

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

AUFGABEN

1) 14P

- a) Was ist eine Klasse ?
- b) Was ist ein Objekt?
- c) Was enthält eine Klasse?
- d) Was ist der Sinn einer Methode?
- e) Was ist ein Attribut?
- f) Was bedeutet private?
- g) Was bedeutet public?

2) 10P

a) 2P

Wann und wie wird ein Konstruktor aufgerufen?

Bitte Beispiel geben (nur Aufruf angeben, keine Klasse implementieren).

b) 3P

Herr X behauptet : "Man braucht keine set-Methoden, da der Konstruktor genau das gleiche machen kann". Nehmen Sie dazu Stellung.

c) 3P

Angenommen, ein Programmierer hat keinen Konstruktor erzeugt.

Wann gibt es eine Fehlermeldung beim Kompilieren?

d) 2P

Was geschieht, wenn die Attribute einer Klasse nicht initialisiert werden und ein Objekt erzeugt wird? (Fehlermeldung?, Wert der Attribute ?, ...). Bitte genaue Beschreibung.

Auf der Rückseite geht es weiter !!!!!!!

3) (Alle Teilaufgaben müssen in **einem** Programm realisiert werden) 28P

a) Erstellen Sie die Klasse Konto, (mit genau 2 Attributen, genau 2 set-Methoden, genau 2 get-Methoden und genau 2 Konstruktoren), die ein Sparkonto mit einem Kontostand und einem Zinssatz repräsentieren soll.

b) Erzeugen Sie ein Konto mit dem Kontostand von 5 (Euro) und einem Zinssatz von 3%

c) Angenommen, man kennt nicht den aktuellen Kontostand und den Zinssatz .

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

d) Ermitteln Sie mit Hilfe der entsprechenden Methoden den neuen Kontostand und den neuen Zinssatz und geben diese auf dem Bildschirm aus.

e) Erzeugen Sie ein anderes, neues Konto mit dem Kontostand von 10 (Euro) und einem Zinssatz von 5%.

Lösung:

1) 14 Punkte (jede Teilaufgabe 2 Punkte)

- a) Ein Bauplan, nach dem ein Objekt erstellt wird
- b) Ein nach einem Bauplan im Arbeitsspeicher angelegter, reservierter Speicher.
- c) Attribute und Methoden.
- d) Attribute zu lesen und zu beschreiben.
- e) Daten (Eigenschaften) in einer Klasse.
- f) Auf private Member darf nur innerhalb einer Klasse zugegriffen werden.
- g) Auf public Member darf innerhalb und außerhalb einer Klasse zugegriffen werden.

2) 10 Punkte

a) 2P

Ein Konstruktor wird aufgerufen, wenn ein Objekt mit new erzeugt wird.

Hund hund1 = new Hund("rex",45);

b) 3P

Diese Ansicht ist falsch:

Mit einem Konstruktor kann man nur einmal (beim Erzeugen des Objekts) die Attribute dieses Objekts festlegen. Will man diese später verändern, braucht man eine set-Methode.

c) 3P

Wenn der Programmierer einen Konstruktor mit mindestens einem Parameter erstellt und damit ein dazu entsprechendes Objekt erzeugt.

d) 2P

Es gibt keine Fehlermeldung, denn die Attribute des angelegten Objekts werden standardmäßig mit 0 vorbelegt.

3)

```
public class MainKonto1 {
    public static void main(String[] args) {
        Konto k1, k2;
        // 2 P
        k1= new Konto(5,3);
        // 6 P
        k1.setKontostand(k1.getKontostand()+1000);
        k1.setZinssatz(k1.getZinssatz()-2);
        // 4 P
        System.out.println("neuer Kontostand="
                           "+k1.getKontostand());
        System.out.println("neuer Zinssatz="
                           "+k1.getZinssatz());

        // 2 P
        k2= new Konto(10,5);
    }
}
```

```
class Konto{
    // 2 Punkte
    private double kontostand;
    private double zinssatz;

    // 2 Punkte
    public Konto(){
        kontostand = 0;
        zinssatz = 0;
    }

    // 2 Punkte
    public Konto(double pKontostand, double pZinssatz){
        kontostand = pKontostand;
        zinssatz = pZinssatz;
    }

    // 2 Punkte
    public double getKontostand() {
        return kontostand;
    }

    // 2 Punkte
    public double getZinssatz() {
        return zinssatz;
    }

    // 2 Punkte
    public void setKontostand(double kontostand) {
        this.kontostand = kontostand;
    }

    // 2 Punkte
    public void setZinssatz(double zinssatz) {
        this.zinssatz = zinssatz;
    }
}
```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

AUFGABEN

I)

52P

Bemerkung:

Alle folgenden Teilaufgaben müssen (soweit sie die Programmierung betreffen) in genau **einem** einzigen Programm realisiert werden.

