OOP – Aufträge für E2-Praktische Informatik, Günzel Sommer 2020

Inhaltliche Hinweise zur Bearbeitung:

- *Falls nichts anderes notwendig ist, verwenden Sie für **Tastatureingaben** die Klassen-Methode **Einlesen.readInt (String text)** (siehe **Klasse Einlesen.java** – die Ihnen zur Verfügung steht) oder die **Eingabe mit dem Scanner-Objekt**.*

- *Die Aufgaben sind mit dem **Java-Editor** zu bearbeiten. Zu jedem Java-Projekt ist zusätzlich zu den Java-Quellcode-Dateien **eine UML-Datei abzugeben**, die vor oder nach erfolgreichem Testlauf der Java-Dateien erzeugt werden kann.*

- *Die Aufgaben basieren auf Ihrem Wissen, das Sie bei Bedarf durch Studieren der **Kapitel 10.1 bis 10.8 im Lehrbuch** ergänzen sollten; und auf den Übungen und Hausaufgaben des Unterrichts.*

Aufgabe 1

Achtung: *Alle Attribute beginnen in Java* und nachfolgend in den Aufgaben *mit einem Kleinbuchstaben* (im Gegensatz zur deutschen Rechtschreibung)!

Implementieren Sie eine **Klasse Mensch**, die *private* **Attribute** beinhaltet, um den vorNamen (String), den nachNamen (String), das alter (int) und das geschlecht (**boolean**, mit **true** für männlich) einer Person zu speichern.

Statten Sie die Klasse mit dem Standardkonstruktur aus und nachfolgend auch mit einem weiteren Konstruktor, der als Parameter das Alter als **int**-Wert, das Geschlecht als **boolean**-Wert und den Vor- und Nachnamen als String-Werte hat.

Ergänzen Sie die Klasse außerdem durch folgende Methoden:
a) **public int** getAlter ( )
Diese Methode soll das Alter des Objekts zurückliefern.

b) **public void** setAlter (**int** neuesAlter)
Diese Methode soll das Alter des Objekts auf den Wert neuesAlter setzen.

c) **public boolean** getIstMaennlich ( )
Diese Methode soll den **boolean**-Wert (die Angabe des Geschlechts) des Objekts zurückliefern.

d) **public boolean** aelterAls (Mensch m) //Übergabeparameter ist ein Objekt
Wenn der Mensch m älter ist als das aktuelle Objekt, soll diese Methode den Wert **true** zurückliefern, andernfalls den Wert **false**.

e) **public String** toString ( )
Diese Methode soll eine Zeichenkette zurückliefern, die sich aus dem Vornamen, dem Nachnamen, dem Alter und dem Geschlecht des Objekts zusammensetzt (Methode ***siehe Aufgabe 2***).

f) Zum Test Ihrer Klasse Mensch entwickeln Sie eine **Klasse TestMensch**, die mit Objekten der Klasse Mensch arbeitet, die den obigen Konstruktor und alle Methoden der Klasse Mensch testet. Verwenden Sie auch Tastatureingaben (siehe oben)!

g) Testen Sie, ob der Compiler wirklich Zugriffe auf die *private* Attribute verweigert. Testen Sie dabei auch, ob der Compiler für ein Objekt **m** der Klasse Mensch tatsächlich bei einer Anweisung

*System.out.println(**m**);*

automatisch eine toString( )-Methode aufruft!

Eränzung zur Aufgabe 1

1.1)
Ergänzen Sie die beiden Klassen entsprechend um

- das *private* Attribut einer fortlaufenden Nummer **int** nr
- eine *public static* **Klassenvariable** namens *gesamtZahl* (zur Information über die Anzahl der bereits erzeugten Objekte der Klasse), die mit dem Wert 0 zu initialisieren ist.

Im Konstruktor muss dabei der Objektzähler gesamtZahl um 1 erhöht und anschließend die laufende Nummer des Objekts auf den neuen Wert von gesamtZahl gesetzt werden. …In der Testklasse Objekte erzeugen und gesamtZahl ausgeben…

*public **static** gesamtZahl*

1.2)
Erzeugen Sie in der Klasse **TestMensch** mindestens 4 „Mensch-Objekte“ sowie ein **Array vom Typ Mensch**. Füllen sie das **Array** mit diesen **4 Referenzvariablen**. Jede Methode, die keinen Parameter enthält, lässt sich über eine **foreach - Schleife** nacheinander für alle Objekte des Arrays ohne Probleme aufrufen. Testen Sie dies für die toString- Methode.
Aber zum Erzeugen von Objekten in einer Wiederholung nur die for-Schleife verwenden!

