Organisatorisches

- mebis-Einschreibeschlüssel: computer
  (Unterricht – Informatik – Oberstufe – HGW Inf Q11 wege)
- Notengebung:
  Schulaufgabe
  Rechenschaftsablagen, praktisches Arbeiten am Rechner
  KA oder Ex?
  Kurztests
- Stoff:
  Datenstrukturen: Listen, Bäume, Graphen
  Durchführung von Softwareprojekten, Softwareentwicklung

Noch fit?

- Welche Datentypen gibt es in Java?
- Gib eine sinnvolle Klassenkarte für die Klasse ZYLINDER an.

- Implementiere die Klasse Zylinder in Java.

```
public class Zylinder {

    private double radius;
    private double hoehe;

    public Zylinder(double r, double h){
        radius = r;
        hoehe = h;
    }

    public double oberflaecheBerechnen(){
        return 2*radius*radius*Math.PI + 2*radius*Math.PI*hoehe;
    }

    public double volumenBerechnen(){
        return radius*radius*Math.PI*hoehe;
    }

    public void radiusSetzen(double r_neu){
        radius = r_neu;
    }

    public void hoeheSetzen(double h_neu){
        hoehe = h_neu;
    }
}
```

- Welche Zugriffsmodifikatoren gibt es in Java und wie werden sie verwendet?
  - private
  - public
  - protected

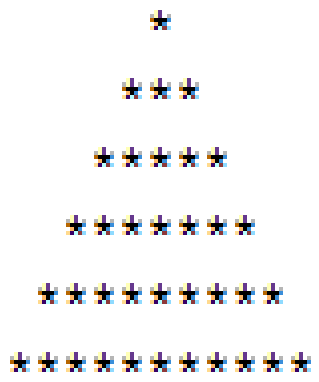
- Vergleiche die beiden Methoden:

```
public double oberflaecheBerechnen(){
    return 2*radius*radius*Math.PI + 2*radius*Math.PI*hoehe;
}

public double oberflaeche_Berechnen(){
    double oberflaeche;
    oberflaeche = 2*radius*radius*Math.PI;
    oberflaeche = oberflaeche + 2*radius*Math.PI*hoehe;
    return oberflaeche;
}
```

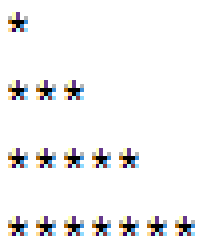
- Welche Kontrollstrukturen stehen zur Verfügung?

  Sequenz
  Alternative, bedingte Anweisung
  Wiederholung mit fester Anzahl
  Wiederholung mit Bedingung

- Implementiere eine Methode baumAusgeben(int b), die einen Baum aus Sternchen ausgibt. Es wird eine ungeradzahlige Eingabe erwartet, ansonsten wird eine Fehlermeldung ausgegeben.



oder einfacher:



```
public void baumAusgeben(int b) {
    if (b%2==0){
        System.out.println("Die Zahl muss ungerade sein!");
    }
    else {
        for (int i=0;i<b;i=i+2){
            for (int j=0;j<(b-i)/2;j++) {
                System.out.print(" ");
            }
            for (int k=0;k<=i;k++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Ganzzahldivision und Modulo

In der Grundschule: 17 : 5 = 3 R2

Ganzzahldivision in Java: 17 / 5 liefert 3

Werden in Java zwei Integer-Werte dividiert, ist das Ergebnis der ganzzahlige Anteil der Division.

Ist einer der Werte vom Datentyp double (bzw. float), erhält man das vom Taschenrechner gewohnte Ergebnis:

17.0/5 liefert 3.4

Ganzzahldivision und Modulo

Oft ist es nützlich, gerade den Rest einer Division zu erhalten. Dies ist mit dem modulo-Operator (%) möglich:

```
17%5 = 2
14%7 = 0
13%2 = 1
```

```
if (b%2==0){
        System.out.println("Die Zahl muss ungerade sein!");
    }
```

Falls die Zahl b gerade ist, wird die Textmeldung ausgegeben.

Textausgabe

Mit System.out.print("A") wird ein A ausgegeben.
Mit System.out.println("A") wird ein A ausgegeben und danach in die nächste Zeile gewechselt.

Der Text in Anführungsstrichen wird direkt ausgegeben. Es können aber auch Variablen (dann ohne Anführungsstriche) benutzt werden:

