JAVA-ÜBUNGSAUFGABEN KLASSEN 1

Bemerkungen:
B1) Erzeugen Sie jeweils in jeder Klasse die Methode printAllAttributs(), die - zu Testzwecken - alle Werte der Attribute der jeweiligen Klasse auf dem Bildschirm ausgibt.

B2) Erzeugen Sie für jede Klasse ein entsprechendes UML-Diagramm.

1) Hund

a) Erzeugen Sie zusätzlich zu dem einen Hund (siehe Präsentation) einen zweiten Hund.
Dieser soll Rex heißen und 20 kg wiegen.
Geben Sie den Namen und das Gewicht dieses Hundes (zur Kontrolle) auf dem Bildschirm aus.

b) Da Rex ziemlich viel arbeitet (jagt), nimmt er 3 kg ab. Gehen Sie davon aus, dass Sie das alte Gewicht von Rex nicht mehr wissen.
Geben Sie den Namen und das Gewicht dieses Hundes (zur Kontrolle) auf dem Bildschirm aus.

c) Erzeugen Sie zusätzlich noch einen dritten Hund.
Der Anwender soll den Namen und das Gewicht dieses Hundes über Tastatur eingeben.
Geben Sie den Namen und das Gewicht dieses Hundes (zur Kontrolle) auf dem Bildschirm aus.

d) Geben Sie mit Hilfe der Methode printAllAttributs()alle Attribute der Hunde in den Aufgabenteilen a), b) c) auf dem Bildschirm aus.

e) **(schwierig)**
Schreiben Sie in der Klasse Hund die Methode vergleichen(...) , die das Gewicht zweier Objekte der Klasse Hund vergleicht.

2) Abschneiden

Erstellen Sie die Klasse MyMath. Diese enthält folgende Methode:
double myTruncate(double zahl, int anzNachkommastellen)
Diese schneidet von einer Fließkommazahl bis auf anzNachkommastellen alle Nachkommastellen ab.

Beispiele:
erg = myTruncate(146.123456678, 2); ---> erg = 146.12
erg = myTruncate(-12.345678912, 1); ---> erg = -12.3
erg = myTruncate(3.1415926535 8979323846 6, 1); ---> erg = 3.1415

3) Bauernhof

Erstellen Sie in Ihrem letzten Programm zusätzlich zur Klasse Hund noch die Klasse Kuh. Überlegen Sie sich, welche Attribute und Methoden die Klasse Kuh haben soll.

a) Erzeugen Sie eine Kuh mit dem Namen Elsa, die eine täglichen Milchleistung von 20 Liter hat. Schreiben Sie zu Testzwecken die Methode printAllAttributs(...), die alle Attribute einer Kuh auf dem Bildschirm ausgibt. Geben Sie mit Hilfe der Methode printAllAttributs(...) alle Attribute (zur Kontrolle) dieser Kuh auf dem Bildschirm aus.

b) Erzeugen Sie eine zweite Kuh mit Namen Milka, die eine täglichen Milchleistung von 60 Liter hat.
Da diese Kuh Kraftfutter bekommt, verdoppelt sich ihre tägliche Milchleistung.
Gehen Sie davon aus, dass Sie die alte tägliche Milchleistung der Kuh Milka nicht mehr wissen.
Geben Sie mit Hilfe der Methode printAllAttributs(...) alle Attribute (zur Kontrolle) dieser Kuh auf dem Bildschirm aus.

c) Erzeugen Sie zusätzlich noch eine dritte Kuh.
Der Anwender soll den Namen und die tägliche Milchleistung dieser Kuh über Tastatur eingeben.
Geben Sie mit Hilfe der Methode printAllAttributs(...) alle Attribute (zur Kontrolle) dieser Kuh auf dem Bildschirm aus.

4) Konto

Erstellen Sie die Klasse Bankkonto mit den entsprechenden Attributen und Methoden.
Zusätzlich soll noch eine Methode verzinse() implementiert werden, die auf einem Kontostand die Zinsen gutschreiben lässt.
a) Erzeugen Sie außerdem jeweils in der Klasse die Methode printAllAttributs(), die - zu Testzwecken - alle Werte der Attribute in der Klasse auf dem Bildschirm ausgibt.
b) Erstellen Sie ein Konto mit dem Anfangskontostand 100 Euro und 10% Zinnsatz.
c) Der Kontostand soll verzinst werden.
d) Dieser neue Kontostand soll wieder verzinst werden.
e) Geben Sie den neuen Kontostand auf dem Bildschirm aus.
f) Erstellen Sie die Methode abheben(...), mit der man vom Konto einen bestimmten Betrag abheben kann. Allerdings darf der Kontostand nie mehr als 5000 Schulden Euro fallen.
g) Erstellen Sie ein Menü, mit dem man
(1) ein Konto eröffnen kann,
(2) den Inhaber der Kontos ändern kann,
(3) vom Konto abheben kann (Kontostand darf nie mehr als 5000 Euro Schulden betragen),
(4) Die Kontodaten auf dem Bildschirm ausgeben kann,
(5) das Konto auflösen kann,
(6) das Programm beenden kann
Bem: Die Eingabe der Zahl 1 bedeutet z.B: Konto eröffnen, usw.