Aufgabe 2

Implementieren Sie eine **Klasse Artikel**, die *private* **Attribute** beinhaltet, um die bezeichnung (String), die artikelNummer (String), den ekPreis (double) und den lagerBestand (int) eines Artikels zu speichern.

Nach dem Standardkonstruktor der Klasse fügen Sie einen Konstruktor mit Parametern ein, der bei seinem Aufruf in der Anwendungsklasse ein Artikel-Objekt mit allen gewünschten Attributwerten erzeugen kann.

Implementieren Sie alle getter- und setter-Methoden für das Auslesen bzw. für das Verändern der Attributwerte.

Übernehmen Sie auch die nachfolgende Methode in die Klasse Artikel:

```
public String toString( ) {
   return "ArtikelNr: "+ artikelNummer +", "+ bezeichnung +", Preis: "+ ekPreis +", L-Bestand: "+ lagerBestand; }
```

Zum Testen Ihrer Klasse Artikel implementieren Sie eine **Klasse AnwendArtikel**. Verwenden Sie für die Erzeugung von Objekten einmal den Aufruf des Standardkontruktors und setter-Methoden, ein anderes Mal den komfortableren Weg über den Aufruf des Parameter-Konstruktors.

Folgende Artikel sollen eingekauft werden:

artikel1: („Maus", „M001“, 10.00, 5)

artikel2: („Tastatur", „T001“, 20.00, 7)

artikel3: („CPU",“C001“, 90.00, 3),

was durch entsprechende Printanweisungen mit Verwendung der toString-Methode zu belegen ist,

z.B. ***System.out.println (artikel1.toString( ));***

Zusatzteilaufgabe:

Erzeugen Sie **ein Array vom Typ Artikel** und füllen es mit 3 bereits fertigen Artikeln. Geben Sie über eine foreach-Schleife alle Eigenschaften der in Ihrem Array enthaltenen Artikel aus.

Aufgabe 3

Aus dem Mathematikunterricht kennen Sie die Koordinaten x_p und y_p von einem beliebigen Punkt $p(x_p|y_p)$. Zwei Punkte p und q bestimmen die Länge einer Strecke (der Verbindungslinie) zwischen ihnen. Die Länge dieser Strecke berechnet man mit der Formel

$$länge = \sqrt{(x_q - x_p)^2 + (y_q - y_p)^2}$$

Entwickeln Sie zur Darstellung und Bearbeitung von Punkten eine **Klasse Punkt**, zur Darstellung und Bearbeitung von Strecken eine **Klasse Strecke** und schließlich für den Test bzw. die Anwendung dieser beiden Klassen eine **Klasse Punkt_Strecke**. Gehen Sie wie folgt vor:

a) Implementieren Sie die **Klasse Punkt** mit zwei *private* Attributen x und y vom Typ double, die die x- und y-Koordinaten eines Punktes repräsentieren. Statten Sie die Klasse Punkt mit Konstruktoren und Instanzmethoden aus:

- einen Konstruktor mit zwei double-Parametern (die x- und y-Koordinaten des Punktes),
- eine Methode getX( ), die die x-Koordinate des Objekts (des Punktes) zurückliefert,
- eine Methode getY( ), die die y-Koordinate des Objekts zurückliefert,
- eine **void**-Methode read( ), die die x- und y-Koordinaten des Objekts **einliest**.

b) Implementieren Sie die **Klasse Strecke** mit zwei *private* Attributen p und q vom Typ **Punkt**, die die beiden Randpunkte einer Strecke repräsentieren. Diese beiden Attribute nennt man Referenzvariablen, da sie Platzhalter für Objekte sind.