```
String wort = "Info";
System.out.println(wort);
```

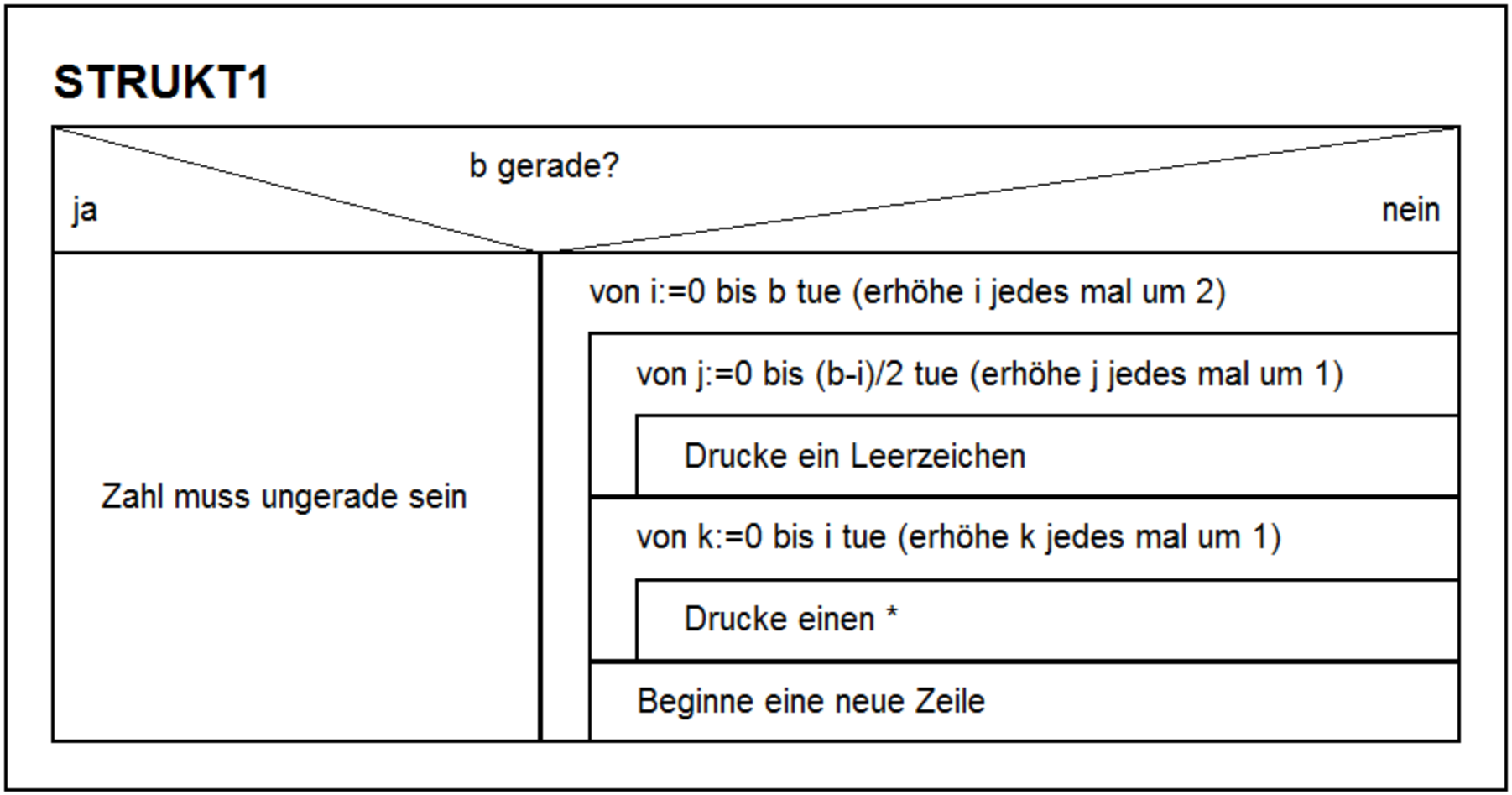
Ausgabe: Info

Es können auch Kombinationen mit + gebildet werden:

```
String wort = "Info";
System.out.println(wort + "rmatik");
```

Ausgabe: Informatik

Aber: System.out.println(1 + 2) liefert: 3


STRUKT1
b gerade?
ja
nein
Zahl muss ungerade sein
von i:=0 bis b tue (erhöhe i jedes mal um 2)
von j:=0 bis (b-i)/2 tue (erhöhe j jedes mal um 1)
Drucke ein Leerzeichen
von k:=0 bis i tue (erhöhe k jedes mal um 1)
Drucke einen *
Beginne eine neue Zeile