Zur Information:

Eine zweispurige Strasse wurde auf einer Teilstrecke von einem Kilometer Länge durch einen Bergrutsch so beeinträchtigt, daß nur noch eine Spur zu befahren ist. Um den Verkehr zu steuern wurden 2 Ampeln aufgestellt, so daß die Teilstrecke abwechselnd immer wieder in eine Richtung und dann in die andere Richtung befahren werden kann.

Wenn die Ampelanlage gestartet wird, ist jedem Zeitpunkt auf jeder Ampel genau eine der 3 Farben rot, gelb oder grün zu sehen

Jede Ampel hat die 4 Zustände 1 (rot) , 2 (grün) , 12 (rot-gelb) 21 (grün-gelb), die bekanntermaßen nach folgender Vorschrift auf einander folgen:

1 (rot) --> 12 (rot-gelb) --> 2 (grün) --> 21 (grün-gelb) --> 1 (rot) --> usw.

Außerdem hat jede Ampel 3 Lampen, von denen jede die 4 Farben rot, gelb, grün oder schwarz annehmen kann, wobei schwarz eine ausgeschaltete Lampe bedeutet.

Der Zustand 1 bedeutet: rote Lampe an, restliche Lampen schwarz.

Der Zustand 2 bedeutet: grüne Lampe an, restliche Lampen schwarz.

Der Zustand 12 bedeutet: gelbe Lampe an, restliche Lampen schwarz.

Der Zustand 21 bedeutet: gelbe Lampe an, restliche Lampen schwarz.

Es gibt die 3 Klassen "Lampe", "Ampel" und "Steuerung".

1) 9P
Implementieren Sie die Klasse "Lampe" mit genau (und nur) dem Attribut farbe und dem Datentyp String. Zusätzlich müssen noch folgende Methoden erzeugt werden:

- a) ... getFarbe(...) :
gibt die Farbe einer Lampe zurück.
- b) ... einschalten(...) :
setzt eine Lampe auf eine bestimmte Farbe.
- c) ... ausschalten(...) :
schaltet eine Lampe aus
- d) einen Konstruktor der Klasse

2) 32P
Die Klasse Ampel hat als Attribute genau 3 Lampen und das Attribut "zustand". Implementieren Sie die Klasse "Ampel" mit genau (und nur) diesen Attributen. Zusätzlich müssen noch folgende Methoden erzeugt werden:

- a) ...setZustand(...) :
Beachten Sie, daß beim Setzen eines Zustand zusätzlich noch die zugehörige Farbe der entsprechenden Lampe eingeschaltet werden muß.
- b) ... getZustand(...) :
gibt den Zustand der Ampel zurück
- c) ...rotSchalten(...) :
schaltet die rote Lampe auf rot und die anderen aus.
- d) ... gelbSchalten(...) :
entsprechendes für die gelbe Lampe.
- e) ... grünSchalten(...) :
entsprechendes für die grüne Lampe.
- f) ... weiterSchalten(...) :
schaltet eine Ampel in den nächsten Zustand.
- g) einen Konstruktor der Klasse

3) 10P
Die Klasse Steuerung hat als Attribute genau 2 Ampeln. Implementieren Sie die Klasse "Steuerung" mit genau (und nur) diesen Attributen. Zusätzlich müssen noch folgende Methoden erzeugt werden:

- a) einen Konstruktor der Klasse
- b) ...start(...) :
beginnt den Steuerungsvorgang:
Ampel1 rot, Ampel2 grün --> Ampel1 gelb, Ampel2 gelb --> Ampel1 grün, Ampel2 rot --> Ampel1 gelb, Ampel2 gelb ---> usw.

