse):

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

Was veranlassen diese zwei Zeilen Programmcode im Arbeitsspeicher?

Bezeichnung Adresse Inhalt

myh1	0800	

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

32P

- a) Erstellen Sie die Klasse Punkt, (mit den entsprechenden Attributen, Methoden und zwei Konstruktoren), die einen Punkt (Pixel) in einer grafischen zweidimensionalen Oberfläche repräsentieren soll.
- b) Erzeugen Sie einen Punkt mit der x-Koordinate 5 und der y-Koordinate 10.
- c) Verändern Sie die Koordinaten um den Wert +3 in x- und den Wert -7 in y-Richtung.
- d) Ermitteln Sie mit Hilfe der entsprechenden Methoden die neuen Koordianten des Punktes und geben diese auf dem Bildschirm aus.
- e) Erzeugen Sie einen anderen, neuen Punkt mit der x-Koordinate 6 und der y-Koordinate 11.

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) 4 Punkte

Bezeichnung	Adresse	Inhalt
myh1	0800	0900
	...	
	0900	Attribute
		des
		Objekts

3)

```
public class MainKlassen6 {
    public static void main(String[] args) throws Exception{
        Punkt p1, p2;
        // 3 P
        p1= new Punkt(5,10);
        // 3 P
        p1.setX(p1.getX()+3);
        p1.setY(p1.getY()-7);
        // 3 P
        System.out.println("x= "+p1.getX());
        System.out.println("y= "+p1.getY());
        // 3 P
        p2= new Punkt(6,11);
    }
}

class Punkt{
    private double x; // 1P
    private double y; // 1P

    public Punkt(){ // 3P
        x=0;
        y=0;
    }

    public Punkt(double px, double py){ // 3P
        setPunkt(px,py);
    }

    public void setX (double px){ // 3P
        x=px;
    }

    public void setY (double py){ // 3P
        y=py;
    }

    public double getX(){// 3P
        return(x);
    }

    public double getY(){// 3P
        return(y);
    }
}
```

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) 10P

Gegeben ist ein Feld, in dem die Noten im Fach Englisch enthalten sind.

Damit die letzte Note des Feldes erkannt wird, kommt nach ihr eine negative Zahl.

Geben Sie einen **Programmausschnitt** an, der den Durchschnitt (arithmetische Mittelwert) der Noten berechnet und auf dem Bildschirm ausgibt.

2) In einem Verwaltungsprogramm eines Zoos sollen u.a. die Löwen (bzw. Löwinnen) eines Löwenrudels verwaltet werden.

Der Einfachheit halber soll die Klasse Löwe nur das Attribut name haben.

a) 10P

Erstellen (implementieren) Sie die Klasse Loewe (Attribute, Methoden, Konstruktor)

b) Betrachten Sie den folgenden Programmausschnitt:

```
public static void main(String argv[]) throws IOException{
    final int len=2;
    Loewe[] loewenrudel;
    loewenrudel=new Loewe[len];
    loewenrudel[0]=new Loewe("Leo");
    loewenrudel[1]=new Loewe("Simba");
}
```

Was veranlassen die folgenden Deklaration bzw. Anweisungen im Arbeitsspeicher?

Geben Sie dazu einen genauen Ausschnitt aus dem Arbeitsspeicher (mit Kommentar, siehe Rückseite) an.

b1) 5P

```
Loewe[] loewenrudel;
```

b2) 5P

```
loewenrudel=new Loewe[len];
```

b3) 10P

```
loewenrudel[0]=new Loewe("Leo");
```

b4) 10P

```
loewenrudel[1]=new Loewe("Simba");
```

Bemerkung:

Ein Ausschnitt aus dem Arbeitsspeiche muss wie folgt in einer Tabelle angegeben werden:

[illegible]

Lösungen:

1) Programmausschnitt:

```
...
int i = 0;
double sum=0;
double mittelwert=0;
double [] noten = {2, -13, -4, -3};
while (noten[i]>=0){
    sum=sum+noten[i];
    i=i+1;
}
mittelwert=sum/i;
System.out.println("mittelwert="+mittelwert);
...
```

2)

a)

```
class Loewe{
    private String name;

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

    public void setName(String pName){
        name = pName;
    }

    public String getName(){
        return(name);
    }
}
```


b1)

Bezeichnung	Adresse	Inhalt
loewenrudel	0100	?

b2)

Bezeichnung	Adresse	Inhalt
loewenrudel	0100	0200
	...	
	0200	null
	0204	null

b3)

Bezeichnung	Adresse	Inhalt	Kommentar
loewenrudel[0]	0200	0300	Objekt Leo
	...		
	0300	0400	Adresse des strings "Leo"
	0400	"Leo"	String "Leo"

b4)

Bezeichnung	Adresse	Inhalt	Kommentar
loewenrudel[1]	0204	0500	Objekt Simba
	...		
	0500	0600	Adresse des strings "Simba"
	0600	"Simba"	String "Simba"

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) Es soll ein kleines Demoprogramm zum Testen der Klasse Rechteck und deren Methoden entwickelt werden.

Dazu muß eine Klasse "Rechteck" erzeugt werden.

a) Erzeugen Sie die Klasse "Rechteck".

Diese Klasse hat genau die folgenden Attribute (Datentyp bitte selbst bestimmen) und sonst keine anderen:

seiteA, seiteB

Diese Klasse hat genau die folgenden Methoden (Datentyp und formale Parameter bitte selbst bestimmen) und sonst keine anderen:

...setSeiteA...

...setSeiteB...

...getSeiteA...

...getSeiteB...