```
public void baumAusgeben(int b){
        if (b%2==0){
            System.out.println("Die Zahl muss ungerade sein!");
        }
        else {
            for (int i=0;i<b;i=i+2){
                for (int k=0;k<=i;k++) {
                    System.out.print("*");
                }
                System.out.println();
            }
        }
    }
```

- **Zusammengesetzter Datentyp: Feld**
  Die Höchsttemperaturen für den Monat August werden untersucht. Dabei sollen der heißeste Tag, sowie die Durchschnittstemperatur ausgegeben werden können. Implementiere ein Feld entsprechender Länge und entsprechende Methoden. Die Temperaturen sollen als Zufallszahlen (z.B. zwischen 10°C und 40°C) eingetragen werden.

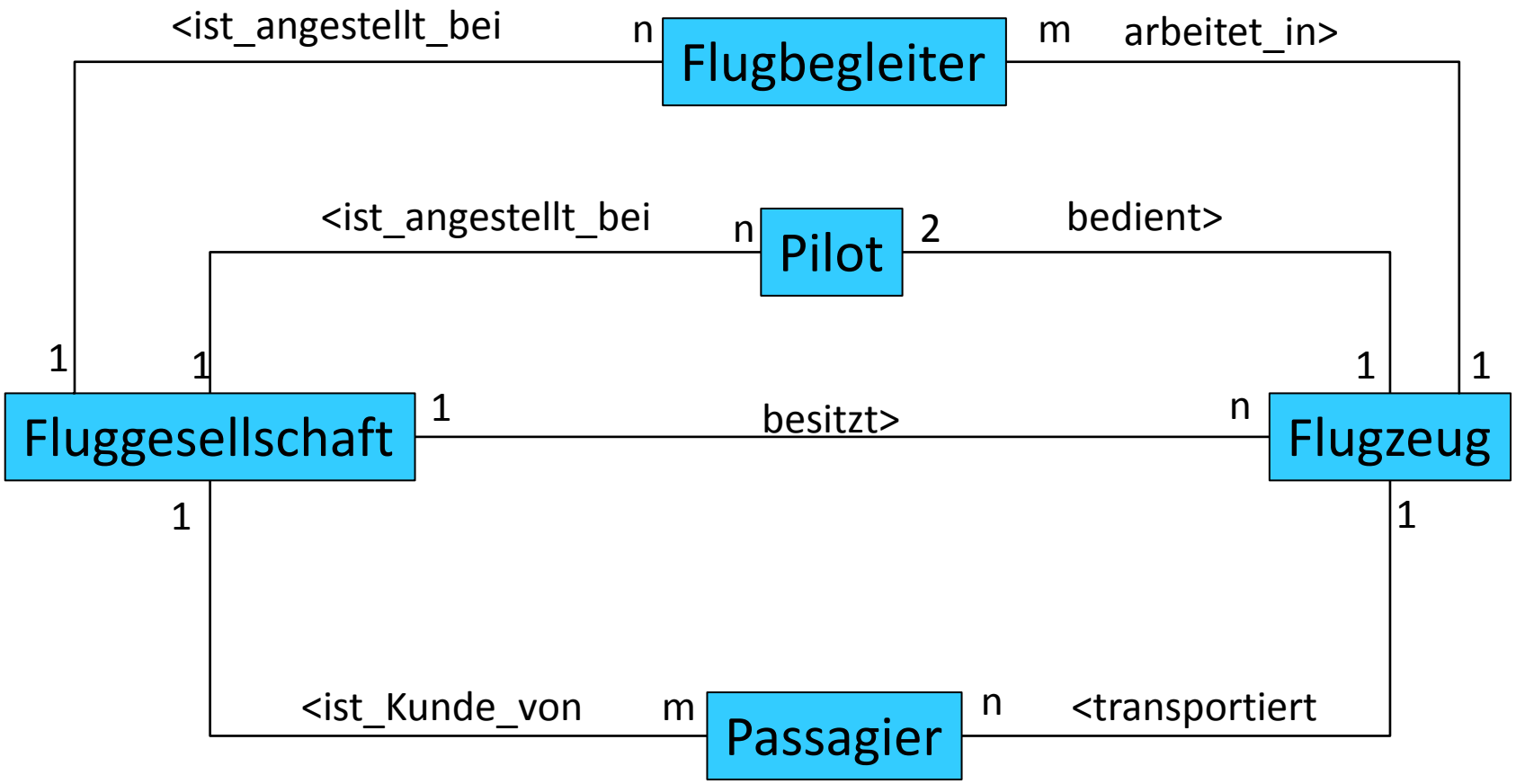
  ```
  private double[] temperatur  = new double[31];
  ```

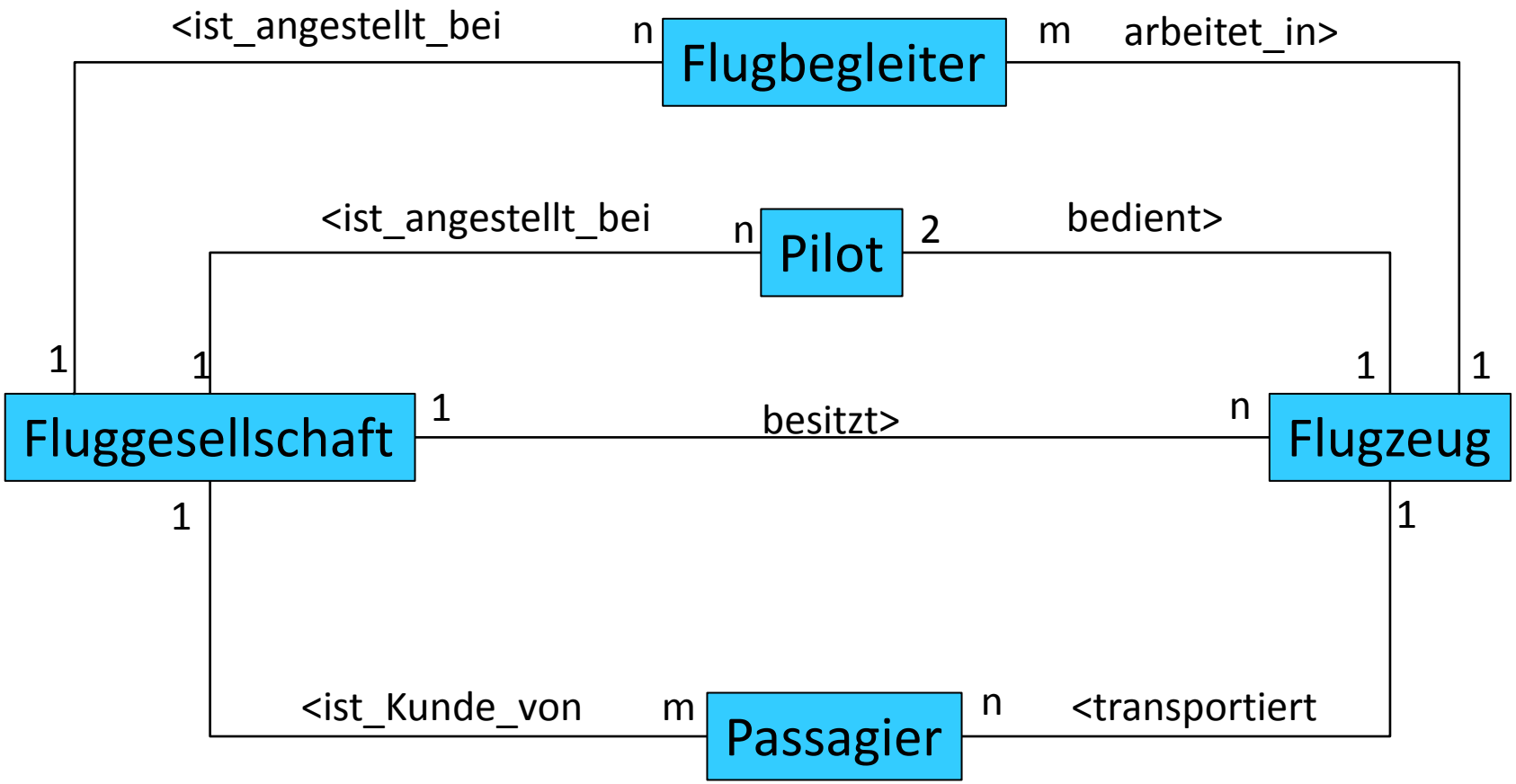
```
public class TempMessung {
    private double[] temperatur = new double[31];