Lösung:

```
class Lampe{ // 9P
    private String farbe; // 1P

    public String getFarbe(){ // 2P
        return farbe;
    }

    public void einschalten(String pFarbe){ // 2P
        farbe=pFarbe;
    }

    public void ausschalten(){ // 2P
        farbe="schwarz";
    }

    public Lampe(){ // 2P
        farbe="schwarz";
    }
}

class Ampel{ // 32P
    private Lampe grünLampe; // 1P
    private Lampe gelbLampe; // 1P
    private Lampe rotLampe; // 1P
    private int zustand; // 1P

    public Ampel(){ // 3P
        grünLampe = new Lampe();
        gelbLampe = new Lampe();
        rotLampe = new Lampe();
    }

    public void setZustand(int pZustand){ // 8P
        if(pZustand == 1){ // rot
            rotSchalten();
        }
        else if(pZustand == 12){ // rot-gelb
            gelbSchalten();
        }
        else if(pZustand == 2){ // grün
            grünSchalten();
        }
        else if(pZustand == 21){
            gelbSchalten();
        }
        else { // Programmierfehler
            zustand=-1;
            return;
        }
        zustand=pZustand;
    }
}
```

```

public int getZustand(){                                // 2P
    return zustand;
}

private void rotSchalten(){                             // 3P
    grünLampe.ausschalten();
    gelbLampe.ausschalten();
    rotLampe.einschalten("rot");
}

private void gelbSchalten(){                           // 3P
    grünLampe.ausschalten();
    gelbLampe.einschalten("gelb");
    rotLampe.ausschalten();
}

private void grünSchalten(){                          // 3P
    grünLampe.einschalten("grün");
    gelbLampe.ausschalten();
    rotLampe.ausschalten();
}

public void weiterSchalten(){                          // 6P
    if(zustand == 1){
        setZustand(12);
    }
    else if(zustand == 12){
        setZustand(2);
    }
    else if(zustand == 2){
        setZustand(21);
    }
    else if(zustand == 21){
        setZustand(1);
    }
    else { // Programmierfehler
        zustand=-1;
    }
}
}

```



```
class Steuerung{ // 10P
    private Ampel ampel1; // 1P
    private Ampel ampel2; // 1P

    public Steuerung(){ // 2P
        ampel1 = new Ampel();
        ampel2 = new Ampel();
    }

    public void start(){ // 6P
        ampel1.setZustand(1);
        ampel2.setZustand(2);

        while (true){
            ampel1.weiterSchalten();
            ampel2.weiterSchalten();
        }
    }

    public void stopp(){
        ampel1.setZustand(1);
        ampel2.setZustand(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

I) 52P

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

1)

a) 35P

Erstellen Sie die Klasse Bruch, (mit genau den 2 Attributen zaehler und nenner, genau 2 set-Methoden, genau 2 get-Methoden und genau einem Konstruktor mit 2 Parametern), die eine mathematische Bruchzahl mit einem ganzzahligen Zähler und einem ganzzahligen Nenner repräsentieren soll.

Zusätzlich sollen genau noch folgende Methoden (und nur diese) erzeugt werden:

b) 2P

... printBruch (...)

gibt Zähler und Nenner in Bruchschreibweise auf den Bildschirm aus.

8 / 40

c) 4P

... kuerzen(int teiler)

kürzt den Bruch durch den wert teiler

Beispiel: 8 / 40 --- Teiler 4 --> 2 / 10

d) 4P

... alsKommazahl(...)

stellt den Bruch durch eine Kommazahl dar.

Beispiel: 1/2 ---> 0.5

e) 4P

... void quadrieren()

bildet das Quadrat eines Bruchs.

Beispiel: 2 / 3 ---> 4 / 9

f) 4P

... Bruch multiplizieren (Bruch bruch)

multipliziert 2 Brüche miteinander

Beispiel: $3/4 * 5/6 \rightarrow 15/24$

Tipp: $\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$

g) 4P

... Bruch kleiner Vergleich (Bruch bruch)

vergleicht 2 Brüche bzgl ihrer Größe miteinander und gibt den kleineren Bruch zurück.

Beispiel: $3/4, 2/3 \rightarrow 2/3$

2)

17P

Realisieren Sie programmtechnisch (mit Hilfe der entsprechenden Methode der Klasse Bruch) folgende Aufgaben in der Methode main(..) der Startklasse:

a) 3P

Legen Sie die Brüche b0 (Wert: 7/8), b1 (Wert: 2/3) und b2 (Wert : 4/5) an.

b) 2P

Geben Sie den Bruch b0 auf dem Bildschirm aus.

c) 3P

Speichern Sie die Kommazahl (mit der entsprechenden Methode) von b0 in der Variablen wert ab.

d) 3P

Berechnen Sie (mit der entsprechenden Methode) das Quadrat von b0.

Wo wird das Ergebnis gespeichert?

e) 3P

Berechnen Sie (mit den entsprechenden Methoden) $b1 * b2$ und speichern Sie das Ergebnis in der Variablen b4 (mit dem Datentyp Bruch) ab.

f) 3P

Berechnen Sie (mit den entsprechenden Methoden), ob b1 kleiner als b2 ist und speichern Sie den kleineren Bruch in das Ergebnis in der Variablen b5 (mit dem Datentyp Bruch) ab.

Lösungen:

1)

