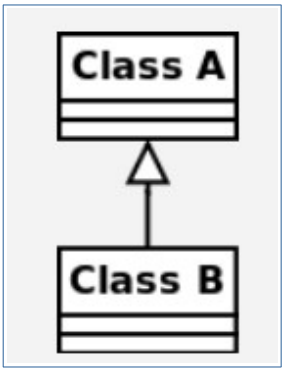
„Types of Relationships“

1. Verebung: ist eine Beziehung:

Vererbung ist ein Konzept, bei dem eine Klasse (Unterklasse) die Eigenschaften und Methoden einer anderen Klasse (Oberklasse) erbt. Dies ermöglicht die Wiederverwendung von Code und die Erweiterung der Funktionalität einer Klasse.



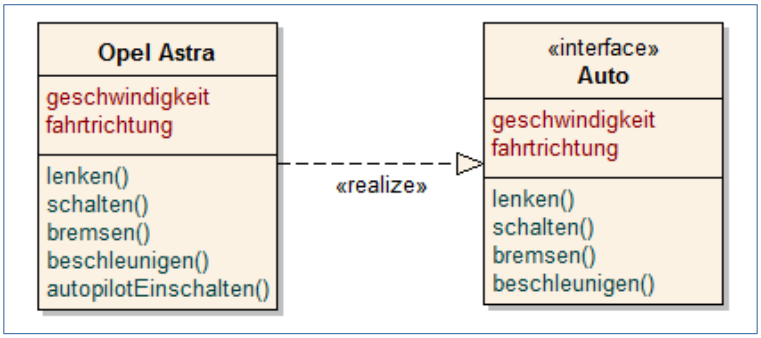
Mehrfachvererbung (Schnittstellen) und abstrakte Klassen:

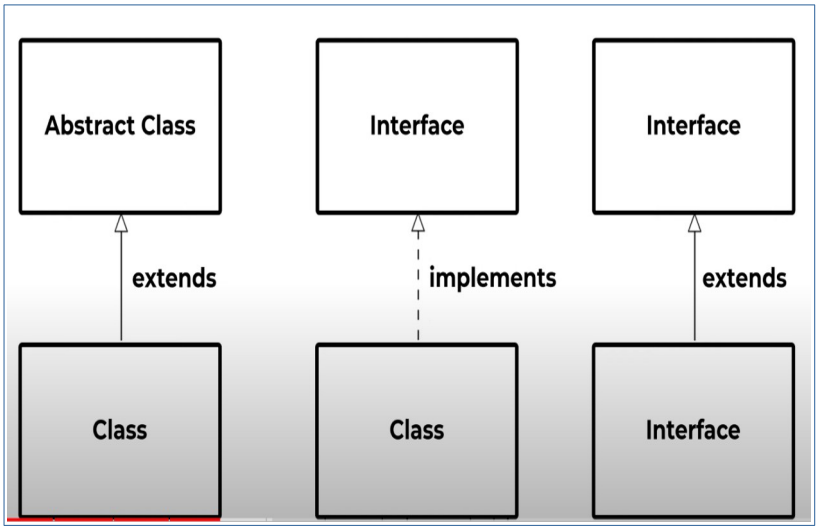
Zweck des Begriffes ist es, eine abgeschwächte Form der Mehrfachvererbung zuzulassen. Dazu wird das neue Schlüsselwort implements eingeführt. Eine Klasse kann nur einmal von einer anderen Klasse erben, aber beliebig viele Interfaces implementieren

```
public class Child extends Parent implements Face1, Face2, Face3
{

}
```