    public void tempeintragen(){
        for (int i= 0; i<31; i++){
            temperatur[i]= 10+ Math.random()*30;
        }
    }
```

```
public int heissesterTag(){
    int tag=0;
    double maxtemp = temperatur[0];
    for (int i=1; i<31; i++){
        if (temperatur[i]>maxtemp) {
            maxtemp=temperatur[i];
            tag=i;
        }
    }
    return tag;
}
```

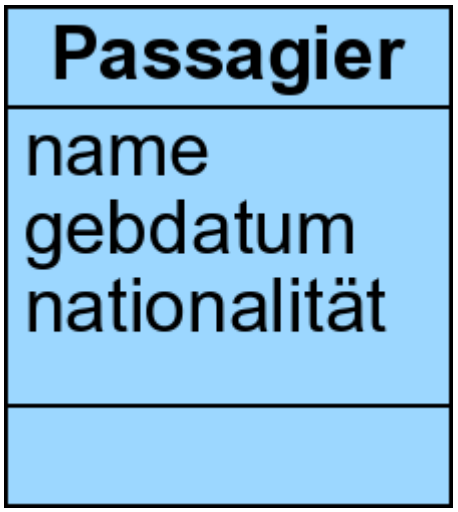
```
public double durchschnittsTemperatur(){
        double summe=0.0;
        for (int i=0; i<temperatur.length; i++){
            summe=summe+temperatur[i];
        }
        return summe/temperatur.length;
    }