...berechneFlaeche... (berechnet die Fläche des Rechtecks)

...vergroessere... (vergroessert alle zwei Seiten um einen bestimmten Faktor)

...printAll... (gibt die zwei Seitenlängen und die Fläche des Rechtecks auf dem Bildschirm aus)

Bemerkung:

Die Methoden müssen Zusicherungen machen (z.B. dürfen die Seitenlängen des Rechtecks nicht negativ werden)

b) Schreiben Sie einen Programmausschnitt (Demoprogramm), in dem ein Rechteck angelegt wird (der Variablenname des Rechtecks sei "tisch").

b1) Dann sollen zwei Zahlen über Tastatur eingegeben werden.

b2) Mit den entsprechenden Methoden sollen diese Zahlen die zwei Seitenlängen des Rechtecks festlegen.

b3) Mit der entsprechenden Methode sollen die zwei Seiten des Rechtecks verdoppelt werden.

b4) Mit der entsprechenden Methode sollen die zwei aktuellen Seiten und die aktuelle Fläche auf dem Bildschirm ausgegeben werden.

b5) Mit der entsprechenden Methode soll der aktuelle Umfang des Rechtecks berechnet, in einer Variable abgespeichert und dann auf dem Bildschirm ausgegeben werden.

Lösungen:

```
package e2fi_9_1_09_nach1;

import java.io.*;
import java.util.Scanner;

public class MainE2FI_9_1_09_nach1 {
    public static void main(String[] args) {
        double seite1=0;
        double seite2=0;
        double umfang=0;
        Rechteck tisch=new Rechteck();
        Scanner scanner = new Scanner(System.in);

        System.out.println("1. Seitenlaenge eingeben");
        seite1=scanner.nextDouble();
        System.out.println("2. Seitenlaenge eingeben");
        seite2=scanner.nextDouble();

        tisch.setSeiteA(seite1);
        tisch.setSeiteB(seite2);

        tisch.vergroessereSeiten(2);
        tisch.printAll();
        umfang=2*(tisch.getSeiteA()+tisch.getSeiteB());
        System.out.println("Umfang="+umfang);
    }
}
```

```
class Rechteck{
    private double seiteA;
    private double seiteB;

    public void setSeiteA(double s){
        if(s>0){ // !!! Zusicherung machen !!!
            seiteA=s;
        }
    }

    public void setSeiteB(double s){
        if(s>0){ // !!! Zusicherung machen !!!
            seiteB=s;
        }
    }

    public double getSeiteA(){
        return(seiteA);
    }

    public double getSeiteB(){
        return(seiteB);
    }

    public double berechneFlaeche(){
        return(seiteA*seiteB);
    }
}
```

```

public void vergroessereSeiten(double faktor){           6P
    if(faktor>0){ // !!! Zusicherung machen !!!
        seiteA=seiteA*faktor;
        seiteB=seiteB*faktor;
    }
}

public void printAll(){                                  6P
    System.out.println("Seite A="+seiteA);
    System.out.println("Seite B="+seiteB);
    System.out.println("Flaeche="+berechneFlaeche());
}
}

```

Bemerkungen:

1) Statt die Seiten des Rechtecks im Hauptprogramm einzugeben, ist es besser eigene Methoden in der Klasse Rechteck zu basteln (siehe oben):

```

void Rechteck::scanSeiteA();
void Rechteck::scanSeiteB();

```

2) Statt den Umfang umständlich mit:

```

umfang=2*(tisch.getSeiteA()+tisch.getSeiteB());

```

im Hauptprogramm auszurechnen, ist es besser eigens eine Methode in der Klasse Rechteck zu basteln (siehe oben):

```

double Rechteck::berechneUmfang() {

```

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

Bemerkung:

Die ganze Aufgabe muss als genau ein Programm dargestellt werden.

1)

a) 20P

Auszubildende Fachinformatiker im 1. Ausbildungsjahr haben gerade gelernt, von Serienschaltungen und Parallelschaltungen den Ersatzwiderstand zu berechnen.

Mit "Serie3" wird z.B. eine Serienschaltungen mit 3 Widerständen, mit "Parallel2" z.B. eine Parallelschaltungen mit 2 Widerständen bezeichnet.

Erstellen Sie die Klassen "Parallel2" und "Serie3" mit genau den Attributen R1 und R2 (bzw. R1, R2 und R3) und den entsprechenden Methoden wie z.B. getErsatzwiderstand().

Bemerkung:

Ersatzwiderstand Serienschaltung: $R_{\text{Ersatz}} = R1 + R2 + R3$

Ersatzwiderstand Parallelschaltung: $1/R_{\text{Ersatz}} = 1/R1 + 1/R2$

b) 16P

Da die Fachinformatiker noch nicht gelernt haben, die Leistung zu berechnen, die an so einer Schaltung erzeugt wird, sind sie sehr erfreut daran, dass die Fachinformatiker im 3.

Ausbildungsjahr die Klasse "Schaltung" gebaut haben, die eine Methode getLeistung() besitzt, die die Leistung (am Ersatzwiderstand) berechnen kann. Die Methode getLeistung() muss allerdings mit Hilfe der abstrakten Methode getErsatzwiderstand() berechnet werden.