Statten Sie die Klasse Strecke mit Konstruktoren und Instanzmethoden aus:

- einen Konstruktor mit zwei Punkt-Parametern (die die Randpunkte der Strecke bilden),
- eine **void**-Methode read( ), die die beiden Randpunkte p und q des Objekts einliest (verwenden Sie dazu den Aufruf der Instanzmethode read( ) der Objekte p und q),
- eine double-Methode getLaenge( ), die unter Verwendung der Instanzmethoden
- getX( ) und getY( ) der Randpunkte die Länge des Strecken-Objekts berechnet und zurückliefert.

c) Für die Tests Ihrer Implementationen ist Ihnen die nachfolgende Klasse vorgegeben:

```
1  public class Punkt_Strecke {
2  public static void main(String[ ] args) {

3  Punkt ursprung =  new Punkt(0.0,0.0);
4  Punkt endpunkt = new Punkt(4.0,3.0);

5  Strecke s = new Strecke(ursprung,endpunkt);

6  System.out.println("Die Laenge der Strecke betraegt " + s.getLaenge( ) + ".");
7  System.out.println( );
8  System.out.println("Strecke s eingeben:");
9  s.read( );
10 System.out.println( );
11 System.out.println("Die Laenge der Strecke betraegt " + s.getLaenge( ) + ".");
12   }
13 }
```

Aufgabe 4

Ihnen wird die nachfolgende **Klasse *Reifen*** vorgelegt.

a) Beschreiben Sie die unten aufgeführte Klasse mit eigenen Worten hinsichtlich Attribut, Konstruktor und Methode. Implementieren Sie anschließend diese Klasse zuzüglich Standardkonstruktor und set-Methode.

```
1  public class Reifen {
2
3  private double druck;
4
5  public Reifen (double luftdruck) {
6  this.druck = luftdruck; }
7
8   public double getAktuellerDruck ( ) {
9   return this.druck; }
10  }
```

b) Für welche übergeordneten Probleme/Klassen lässt sich eine solche Klasse Reifen verwenden?

c) Implementieren Sie eine Klasse ***Fahrzeug***, die die Klasse Reifen verwendet und folgendes beinhaltet:

- *private* Attribute bezeichnung vom Typ String, anzahlReifen vom Typ int**, reifen** vom Typ **Reifen** und faehrt vom Typ **boolean** (für die Information über den momentanen Zustand)

- einen Konstruktor, der mit Parametern für bezeichnung, reifenanzahl und **druck** ausgestattet ist, in seinem Rumpf die entsprechenden Attribute des Objekts belegt und außerdem das Fahrzeug in den Zustand "fährt nicht" versetzt
  // this.**reifen** = new Reifen(**druck**);

- eine Instanzmethode fahreLos( ), die die Variable faehrt des Fahrzeug-Objektes auf **true** setzt

- eine Instanzmethode halteAn( ), die die Variable faehrt des Fahrzeug-Objektes auf **false** setzt

- eine Instanzmethode status( ), die einen Informations-String über Bezeichnung, Fahrzustand, Reifenzahl und Reifendruck des Fahrzeug-Objektes ausgibt. (Welche Bezeichnung für eine solche Methode kennen Sie bereits?)

d) Entwickeln und implementieren Sie ein Testprogramm, das zuerst ein Fahrrad (verwenden Sie Reifen mit 4.5 bar) und anschließend einen PKW (verwenden Sie Reifen mit 1.9 bar) in Form von Objekten der Klasse Fahrzeug erzeugt und anschließend folgende Vorgänge durchführt:

1. mit dem Fahrrad losfahren, 2. mit dem PKW losfahren,
3. mit dem Fahrrad anhalten, 4. mit dem PKW anhalten.

Unmittelbar nach jedem der vier Ereignisse soll jeweils mittels der Methode status( ) der aktuelle Fahrzustand *beider* Fahrzeuge ausgegeben werden.
Eine Ausgabe hat das folgende Aussehen:

Vorgang 1:
Fahrrad faehrt auf 2 Reifen mit je 4.5 bar
PKW steht auf 4 Reifen mit je 1.9 bar
Vorgang 2:
Fahrrad faehrt auf 2 Reifen mit je 4.5 bar
PKW faehrt auf 4 Reifen mit je 1.9 bar
Vorgang 3:
Fahrrad steht auf 2 Reifen mit je 4.5 bar
PKW faehrt auf 4 Reifen mit je 1.9 bar
Vorgang 4:
Fahrrad steht auf 2 Reifen mit je 4.5 bar
PKW steht auf 4 Reifen mit je 1.9 bar