}
```

- **Referenzen**

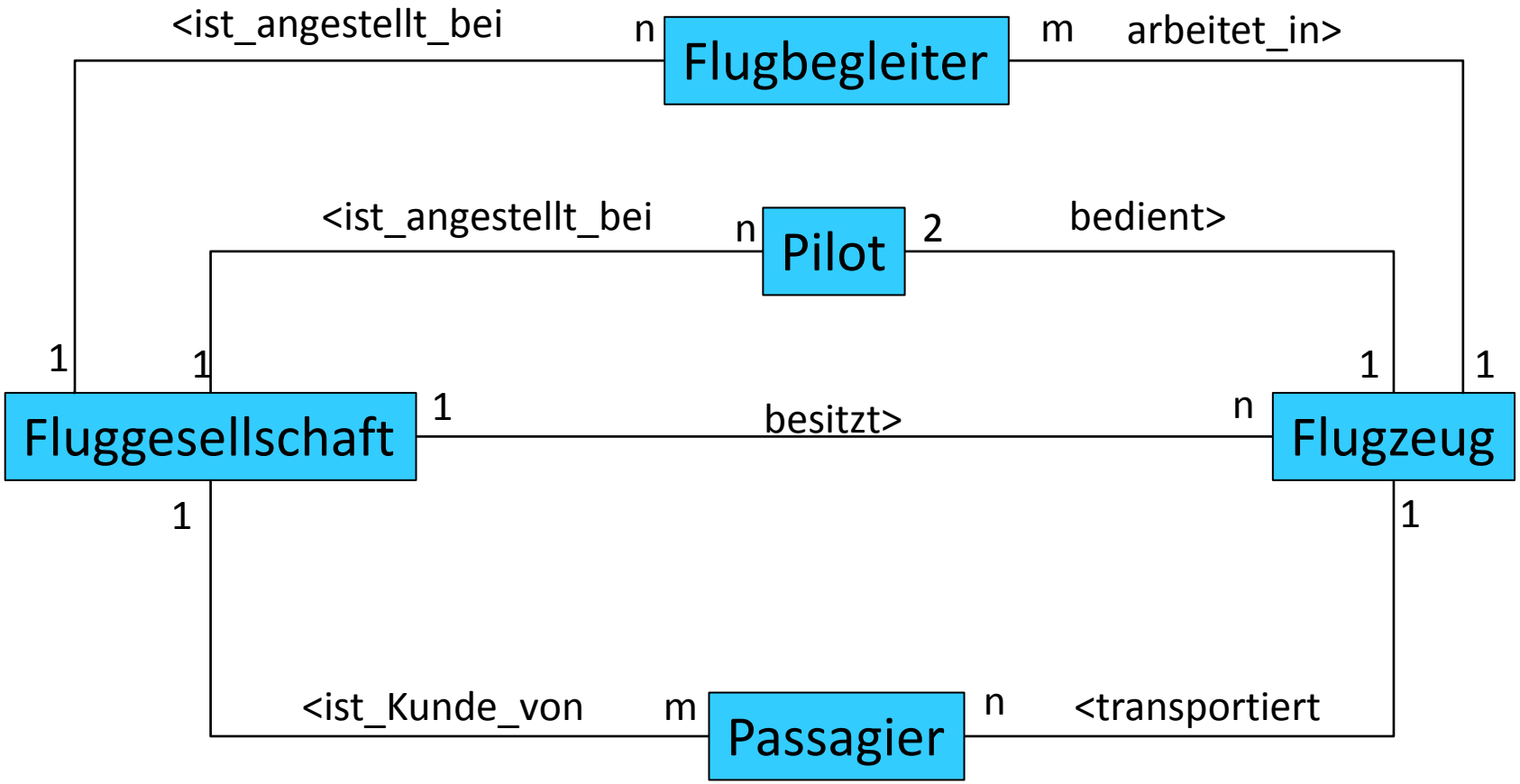

<ist_angestellt_bei
n
Flugbegleiter
m
arbeitet_in>
<ist_angestellt_bei
n
Pilot
2
bedient>
1
1
1
1
Fluggesellschaft
1
besitzt>
n
Flugzeug
1
1
<ist_Kunde_von
m
Passagier
n
<transportiert



Implementiere die Klasse Passagier. Setze dabei auch die Assoziationen "ist_Kunde_von" und "wird_transportiert_von" geeignet um.



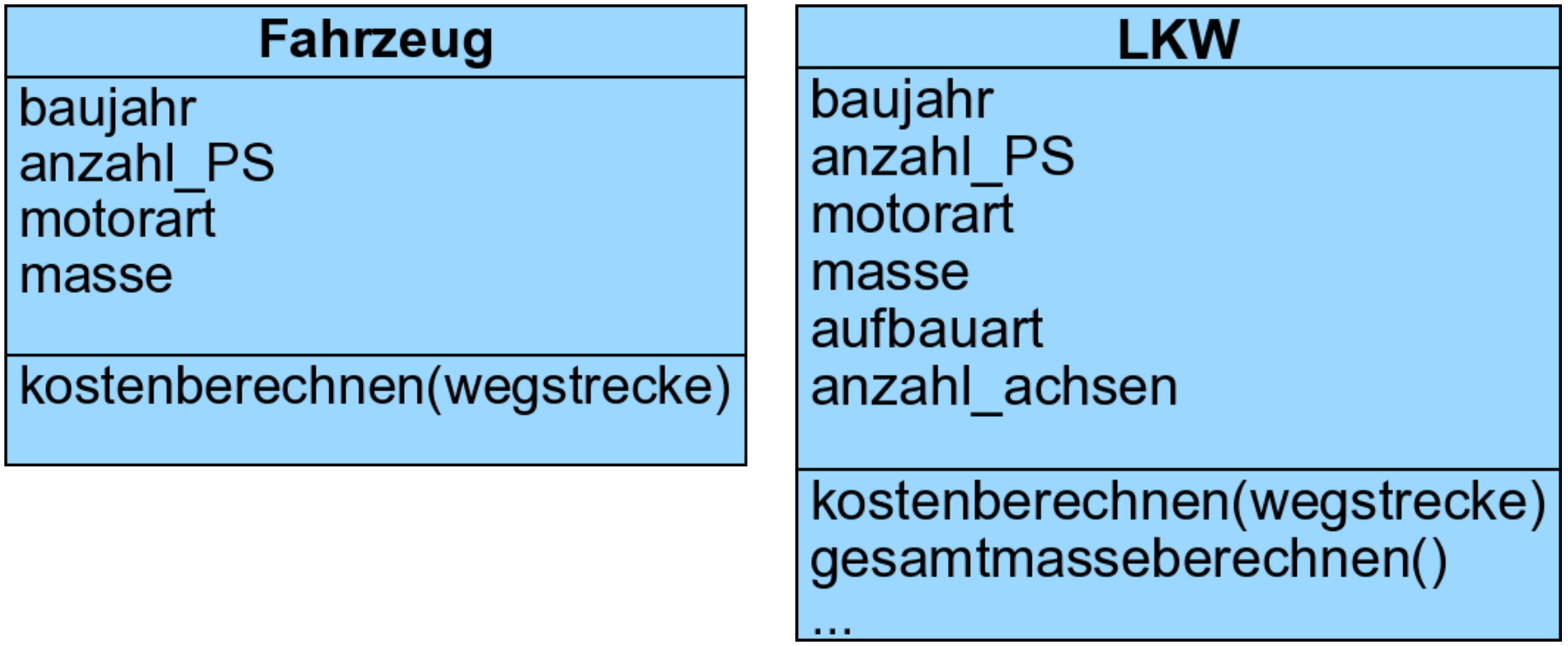

