

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

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

Konkret:

a) (22P)

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

b) (4P)

Erzeugen Sie in main zwei Kreise:
k1 mit Radius 2 und k2 mit Radius 10.

c) (6P)

Geben Sie die Daten (Radius, Umfang, Fläche) von k2 (durch Verwendung der entsprechende(n) Methoden) auf dem Bildschirm aus.

d) (8P)

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)

(6P)

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

f)

(4P)

Berechnen Sie (durch Verwendung der entsprechende(n) Methoden), um das Wievielfache sich dann der Flächeninhalt erhöht hat und geben Sie dieses Ergebnis auf dem Bildschirm aus.

Lösung:

```
public class MainTest4 {
    public static void main(String[] args){
        double verhaeltnis;
        // Teilaufgabe b)
        Kreis k1 = new Kreis(2);                // 2P
        Kreis k2 = new Kreis(10);               // 2P
        // Teilaufgabe c)
        System.out.println("k2.getRadius()= "+k2.getRadius()); // 2P
        System.out.println("k2.berechneUmfang()="+k2.berechneUmfang()); //2P
        System.out.println("k2.berechneFlaeche()="+k2.berechneFlaeche()); //2P
        // Teilaufgabe d)
        k2.setRadius(k1.getRadius()*2); // 8P
        // Teilaufgabe e)
        System.out.println("k2.getRadius()= "+k2.getRadius()); // 2P
        System.out.println("k2.berechneUmfang()="+k2.berechneUmfang()); // 2P
        System.out.println("k2.berechneFlaeche()="+k2.berechneFlaeche()); //2P
        // Teilaufgabe f)
        verhaeltnis = k2.berechneFlaeche()/k1.berechneFlaeche(); // 4P
        System.out.println("verhaeltnis= "+verhaeltnis); // 2P
    }
}
```

```
class Kreis{
    private double radius;                // 2P
    public static final double pi = 3.14;

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

    public Kreis(){                      // 2P
    }

    public void setRadius(double pRadius){ // 4P
        radius = pRadius;
    }

    public double getRadius(){           // 4P
        return(radius);
    }

    public double berechneUmfang(){      // 4P
        return(2*pi*radius);
    }

    public double berechneFlaeche(){    // 4P
        return(pi*radius*radius);
    }
}
```

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

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

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

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

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

        // b5)
        // Radius=13
        // kreisSicherung[10] und kreise [10] zeigen auf das
        // 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) 15P
Erstellen Sie die Klasse Konto mit genau (nur) dem Attribut `kontostand` und den entsprechenden set- und get-Methoden und einem Konstruktor mit genau einem Parameter.

2) Die Klasse Konto existiere schon. Sie enthält der Einfachheit halber nur das Attribut `kontostand` und die entsprechenden set- und get-Methoden.

a) 5P
Was veranlaßt dieser Programmausschnitt im Arbeitsspeicher?
Beschreiben Sie dies mit Hilfe einer Zeichnung oder mit Worten

```
...  
int i;  
Konto[] konten = new Konto[10];  
for(i=0;i<10;i++){  
    konten[i] = new Konto(i);  
}  
...
```

b) 10P
Die Kontenstände aller Konten sollen verdreifacht werden.
Beurteilen Sie dazu folgende Lösungen, d.h:
Was veranlaßt dieser Programmausschnitt im Arbeitsspeicher?
Beschreiben Sie dies mit Hilfe einer Zeichnung oder mit Worten

```
...  
for(i=0;i<10;i++){  
    konten[i].setKontostand(konten[i].getKontostand()*3);  
}  
...
```

oder

```
...  
for(i=0;i<10;i++){  
    konten[i]=new Konto(konten[i].getKontostand()*3);  
}  
...
```

c) 5P

Die obigen Konten sollen in einem Feld mit dem Namen `kontenSic` zusätzlich nochmals (durch das folgende Programm) gespeichert (gesichert) werden:

```
...
Konto[] kontenSic = new Konto[10];
for(i=0;i<10;i++){
    kontenSic[i] = konten[i];
}
// Das war die Sicherung. Jetzt kommen 2 Programmierzeilen:
kontenSic[7].setKonto(123);
System.out.println(konten[7].getKontostand());
```

Was wird auf dem Bildschirm ausgegeben? Begründen Sie !

