Elektronikschule Tettnang	Softwareentwicklung	Datum:	Klasse:
	Assoziationen	Name:	

Eine wichtige Eigenschaft objektorientierter Systeme ist der gegenseitige Zugriff zwischen Objekten. Dazu »kennt« ein Objekt andere Objekte und kann Anforderungen weitergeben. Diese Verbindung nennt sich Assoziation und ist ein wichtigste Werkzeug bei der Bildung von Objektverbänden.

Assoziationstypen:

Bei Assoziationen ist zu unterscheiden, ob nur eine Seite die andere kennt oder ob eine Navigation in beide Richtungen möglich ist. Dies nennt man auch Navigierbarkeit.

- Eine unidirektionale Beziehung geht nur in eine Richtung (ein Fan kennt seine Band, aber nicht umgekehrt).
- Eine bidirektionale Beziehung geht in beide Richtungen (Computerspiel: Raum kennt Spieler und Spieler kennt Raum). Eine bidirektionale Beziehung ist natürlich ein großer Vorteil, da die Anwendung die Assoziation in beliebiger Richtung ablaufen kann.

Daneben gibt es bei Beziehungen die Multiplizität. Sie sagt aus, mit wie vielen Objekten eine Seite eine Beziehung haben kann. Sie werden auf beiden Seiten angegeben.

0..* oder *	beliebig viele
1	genau eins
1..*	mindestens eins
4..*	mindestens 4
3..5	3 bis 5

Beispiel einer Assoziation zwischen den Klassen Auto und Fahrer:

unidirektionale Assoziation mit der Multiplizität 1 zu 1

```
public class Auto {
    private String farbe;
    ... // weitere Eigenschaften und Methoden
}


public class Fahrer {
    private Auto auto;
    ... // weitere Eigenschaften und Methoden

    public void setAuto(Auto auto) {
        this.auto = auto;
    }

    public Auto getAuto() {
        return auto;
    }
}

public class TestUnidirektionaleAssoziation {
    public static void main(String args[]) {
        Auto golf = new Auto();
        Fahrer michael = new Fahrer();

        michael.setAuto(golf);
    }
}
```

bidirektionale Assoziation mit der Multiplizität 1 zu 1

```
public class Auto {
    private Fahrer fahrer;
    private String farbe;
    ... // weitere Eigenschaften und Methoden

    public void setFahrer(Fahrer fahrer) {
        this.fahrer = fahrer;
    }

    public Fahrer getFahrer() {
        return fahrer;
    }
}

public class Fahrer {
    private Auto auto;
    private String name;
    ... // weitere Eigenschaften und Methoden

    public void setAuto(Auto auto) {
        this.auto = auto;
    }

    public Auto getAuto() {
        return auto;
    }

    public String getName() {
        return name;
    }
}

public class TestBidirektionaleAssoziation
{
    public static void main(String args[])
    {
        Auto golf = new Auto();
        Fahrer michael = new Fahrer();

        michael.setAuto(golf);
        golf.setFahrer(michael);

        // Zugriff
        System.out.println(golf.getFahrer().getName());
    }
}
```

unidirektionale Assoziation mit der Multiplizität 1 zu *

- bei der 1 zu * Assoziation „kennt“ ein Objekt mehrere Objekte. Eine einfache Variable für die Assoziation reicht dann nicht mehr aus. Es muss eine Datenstruktur verwendet werden, die mehrere Objekte speichern kann, z.B. ein Array oder eine ArrayList.

```
public class Auto {
    private String farbe;
    ... // weitere Eigenschaften und Methoden
}


public class Fahrer {
    private ArrayList<Auto> autos;
    ... // weitere Eigenschaften und Methoden

    public void addAuto(Auto auto) {
        this.autos.add(auto);
    }

    public ArrayList<Auto> getAutos() {
        return autos;
    }
}

public class TestUnidirektionaleAssoziation {
    public static void main(String args[]) {
        Auto golf = new Auto();
        Auto bmw = new Auto();
        Fahrer michael = new Fahrer();

        michael.addAuto(golf);
        michael.addAuto(bmw);

        // späterer Zugriff
        for(Auto auto : autos) {
            System.out.println(auto.getKennzeichen());
        }
    }
}
```