```
public class Passagier {

    private String name;
    private Datum gebdatum;
    private String nationalitaet;
    private Fluggesellschaft ist_Kunde_von;
    private Flugzeug wird_transportiert_von;

    public Passagier(String n, Datum geb, String nat,
                                Fluggesellschaft fg, Flugzeug f) {
        name = n;
        gebdatum = geb;
        nationalitaet = nat;
        ist_Kunde_von = fg;
        wird_transportiert_von = f;
    }
}
```

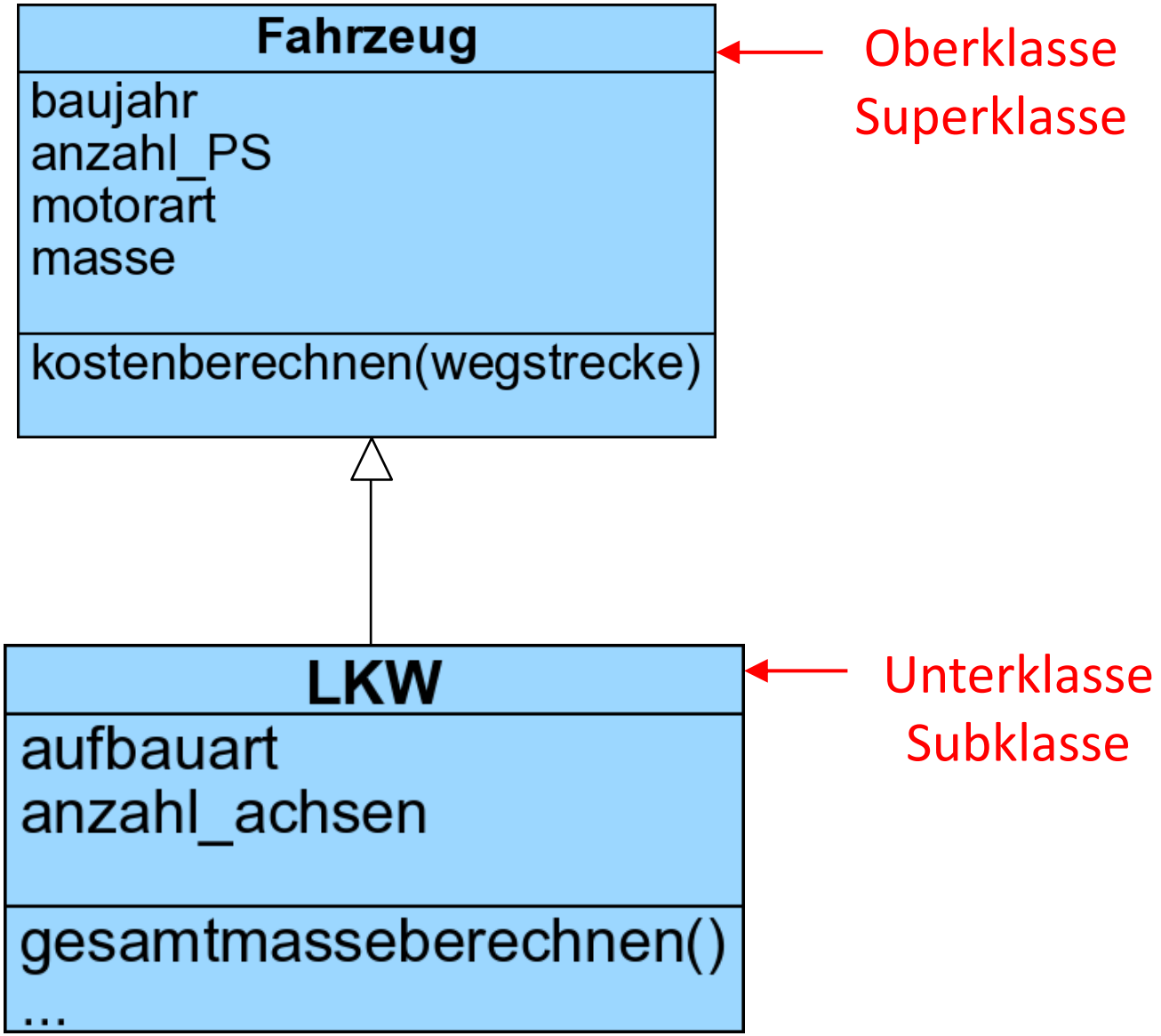


Implementiere die Klasse Fluggesellschaft. Setze dabei nur die Assoziation "besitzt" um. Gehe dabei davon aus, dass die Fluggesellschaft maximal 100 Flugzeuge besitzen kann.