Die entsprechende Formel dazu ist:

$\text{Leistung} = \text{Spannung}^2 / \text{Ersatzwiderstand}$

Erstellen Sie die Klasse "Schaltung" mit dem Attribut U und den entsprechenden Methoden, so dass die Fachinformatiker des 1. Ausbildungsjahres mit ihren Klassen "Parallel2" und "Serie3" die Methoden der Klasse "Schaltung" nutzen können.

c) 14P

Machen Sie im "Hauptprogramm" main() folgendes:

- c1) Erzeugen Sie eine serielle Schaltung mit den Widerstandswerten 1, 2, 7.
- c2) Erzeugen Sie eine parallele Schaltung mit den Widerstandswerten 3, 6.
- c3) An die serielle Schaltung wird eine Spannung von 10 angelegt.
- c4) An die parallele Schaltung wird eine Spannung von 4 angelegt.
- c5) Berechnen Sie die an der seriellen Schaltung verbrauchte Leistung.
- c6) Geben Sie diese verbrauchte Leistung auf dem Bildschirm aus.
- c7) Berechnen Sie die an der parallelen Schaltung verbrauchte Leistung.
- c8) Geben Sie diese verbrauchte Leistung auf dem Bildschirm aus.

Lösungen

```
public class MainAbstract6{
    public static void main(String[] args) {
        double P1,P2;
        Seriell3 mySeriell3 = new Seriell3(1,2,7);      2P
        Parallel2 myParallel2 = new Parallel2(3,6);    2P

        mySeriell3.setU(10);                          2P
        myParallel2.setU(4);                          2P
        P1=mySeriell3.getP();                          2P
        P2=myParallel2.getP();                        2P

        System.out.println("P1= "+P1);                1P
        System.out.println("P2= "+P2);                1P
    }
}

abstract class Schaltung {                          3P
    private double U;                                1P

    public Schaltung(double pU){                      3P
        U = pU;
    }

    public void setU(double pU){                      3P
        U = pU;
    }

    public double getU(){                             3P
        return(U);
    }

    public double getP(){                             3P
        double R;
        R=getErsatzwiderstand();
        return(U*U/R);
    }

    abstract public double getErsatzwiderstand();    3P
}

// Parallelschaltung mit 2 Widerständen
class Parallel2 extends Schaltung{                  3P
    private double R1, R2;                          1P

    public Parallel2(double pR1, double pR2){        1P
        super(0);
        setR1R2(pR1,pR2);
    }
}
```

```

    public void setR1R2(double pR1, double pR2){          1P
        R1=pR1;
        R2=pR2;
    }

    public double getR1(){                                1P
        return R1;
    }

    public double getR2(){                                1P
        return R2;
    }

    public double getErsatzwiderstand(){                  2P
        double temp;
        temp=1/(1/R1+1/R2);
        return temp;
    }
}

// Serienschaltung mit 3 Widerständen
class Seriell3 extends Schaltung{                        3P
    private double R1, R2, R3;                            1P

    public Seriell3(double pR1, double pR2, double pR3){  1P
        super(0);
        setR1R2R3(pR1,pR2,pR3);
    }

    public void setR1R2R3(double pR1, double pR2, double pR3){  1P
        R1=pR1;
        R2=pR2;
        R3=pR3;
    }

    public double getR1(){                                1P
        return R1;
    }

    public double getR2(){                                1P
        return R2;
    }

    public double getR3(){                                1P
        return R3;
    }

    public double getErsatzwiderstand(){                  1P
        double temp;
        temp=R1+R2+R3;
        return temp;
    }
}

```


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)

a) Was ist Polymorphie ? 5P

b) Geben Sie ein Beispiel für die Verwendung von Polymorphie
(kurze Beschreibung, aber nur in Worten) 5P

c) Warum ist es sinnvoll im Zusammenhang mit Polymorphie abstrakte Methoden zu
benutzen? 4P

d) Die Klassen A und B binden (implements) das Interface CIF ein.
Stellen Sie dies in UML dar. 4P

2)

Bewerten Sie folgende Aussagen (ohne Begründung) mit wahr oder falsch:

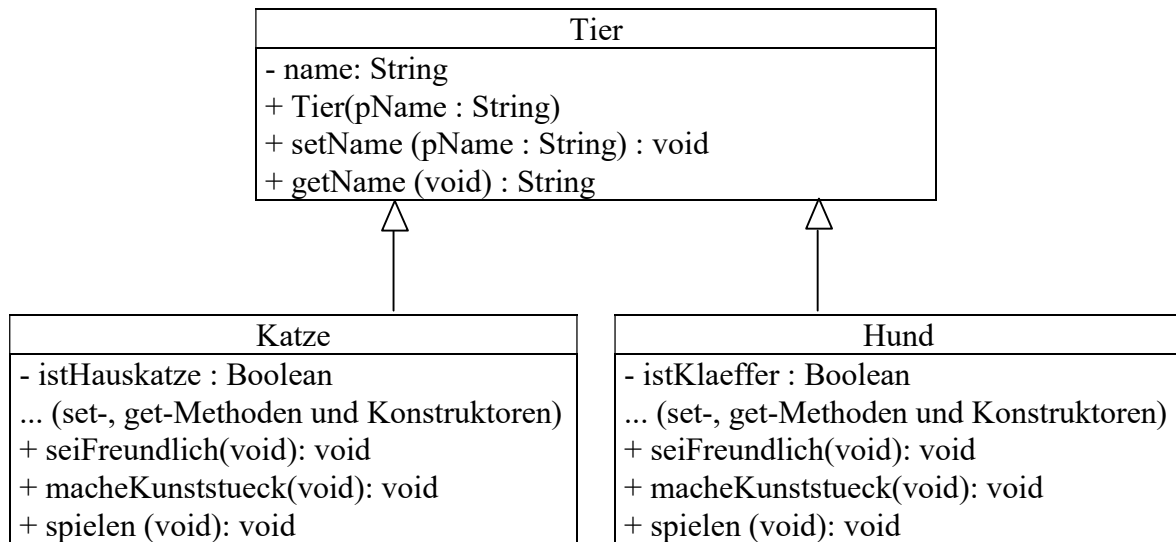
b1) Wenn eine abstrakte Methode in einer Klasse verwendet wird, dann muss diese Klasse
eine abstrakte Klasse sein. 3P