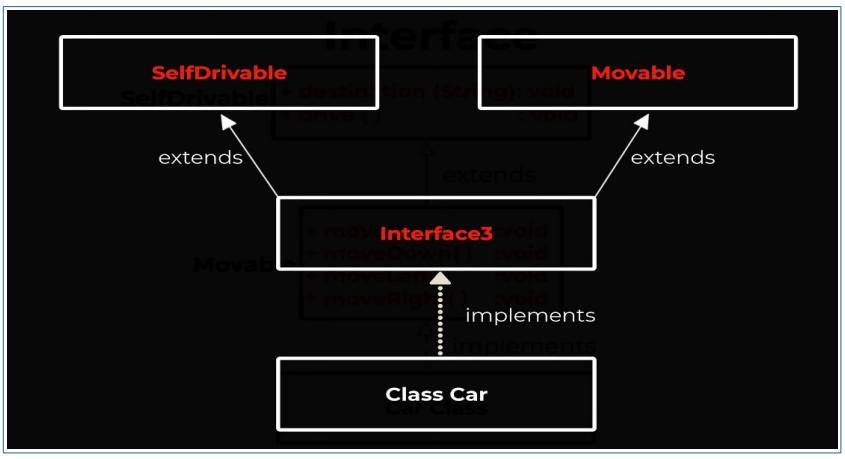
Interfaces haben in java zwei Aufgaben:

1. dienen für Mehrfachvererbung
2. dienen als abstrakte klassen

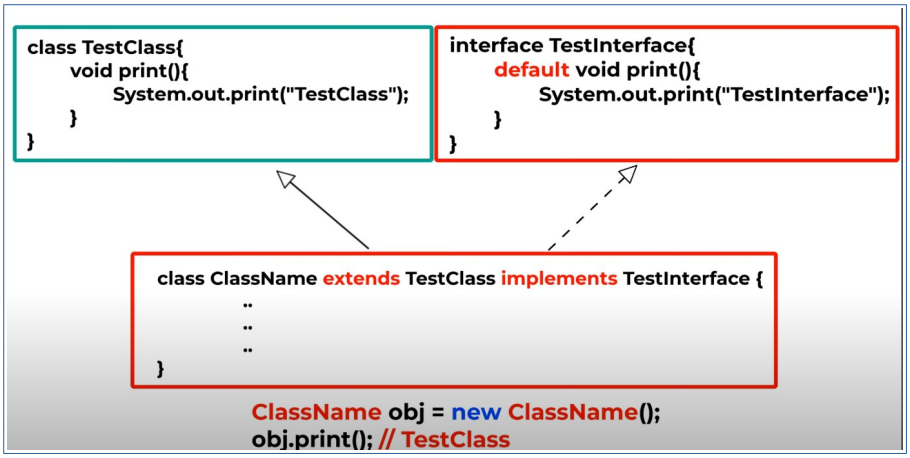




Interfaces3 erbt die zwei Interface Klassen und die klasse Car implements die interface klasse 3



Wenn eine Klasse extends eine andere klasse und gleichzeitig implements ein Interface. Das Objekt der Unterklasse wir die Methoden der extendes Klasse aufrufen, da extends stärker als Interface ist



2. Association:

Assoziation beschreibt eine Beziehung zwischen zwei unabhängigen Klassen. Diese Beziehung kann in beide Richtungen bestehen und beschreibt, wie Objekte der beiden Klassen miteinander interagieren.

Multiplicity in der Objektorientierten Programmierung:

- –..1 keiner oder einer, d.h. optionale Kann-Assoziation
- – 1 genau einer, d.h. Muss-Assoziation
- – * beliebig viele, d.h. optionale Kann-Assoziation
- – 1..* mindestens einer, d.h. Muss-Assoziation

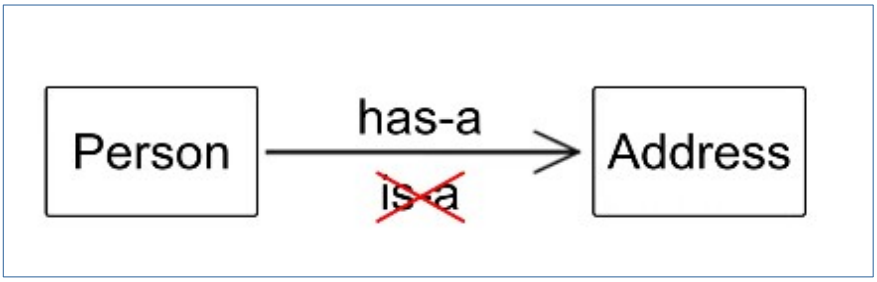
Besteht aus zwei Arten:

2.1 (Unary):

die Klasse Person weißt alles über

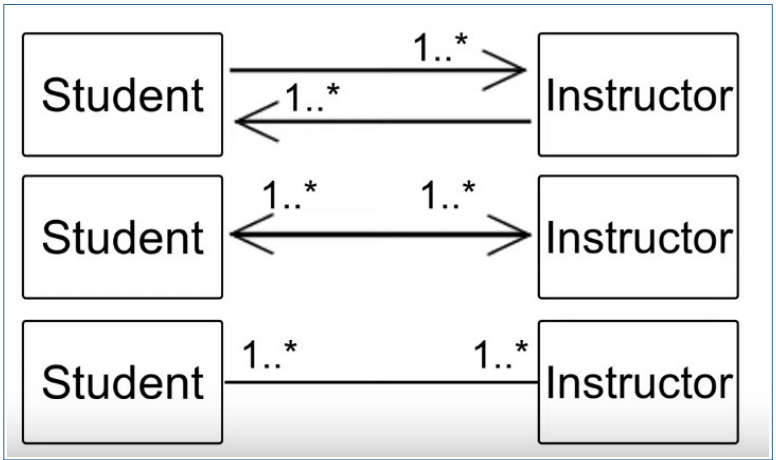
Klasse Adresse aber die Klasse Adresse weiß nicht über die Klasse Person.

Die Klasse Person hat eine Adresse ***(has-a)***



2.2 (Binary):

die beide Klassen kennen einander ein oder mehrere Student hat ein oder mehrere Profs und umgekehrt.

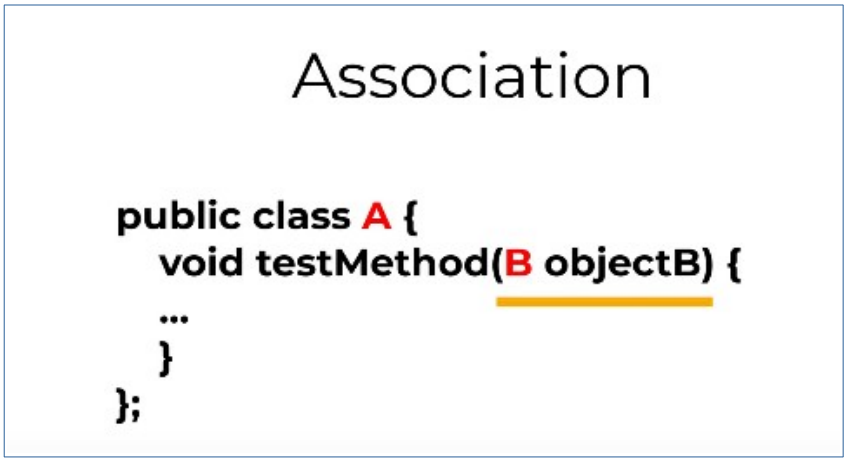


Die drei Arten von Pfeilen sind gültig

(has-a)

Wichtig:

Bei association wird wie im Bild die Klasse A eine Methode erstellen, die ein Objekt aus der Klasse B als Parameter dieser Methode bekommen hat (kein Objekt) definieren!!



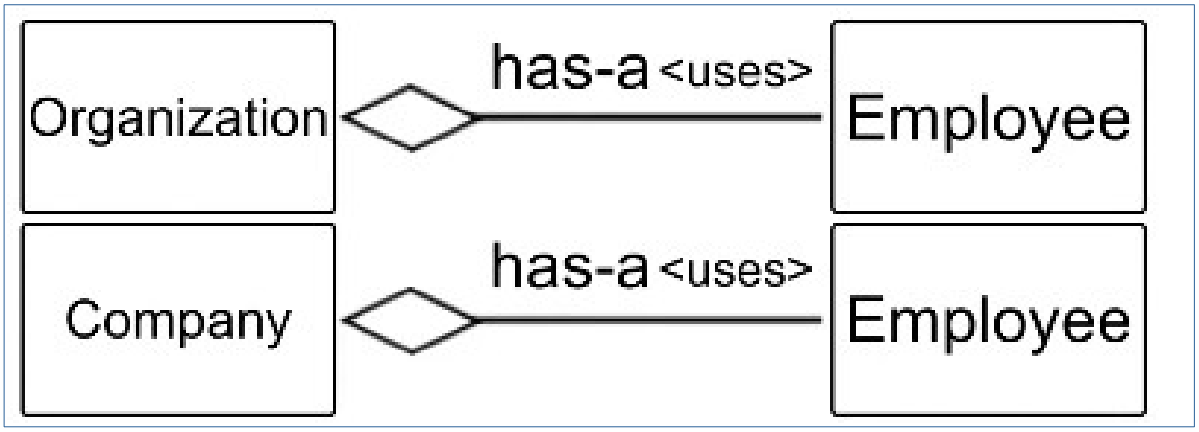
3. Aggregation (has-a) Beziehung:

In einer Aggregation (has-a) Beziehung hat ein Objekt eines Typs ein oder mehrere Objekte eines anderen Typs. Zum Beispiel hat ein Auto einen Motor. Der Motor kann jedoch auch ohne das Auto existieren, deswegen heißt diese Beziehung *(Weak relationship)*

Schaubild 1: Ein organization kann doch ohne Mitarbeiter laufen

Ein Auto hat einen Motor, aber ein Motor kann unabhängig von einem Auto existieren.

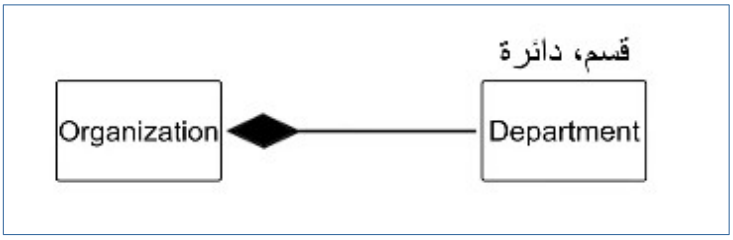
```
class Motor {
    private String typ;

    public Motor(String typ) {
        this.typ = typ;
    }

    public void starten() {
        System.out.println(typ + " Motor gestartet");
    }
}

// Klasse Auto
class Auto {
    private String marke;
    private Motor motor;  // Aggregation: Auto hat einen Motor

    public Auto(String marke, Motor motor) {
        this.marke = marke;
        this.motor = motor;
    }

    public void fahren() {
        System.out.println(marke + " fährt los");
        motor.starten();
    }
}
```



4. Composition (part-a) Beziehung:

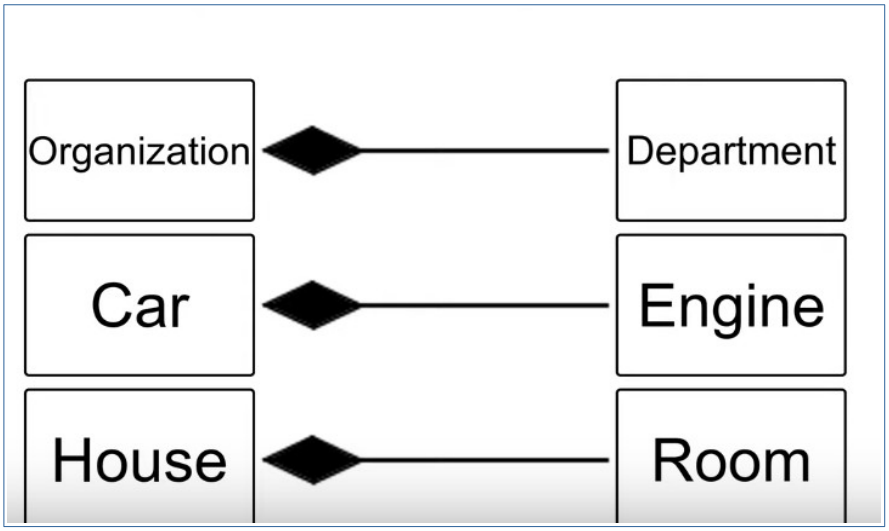
In einer Komposition (has-a) Beziehung hat ein Objekt eines Typs ein oder mehrere Objekte eines anderen Typs *(Strong relationship)*

((bei der Komposition die enthaltenen Objekte nicht unabhängig vom Container existieren können. Wenn der Container zerstört wird, werden auch die enthaltenen Objekte zerstört)).



Wenn die Klasse Catr zerstört, haben wir keine Mototr meh.

Wenn die Klasse House zerstört, haben wir keinen Room mehr



Aggregation vs Compositon: