

## Übungsaufgaben 6

### OOP und Klassen

**Aufgabe 1** Modelliere das Objekt "Katze" als Java-Code, also schreibe eine Klasse, mithilfe der nachfolgenden Beschreibung.

Eine Katze *hat* eine Anzahl an Pfoten, Augen, Ohren und ein Gewicht. Katzen haben von Geburt an Fellfarben. Außerdem gibt es entweder den Zustand *tot* oder *lebendig*.

Die Katze *kann* laufen und fressen.

Beim Fressen kann ihr eine bestimmte Menge an Futter übergeben werden, wodurch sie schwerer wird.

```
public class Katze
{
    public int amountPaws;
    public int amountEyes;
    public int amountEars;

    public double weight;

    public String[] furColours;

    public boolean isAlive;

    public Katze()
    {
        this(4, 2, 2, 5.0, new String[] {"black",
            "white"}, true);
    }

    public Katze(int amountPaws, int amountEyes, int
        amountEars, double weight, String[] furColours,
        boolean isAlive)
    {
        this.amountPaws = amountPaws;
        this.amountEyes = amountEyes;
    }
}
```

```
        this.amountEars = amountEars;
        this.weight = weight;
        this.furColours = furColours;
        this.isAlive = isAlive;
    }

    public void move()
    {
    }

    public void eat(double amountFood)
    {
        this.weight += amountFood;
    }
}
```

## Aufgabe 2

Erweitere die Katze aus Aufgabe 1:

Eine Katze verliert Gewicht, je nachdem wie weit sie gelaufen ist. Falls die Katze schwerer als 10 kg ist, stirbt sie an den Folgen des Übergewichts (Beachte: Eine tote Katze kann nicht mehr fressen, ...).

```
public class Katze
{
    public int amountPaws;
    public int amountEyes;
    public int amountEars;

    public double weight;

    public String[] furColours;

    public boolean isAlive;

    public Katze()
    {
        this(4, 2, 2, 5.0, new String[] {"black",
            "white"}, true);
    }

    public Katze(int amountPaws, int amountEyes, int
        amountEars, double weight, String[] furColours,
        boolean isAlive)
    {
        this.amountPaws = amountPaws;
        this.amountEyes = amountEyes;
        this.amountEars = amountEars;
        this.weight = weight;
        this.furColours = furColours;

        if (weight <= 10.0)
        {
            this.isAlive = isAlive;
        }
    }

    public void move(double distance)
    {
        if (isAlive)
        {
```

```
        this.weight -= distance / 3;
    }
}

public void eat(double amountFood)
{
    if (isAlive)
    {
        this.weight += amountFood;

        if (this.weight > 10.0)
        {
            this.isAlive = false;
        }
    }
}
}
```

**Aufgabe 3** Modelliere das Objekt "Person" als Java Klasse vollständig, d.h. versuche dabei alle möglichen Member und Methoden zu finden.

**Lösung:** Hier gibt es viel Freiheiten, daher ist unten nur ein Lösungsvorschlag einer Teillösung.

```
public class Person
{
    public String name;
    public String address;

    public String sex;

    public int age;

    public Head head;
    public Arm[] arms;
    public Leg[] legs;

    // ...

    public void moveArm(int index, double... angle)
    {
    }

    public void moveLeg(int index, double... angle)
    {
    }

    // ...

    public void fart(double loudness)
    {
    }

    // ...
}
```

**Aufgabe 4** Modellierte die Member und Methoden des Objekts "Bankautomat" als Text. Als Hilfe kannst du die Begriffe *hat* und *kann* verwenden. Danach programmierst du das.

**Lösung:** Ein "Bankautomat" *hat* Display, Eingabebereich, Kartenschlitz, Ausgabeschlitz, Tresor, ...

Ein "Bankautomat" *kann* Karte einziehen und ausgeben, Geld ausgeben, Eingaben entgegennehmen, Ausgaben auf Display machen, ...

```
public class Bankautomat
{
    public Display display;
    public InputDevice inputDevice;
    public Cardholder cardholder;
    public MoneyOutput moneyOutput;
    public Safe safe;

    // ...

    public void putCardIn()
    {
    }

    public void putCardOut()
    {
    }

    public void outputMoney(double amountOfMoney)
    {
    }

    public void handleInput()
    {
    }

    public void print(String text)
    {
    }

    // ...
}
```