b2) Wenn eine Klasse abstrakt ist, dann muss mindestens eine sich darin befindliche
Methode abstrakt sein. 3P

b3) Alle Methoden in Interfaces sind automatisch abstract, aber nicht public 3P

b4) In Java ist Mehrfachvererbung möglich. 3P

3) Das folgende Projekt soll multipersonal entwickelt werden. Deshalb ist die Klasse Tier fest
vorgegeben und darf nicht mehr verändert werden. Hunde und Katzen sollen die Rollen von
Haustieren spielen können. Deshalb hat ein Programmierer die Methoden seiFreundlich(),
macheKunststueck() und spielen () in die Klasse Hund und Katze mit aufgenommen und
folgendes Design entworfen (siehe UML):



```

public static void main(String[] args) {
    int i;
    Tier[] tiere;
    tiere = new Tier[2];
    tiere[0]=new Hund(true, "Rex");
    tiere[1]=new Katze(false, "Kitty");
    for(i=0;i<2;i++){
        tiere[i].seiFreundlich();
        tiere[i].macheKunststueck();
        tiere[i].spielen();
    }
}

```

a) 10P

a1) Ist die obige main() Methode korrekt ? Begründen Sie.

Angenommen die main() Methode wäre falsch. Was müsste in der Klasse Tier gemacht werden, dass die main() Methode richtig wird?

b) Ein anderer Entwickler schlägt eine andere Lösung vor:

Er will für die Methoden seiFreundlich(), macheKunststueck() und spielen () das Interface HaustierIF entwerfen.

b1) Implementieren Sie das Interface HaustierIF 5P

b2) Warum funktioniert die folgende Methode main()? 5P

```

public class MainE2FI_15_6_09_nr3 {
    public static void main(String[] args) {
        int i;
        HaustierIF[] tiere;
        tiere = new HaustierIF[2];
        tiere[0]=new Hund(true, "Rex");
        tiere[1]=new Katze(false, "Kitty");
        for(i=0;i<2;i++){
            tiere[i].seiFreundlich();
            tiere[i].macheKunststueck();
            tiere[i].spielen();
        }
    }
}

```

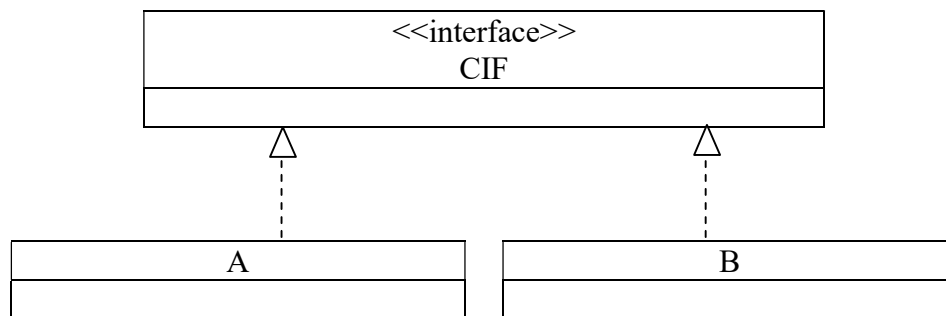
Lösungen:

1) a) Wenn eine Methode, wie z.B. `getTierwert()` für verschiedene Programmteile steht (und z.B. einmal den Tierwert (= Marktwert) verschiedener Klassen wie z.B. Henne oder das andere Mal Kuh auf dem Bildschirm ausgeben kann), dann nennt man dies Polymorphie.

b) In einem Feld werden Objekte der Klasse Dreieck und Kreis gespeichert.

Um in einer Schleife den jeweiligen Flächeninhalt auszugeben (ohne dass man weiß, ob es ein Dreieck oder ein Kreis ist), wird durch die Polymorphe automatisch die richtige Methode ausgewählt (entweder die Methode zur Flächenberechnung des Kreises oder des Dreiecks).

c) Ein Interface wird mit `<<interface>>` bezeichnet und mit einem Pfeil mit gestrichelten Linien auf das Interface gezeigt.



d)

Durch eine abstrakte Methode wird man in der Unterklasse gezwungen, die Methode zu implementieren, so dass es nicht passieren kann, dass die Methode der Oberklasse aufgerufen wird (wenn man vergessen hat, diese in der Unterklasse auszuprogrammieren).

2)

b1) wahr

b2) falsch

b3) falsch

b4) falsch

3)

a) Die Methoden `seiFreundlich()`, `makeKunststueck()` und `spielen()` gibt es in der Klasse Tier nicht (weder abstract noch nicht abstract). Also können dies auch nicht aufgerufen werden. Um diese Methode aufrufen zu können, müssen sie als (am besten abstract) Methoden in der Klasse Tier existieren.

b1)

```
public interface HaustierIF {
    void seiFreundlich();
    void makeKunststueck();
    void spielen();
}
```

b2)

Im Interface HaustierIF gibt es die abstrakten Methoden `seiFreundlich()`, `makeKunststueck()` und `spielen()`. Damit ist dann die Polymorphie realisiert, weil der Programmierer gezwungen wird, in den Klassen, die HaustierIF einbinden (mit `implements`), diese auszuprogrammieren.

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) gegeben sei folgender Programmausschnitt (Hund ist eine Klasse mit einem zweiparametrischen Konstruktor):

Ist der folgende Java-Programmausschnitt syntaktisch korrekt? 5P

Begründen Sie Ihre Antwort.

```
...  
Hund[] hunde;  
hunde[0]=new Hund("Goldi",10);  
hunde[1]=new Hund("Dicki",20);  
...
```

2) Was sind Klassenvariablen und warum benötigt man sie ? 5P

Geben Sie dazu ein Beispiel.

