

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

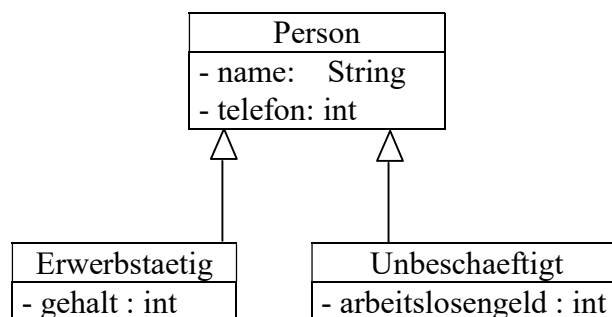
AUFGABEN

ACHTUNG: Die Klassenarbeit besteht aus genau einer Aufgabe (die aus Teilaufgaben besteht).

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

I) 50P

Gegeben ist das folgende UML-Diagramm, in dem nur die Klassennamen und alle Attribute vorkommen (und die nicht durch andere ergänzt werden dürfen). Die zugehörigen Methoden werden hier nicht dargestellt.



1) 32P

Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen und genau den (und NUR den) folgenden Methoden (es dürfen also keine weiteren Methoden erzeugt werden): jeweilige Konstruktoren (keine Standardkonstruktoren), set- und get-Methoden.

2) 9P

Erzeugen Sie in der Methode main (jeweils durch eine Anweisung)

a) den Erwerbstätigen "Schaffer" mit der Telefonnummer 4711, dem Gehalt von 1500 Euro

b) den Unbeschäftigten "Schröder" mit der Telefonnummer 4712 und Arbeitslosengeld von 400 Euro

c) die Person "Schulze" mit der Telefonnummer 12345

d)

Herr Schaffer bekommt 100 Euro mehr Gehalt.

Realisieren Sie dies durch genau eine einzige Anweisung (keine neue Methode implementieren), die zum alten Gehalt die 100 Euro dazu addiert.

In der Anweisung muss also der alte Gehalt ermittelt werden.

3)

9P

a) 6P

Erzeugen Sie zusätzlich in der Klasse "Erwerbstaetig" die Methode

... vergleiche(...)

Diese soll von 2 Erwerbstaetigen den Erwerbstaetigen zurückliefern, der das größere Gehalt hat. Bei 2 Erwerbstaetigen mit dem gleichen Gehalt ist es egal, wer von den beiden

zurückgeliefert wird.

b) 3P

Vergleichen Sie in main mit Hilfe der Methode ... vergleiche(...) den Erwerbstaetigen

"Schaffer" mit sich selbst und speichern Sie das Ergebnis in einer entsprechenden Variablen ab.

Lösung:

1)

```
public class Startklasse {
    public static void main(String[] args){
        Erwerbstaetig e = new Erwerbstaetig("Schaffer", 4711, 1500);    // 2P
        Unbeschaeftigt u = new Unbeschaeftigt("Schroeder", 4712, 400); // 2P
        Person p = new Person("Schulze",12345);                        // 2P
        e.setGehalt(e.getGehalt()+100);                                // 3P
        Erwerbstaetig temp = e.vergleiche(e);                          // 3P
    }
}

class Person{                                                         // 12P
    private String name;      // 1P
    private int telefon;      // 1P

    public Person (String pName, int pTelefon){ // 2P
        name = pName;
        telefon = pTelefon;
    }

    public void setTelefon(int pTelefon){ //2P
        telefon = pTelefon;
    }

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

    public int getTelefon(){ // 2P
        return(telefon);
    }

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

class Unbeschaeftigt extends Person{ //1P                                // 10P
    private int arbeitlosengeld; // 1P

    public Unbeschaeftigt(String pName, int pTelefon, // 4P
        int pArbeitslosengeld){
        super(pName, pTelefon);
        arbeitlosengeld = pArbeitslosengeld;
    }

    public void setArbeitslosengeld(int pArbeitslosengeld){ // 2P
        arbeitlosengeld = pArbeitslosengeld;
    }

    public int getArbeitslosengeld(){ // 2P
        return(arbeitslosengeld);
    }
}
```

```
class Erwerbstaetig extends Person{ // 1P
    private int gehalt ; // 1P

    public Erwerbstaetig(String pName, int pTelefon, int pGehalt){ // 4P
        super(pName, pTelefon);
        gehalt = pGehalt;
    }

    public void setGehalt(int pGehalt){ // 2P
        gehalt = pGehalt;
    }

    public int getGehalt(){ // 2P
        return(gehalt);
    }

    public Erwerbstaetig vergleiche(Erwerbstaetig pErwerbstaetig){ // 6P
        if(this.gehalt < pErwerbstaetig.gehalt){
            return pErwerbstaetig;
        }
        else{
            return this;
        }
    }
}
```