5)

Modellieren Sie einen kleinen Bauernhof mit Hilfe der objektorientierten Programmierung:
Auf einem kleinen Bauernhof gibt es u.a. eine Kuh, eine Henne und einen Hund.
Jedes Tier hat u.a. einen Namen und ein Alter.
Jede Kuh hat eine bestimmte Milchleistung pro Zeiteinheit, erzielt aber auf dem Markt den gleichen Preis für einen Liter Milch.
Jede Henne hat eine bestimmte Legeleistung pro Zeiteinheit, erzielt aber auf dem Markt den gleichen Preis für ein Ei.

a) Erstellen Sie die Klasse Kuh, Henne und Hund mit den entsprechenden Attributen und Methoden.
b) Erstellen Sie die Klasse Bauernhof.
Da der Bauernhof sehr klein ist, gibt es (jeweils als Attribut!) dort nur maximal eine Kuh, maximal zwei Hennen und maximal einen Hund (ein Tier kann sterben oder geschlachtet werden), d.h. es muss jeweils eine Methode geben, die dies realisiert.
c) Erstellen Sie ein Objekt der Klasse Bauernhof mit einer Kuh, zwei Hennen und einem Hund.
d) Der Bauer will in den Urlaub. Deswegen verschenkt er eine Henne und eine Kuh an einen Freund.
Realisieren Sie dies programmtechnisch.
e) Der Bauer kommt wieder vom Urlaub zurück und kauft deswegen eine neue Henne und eine neue Kuh.
Realisieren Sie dies programmtechnisch.

6) Schwierig

Der Hund in dem Programm oben soll noch so modelliert werden, dass er als Attribut ein Objekt der Klasse Schwanz bekommt:

```
class Hund{
  Schwanz schwanz;
  ...
}
```

a) Erstellen Sie die Klasse Schwanz und Hund.
b) Erzeugen Sie ein Objekt myS1 der Klasse Schwanz (Länge: 50 cm, Farbe gelb).
Erzeugen Sie einen Hund myH1. Dieser soll Trex heißen und den Schwanz myS1 haben.
Geben Sie die Eigenschaften (einschliesslich Schwanz) dieses Hundes (zur Kontrolle) auf dem Bildschirm aus.
c) Erzeugen Sie ein Objekt myS2 der Klasse Schwanz (Länge: 100 cm, Farbe braun).
Dieser Schwanz soll der Schwanz myS1 sein. Realisieren Sie dies programmtechnisch.
Geben Sie die Eigenschaften dieses Schwanzes (zur Kontrolle) auf dem Bildschirm aus.
d) Verändern Sie die Länge des Schwanzes myS2 auf 10 cm Länge und die Farbe schwarz.
Geben Sie die Eigenschaften dieses Schwanzes (zur Kontrolle) auf dem Bildschirm aus.
e) Geben Sie die Eigenschaften (einschliesslich Schwanz) des Hundes myH1(zur Kontrolle) auf dem Bildschirm aus und erklären Sie dies.