3) Was versteht man unter dem Begriff "Vererbung" ? 5P

Welchen Vorteil hat man beim Vererben ?

4) Ein Taschenrechner soll wie folgt funktionieren:

Die 1. Zahl, die eingegeben wird, wird zunächst einfach nur in dem Attribut ergebnis der Klasse EinfachTaschenrechner abgespeichert.

Wenn dann z.B. dividiert werden soll, wird der Wert des Attributs ergebnis durch die 2. Zahl dividiert. Wenn danach z.B. multipliziert werden soll, wird der Wert des Attributs ergebnis mit der 3. Zahl multipliziert, usw.

Beispiel: (et sei ein Objekt der Klasse EinfachTaschenrechner)

```
et.setErgebnis(10);           // ergebnis --> 10  
et.dividieren (20);           // ergebnis --> 10 / 20 = 0.5  
et.multiplizieren (100);      // ergebnis --> 100 * 0,5 = 50
```

4.1) Da der einfache Taschenrechner für den "modernen, deutschen Menschen" konzipiert wurde, soll er keine negativen Zahlen verarbeiten können (d.h. der Taschenrechner soll z.B. nicht $-4 * 5$ oder $5 - 10$ verarbeiten können). Wenn der Anwender also z.B. eine negative Zahl verarbeiten will, müssen die entsprechenden Methoden adäquat darauf reagieren. Wie kann man das programmtechnisch realisieren (keine Details, sondern nur Schlagwort(e) angeben). 3P

4.2) Implementieren Sie die Klasse EinfachTaschenrechner, d.h. genau (nur!!) das Attribut ergebnis und genau (nur!!) die folgenden Methoden (Parameter und Rückgabewert selbst überlegen):

- Konstruktor 3P
- setErgebnis(...) 3P
- getErgebnis(...) 3P
- dividieren(...) 7P

4.3) Machen Sie folgendes in der Methode main(...):

a) Erzeugen Sie in der Methode main(..) ein Objekt der Klasse EinfachTaschenrechner, 4P

b) danach soll mit Hilfe der entsprechenden Methode(n) $10 / 20$ berechnet werden

4P

c) und auf dem Bildschirm ausgegeben werden. 4P

4.4) Implementieren Sie - mit Hilfe der Verebung - die Klasse WissenschaftlicherTaschenrechner. 6P

Der Einfachheit halber soll dieser nur dividieren (Zähler, Nenner und Ergebnis der Division dürfen auch negativ sein)

Lösung:

1) Die Variable hunde zeigt auf kein Feld.

2) Da es viel Schreibaufwand macht, für jedes Exemplar einer Klasse, deren Attribute immer den gleichen Wert haben (Beispiel Kuh und Methode Literpreis) den gleichen Wert festzusetzen, ist es viel sinnvoller, dies mit einer einzigen Anweisung zu machen.

Dazu wird eine sogenannte Klassenvariable definiert. Eine Klassenvariable unterscheidet sich von einem "normalen" Attribut durch den Vorsatz "static".

3) Da es viel Schreibaufwand macht, die gleichen Methoden für die Manipulation eines Attributs (wie z.B. dem Alter und dem Namen) zu implementieren, ist es geschickter dies nur einmal in einer eigenen Klasse zu machen und diese Methoden dann in die Unterklassen zu vererben.

4)

4.1) Exceptions

4.2) bis 4.4)

Bem: Bessere Lösung wäre mit Exceptions

```
public class MainTaschenrechner1{
    public static void main(String[] args){
        double erg;
        EinfachTaschenrechner et = new EinfachTaschenrechner();
        WissenschaftlicherTaschenrechner wt = new
                                                    WissenschaftlicherTaschenrechner();

        et.setErgebnis(10);
        et.dividieren(20);
        if(et.getStatus()==0){
            erg = et.getErgebnis();
            System.out.println("Ergebnis einfacher TR= " +erg);
        }
        wt.setErgebnis(-10);
        wt.dividieren(0);
        if(wt.getStatus()==0){
            erg = wt.getErgebnis();
            System.out.println("Ergebnis wissenschaftlicher TR= " +erg);
        }
    }
}

class EinfachTaschenrechner{
    protected double ergebnis;
    protected int status;
    /*
        status = 0    alles ok
        status = -1   Probleme bei der Division
    */

    public EinfachTaschenrechner(){
        ergebnis=0;
        status=-10;
    }

    public void setErgebnis(double pErgebnis){
        ergebnis = pErgebnis;
    }

    public double getErgebnis(){
        return ergebnis;
    }
}
```

```

    public void setStatus(int pStatus){
        status = pStatus;
    }

    public double getStatus(){
        return status;
    }

    public void dividieren(double pZahl){
        if(ergebnis<0 || pZahl<=0){
            status = -1;
        }
        else{
            ergebnis = ergebnis/pZahl;
            status=0;
        }
    }
};

class WissenschaftlicherTaschenrechner extends EinfachTaschenrechner{
    public WissenschaftlicherTaschenrechner(){
        super();
    }

    public void dividieren(double pZahl){
        if(pZahl==0){
            status = -1;
        }
        else{
            ergebnis = ergebnis/pZahl;
            status=0;
        }
    }
}

```

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) Eine Digitalschaltung besteht aus mehreren Eingängen und genau einem Ausgang. Die Anzahl der Eingänge wird beim jeweiligen Erstellen eines Objekts angegeben. Eine Oder-Schaltung bzw. eine Und-Schaltung sind spezielle Digitalschaltungen, also Unterklassen der Klasse Digitalschaltung.

a) Modellieren Sie dies in einem UML-Diagramm

b) Implementieren (in Java) Sie dazu die Klassen:
Digitalschaltung, Undschaltung, Oderschaltung

c) Erzeugen Sie in der Methode main(...) eine Oder-Schaltung mit 3 Eingängen (mit den Werten 1, 0, 1). Von dieser Oderschaltung soll dann der Wert des Ausgangs berechnet werden.