```
package bruch2;
public class Bruch2 {                                // gesamt: 17P
    public static void main(String[] args) {
        Bruch b0=new Bruch(7,8);                    // 1P
        Bruch b1=new Bruch(2,3);                    // 1P
        Bruch b2=new Bruch(4,5);                    // 1P
        Bruch b4;
        double wert;

        b0. printBruch();                            // 2P
        wert=b0.alsKommazahl();                      // 3P
        b0.quadrieren();                             // 3P
        b4=b1.multiplizieren(b2);                   // 3P
        b5=b1.kleinerVergleich (b2);                // 3P
    }
}

class Bruch{                                         // gesamt: 35P
    private int zaehler;                             // 1P
    private int nenner;                              // 1P

    public Bruch(int zaehler, int nenner) {          // 3P
        this.zaehler = zaehler;
        this.nenner = nenner;
    }

    public int getZaehler() {                        // 2P
        return zaehler;
    }

    public void setZaehler(int zaehler) {            // 2P
        this.zaehler = zaehler;
    }

    public int getNenner() {                         // 2P
        return nenner;
    }

    public void setNenner(int nenner) {              // 2P
        this.nenner = nenner;
    }

    public void printBruch(){                        // 2P
        System.out.print(zaehler+"/"+nenner);
    }

    public void kuerzen(int teiler) {                // 4P
        nenner = nenner/teiler;
        zaehler = zaehler/teiler;
    }

    public double alsKommazahl() {                   // 4P
        return ((double) zaehler/(double) nenner);
    }
}
```

```

public void quadrieren(){                                     // 4P
    zaehler=zaehler*zaehler;
    nenner = nenner * nenner;
}

public Bruch multiplizieren(Bruch bruch){                    // 4P
    int z, n;
    z=zaehler*bruch.zaehler;
    n=nenner*bruch.nenner;
    Bruch ergBruch=new Bruch(z,n);
    return ergBruch;
}

public Bruch kleinervergleich (Bruch bruch){                // 4P
    Bruch temp;
    if(this. alsKommazahl()<bruch. alsKommazahl()){
        temp=this;
    }
    else{
        temp=bruch;
    }
    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 Punktabzug !!!!

AUFGABEN

I) 53P

In den Alpen werden in einer Wetterstation verschiedene Messdaten gespeichert. Zu der Messtation gehören also mehrere Messdaten, die in einem Speicher begrenzter Länge (maxAnzahl) hintereinander angefügt werden können. Die aktuelle Anzahl gespeicherter Messwerte (aktuelleAnzahl) verändert sich beim Anfügen bzw. Löschen von Messwerten. Die Messwert werden wie auf einem Stack verwaltet, d.h. es kann nur an das oberste Element angefügt werden, bzw. das oberste Element entfernt werden.

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

1)
geben Sie den Zusammenhang zwischen der Klasse Messwert und der Klasse Messtation mit Hilfe eines Diagramms an.
In allen Klassen bitte die Methoden nicht angeben.

2) 14P
Erstellen Sie die Klasse Messwert (mit genau den 2 Attributen temperatur und windrichtung, genau 2 set-Methoden, genau 2 get-Methoden, genau einem Konstruktor mit 2 Parametern und der Methode printMesswert()). Die Windrichtungen sind "nord", "süd", "west" oder "ost".