7)

```
// Was gibt dieses Programm1 auf dem Bildschirm aus ?
package ohnecopykonstruktor1;
public class MainOhneCopyKonstruktor1 {
    public static void main(String[] args) throws Exception {
        Schwanz mySchwanz = new Schwanz(15);
        Hund myHund1 = new Hund(13, mySchwanz);
        System.out.println("myHund1.getSchwanz().getLaenge()=" +
                                            myHund1.getSchwanz().getLaenge());
        mySchwanz.setLaenge(50);
        Hund myHund2 = new Hund(3, mySchwanz);
        System.out.println("myHund1.getSchwanz().getLaenge()=" +
                                            myHund1.getSchwanz().getLaenge());
        System.out.println("myHund2.getSchwanz().getLaenge()=" +
                                            myHund2.getSchwanz().getLaenge());
    }
}

class Hund {
    private double gewicht;
    private Schwanz schwanz;

    public Hund(double pGewicht, Schwanz pSchwanz) {
        gewicht = pGewicht;
        schwanz = pSchwanz;
    }

    public void setGewicht(double pGewicht) {
        gewicht = pGewicht;
    }

    public double getGewicht() {
        return (gewicht);
    }

    public void setSchwanz(Schwanz pSchwanz) {
        schwanz = pSchwanz;
    }

    public Schwanz getSchwanz() {
        return (schwanz);
    }
}

class Schwanz {
    private int laenge;

    public Schwanz(int pLaenge) {
        laenge = pLaenge;
    }

    public void setLaenge(int plaenge) {
        laenge = plaenge;
    }
    public int getLaenge() {
        return (laenge);
    }
}
```

```
// Was gibt dieses Programm2 auf dem Bildschirm aus ?
package mitcopykonstruktor1;
public class MainMitCopyKonstruktor1 {
    public static void main(String[] args) throws Exception {
        Schwanz mySchwanz = new Schwanz(15);
        Hund myHund1 = new Hund(13, mySchwanz);
        System.out.println("myHund1.getSchwanz().getLaenge()=" +
                                            myHund1.getSchwanz().getLaenge());
        mySchwanz.setLaenge(50);
        Hund myHund2 = new Hund(3, mySchwanz);
        System.out.println("myHund1.getSchwanz().getLaenge()=" +
                                            myHund1.getSchwanz().getLaenge());
        System.out.println("myHund2.getSchwanz().getLaenge()=" +
                                            myHund2.getSchwanz().getLaenge());
    }
}

class Hund {
    private double gewicht;
    private Schwanz schwanz;

    public Hund(double pGewicht, Schwanz pSchwanz) {
        gewicht = pGewicht;
        schwanz = new Schwanz(pSchwanz);
    }

    public void setGewicht(double pGewicht) {
        gewicht = pGewicht;
    }

    public double getGewicht() {
        return (gewicht);
    }

    public void setSchwanz(Schwanz pSchwanz) {
        schwanz = pSchwanz;
    }

    public Schwanz getSchwanz() {
        return (schwanz);
    }
}

class Schwanz {
    private int laenge;

    public Schwanz(Schwanz schwanz) {
        laenge = schwanz.getLaenge();
    }

    public Schwanz(int pLaenge) {
        laenge = pLaenge;
    }

    public void setLaenge(int plaenge) {
        laenge = plaenge;
    }
```

```
    public int getLaenge() {
        return (laenge);
    }
}
```

JAVA-ÜBUNGSAUFGABEN KLASSEN 2

8)

Erstellen Sie die Klassen, die Inverter, AND-Gatter bzw. OR-Gatter simulieren.
Zeigen Sie damit folgende Behauptung:
`NON (A AND B)` besitzt die gleiche Wertetabelle wie **`NON A OR NON B`**

9)

Erstellen Sie die Klasse "Bruch", die Brüche der Form a/b verwaltet, wobei a und b ganze Zahlen sind. Die Klasse soll die Methoden enthalten, mit denen man Brüche kürzt, addiert, subtrahiert, multipliziert und dividiert.

10)

Erstellen Sie die Klasse "Zufall", die durch den Aufruf von rand() eine zufällige Zahl aus einem bestimmten Bereich ausgibt.
Ein möglicher Algorithmus, um eine Folge von Zufallszahlen zu erzeugen, ist durch folgende Formel gegeben:
$x_{n+1} = (x_n \cdot b + 1) \bmod m$
Dabei sind b und m feste Zahlen. Für diese sind zunächst folgende Werte festgesetzt:
b = 3456
m = 10000
Der Anfangswert x_0 kann z.B. durch den Konstruktor festgelegt werden.
a) Erstellen Sie 10 Zufallszahlen und geben dann diese auf dem Bildschirm aus.
b) Versuchen Sie auch eine Methode bereitzustellen, die einen Münzwurf simuliert.
c) Ab wann wiederholen sich die "Zufallszahlen" ?

Bemerkung:
mod bedeutet den Rest bei der Division.
23 mod 5 = 3
weil 5 in 23 4 mal hineinpasst und dabei ein Rest von 3 übrig bleibt.

11)

Modellieren Sie das Nim-Spiel:
Auf dem Tisch liegt ein Haufen von z.B. anzahl = 5 Streichhölzern. Jeder Spieler muss eine Anzahl zwischen 1 und 3 Streichhölzern vom Tisch nehmen. Wer nicht mehr ziehen kann (weil kein Streichholz mehr da ist) hat verloren.

12) Spiel "Black Jacket"

Ihr Bekannter Herr X hat im Internet von einer Spielstrategie erfahren, mit der er mit dem Spiel "Black Jacket" im Spielcasino erfolgreich sein soll. Als vorsichtiger Mensch raten Sie ihm, dieses Vorhaben zuerst mal am Computer zu simulieren, bevor er im Spielcasino viel Geld verliert.