```
public class Fluggesellschaft
{
    private String name;
    private Flugzeug[] flotte;


    public Fluggesellschaft(String n)
    {
        name =n;
        flotte = new Flugzeug[100];
    }
}
```

- **Vererbung**


Fahrzeug
baujahr
anzahl_PS
motorart
masse
kostenberechnen(wegstrecke)
LKW
baujahr
anzahl_PS
motorart
masse
aufbauart
anzahl_achsen
kostenberechnen(wegstrecke)
gesamtmasseberechnen()
...


Fahrzeug
baujahr
anzahl_PS
motorart
masse
kostenberechnen(wegstrecke)
Oberklasse
Superklasse
LKW
aufbauart
anzahl_achsen
gesamtmasseberechnen()
...
Unterklasse
Subklasse

Implementiere die Klasse Fahrzeug. Die Kosten sollen mit 0,50€ pro gefahrenen Kilometer veranschlagt werden.

```
public class Fahrzeug {
   protected int baujahr;
   protected int anzahl_PS;
   protected String motorart;
   protected double masse;

   public Fahrzeug(int b, int a, String ma, double m) {
      baujahr = b;
      anzahl_PS = a;
      motorart = ma;
      masse = m;
    }

   public double kostenberechnen(double wegstrecke) {
      return wegstrecke*0.5;
   }
}
```

protected erlaubt den Zugriff von Objekten einer Unterklasse.

Implementiere die Klasse LKW!

public class LKW **extends Fahrzeug** {...}

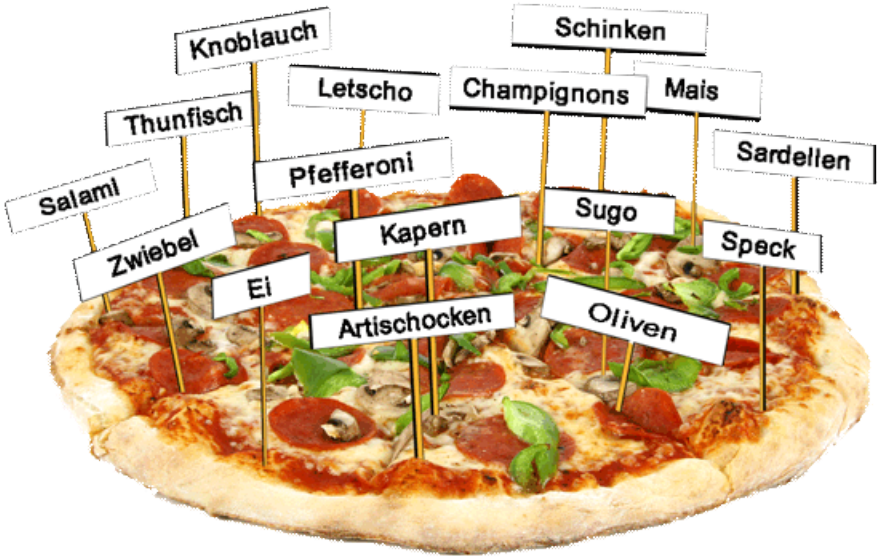
Im Konstruktor: super(*Parameterliste*);

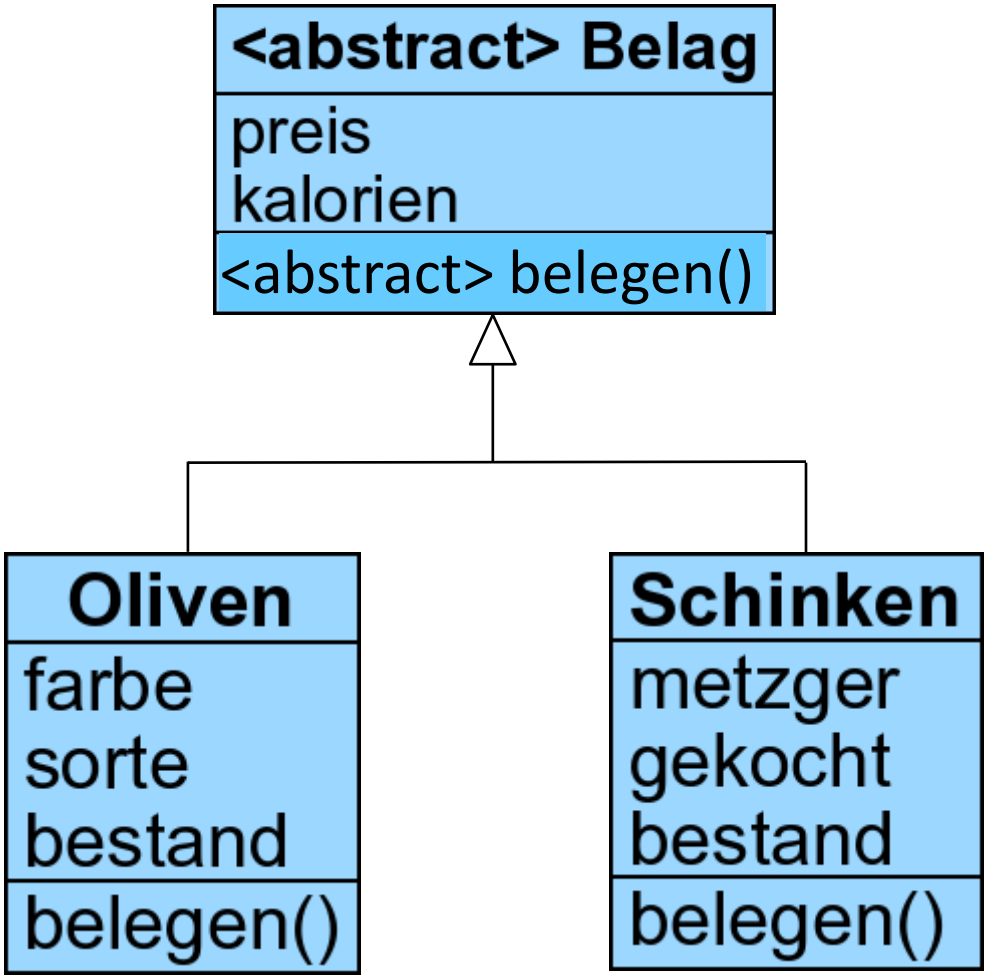
```
public class LKW extends Fahrzeug {
    private String aufbauart;
    private int anzahl_achsen;

    public LKW(int b, int aPS, String ma, double m, String aa,
                    int anza) {
      super(b, aPS, ma, m);
      aufbauart = aa;
      anzahl_achsen = anza;
    }

    public double gesamtmasseberechnen(double ladung){
       return masse+ladung;
    }
}
```

Für einen LKW berechnen sich die Wegkosten anders:
Es wird eine einmalige Bereitstellungsgebühr von 100 € verlangt und pro gefahrenen Kilometer 2 €.
Implementiere eine entsprechende Methode kostenberechnen(double wegstrecke) in der Klasse LKW.

```
public double kostenberechnen(double wegstrecke)
  {
     return 100+wegstrecke*2;
  }
```

- Abstrakte Klassen





```
public abstract class Belag {

    protected double preis;
    protected double kalorien;

    public Belag(double p, double k){
        preis = p;
        kalorien = k;
    }

    public abstract void belegen();
}
```

```
public class Oliven extends Belag {
    private String farbe;
    private String sorte;
    private int bestand;

    public Oliven(double p, double k){
        super(p, k);
        farbe = "grün";
        sorte = "Manzanilla";
        bestand = 100;
    }

    public void belegen(){
        bestand = bestand - 10;
    }
}
```

```
public class Schinken extends Belag {

    private String metzger;
    private boolean gekocht;
    private double bestand;

    public Schinken(double p, double k, boolean g){
        super(p, k);
        metzger = "Kremmel";
        gekocht = g;
        bestand = 1.5;
    }

    public void belegen(){
        bestand = bestand - 0.1;
    }
}
```

Ende - Wiederholung

```
public class Pizzabaecker {

   private Oliven o1;
   private Schinken s_gek;
   private Schinken s_roh;

   public Pizzabaecker() {
      o1 = new Oliven(0.8, 95);
      s_gek = new Schinken(0.9, 200, true);
      s_roh = new Schinken(1.4, 180, false);
    }

    public void backen() {
       o1.belegen();
       s_gek.belegen();
       s_roh.belegen();
    }
}
```