3) 29P
Erstellen Sie die Klasse Messtation mit genau den 3 Attributen maxAnzahl , aktuelleAnzahl und dem Feld messwerte. Zusätzlich müssen noch genau (d.h. nicht weniger und nicht mehr) die folgende Methoden implementiert werden:

a)
Messtation(int maxAnzahl)
In dem Konstruktor wird das Feld mit maximaler Anzahl Elemente erzeugt.

b)
addMesswert(Messwert mw)
Fügt einen Messwert hinzu. Das geht nur, wenn das Feld noch nicht voll ist.

c)

`getMesswert (int index)`

Gibt einen Messwert (d.h. Objekt) zurück, wenn der Index innerhalb des Feldes liegt.

d)

`löscheOberstenMesswert()`

löscht das aktuell oberste Element (d.h. Objekt) des Feldes.

e)

`getMesswertMaxTemperatur()`

gibt den Messwert (d.h. Objekt) mit der größten Temperatur zurück.
Falls es mehrere gibt, wird der erste Messwert im Feld genommen.

f)

`printAllMesswerte()`

Gibt alle gespeicherten Messwerte auf dem Bildschirm aus.

4) 10P

machen Sie in `main()` Folgendes:

a)

Erzeugen Sie die Messtation `ms`, die maximal 10 Messwerte aufnehmen kann.

b)

Speichern Sie den Messwert 30, "nord"

Speichern Sie den Messwert 40, "west"

c)

Geben Sie alle Messwerte auf dem Bildschirm aus.

d)

Löschen Sie alle Messwerte der Messtation.

Lösungen:

1)

3P



2)

14P

```
class Messwert {

    private double temperatur;
    private String windrichtung;

    public Messwert(double temperatur, String windrichtung) {
        this.temperatur = temperatur;
        this.windrichtung = windrichtung;
    }

    public double getTemperatur() {
        return temperatur;
    }

    public void setTemperatur(double temperatur) {
        this.temperatur = temperatur;
    }

    public String getWindrichtung() {
        return windrichtung;
    }

    public void setWindrichtung(String windrichtung) {
        this.windrichtung = windrichtung;
    }

    public void printMesswert() {
        System.out.println(temperatur + "," + windrichtung);
    }

}
```

2)

26P

```
class Messtation {

    private int aktuelleAnzahl;
    private int maxAnzahl;
    private Messwert[] messwerte;

    public Messtation(int maxAnzahl) {
        this.maxAnzahl = maxAnzahl;
        this.messwerte = new Messwert[maxAnzahl];
        this.aktuelleAnzahl = 0;
    }

    // Fügt einen Messwert hinzu, falls Array noch nicht voll ist.
    public boolean addMesswert(Messwert t) {
        boolean erg;
        if (aktuelleAnzahl < maxAnzahl) {
            messwerte[aktuelleAnzahl] = t;
            aktuelleAnzahl++;
            erg = true;
        } else {
            erg = false;
        }
        return erg;
    }

}
```



```

public Messwert getMesswert(int index) {          4P
    Messwert t;
    if (index >= 0 && index < aktuelleAnzahl) {
        t = messwerte[index];
    } else {
        t = null;
    }
    return t;
}

public void löscheOberstenMesswert() {           4P
    if (aktuelleAnzahl > 0) {
        aktuelleAnzahl--;
    }
}

private Messwert getMesswertMaxTemperatur() {    6P
    int i;
    double maxTemperatur;
    Messwert maxMessewert = null;

    // initialisieren des maximums
    maxTemperatur = messwerte[0].getTemperatur();
    i=0;
    if (aktuelleAnzahl > 0) {
        while (i < aktuelleAnzahl) {
            if (messwerte[i].getTemperatur() > maxTemperatur) {
                maxMessewert=messwerte[i];
                maxTemperatur = maxMessewert.getTemperatur();
            }
            i++;
        }
    }
    return maxMessewert;
}

public void printAllMesswerte() {                4P
    int i;
    for (i = 0; i < aktuelleAnzahl; i++) {
        messwerte[i].printMesswert();
    }
}

}

public class Startklasse {
    public static void main(String[] args) {
        int i;
        int maxAnzahl = 1;
        //int maxAnzahl = 10;
        Messtation ms = new Messtation(maxAnzahl);  2P
        ms.addMesswert(new Messwert(40, "nord"));  2P
        ms.addMesswert(new Messwert(50, "west"));  2P
        ms.printAllMesswerte();                    2P
        ms.löscheOberstenMesswert();                1P
        ms.löscheOberstenMesswert();                1P
        // Zum Testen:
        for(i=0;i<1000;i++){
            ms.löscheOberstenMesswert();
        }
    }
}

```

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) 16P

a) 12P

Erstellen Sie die Klasse "Zufall". Darin gibt es genau einen Konstruktor und nur die Methode rand(), die eine zufällige Zahl aus einem bestimmten Bereich ausgibt.