Zur Information:
Das Spiel "Black Jacket" ("17 + 4") funktioniert wie folgt beschrieben:
Das Ziel eines jeden Spielers ist es, möglichst nahe an die 21, aber nicht über die 21 zu kommen.
Spieler_1 "würfelt" so lange mit einem Spezialwürfel (mit dem genau eine der folgenden Zahlen 2, 3, 4, 7, 8, 9, 10, 11 erwürfelt werden kann), bie er aufhören will.
Die einzelnen Würfe sind verdeckt und vom Gegenspieler nicht zu sehen.
Dann macht der Gegenspieler Spieler_2 die gleiche Prozedur.
Wer die größere Summe seiner Würfe hat, hat gewonnen.
Wer eine Summe > 21 hat, hat verloren (wenn die Summe des Gegeners <22).
Bei gleicher Summe ist der Spielausgang unentschieden.
Hat jeder Spieler eine Summe > 21, ist der Spielausgang ebenfalls unentschieden.
Wenn alle 2 Spieler nicht mehr würfeln wollen (bzw. maximal 10 mal gewürfelt haben), wird das Spiel sofort beendet und auf dem Bildschirm die entsprechenden Infos ausgegeben (Gewinner und Verlierer mit jeweiliger Punktzahl).

Weisen Sie jedem Spieler eine Spielstrategie zu (z.B. hört nach 2 Würfen auf).
Simulieren Sie eine bestimmte Anzahl Spiele (Spieler_1 gegen Spieler_2) und berechnen Sie den Prozentsatz gewonnener Spiele von Spieler_1.
Damit kann man entscheiden, welche Strategie erfolgreicher ist.

JAVA-ÜBUNGSAUFGABEN KLASSEN 3

13)

Da es in manchen Gaststätten nach dem Essen ein kostenloses, alkoholhaltiges Getränk gibt (z.B. beim Griechen einen Uso), aber viele Leute nicht abschätzen können, was dieser für eine Wirkung hat, wäre es von Vorteil, wenn man wüßte wieviel halbe Liter Bier die gleiche Wirkung haben.
Ein Programmierer hat deswegen eine Klasse entwickelt, die u.a. die Methode getAnzahlBiere() enthält.

14)

a) Erstellen Sie die Klasse "Punkt", (mit den entsprechenden Attributen, Methoden und zwei Konstruktoren), die einen Punkt (Pixel) in einer grafischen zweidimensionalen Oberfläche repräsentieren soll.
b) Eine Gerade sei durch zwei Punkte gegeben. Erzeugen Sie die Klasse "Gerade".
Erstellen Sie dazu Methoden, die folgendes berechnen:
b1) y-Achsenabschnitt der Geraden
b2) Steigung der Geraden
b3) Ist eine Gerade senkrecht (d.h. parallel zur y-Achse)
b4) Sind zwei Geraden (die durch ihre Punkte festgelegt sind) identisch?
b5) Sind zwei Geraden (die durch ihre Punkte festgelegt sind) orthogonal ?
b6) Ausgabe der Geradengleichung auf dem Bildschirm.
b7) Berechnung des spitzen Winkels α zwischen zweier Geraden. $\tan\alpha = \frac{m_2 - m_1}{1 + m_1 m_2}$

Den Winkel bekommt man mit der Funktion atan in der Klasse Math

b8) Liegt ein Punkt auf einer Geraden?
b9) Schnittpunkt zweier Geraden

15)

Die Addition, Subtraktion, Multiplikation und Division zweier ganzen Zahlen des Datentyps int bzw. long ist leider nur innerhalb eines bestimmten Datenbereichs möglich.
Erstellen Sie eine Klasse, die diese elementaren Operationen innerhalb eines größeren Datenbereichs macht.

16)

Erstellen Sie eine 2D-Bibliothek, d.h. eine Vielzahl von Funktionen liefern, die den 2D betrifft: Schnittpunkt(e) zweier Gerade, Senkrechte, usw.

17) Mathematik (Primzahlen bestimmen)
Die folgende Funktion p(n) ermittelt die n-te Primzahl

$$p(n) = \sum_{m=1}^{2^n} \left[\sqrt[n]{n} \cdot \left(\sum_{k=1}^{m} \left[(\cos^2(\pi \frac{(k-1)!+1}{k})) \right] \right)^{-\frac{1}{n}} \right]$$

Bem: Die eckige Klammer bedeutet die Abrundungsfunktion
Beispiele: p(1) = 2, p(2) = 3, p(3) = 5
[2,8] = 2 [3,1415] = 3

Bem:
∑ bedeutet das Summenzeichen.
! bedeutet die Fakultät.
Für sehr große Zahlen empfiehlt sich der Datentyp BigDecimal bzw.
BigDecimalMath (diesen gibt es im Internet zum Download).