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) 51P
Die Verwaltung eines Bauernhofs soll mit Hilfe der OOP auf EDV umgestellt werden.
In einem Gespräch zwischen der EDV-Firma „Makall“ und den Besitzern des Bauernhofs wurde folgendes festgehalten:
Der Bauernhof gehört genau 2 Besitzern d.h. Personen und besteht aus genau einem Bauernhaus und einem Kuhstall, in dem maximal 10 Kühe untergebracht werden können..
Die Klasse Kuh wurde schon implementiert (mit den Attributen name und alter und den entsprechenden Methoden).
Aus Gründen der Vereinfachung besitzen die Klassen "Besitzer", "Person" und "Bauernhaus" jeweils genau ein Attribut
- 1) 16P
Erstellen Sie ein UML-Diagramm (mit den entsprechenden Attributen, aber ohne Methoden), das diesen Fall modelliert.
- 2) 7P
Implementieren Sie die Klasse "Person" mit den üblichen Attributen und Methoden.
- 3) 9P
Implementieren Sie die Klasse "Besitzer" mit den üblichen Attributen und Methoden.
- 4) 4P
a)
Erzeugen Sie die Klasse "Bauernhof" mit allen nötigen Attributen.
- b) 5P
Erzeugen Sie genau einen Konstruktor (mit genau 4 Parametern). Falls ein Array benötigt wird, soll dieses nicht dynamisch sein und die Länge 10 haben.

c)

10P

Erstellen Sie in der Klasse Bauernhof eine Methode

... anfüegenKuh (Kuh pKuh) :

fügt eine Kuh an das Ende der bisherigen gespeicherten Kühe an.

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

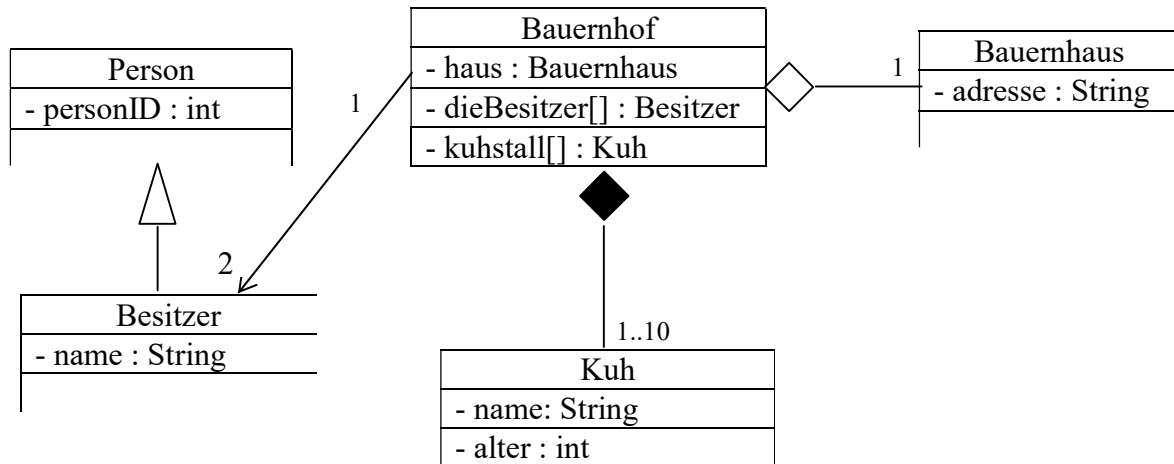
Bemerkung:

In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe-Methoden.