Die Aufgabe soll möglichst optimal nach den Grundsätzen der OOP gelöst werden.

Bemerkung:

Wertetabelle einer Oderschaltung mit 3 Eingängen

E1	E2	E3	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

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

Bemerkung: Jede Teilaufgabe entspricht einem Programmteil. Die ganze Aufgabe soll durch **ein** Projekt realisiert werden.

I)

1) 13P

Nichtmathematikinteressierte Fachinformatiker sollen eine Klasse "GeoForm" entwickeln, die eine Methode `getOberfläche()` besitzt, die die Oberfläche (= Fläche der Unterseite + Fläche der Oberseite) eines zweidimensionalen Gebildes (wie z.B. Kreis, Rechteck, Parallelogramm, Dreieck, usw.) berechnet.

Die Methode `getOberfläche()` muß allerdings mit Hilfe der abstrakten Methode `getFlaeche()` berechnet werden, die die Fläche eines zweidimensionalen Gebildes (wie z.B. Kreis, Rechteck, Parallelogramm, Dreieck, usw.) berechnet.

Die abstrakte Methode `getUmfang()` wird auch benötigt.

Erstellen Sie die Klasse "GeoForm" den entsprechenden Methoden und - falls nötig - mit den entsprechenden Attributen.

2) 21P

Mathematikinteressierte Fachinformatiker sollen die Klassen "Rechteck" und "Kreis" entwickeln (unter Zuhilfenahme der Vererbung), mit den entsprechenden Methoden (und Nicht Standard-Konstruktoren) und mit den entsprechenden Attributen.

Bemerkung:

Umfang Kreis = $2 * \pi * \text{Radius}$

Fläche Kreis = $\pi * \text{Radius} * \text{Radius}$

3) 16P

Machen Sie im "Hauptprogramm" `main()` folgendes:

- a) Erzeugen Sie ein Rechteck Mit Länge 1 und Breite 2
- b) Erzeugen Sie einen Kreis mit Radius 3
- c) Berechnen Sie die Oberfläche des Rechtecks.
- d) Berechnen Sie den Umfang des Rechtecks.
- e) Berechnen Sie die Oberfläche des Kreises.
- f) Berechnen Sie den Umfang des Kreises.

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

$$U = 2\pi r$$

$$A = \pi r^2$$

Bemerkung:

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

Konkret:

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

Erzeugen Sie die Methode berechneFlaeche und berechneUmfang.

b) Erzeugen Sie in main zwei Kreise:

k1 mit Radius 2 und k2 mit Radius 10.

c) Geben Sie die Daten (Radius, Umfang, Fläche) von k2 auf dem Bildschirm aus

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

e) Geben Sie die Daten (Radius, Umfang, Fläche) von k2 nach der Verdopplung aus.

f) Berechnen Sie, um das Wievielfache sich dann der Flächeninhalt erhöht hat und geben Sie dieses Ergebnis auf dem Bildschirm aus.