Die Formel, mit der eine Folge von Zufallszahlen simuliert wird, ist durch folgende Formel gegeben:

$$x_{n+1} = (a \cdot x_n + b) \bmod m$$

Dabei sind a, b und m feste Zahlen. Für diese sind zunächst folgende Werte festgesetzt:

$$a = 17$$

$$b = 3791$$

$$m = 513$$

Die Werte von a, b, m und der Anfangswert x_0 der 1. simulierten Zufallszahl muß durch einen Konstruktor festgelegt werden.

b) 2P

Erstellen Sie in main() durch den Aufruf der entsprechenden Methode hintereinander 10 Zufallszahlen und geben diese auf dem Bildschirm aus.

c) 2P

Erstellen Sie in main() 10 Zufallszahlen, die einen Würfel simulieren (der Würfel ist mit den Ziffern 0 bis 5 beschriftet).

Bemerkung:

mod bedeutet den Rest bei der Division.

$$23 \bmod 5 = 3$$

weil 5 in 23 insgesamt 4 mal hineinpasst und dabei ein Rest von 3 übrig bleibt.

2) 10P

Erstellen Sie die Klasse "Lottozahlen".

Darin gibt es genau einen Konstruktor und nur die Methode ziehung (), die 6 zufällige ganze Zahlen zwischen 0 und 48 (je einschließlich) erzeugt und in einem Feld abspeichert.

Diese Klasse "Lottozahlen" ist eine Spezialisierung der Klasse "Zufall".

3)

25P

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

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

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

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

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

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

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

a)

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

b)

Erstellen Sie die außerdem die folgenden Methoden:

b1)

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

b2)

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

b3)

```
public void printFeld()
```

c)

In `main()` müssen in einem "dynamischen" Feld die Zahlen 3,4,5,6,7,8 abgespeichert werden. Implementieren Sie dies mit Hilfe der Klasse `DynamischerSpeicher` und der entsprechenden Methoden.

Lösung:

1)

a)

```
class Zufall{
    private int xn;           // 1P
    private int a;           // 1P
    private int b;           // 1P
    private int n;           // 1P

    public Zufall(int a, int b, int n, int x0){ // 4P
        xn=x0;
        this.a=a;
        this.b=b;
        this.n=n;
    }

    public int rand(){        // 4P
        xn = (a * xn + b) % n;
        return xn;
    }
}

public static void main(String[] args) { // b+c) 4P
    int i;
    int erg;
    Zufall z = new Zufall(1367,1519,6,715);

    System.out.println("Zufallszahlen=");
    for(i=0;i<6;i++){
        erg = z.rand();
        System.out.println(erg);
    }
}

class Lottozahlen extends Zufall{
    private int [] feld;      // 1P
    public Lottozahlen(){    // 4P
        super(1367,1519,49,715);
        feld=new int[6];
    }

    public void ziehung(){    // 5P
        int i;
        for(i=0;i<6;i++){
            feld[i]=rand();
        }
    }

    public void printZiehung(){// nur als Test, nicht verlangt!
        int i;
        System.out.println("Ziehung=");
        for(i=0;i<6;i++){
            System.out.print(feld[i]+" ");
        }
        System.out.println("");
    }
}
```

3)

25P

```
package dynamischerspeicher1;
public class MainDynamischerSpeicher1 {
    public static void main(String[] args) {    // 4P
        int j;
        DynamischerSpeicher ds;
        ds=new DynamischerSpeicher(3);
        for(j=0;j<5;j++){
            ds.anfuegen(j+1);
        }
        ds.printFeld();
    }
}

class DynamischerSpeicher{
    private int[] feld;                // 1P
    private int[] feldTemp;           // 1P
    private int len;                  // 1P

    public DynamischerSpeicher(int zahl){    // 6P
        feld = new int[1];
        feld[0]=zahl;
        len=1;
    }

    public void anfuegen(int zahl){        // 9P
        int i;
        feldTemp=new int[len];
        for(i=0;i<len;i++){
            feldTemp[i]=feld[i];
        }
        feld = new int[len+1];
        for(i=0;i<len;i++){
            feld[i]=feldTemp[i];
        }
        feld[len]=zahl;
        len=len+1;
    }

    void printFeld(){                    // 3P
        int i;
        for(i=0;i<len;i++){
            System.out.print(feld[i]+" ");
        }
    }
}
```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