Alle Teilaufgaben - sofern es sich um Programmieraufgaben handelt - müssen in **einem** Programm realisiert werden

Lösung:

1)



2)

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;
    }
}
```

3)

9P

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

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

    public String getName() {
        return name;
    }

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

4)

```
class Bauernhof{
    private String adresse;
    private Kuh[] kuhstall;
    private Besitzer[] dieBesitzer;
    private Bauernhaus haus;

    public Bauernhof(String adresse, Besitzer b1, Besitzer b2,
                      Bauernhaus haus){
        this.adresse = adresse;
        dieBesitzer=new Besitzer[2];
        dieBesitzer[0]=b1;
        dieBesitzer[1]=b2;
        this.haus=haus;
        kuhstall = new Kuh[10];
    }

    public void anfuegenKuh(Kuh k){
        for(int i=0;i<10;i++){
            if(kuhstall[i]==null){
                kuhstall[i]=k;
                return;
            }
        }
    }

    public void printKuhstall(){
        for(int i=0;i<10;i++){
            if(kuhstall[i]!=null){
                sout ("Kuhname="+kuhstall[i].getName());
                sout ("Kuhgewicht="+kuhstall[i].getGewicht());
            }
        }
    }
}
```


Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

AUFGABEN

1) 3 P

Wie werden programmtechnisch Beziehungen zwischen verschiedenen Objekten hergestellt?
Bitte kurze verbale Beschreibung.

2) 47P

In einem Planungsbüro hat ein Lehrling folgende Gesprächsschnipsel gehört:

Es soll ein Kiosk gebaut werden. Dieses soll einen Namen haben.

Zu diesem Kiosk gibt es genau ein Lager. In dem Lager sollen eine bestimmte Anzahl Bierdosen gelagert werden, die nur für dieses Kiosk reserviert sind. D.h:

Zu einem Kiosk gibt es genau ein Lager und zu einem Lager gibt es genau ein Kiosk.

Dieser Sachverhalt soll objektorientiert modelliert werden, wobei die Navigation in alle 2 Richtungen geht.

Machen Sie dazu Folgendes:

a) 8P

Zeichnen Sie dazu das passende UML mit den entsprechenden Attributen und ohne die Methoden (einschließlich Name der Assoziation mit Leserichtung und Kardinalitäten).

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

b) 24P

Erstellen Sie die Klasse Kiosk und Lager (mit jeweils genau 2 Attributen, genau 2 set-Methoden, genau 2 get-Methoden und genau 1 Konstruktor).

c) 8P

Erzeugen Sie einen Kiosk mit dem Namen "standerl" und ein Lager, das einen Anfangsbestand von 1000 Bierdosen hat (einschließlich "Verlinkung").

d)

5P

Bei einer heftigen Zecherei hatten sich die dabei anwesenden Zeher 100 Bierdosen einverleibt.

Verändern Sie (**vom Kiosk ausgehend**) den entsprechenden Bierbestand des Lagers.

Bemerkung:

Dabei darf der neue Bierbestand nicht mit einem Taschenrechner (oder im Kopf) berechnet werden, also $1000-100$, sondern dies muß programmtechnisch mit Hilfe der Benutzung der gegebenen Methoden realisiert werden.

e)

2P

Geben Sie (**vom Lager ausgehend**) den neuen Bierbestand auf dem Bildschirm aus.

Siehe Bemerkung bei d)

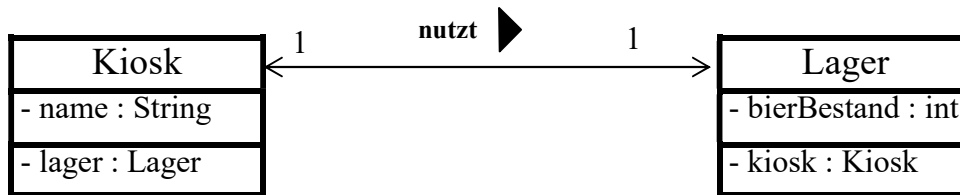
Lösungen:

1)
Ein Attribut der einen Klasse ist ein Objekt vom Typ der anderen Klasse

3P

2a)

8P