Lösung:

```
public class MainTest4 {
    public static void main(String[] args){
        double verhaeltnis;
        // Teilaufgabe b)
        Kreis k1 = new Kreis(2);
        Kreis k2 = new Kreis(10);
        // Teilaufgabe c)
        System.out.println("k2.getRadius()= "+k2.getRadius());
        System.out.println("k2.berechneUmfang()=
                                "+k2.berechneUmfang());
        System.out.println("k2.berechneFlaeche()=
                                "+k2.berechneFlaeche());

        // Teilaufgabe d)
        k2.setRadius(k1.getRadius()*2);
        // Teilaufgabe e)
        System.out.println("k2.getRadius()= "+k2.getRadius());
        System.out.println("k2.berechneUmfang()=
                                "+k2.berechneUmfang());
        System.out.println("k2.berechneFlaeche()=
                                "+k2.berechneFlaeche());

        // Teilaufgabe f)
        verhaeltnis = k2.berechneFlaeche()/k1.berechneFlaeche();
        System.out.println("verhaeltnis= "+verhaeltnis);
    }
}

class Kreis{
    private double radius;
    public static final double pi = 3.14;

    public Kreis(double pRadius){
        radius = pRadius;
    }

    public Kreis(){
    }

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

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

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

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

Name, Vorname:

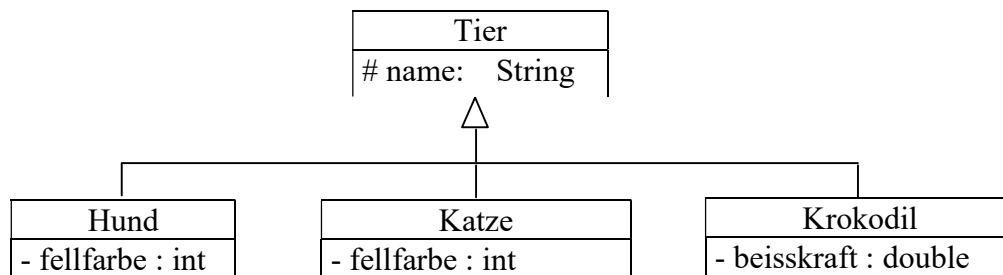
Hilfsmittel:
keine

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

1) Gegeben ist das folgende abgespeckte, **fest vorgegebene** UML-Diagramm.
Die Klassenhierarchie darf also **nicht** verändert werden.



a) 39P

Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen mit den jeweiligen Konstruktoren, set- und get-Methoden. Keine weiteren Attribute einfügen!

Die Konstruktoren müssen jeweils Parameter enthalten.

Wichtig: Jede Farbe wird durch einen Integer-Wert dargestellt, wie z.B:

... -100: weiß -99:gelb, ...

Umgekehrt entspricht jedem Integer-Wert eine Farbe!

Bitte keine "Umrechnungstabelle" der Zahlenwerte in tatsächliche Farben erstellen!

Hier ist eine Farbe ein Zahlenwert, mehr nicht!

b) 11P

Das Programm soll multipersonal entwickelt werden: Zu Testzwecken sollen die Entwickler der einzelnen Klassen gezwungen werden, die Methode

void getBeschreibung()

zu entwickeln, die die Namen der Attribute mit den zugehörigen Werten auf dem Bildschirm ausgibt.

In einem Feld sollen verschiedene Tiere (Hunde, Katzen, Krokodile) abgespeichert werden.

Danach sollen die Werte der Attribute mit der Methode getBeschreibung() ausgegeben werden. Machen Sie folgendes in der Methode main()

b1) Erzeugen Sie einen Hund, eine Katze und ein Krokodil.

b2) Speichern Sie diese 3 Tiere in dem Feld.

b3) Geben Sie mit Hilfe einer Schleife und der Methode `getBeschreibung()` die Eigenschaften der jeweiligen Tiere auf dem Bildschirm aus.

Lösung:

public class MainKlassenarbeit {		11P
public static void main(String[] args) {		
int i;		
Katze katze;		
Hund hund;		
Krokodil krokodil;		
Tier tiere[];	1P	
tiere = new Tier[3];	1P	
katze = new Katze("Ute", 2);	1P	
tiere[0] = katze;	1P	
hund = new Hund("Rex", 3);	1P	
tiere[1] = hund;	1P	
krokodil = new Krokodil("Krok", 30);	1P	
tiere[2] = krokodil;	1P	
System.out.println("Beschreibung der Tiere:");		
for(i=0; i<tiere.length; i++){	3P	
System.out.println(tiere[i].getBeschreibung());		
}		
}		
}		
 abstract class Tier{	1P	11P
protected String name;	1P	
 public Tier(String pName){	2P	
name = pName;		
}		
 public void setName(String pName){	2P	
name=pName;		
}		
 public String getName(){	2P	
return(name);		
}		
 abstract public String getBeschreibung();	3P	
}		
 class Katze extends Tier {		10P
int fellfarbe;		
 public Katze(String pName, int pFellfarbe){		
super(pName);		
fellfarbe = pFellfarbe;		
}		
 public String getBeschreibung(){		
String s;		
s="Name="+name+" Fellfarbe="+fellfarbe;		
return s;		
}		
 public void setFellfarbe(int pFellfarbe){		
fellfarbe = pFellfarbe;		
}		

```

    public int getFellfarbe() {
        return fellfarbe;
    }
}

```

```

class Hund extends Tier {
    private int fellfarbe;

```

10P

```

    public Hund(String pName, int pFellfarbe) {
        super(pName);
        fellfarbe = pFellfarbe;
    }

```

```

    public String getBeschreibung() {
        String s;
        s="Name="+name+" Fellfarbe="+fellfarbe;
        return s;
    }

```

```

    public void setFellfarbe(int pFellfarbe) {
        fellfarbe = pFellfarbe;
    }

```

```

    public int getFellfarbe() {
        return fellfarbe;
    }
}

```

```

class Krokodil extends Tier{
    private double beisskraft;

```

10P

```

    Krokodil(String pName, double pBeisskreaft){
        super(pName);
        beisskraft = pBeisskreaft;
    }

```

```

    public String getBeschreibung() {
        String s;
        s="Name="+name+" Beisskraft="+beisskraft;
        return s;
    }

```

```

    public void setBeisskraft(double pBeisskraft) {
        beisskraft = pBeisskraft;
    }

```