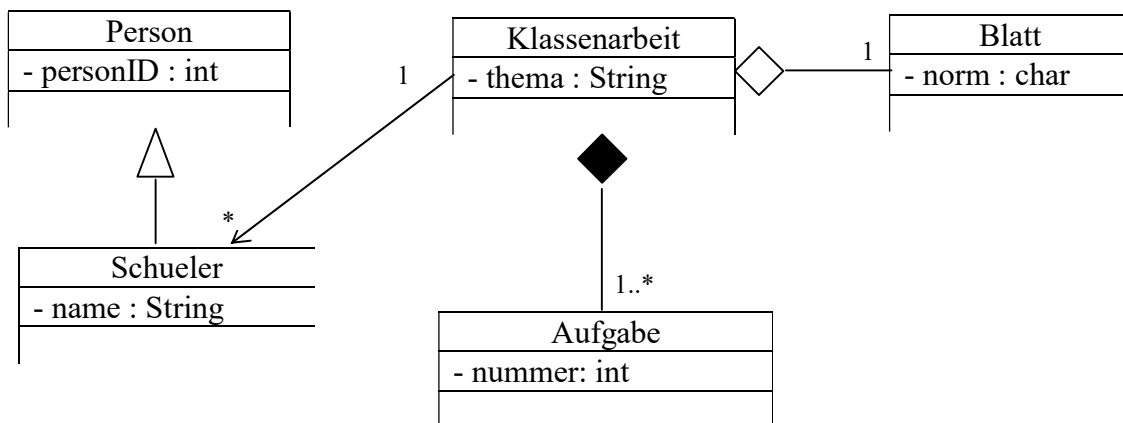
- Alle Aufgaben müssen bearbeitet werden.
- Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- Aufgabenblätter bitte auch abgeben.
- Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- Die rote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

AUFGABEN

I)

50P

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



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

// 1)

bzw.

// 1a)

- 1) 4P
Was bedeuten die 2 verschiedenen Pfeile (offene bzw. geschlossene Pfeilspitze) und die Rauten (weiß und schwarz). Jeweils nur ein Stichwort angeben.
- 2) 7P
Erzeugen Sie die Klasse Blatt mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).
- 3) 7P
Erzeugen Sie die Klasse "Person" mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).
- 4) 9P
Erzeugen Sie die Klasse "Schueler" mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau zwei Parametern).
- 5) 7P
Erzeugen Sie die Klasse "Aufgabe" mit genau dem im UML angegebenen Attribut (und nur diesem Attribut) und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).
- 6) 4P
a) Erzeugen Sie die Klasse "Klassenarbeit" mit allen nötigen (genau) 4 Attributen.
- b) 4P
Erzeugen Sie genau einen Konstruktor (mit genau zwei Parametern). Falls ein Array benötigt wird, soll dieses nicht dynamisch sein und im Konstruktor das Feld die Länge 10 erhalten).
- c) 4P
Erzeugen Sie die folgende Methode:
... insertAufgabe(...) :
fügt eine Aufgabe an eine bestimmte Stelle des Feldes ein.
Beachten Sie dazu, daß eine Komposition und eine Aggregation anders implementiert werden.
- d) 4P
Erzeugen Sie die folgende Methode:
... insertBlatt (...) :
fügt ein Blatt ein.
Beachten Sie dazu, daß eine Komposition und eine Aggregation anders implementiert werden.

Lösungen:

1) 4P
geschlossene Pfeilspitze: Vererbung, geöffnete Pfeilspitze: Assoziation,
schwarze Raute: Komposition, weiße Raute: Aggregation

2) 7P

```
class Blatt {
    private char norm;

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

    public char getNorm() {
        return norm;
    }

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

3) 7P

```
class Person{
    private int personID;

    public Person(int personID) {
        this.personID = personID;
    }

    public int getPersonID() {
        return personID;
    }

    public void setPersonID(int personID) {
        this.personID = personID;
    }
}
```

4) 9P

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

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

    public String getName() {
        return name;
    }

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

5) 7P

```
class Aufgabe{
    private int nummer;

    public int getNummer() {
        return nummer;
    }

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

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


6)

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

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

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

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

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

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

Bemerkungen:

Falls in der Lösung Felder verwendet werden, dürfen diese keine dynamischen Felder sein (wie z.B. ArrayList).

Die Programme müssen den Prinzipien der OOP entsprechen.

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

// 1)

bzw.

// 1a)

I)

52 P

Ein Haus hat mehrere Räume. Dies soll modelliert werden.