d) 5P

Statt c) wird folgende Lösung angeboten:

```
...
Konto[] kontenSic = new Konto[10];
for(i=0;i<10;i++){
    kontenSic[i] = new Konto(konten[i].getKontostand());
}
// Das war die Sicherung. Jetzt kommen 2 Programmierzeilen:
// Was wird auf dem Bildschirm ausgegeben?
kontenSic[7].setKontostand(123);
System.out.println(konten[7].getKontostand());
```

Was wird auf dem Bildschirm ausgegeben? Begründen Sie !

e) 10P

Statt c) wird folgende Lösung angeboten:

```
...
Konto[] kontenSic;
kontenSic = konten;
for(i=0;i<10;i++){
    kontenSic[i] = konten[i];
}
// Das war die Sicherung. Jetzt kommen 2 Programmierzeilen:
kontenSic[7].setKonto(123);
System.out.println(konten[7].getKontostand());
```

Was wird auf dem Bildschirm ausgegeben? Begründen Sie !

Lösung

1)

```
class Konto {
    double kontostand;

    public Konto(double pKontostand) {
        kontostand = pKontostand;
    }

    public double getKontostand() {
        return kontostand;
    }

    public void setKontostand(double kontostand) {
        this.kontostand = kontostand;
    }
}
```

2a)

```
int i;
Konto[] konten = new Konto[10];
for(i=0; i<10; i++) {
    konten[i] = new Konto(i);
}
```

Im Arbeitsspeicher werden 10 Objekte (Speicherplätze) reserviert.
konten[0], ..., konten[9] sind Zeiger auf diese Objekte.

b)

b1)

```
for(i=0; i<10; i++) {
    konten[i].setKontostand(konten[i].getKontostand() * 3);
}
```

Die jeweiligen "Schubladen" kontostand dieser Objekte werden verdreifacht.

b2)

```
for(i=0; i<10; i++) {
    konten[i] = new Konto(konten[i].getKontostand() * 3);
}
```

Im Arbeitsspeicher werden wieder 10 Objekte (Speicherplätze) reserviert. Diese haben dann natürlich andere Adressen als die alten Objekte.

konten[0], ..., konten[9] werden mit den neuen Adressen dieser (neuen) Objekte überschrieben.

Auf die alten Objekte kann nicht mehr zugegriffen werden. Sie werden irgendwann von der javaeigenen Müllabfuhr (= Garbage Collector) entsorgt.

c)

```
Konto[] kontenSic = new Konto[10];
for(i=0;i<10;i++){
    kontenSic[i] = konten[i];
}
kontenSic[7].setKonto(123);
System.out.println(konten[7].getKontostand());
```

kontenSic[7] zeigt auf das gleiche Objekt wie konten[7]. Da der Kontostand des Objekts auf das kontenSic[7] zeigt, auf 123 gesetzt wurde, ist also der Kontostand des Objekts auf das konten[7] zeigt (es zeigt auf das gleiche Objekt!) auch 123. Also wird 123 ausgegeben.

d)

```
Konto[] kontenSic = new Konto[10];
for(i=0;i<10;i++){
    kontenSic[i] = new Konto(konten[i].getKontostand());
}
kontenSic[7]. setKontostand(123);
System.out.println(konten[7].getKontostand());
```

Da für kontenSic[7] jetzt neuer Speicherplatz reserviert wurde (er zeigt nicht schon auf vorhandenen, reservierten Speicher), der mit konten[7].getKontostand(), also 21 vorbelegt wurde, wird 21 ausgegeben.

e)

```
Konto[] kontenSic;
kontenSic = konten;
```

konten ist ein Zeiger auf einen Speicherplatz, der wiederum aus Zeigern auf die Elemente des Feldes besteht.

Für das Feld kontenSic wird nicht eigener Speicher reserviert, sondern der Zeiger kontenSic zeigt auf den gleichen Speicher wie der Zeiger konten .

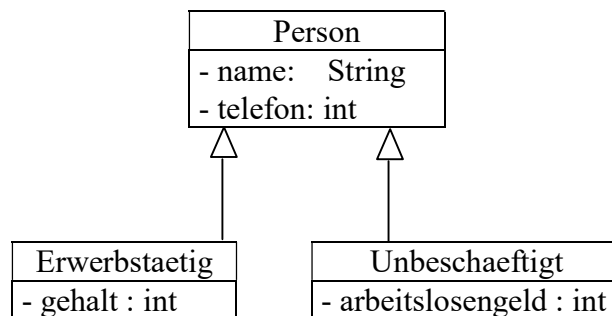
*Name, Vorname:*Hilfsmittel:
keine

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_11_4_11_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 Punktabzug !!!!