```

package e2fi_7_1_2013_nr1;
public class MainE2FI_7_1_2013_nr1 {
    public static void main(String[] args) {
        Lager myLager = new Lager(1000); // 2P
        Kiosk myKiosk = new Kiosk("standerl"); // 2P
        myLager.setKiosk(myKiosk);
        myKiosk.setLager(myLager); // 4P

        int anzahl; // 1P
        anzahl = myKiosk.getLager().getBierBestand();
        myKiosk.getLager().setBierBestand(anzahl - 100); // 4P
        System.out.println("neuer Bierbestand=" +
            myLager.getBierBestand()); // 2P
    }
}

class Kiosk {
    private String name; // 1P
    private Lager lager; // 1P

    public Kiosk(String name) { // 2P
        this.name = name;
    }

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

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

    public void setLager(Lager lager) { // 2P
        this.lager = lager;
    }

    public Lager getLager() { // 2P
        return lager;
    }
}
  
```

```
class Lager {
    private int bierBestand;           // 1P
    private Kiosk kiosk;               // 1P

    public Lager(int bierBestand) {    // 2P
        this.bierBestand = bierBestand;
    }

    public int getBierBestand() {      // 2P
        return bierBestand;
    }

    public void setBierBestand(int bierBestand) { // 2P
        this.bierBestand = bierBestand;
    }

    public void setKiosk(Kiosk kiosk) { // 2P
        this.kiosk = kiosk;
    }

    public Kiosk getKiosk() {          // 2P
        return kiosk;
    }
}
```

Name, Vorname:

Hilfsmittel:
keine

Hinweise (unbedingt beachten):

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

AUFGABEN

I)

52P

Bemerkung:

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

Zur Information:

Das Spiel "Flaschen drehen" funktioniert wie folgt beschrieben:

Ein Kreis wird in 10 durchnummerierte Kreisausschnitte unterteilt. In der Mitte des Kreises befindet sich eine Flasche, die jeder Spieler, der am "Zug" ist drehen muß und die danach auf genau einen Kreisausschnitt zeigt. Jeder Spieler, der am "Zug" ist, muß vor jedem Drehvorgang mindestens einen Cent auf genau einen Kreisausschnitt setzen (auf dem auch schon mehrere Cents liegen können). Sein Vermögen vermindert sich dann um diesen gesetzten Betrag,

Nach jedem Drehvorgang darf der drehende Spieler alle Cents von dem Kreisausschnitt nehmen, auf den die Flaschenöffnung zeigt. Dann muß der nächste Spieler die Flasche drehen.

Bei 2 Spielern ergibt sich folgende dynamische Entwicklung des Spiels:

Spieler 1 setzt auf einen Kreisausschnitt seiner Wahl eine bestimmte Anzahl Cents

Spieler 1 dreht die Flasche, die danach auf einen bestimmten Kreisausschnitt zeigt.

Für Spieler 1 wird die Gewinnauswertung durchgeführt.

Spieler 2 setzt auf einen Kreisausschnitt seiner Wahl eine bestimmte Anzahl Cents

Spieler 2 dreht die Flasche, die danach auf einen bestimmten Kreisausschnitt zeigt.

Für Spieler 2 wird die Gewinnauswertung durchgeführt.

weiter mit Spieler 1...

1)

11P

Erzeugen Sie die Klasse "Kreis" mit genau dem Array "dieWerte" als Attribut, einem Konstruktor (ohne Parametern), der alle Kreisausschnitte mit 0 Cents vorbelegt und die folgenden Methoden:

...getWert(...)

gibt die Belegung (Geldwert) eines Kreisausschnitts zurück.

... setWert(...)

belegt einen Kreisausschnitt mit einem bestimmten Einsatz.

Der Geldbetrag auf diesem Kreisausschnitt wird um diesen Einsatz vermehrt.

... reset(...)

belegt einen Kreisausschnitt mit dem Einsatz 0

2)

5P

Erzeugen Sie die Klasse "Flasche" mit genau dem Attribut "kreis" als Attribut, einem Konstruktor (mit genau einem Parameter) und die Methode

...drehen(), die einen Wert zwischen 0 und 9 zurückgibt.

Dazu dürfen Sie die Methode MyMath.random() verwenden, die Ihnen ein arbeitsloser Mathematiker spendiert hat und die ganze Zahlen zwischen 0 und 9 zurückgibt.

3)

27P

Erzeugen Sie die Klasse "Spieler" mit genau den Attributen "name", "vermögen", "kreis" als Attribut, einem Konstruktor (mit genau 3 Parametern), den get- und set-Methoden und den folgenden, weiteren Methoden:

... geldSetzen(...)

setzt einen Einsatz auf einen bestimmten Kreisausschnitt. Das gesamte Vermögen wird um diesen Einsatz vermindert.

... spielAuswerten(...)

gibt den aktuellen Gewinn und zusätzlich noch das Vermögen auf dem Bildschirm aus.

4)

9P

Erstellen Sie ein UML-Diagramm ohne Attribute und ohne Methoden Navigierbarkeit, Multiplizität jeweils angeben.

Lösung:

```
// 11 P
class Kreis{
    private int[] dieWerte;                                //2P

    public Kreis(){                                        //2P
        dieWerte = new int[10];
    }

    public int getWert(int i) {                            //2P
        return dieWerte[i];
    }

    public void setWert(int einsatz, int i) {              //3P
        dieWerte[i]=dieWerte[i]+ einsatz;
    }

    public void reset (int i) {                            //2P
        dieWerte[i]=0;
    }
}

// 5P
class Flasche{
    private Kreis kreis;                                    //1P

    public Flasche(Kreis kreis){                            //2P
        this.kreis = kreis;
    }

    public int drehen(){                                    //2P
        double zufall;
        int erg;
        zufall = Math.random();
        erg=(int) (10*zufall);
        if(erg>9)
            erg=9;
        //System.out.println("gedreht= "+erg);
        return erg;
        //kreis.setNr(erg);
    }
}

// 27P
class Spieler{
    private String name;                                    //1P
    private int vermögen;                                    //1P
    private Kreis kreis;                                    //1P

    //3P
    public Spieler(String name, int vermögen, Kreis kreis){
        this.name = name;
        this.vermögen = vermögen;
        this.kreis = kreis;
    }
}
```

```

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

public void setName(String name) {                           //2P
    this.name = name;
}

public int getVermögen() {                                    //2P
    return vermögen;
}

public void setVermögen(int vermögen) {                      //2P
    this.vermögen = vermögen;
}

public void geldSetzen(int einsatz, int nr) {                //4P
    kreis.setWert(einsatz, nr);
    vermögen = vermögen - einsatz;
}

public Kreis getKreis() {                                    //2P
    return kreis;
}

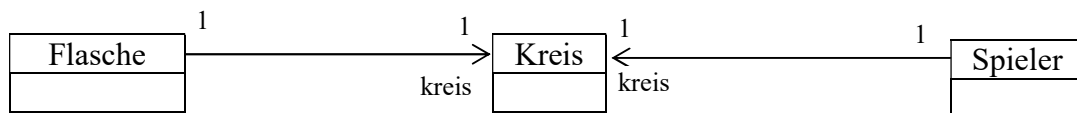
public void setKreis(Kreis kreis) {                          //2P
    this.kreis = kreis;
}

// Spiel auswerten für Kreisausschnitt i
public void spielAuswerten(int i){                          //5P
    int wert;
    wert=kreis.getWert(i);
    vermögen = vermögen + wert;
    // Wert zurücksetzen
    kreis.reset(i);
    System.out.print("Spieler "+name);
    System.out.print(" hat gewonnen: "+wert);
    System.out.println(" hat Gesamtvermögen: "+vermögen);
}
}

```

4)

9P



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);
    }
}
```