```

    public double getBeisskraft() {
        return beisskraft;
    }
}

```

Name, Vorname:

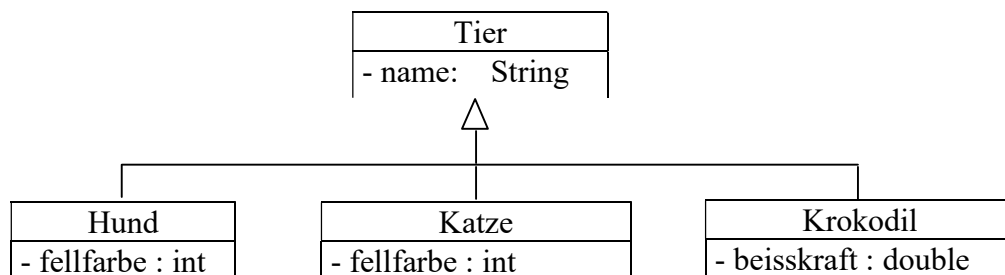
Hilfsmittel:
keine

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) Gegeben ist das folgende abgespeckte, **fest vorgegebene** UML-Diagramm.
Die Klassenhierarchie darf also **nicht** verändert werden.



a) Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen mit den jeweiligen Konstruktoren, set- und get-Methoden. Keine weiteren Attribute einfügen!
Wichtig: Jede Farbe wird durch einen Integer-Wert dargestellt, wie z.B:

-100: weiß 1:gelb, usw.

Umgekehrt entspricht jedem Integer-Wert eine Farbe!

39P

Bitte keine "Umrechnungstabelle" der Zahlenwerte in tatsächliche Farben erstellen!

Hier ist eine Farbe ein Zahlenwert, mehr nicht!

b)

11P

Das Programm soll multipersonal entwickelt werden: Zu Testzwecken sollen die Entwickler der einzelnen Klassen gezwungen werden, die Methode
void getBeschreibung()

zu entwickeln, die die Namen der Attribute mit den zugehörigen Werten auf dem Bildschirm ausgibt.

In einem Feld sollen verschiedene Tiere (Hunde, Katzen, Krokodile) abgespeichert werden.

Danach sollen die Werte der Attribute mit der Methode getBeschreibung() ausgegeben werden. Machen Sie folgendes in der Methode main()

b1) Erzeugen Sie einen Hund, eine Katze und ein Krokodil.

b2) Speichern Sie diese 3 Tiere in dem Feld.

b3) Geben Sie mit Hilfe einer Schleife und der Methode `getBeschreibung()` die Eigenschaften der jeweiligen Tiere auf dem Bildschirm aus.

c)

12,5P

Beachten Sie:

Ein Krokodil hat - nach heutigem biologischen Wissenstand - kein Fell.

Dagegen besitzt eine Katze bzw. ein Hund ein Fell.

In dem Feld "felktiere" sollen die im vorigen Programmteil erstellte Katze und Hund abgespeichert werden und dann mit Hilfe einer Schleife und der Methode `getFellfarbe()` die Farbe des jeweiligen Tiers auf dem Bildschirm ausgegeben werden.

c1) Was wäre der Nachteil der Lösung, in der man die Methode `getFellfarbe()` in der Klasse `Tier` implementiert?

c2) Beurteilen Sie die folgende Lösung:

Die Methode `getFellfarbe()` wird nur (und sonst nirgends) in die Klasse `Hund` und `Katze` implementiert.

c3) Was ist die beste Lösung ? (keine Implementierung, nur verbale Beschreibung und Begründung).

Bewertung:

62,5P	Note 1
50P	Note 2
37,5P	Note 3
25P	Note 4
12,5P	Note 5
0P	Note 6

Lösung:

```
public class MainKlassenarbeit {                                     11P
    public static void main(String[] args) {
        int i;
        Katze katze;
        Hund hund;
        Krokodil krokodil;
        Tier tiere[];                                             1P
        tiere = new Tier[3];                                     1P
        katze = new Katze("Ute", 2);                             1P
        tiere[0] = katze;                                       1P
        hund = new Hund("Rex", 3);                               1P
        tiere[1] = hund;                                         1P
        krokodil = new Krokodil("Krok", 30);                    1P
        tiere[2] = krokodil;                                     1P
        System.out.println("Beschreibung der Tiere:");
        for(i=0;i<3;i++){                                       3P
            System.out.println(tiere[i].getBeschreibung());
        }
    }
}

abstract class Tier{                                             1P
    protected String name;                                     1P

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

    public void setName(String pName){                           2P
        name=pName;
    }

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

    abstract public String getBeschreibung();                   3P
}

class Katze extends Tier {                                       10P
    int fellfarbe;

    public Katze(String pName, int pFellfarbe){
        super(pName);
        fellfarbe = pFellfarbe;
    }

    public String getBeschreibung(){
        String s;
        s="Name="+name+" Fellfarbe="+fellfarbe;
        return s;
    }

    public void setFellfarbe(int pFellfarbe){
        fellfarbe = pFellfarbe;
    }
}
```

```

    public int getFellfarbe() {
        return fellfarbe;
    }
}

```

```

class Hund extends Tier {
    private int fellfarbe;

```

10P

```

    public Hund(String pName, int pFellfarbe) {
        super(pName);
        fellfarbe = pFellfarbe;
    }

```

```

    public String getBeschreibung() {
        String s;
        s="Name="+name+" Fellfarbe="+fellfarbe;
        return s;
    }

```

```

    public void setFellfarbe(int pFellfarbe) {
        fellfarbe = pFellfarbe;
    }

```

```

    public int getFellfarbe() {
        return fellfarbe;
    }
}

```

```

class Krokodil extends Tier{
    private double beisskraft;

```

10P

```

    Krokodil(String pName, double pBeisskreaft){
        super(pName);
        beisskraft = pBeisskreaft;
    }

```

```

    public String getBeschreibung() {
        String s;
        s="Name="+name+" Beisskraft="+beisskraft;
        return s;
    }

```

```

    public void setBeisskraft(double pBeisskraft) {
        beisskraft = pBeisskraft;
    }

```

```

    public double getBeisskraft() {
        return beisskraft;
    }
}

```

c1)

4P

Dadurch erbt die Klasse Krokodil u.a. die Methode getFellfarbe().

Dadurch kann man die Fellfarbe eines Krokodils herausfinden, obwohl ein Krokodil keine Fellfarbe besitzt.

c2) Beurteilen Sie die folgende Lösung:

4P

Die Methode getFellfarbe() werden nur (und sonst nirgends) in die Klasse Hund und Katze implementiert.

Da in einer Schleife z.B. mit felltiere. getFellfarbe() die Farbe abgefragt wird, muss die Methode auch in der Klasse Tier existieren. Da dies nicht der Fall ist, meldet der Compiler einen Fehler.

c3)

4,25P

Katze und Hund implementieren das Interface FelltierIF, wo die Methodenköpfe getFellfarbe() und setFellfarbe(...) angegeben werden.

Nur in den Klassen Hund und Katze müssen diese Methoden dann ausprogrammiert werden.