Beachten Sie: Wenn das Haus zerstört (z.B. durch Abriss) wird, werden auch die Räume des Hauses zerstört.

1) // 9P

Erzeugen Sie die Klasse Raum mit genau dem Attribut (und nur diesem Attribut) "name" und den dazugehörigen get-und set-Methoden und genau einem Konstruktor (mit genau einem Parameter).

2) // 24P

Erzeugen Sie die Klasse Haus mit mindestens 2 und maximal 4 Attributen. Ein Attribut davon ist "adresse". Die weiteren Attribute dürfen nur zur Verwaltung der Räume des Hauses verwendet werden.

Erzeugen Sie außerdem die zugehörigen get- und set-Methoden und genau einen Konstruktor mit den entsprechenden, sinnvollen Parametern.

Erstellen Sie in der Klasse Haus noch zusätzlich genau die folgende Methoden:

3)

19P

a) 3P

... anfüegen(...) :

fügt einen Raum an das Ende der bisherigen Räume an.

Wenn das Feld voll ist, werden keine neuen Räume angefügt.

b) 3P

... gibRaum (...) :

gibt den Raum eines Hauses zurück, der in einer bestimmten Zelle abgespeichert wurde.

Dazu muß auch überprüft werden, ob sich die Zelle (deren Inhalt zurückgegeben werden soll) innerhalb der bis jetzt angelegten Räume befindet.

c) 3P

... aendern(...)

Falls ein falscher Raum in eine Zelle eingetragen wird, kann man diesen nachträglich abändern. Dazu muß auch überprüft werden, ob sich die Zelle (deren Inhalt geändert werden soll) innerhalb der bis jetzt angelegten Räume befindet.

d) 3P

... loeschen(...)

Löscht den letzten Raum aus dem Feld. Das bedeutet nicht, daß der letzte Raum 0 wird, sondern daß der letzte Eintrag getilgt wird.

e) 3P

... ausgabe(...)

Gibt alle Namen der Räume des Hauses auf dem Bildschirm aus.

f) 4P

Erstellen Sie in main() das Haus myHaus mit der Adresse "Mesk", das maximal 100 verschiedene Räume mit Schulklassen belegen kann.

Fügen Sie dort den Raum mit Namen "B1.21" an.

Lösungen:

```
class Raum{ // 9P
    String name;

    public String getName() {
        return name;
    }

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

    public Raum(String name) {
        this.name = name;
    }
}

class Haus{ // 24P
    private String adresse; // 1P
    private Raum[] dieRäume; // 1P
    private int ende; // 1P
    private int gesamtLaenge; // 1P

    public Haus(String adresse,int gesamtLaenge){ // 4P
        this.adresse = adresse;
        this.gesamtLaenge=gesamtLaenge;
        ende=0;
        this.dieRäume = new Raum[gesamtLaenge];
    }

    public int getEnde() { // 2P
        return ende;
    }

    public void setEnde(int ende) { // 2P
        this.ende = ende;
    }

    public int getGesamtLaenge() { // 2P
        return gesamtLaenge;
    }

    public void setGesamtLaenge(int gesamtLaenge) { // 2P
        this.gesamtLaenge = gesamtLaenge;
    }

    public String getAdresse() { // 2P
        return adresse;
    }

    public void setAdresse(String adresse) { // 2P
        this.adresse = adresse;
    }

    public Raum[] getRaum() { // 2P
        return dieRäume;
    }

    public void setRaum(Raum Raum) { // 2P
        this.dieRäume = dieRäume;
    }
}
```

```

// 3) // 19P
public Raum gibRaum(int index){ // 3P
    if(index >=0 && index < ende){
        return dieRäume[index];
    }
    else{
        return null;
    }
}

public void anfüegen(Raum raum){ // 3P
    if(ende<=gesamtLaenge-1){
        dieRäume[ende]=raum;
        ende++;
    }
}

public void ändern(int index, Raum raum){ // 3P
    if(index>=0 && index < ende){
        dieRäume[index]=raum;
    }
}

public void löschen(){ // 3P
    if(ende>0){
        ende--;
    }
}

public void ausgabe(){ // 3P
    int i;
    for(i=0;i<ende;i++){
        System.out.println("Raum["+i+"]="+dieRäume[i].getName());
    }
}

}

public class Startklasse {
    public static void main(String[] args) {
        Haus myHaus = new Haus("Mesk",2); // 2P
        myHaus.anfüegen(new Raum("B1.21")); // 2P
    }
}

```