AUFGABEN

1) 6P

a) Was ist Polymorphie ?

b) Warum ist es sinnvoll im Zusammenhang mit Polymorphie abstrakte Methoden zu benutzen?

2) 12P

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

3) 14P

Erzeugen Sie die Klasse Rechteck mit genau den Attribut (und nur diese Attributen) "breite", "laenge" den dazugehörigen get-und set-Methoden und einem Konstruktor.

4) 18P

Erzeugen Sie einen Kreis (mit Radius 1) und ein Rechteck (mit Breite 2 und Länge 3) und speichern diese in einem Feld.

Dann geben Sie mit Hilfe der Polymorphie den Flächeninhalt und den Umfang auf dem Bildschirm aus.

Erzeugen Sie so ein Design, daß der Programmierer gezwungen wird, in den jeweiligen Klassen die Methoden zur Berechnung der Fläche und des Umfangs zu implementieren.

Bemerkung:

$$U = 2\pi r$$

$$A = \pi r^2$$

Lösung:

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) 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).

```
public class MainE2fi_28_06_11_nr1 {  
    public static void main(String[] args) {  
        int i;  
        GeoForm[] feld;  
        feld = new GeoForm[2];  
        feld[0]=new Rechteck(2,3);  
        feld[1]=new Kreis(1);  
        for (i = 0; i < 2; i++) {  
            System.out.println("Fläche des"+i+"-ten Objekts= "+feld[i].berechneFlaeche());  
            System.out.println("Umfang des"+i+"-ten Objekts= "+feld[i].berechneUmfang());  
        }  
    }  
}
```

ges: 10P
2P
2P
2P
2P
4P

```
class Rechteck extends GeoForm {  
    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));  
    }  
}
```

3P ges: 14P
0,5P
0,5P
2P
1P
1P
1P
1P
2P
2P

```
}  
}
```

```
class Kreis extends GeoForm {  
    private double radius;
```

3P ges: 12P
1P

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

2P

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

1P

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

1P

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

2P

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

2P

```
abstract class GeoForm {  
    abstract public double berechneFlaeche();  
    abstract public double berechneUmfang();  
}
```

4P ges: 8P
2P
2P

Name, Vorname:

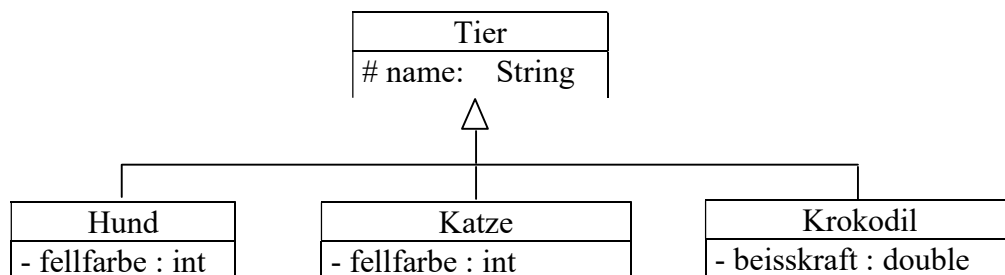
Hilfsmittel:
keine

Hinweise (unbedingt beachten):

- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote Farbe darf nicht benutzt werden.
- Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- Selbsterklärende Variablennamen benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden 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) 31P
Implementieren Sie (unter Zuhilfenahme des obigen folgenden UML-Diagramms) die Klassen Hund und Krokodil (Klasse Katze bitte nicht implementieren: Sie wird analog Klasse Hund implementiert!) mit den jeweiligen Konstruktoren, set- und get-Methoden. Keine weiteren Attribute einfügen!

Aus dem obigen UML-Diagrammen soll nicht hervorgehen, ob es sich um abstrakte oder "normale" Klassen handelt.

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

-100: weiß 1:gelb, usw.

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)

8P

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)

12P

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).

Lösung:

public class MainKlassenarbeit {		8P
public static void main(String[] args) {		
int i;		
Tier tiere[];	1P	
tiere = new Tier[3];	1P	
tiere[0] = new Katze("Ute", 2);	1P	
tiere[1] = new Hund("Rex", 3);	1P	
tiere[2] = new Krokodil("Krok", 30);	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 Hund extends Tier {		10P
private int fellfarbe;		
 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;

    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) 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) 4P
 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.