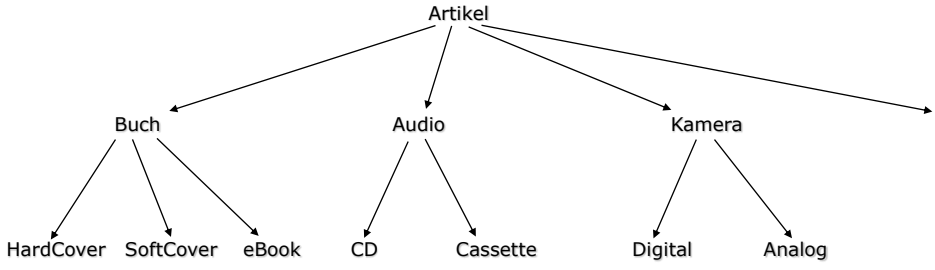
Vererbung

Klassifikation

Dinge der realen Welt lassen sich oft klassifizieren

z. B. Artikel eines Web- Shops



Man beachte

- Ein eBook hat alle Eigenschaften eines Buchs; zusätzlich hat es ...
- Ein Buch hat alle Eigenschaften eines Artikels; zusätzlich hat es ...
- CD und Cassette lassen sich gleichermaßen als Audio behandeln

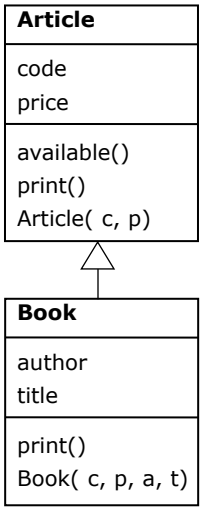
Buch, Audio und Kamera lassen sich gleichermaßen als Artikel behandeln

Vererbung

Vererbung

```
class Article {
    int code;
    int price;
    bool available() {...}
    void print() {...}
    Article( int c, int p) {...}
}
```

Oberklasse
Basisklasse




```
class Book extends Article {
    String author;
    String title;
    void print() {...}
    Book( int c, int p,
        String a, String t) {...}
}
```

Unterklasse

erbt: code, price, available, print
ergänzt: author, title, Konstruktor
überschreibt: print

Wenn keine Oberklasse angegeben wird, ist sie *Object*

Überschreiben von Methoden

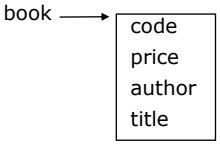
```
class Article {
    ...
    void print() {
        Out. print( code + " " + price + " ");
    }
    Article( int c, int p) {
        code = c; price = p;
    }
}
```

```
class Book extends Article {    ...
    void print() {
        super. print();
        Out. print( author + " " + title+ " ");
    }
    Book( int c, int p, String a, String t) {
        super( c, p);
        author = a; title = t;
    }
}
```

Benutzung

```
Book book = new Book( code, price, author, title);
```

erzeugt *Book*- Objekt
Article- Konstruktor
Book- Konstruktor

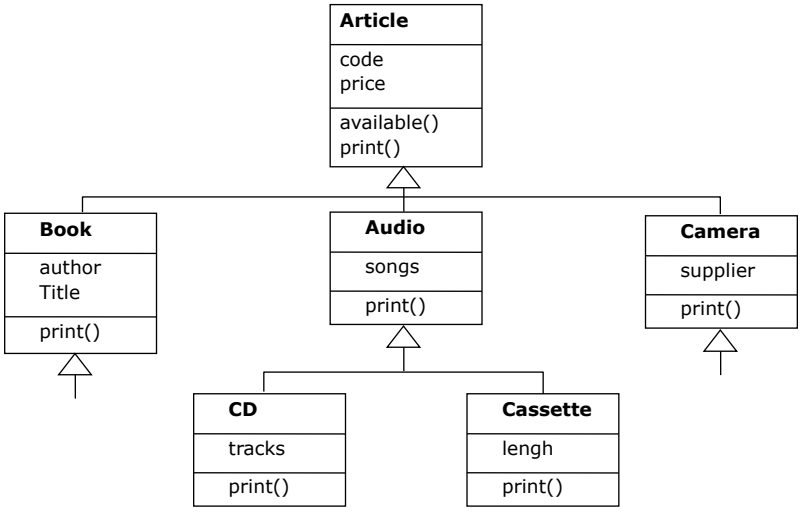



```
book. print();
```

print aus Article
print aus Book

Ausgabe: code price author title

Klassenhierarchien



Jedes Buch ist ein Artikel
Aber: nicht jeder Artikel ist ein Buch

Kompatibilität zwischen Klassen

Unterklassen sind Spezialisierungen ihrer Oberklassen

Book- Objekte können *Article*- Variablen zugewiesen werden

```
Article a = new Book( code, price, author, title);
```

a → code, price, author, title

nur *Article*- Felder sind über *a* zugreifbar
- a.code
- a.price

```
if (a instanceof Book)          // Laufzeittyptest
   Book b = (Book) a;           // Typumwandlung mit Laufzeittypprüfung
```

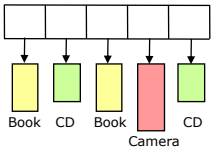
b → code, price, author, title

alle Book- Felder sind über *b* zugreifbar
- b.code
- b.price
- b.author
- b.title

Dynamische Bindung

Heterogene Datenstruktur

Article[] a;



Alle Varianten können als Artikel behandelt werden

```
void printArticles() {
    for (int i = 0; i < a. length; i++) {
        if (a[i].available()) {
            a[i].print();
        }
    }
}
```

ruft geerbtes *available()* aus Article auf

ruft je nach Artikelart das *print()* aus *Book, CD* oder *Camera* auf

Dynamische Bindung

a.print() ruft die print- Methode des Objekts auf, auf das a gerade zeigt