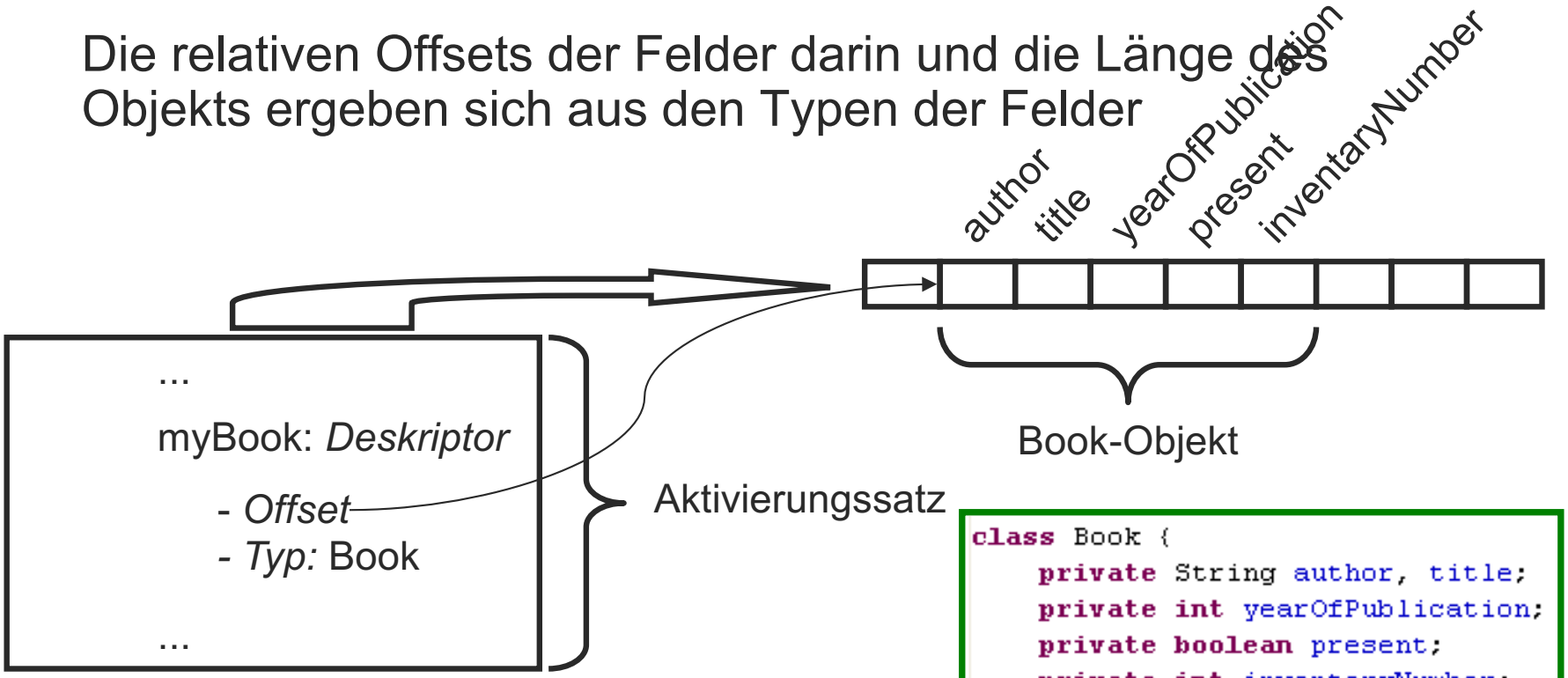
Speicherverwaltung und bekannte Objekte

- Verwaltung von Objekten im Speicher
- Zuweisungen von Objekten vs.
  Zuweisungen einfacher Daten
- Bekannte Objekte
- Schachtelung von Objekten

Verwaltung von Objekten

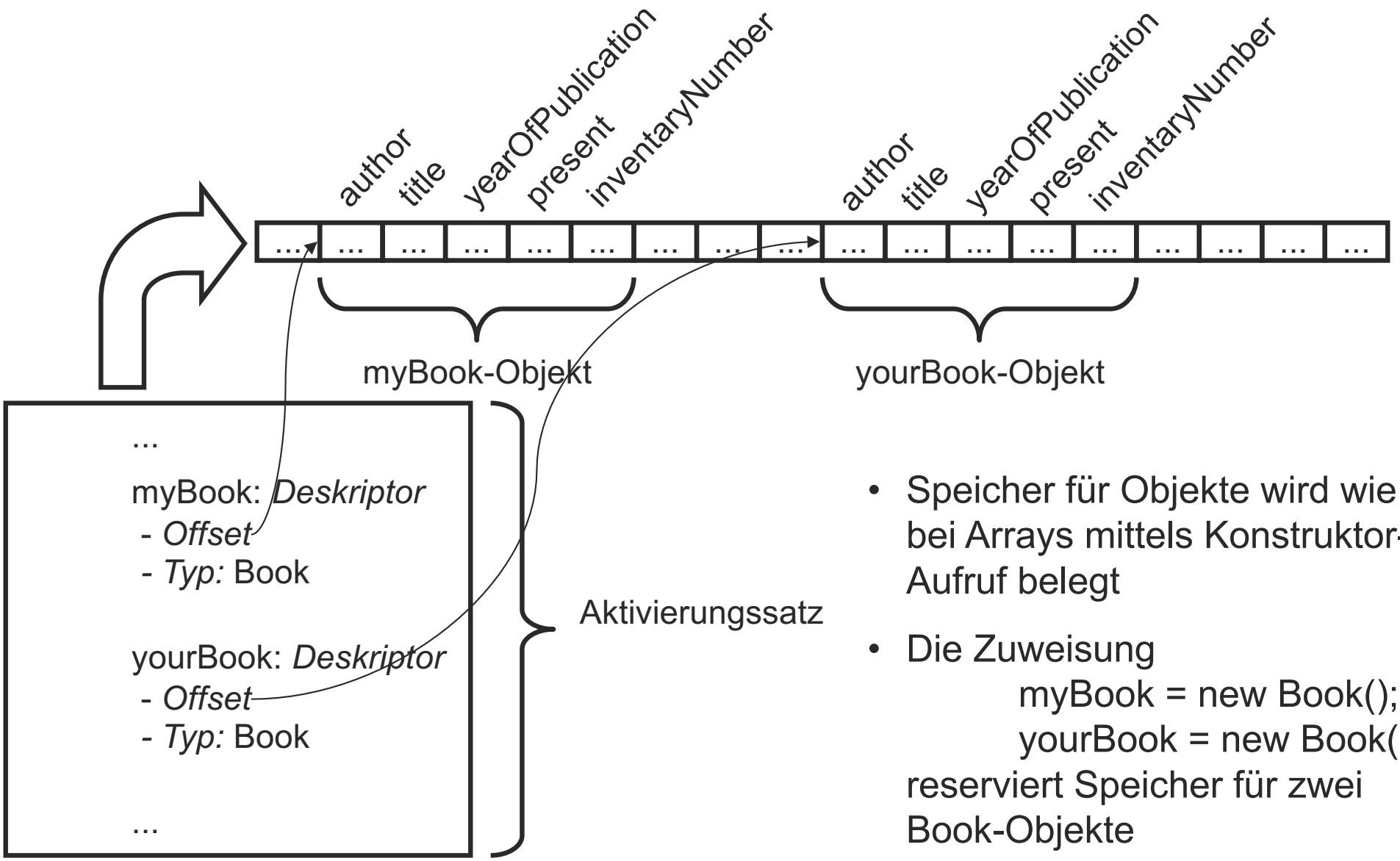
- Objekte werden, ebenso wie Arrays, mittels *Deskriptoren* verwaltet
- Allerdings ist im Deskriptor für Objekte nicht wie beim Array die Länge interessant, sondern
  - neben dem *Offset* des Objekts noch
  - der *Typ* des Objekts (also seine Klasse)

Die relativen Offsets der Felder darin und die Länge des Objekts ergeben sich aus den Typen der Felder




```
class Book {
    private String author, title;
    private int yearOfPublication;
    private boolean present;
    private int inventaryNumber;

}
```

Anlegen von Objekten mittels Konstruktor

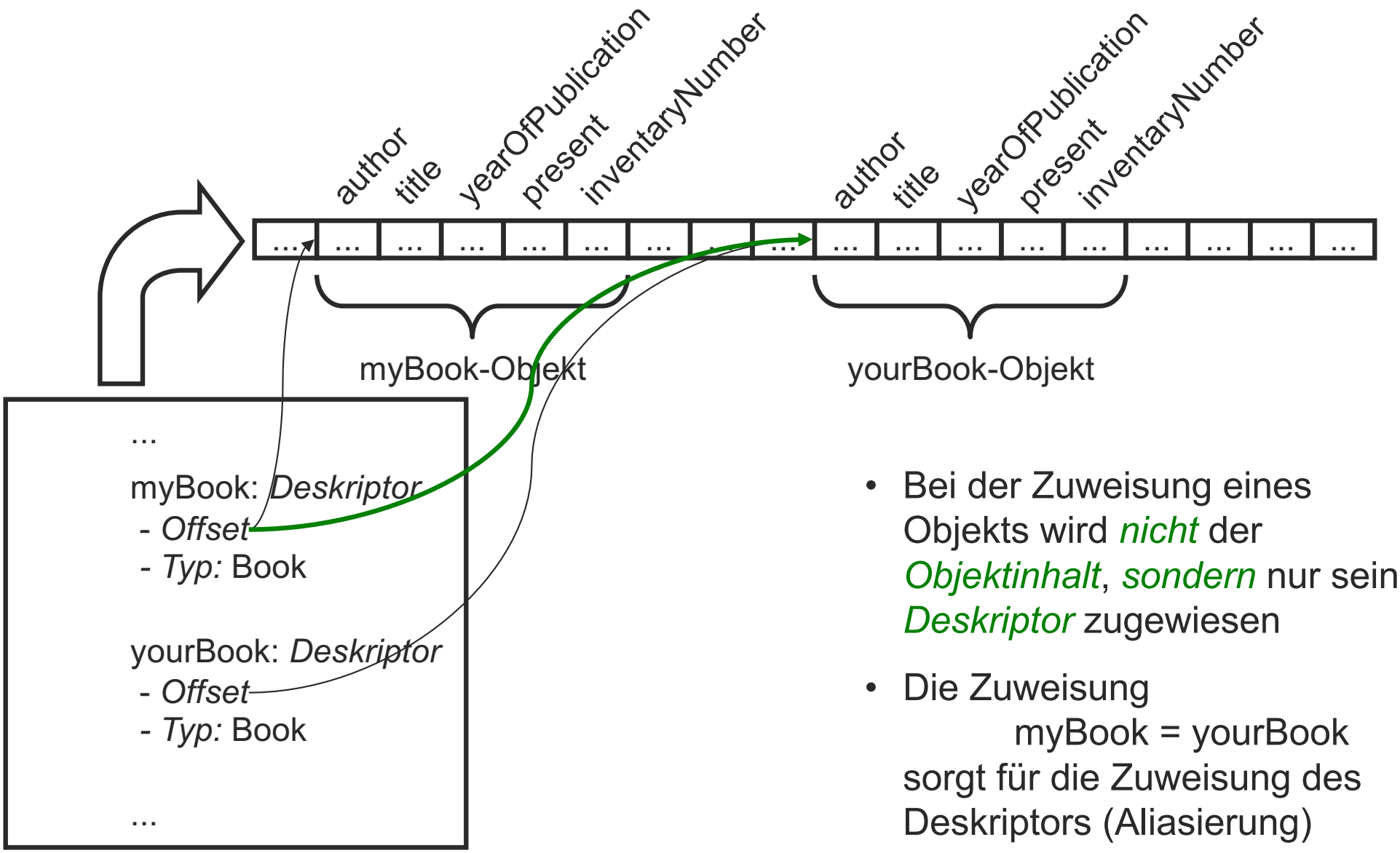


- Speicher für Objekte wird wie bei Arrays mittels Konstruktor-Aufruf belegt
- Die Zuweisung

  ```
  myBook = new Book();
  yourBook = new Book();
  ```

  reserviert Speicher für zwei Book-Objekte

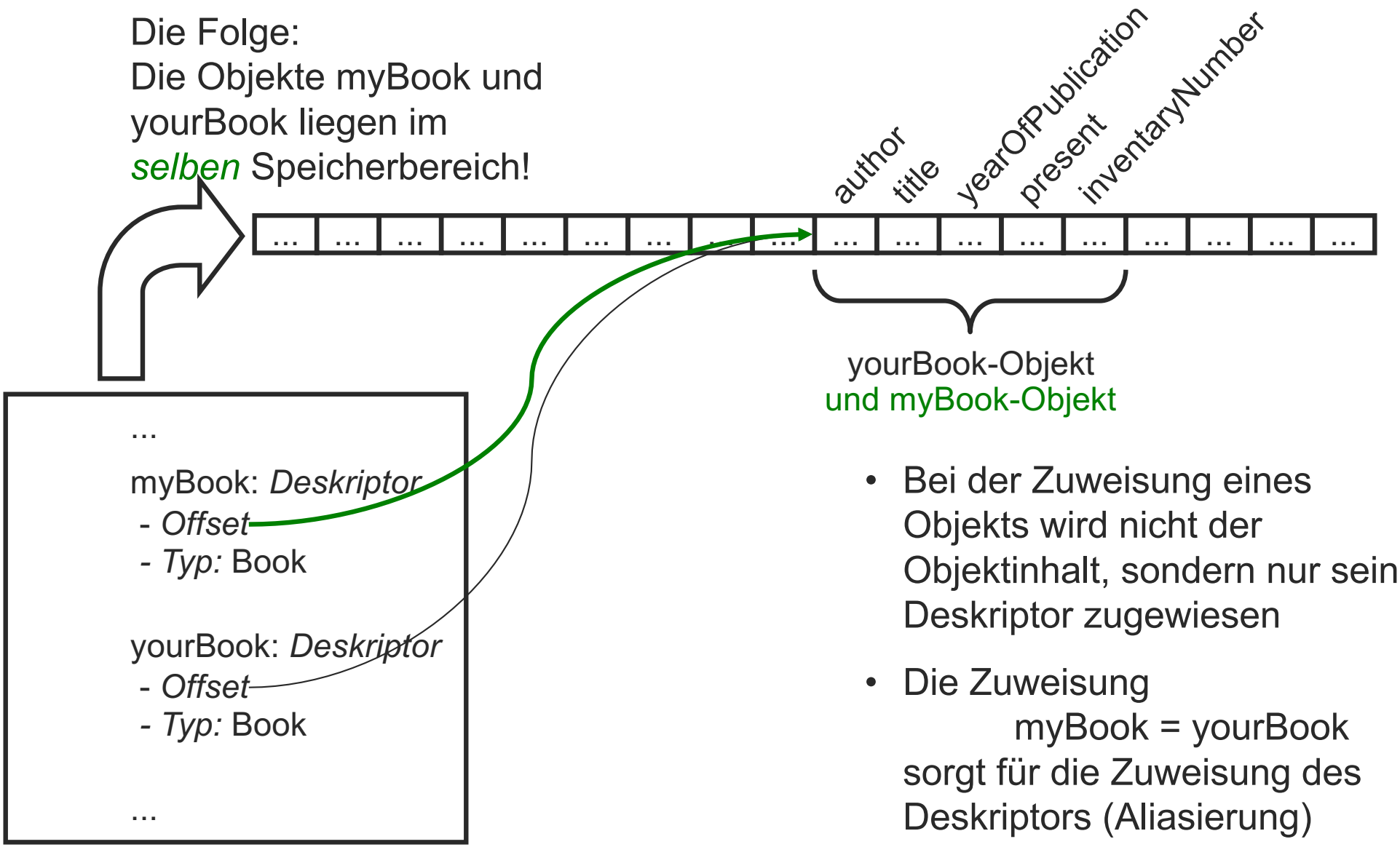
Zuweisung ganzer Objekte



- Bei der Zuweisung eines Objekts wird *nicht* der *Objektinhalt*, *sondern* nur sein *Deskriptor* zugewiesen
- Die Zuweisung
  myBook = yourBook
  sorgt für die Zuweisung des Deskriptors (Aliasierung)

Zuweisung ganzer Objekte

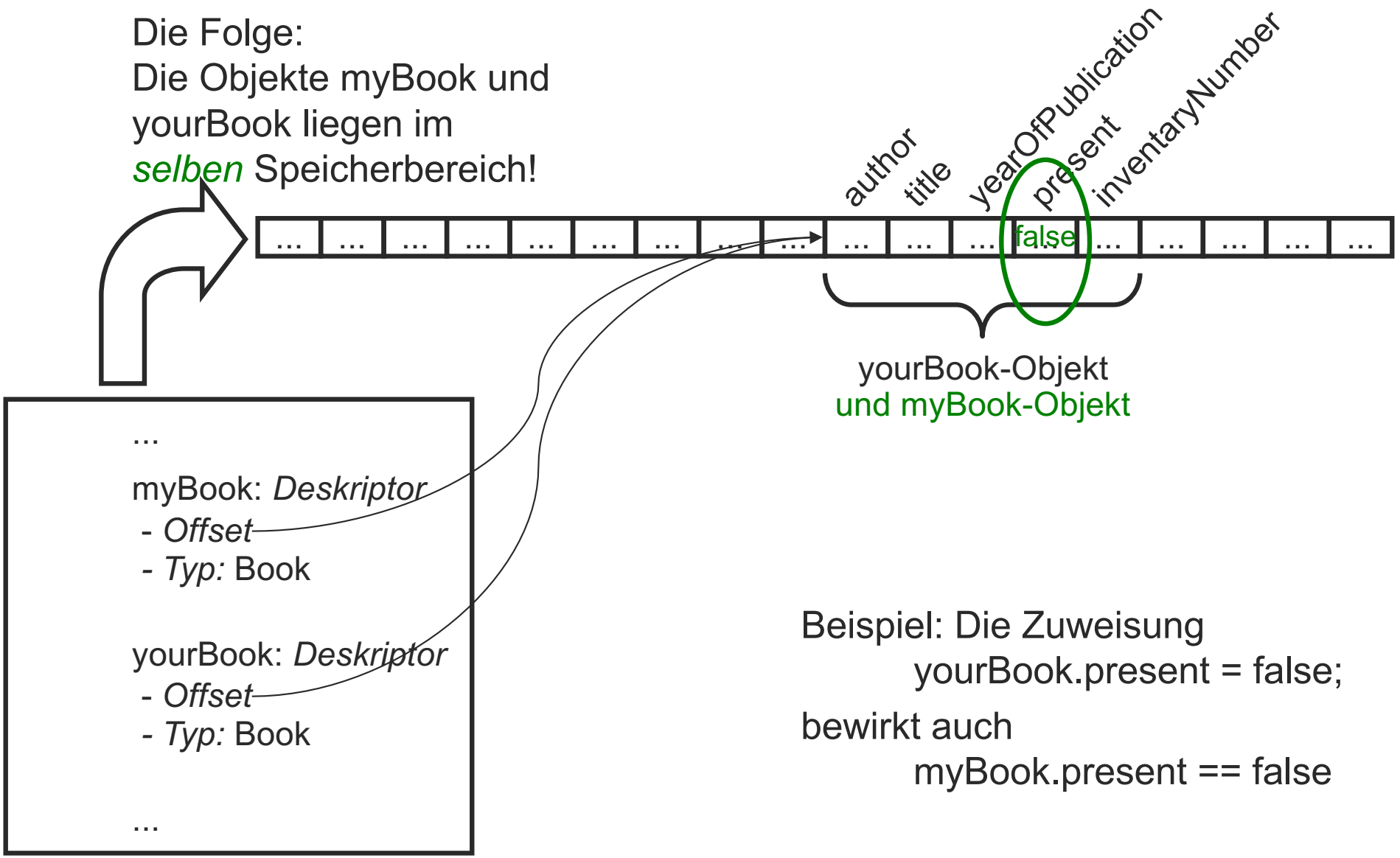
Die Folge:
Die Objekte myBook und yourBook liegen im *selben* Speicherbereich!



- Bei der Zuweisung eines Objekts wird nicht der Objektinhalt, sondern nur sein Deskriptor zugewiesen
- Die Zuweisung
  myBook = yourBook
  sorgt für die Zuweisung des Deskriptors (Aliasierung)

Zuweisung ganzer Objekte

Die Folge:
Die Objekte myBook und yourBook liegen im *selben* Speicherbereich!



Beispiel: Die Zuweisung

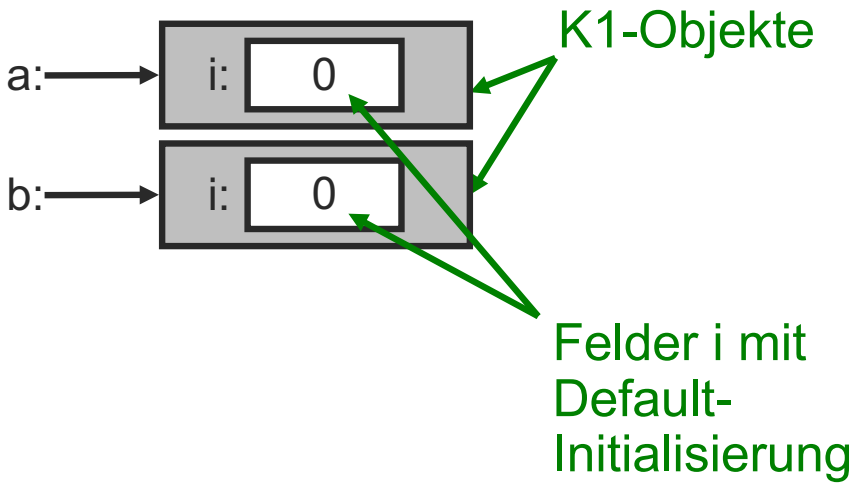
```
yourBook.present = false;
```

bewirkt auch

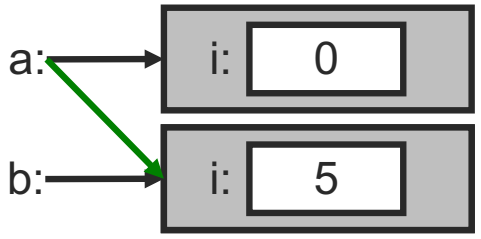
```
myBook.present == false
```

Zuweisung *ganzer* Objekte: schematisch dargestellt

```
class K1 { private int  i;

...
public static void main ( ... ) {

   K1       a = new K1(),
            b = new K1();
```

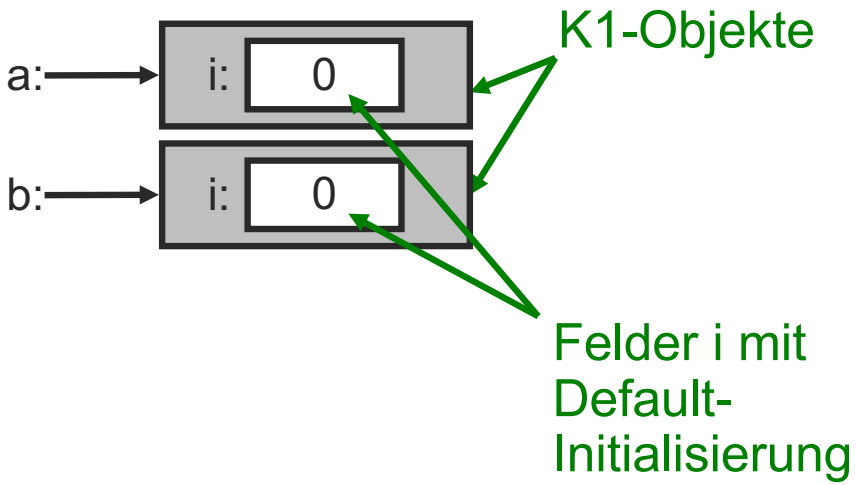



```
   b.i = 5;

   a = b;
```

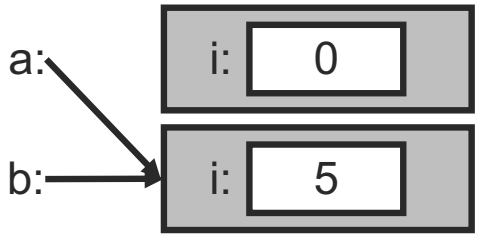


Zuweisung *ganzer* Objekte: schematisch dargestellt

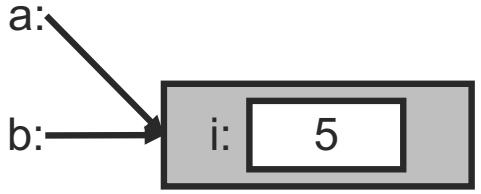
```
class K1 { private int  i;

...
public static void main ( ... ) {

   K1       a = new K1(),
            b = new K1();
```




```
   b.i = 5;

   a = b;
```

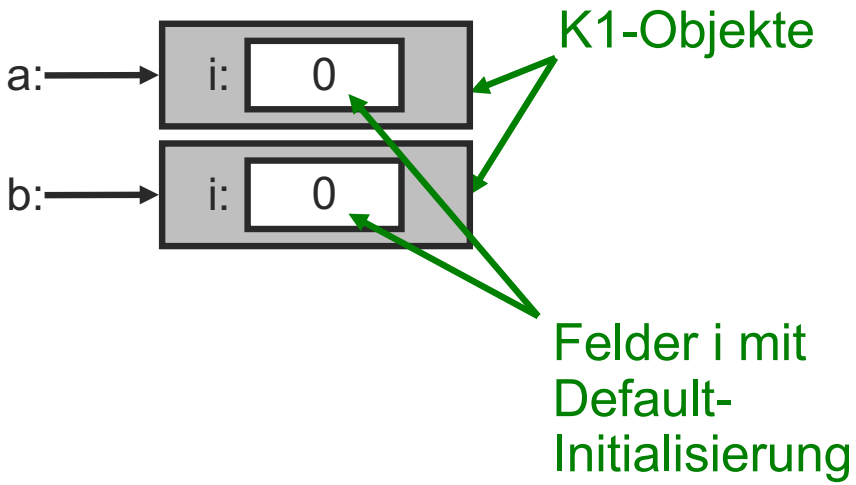



```
   b.i = 4;
```

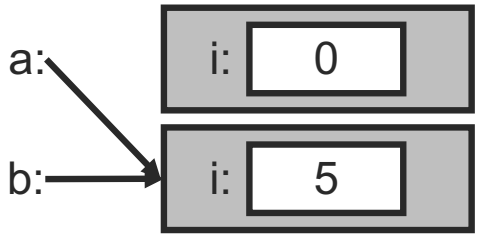


Zuweisung *ganzer* Objekte: schematisch dargestellt

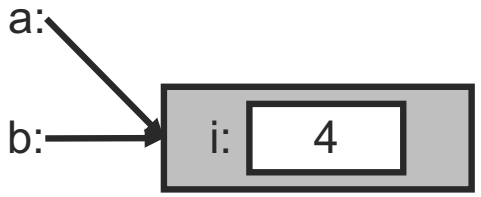
```
class K1 { private int  i;

...
public static void main ( ... ) {

   K1       a = new K1(),
            b = new K1();
```




```
   b.i = 5;

   a = b;
```

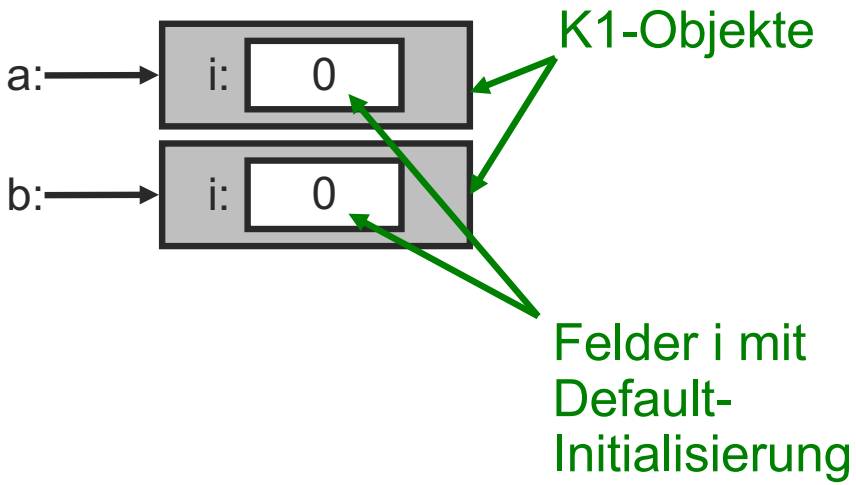



```
   b.i = 4;     Folge: a.i == 4

   a.i = 3;
```

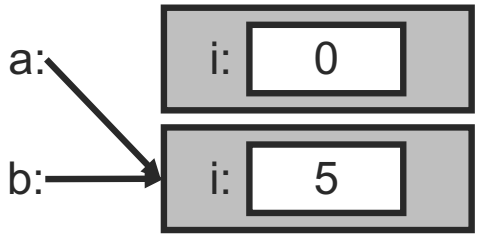


Zuweisung *ganzer* Objekte: schematisch dargestellt

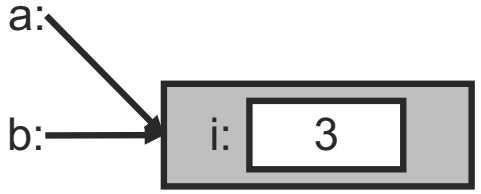
```
class K1 { private int  i;

...
public static void main ( ... ) {

   K1       a = new K1(),
            b = new K1();
```




```
   b.i = 5;

   a = b;
```

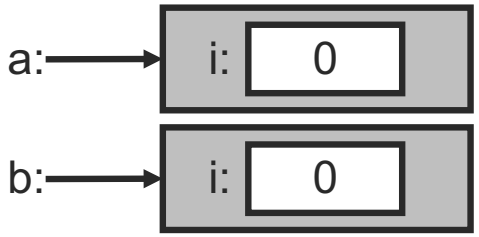



```
   b.i = 4;     Folge: a.i == 4

   a.i = 3;     Folge: b.i == 3
```

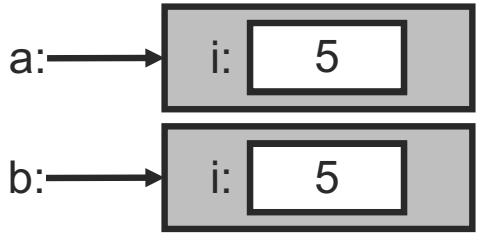


Zuweisung *einfacher* Werte *in* Objekten: schematisch dargestellt

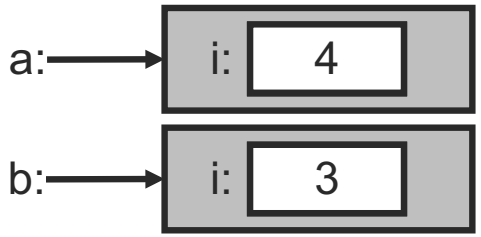
```
class K1 { private int  i;

...
public static void main ( ... ) {

  K1      a = new K1(),
          b = new K1();
```




```
  b.i = 5;

  a.i = b.i;
```




```
  a.i = 4;

  b.i = 3;
```



Semantik bei der Zuweisung von Objekten

- Wie für Arrays gilt auch für *Objekte* bei der Zuweisung: Zugewiesen werden die Deskriptoren der Objekte mit den *Referenzen* auf die eigentlichen Objekte (auf dem Heap)
- Referenzieren zwei (oder mehr) Objekte den gleichen Speicherbereich, spricht man von ***Aliasierung***
- Für Felder in einem Objekt, die einen *einfachen Typ* haben, werden direkt die Werte *kopiert*

Bekannte Objekte: Arrays

- Wir wissen, dass Objekte und Arrays intern von der VM durch Deskriptoren dargestellt werden
- Tatsächlich *sind Arrays* in Java (*spezielle*) *Objekte*

  Sie bieten nur eine besondere Schreibweise, nämlich die Indizierung mittels eckiger Klammern, um auf ihre Elemente zuzugreifen (Bequemlichkeit)

Bekannte Objekte: Strings

- Auch *Strings sind* in Java (*spezielle*) *Objekte*, sie werden ebenfalls durch Deskriptoren dargestellt
- Auch sie bieten eine besondere Schreibweise, nämlich die Notation in doppelten Hochkommata

- Strings und Arrays sind in Java nur deshalb „dynamischer“ als in den meisten anderen Programmiersprachen, weil sie als Objekte realisiert sind, also auf dem Heap gespeichert werden
- Hier unterscheidet sich Java von vielen anderen Programmiersprachen

Geschachtelte Objekte

Achtung:

- Objekte können *Felder* enthalten, die einen *nicht-einfachen Typ* haben, also zum Beispiel ein Array oder Objekte einer (anderen oder derselben) Klasse

  Man spricht dann von ***geschachtelten Objekten***

- Ebenso können Arrays als *Elementtyp* einen *nicht-einfachen Typ* haben, also ein Objekt oder ein weiteres Array

- Für solche geschachtelten Strukturen gelten bei der Zuweisung die gleichen Regeln wie für nicht-geschachtelte Objekte:
  Es werden die Deskriptoren kopiert, nicht die Inhalte

Beispiel für ein geschachteltes Objekt 1/2

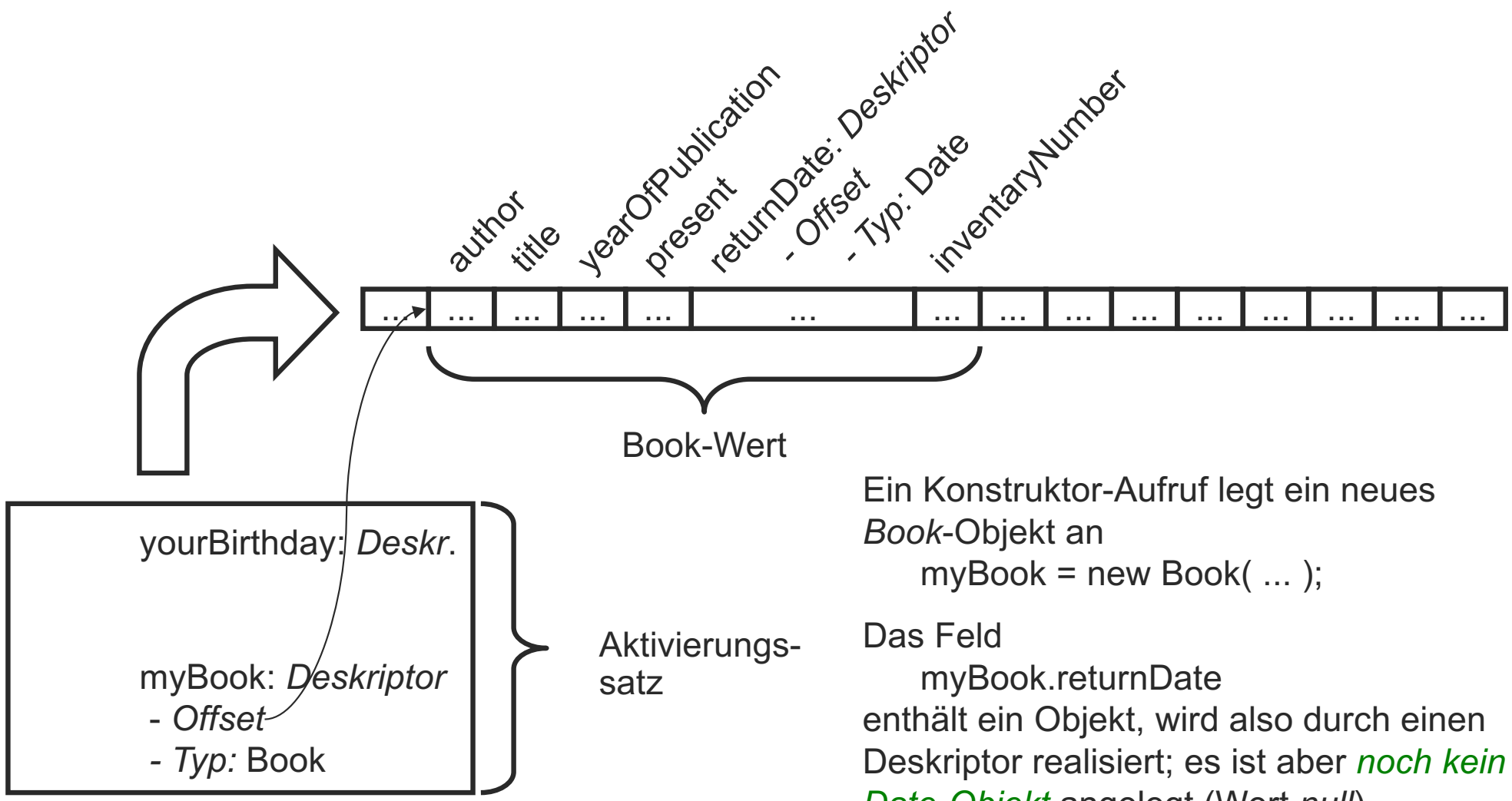
- Wir können der Buch-Klasse ein weiteres Feld hinzufügen für das Datum, an dem es wieder in die Bibliothek zurückgebracht werden soll, falls es gerade ausgeliehen ist
- Wir benutzen die bereits bekannte Deklaration für eine Datumsklasse

```
class Date {
    int  day;
    int  month;
    int  year;
}
```

- Die erweiterte Buch-Klasse sieht dann wie folgt aus

```
class Book {
    private   String    author;
    private   String    title;
    private   int       yearOfPublication;
    private   boolean   present;
    private   Date      returnDate;
    private   int       inventaryNumber;
}
```

Beispiel für ein geschachteltes Objekt 2/2



Ein Konstruktor-Aufruf legt ein neues *Book*-Objekt an

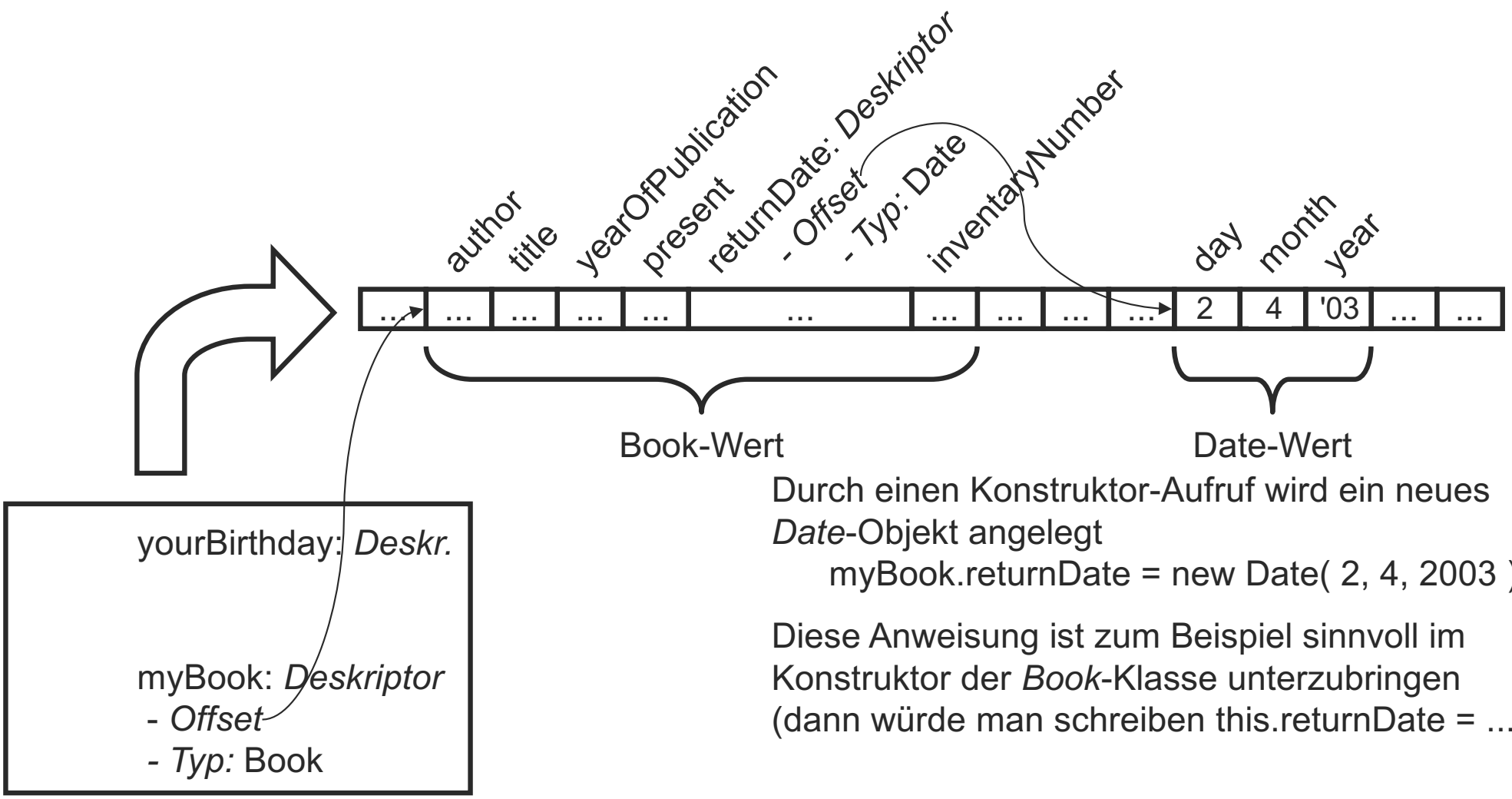
```
myBook = new Book( ... );
```

Das Feld

```
myBook.returnDate
```

enthält ein Objekt, wird also durch einen Deskriptor realisiert; es ist aber *noch kein Date-Objekt* angelegt (Wert *null*)

Beispiel für ein geschachteltes Objekt 2/2

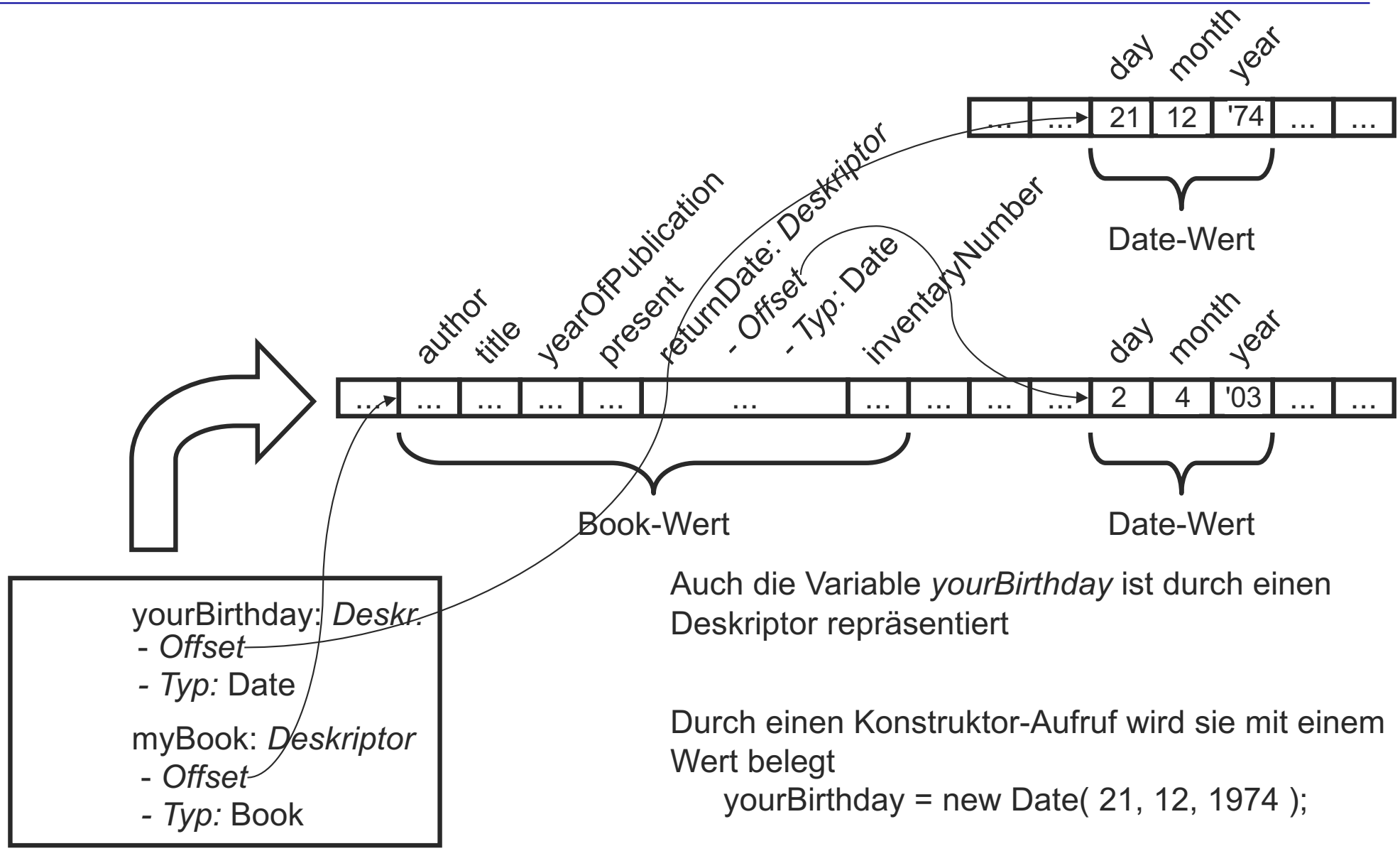


Durch einen Konstruktor-Aufruf wird ein neues *Date*-Objekt angelegt

myBook.returnDate = new Date(2, 4, 2003);

Diese Anweisung ist zum Beispiel sinnvoll im Konstruktor der *Book*-Klasse unterzubringen (dann würde man schreiben this.returnDate = ...)

Beispiel für ein geschachteltes Objekt 2/2

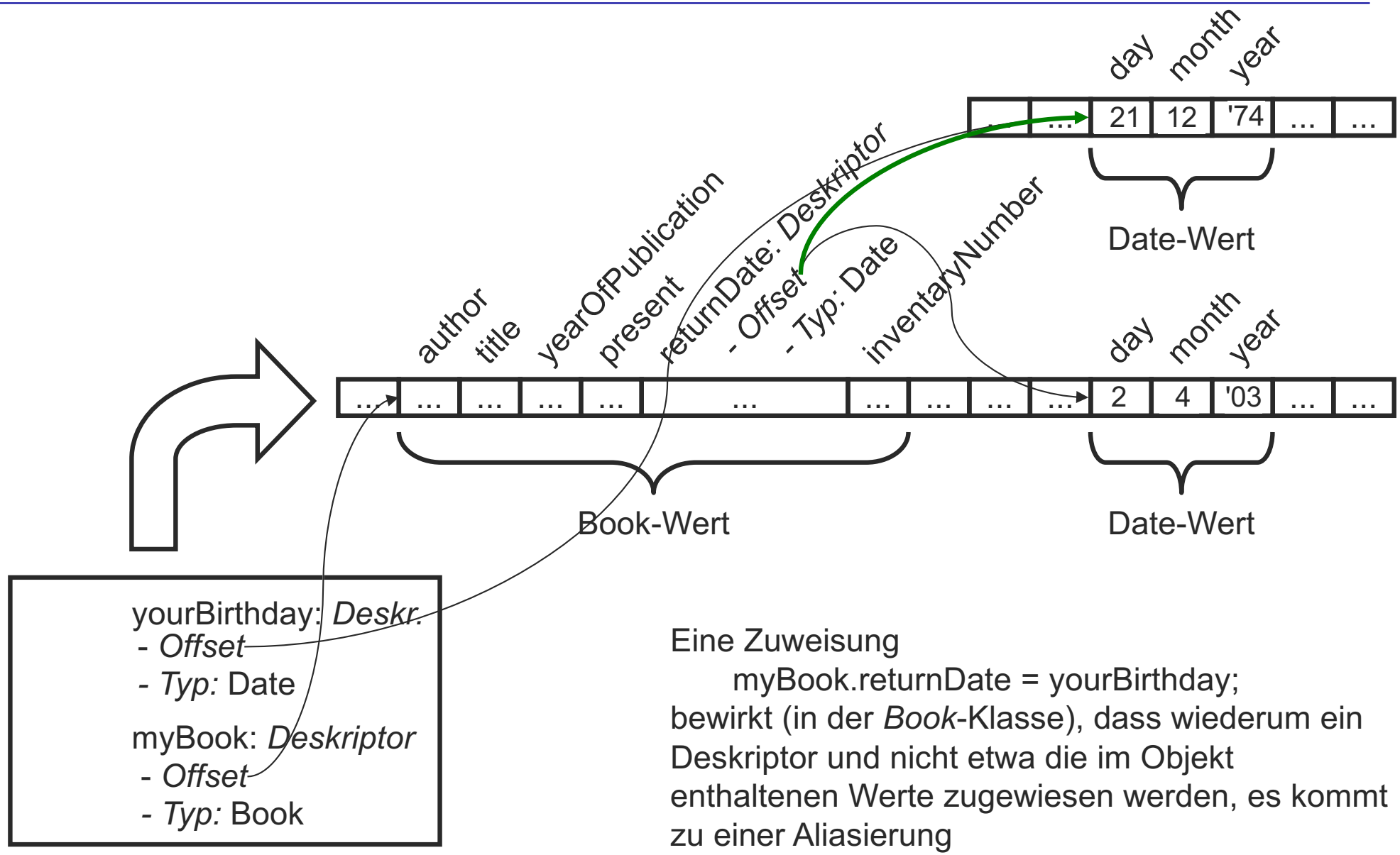


Auch die Variable *yourBirthday* ist durch einen Deskriptor repräsentiert

Durch einen Konstruktor-Aufruf wird sie mit einem Wert belegt

```
yourBirthday = new Date( 21, 12, 1974 );
```

Beispiel für ein geschachteltes Objekt 2/2

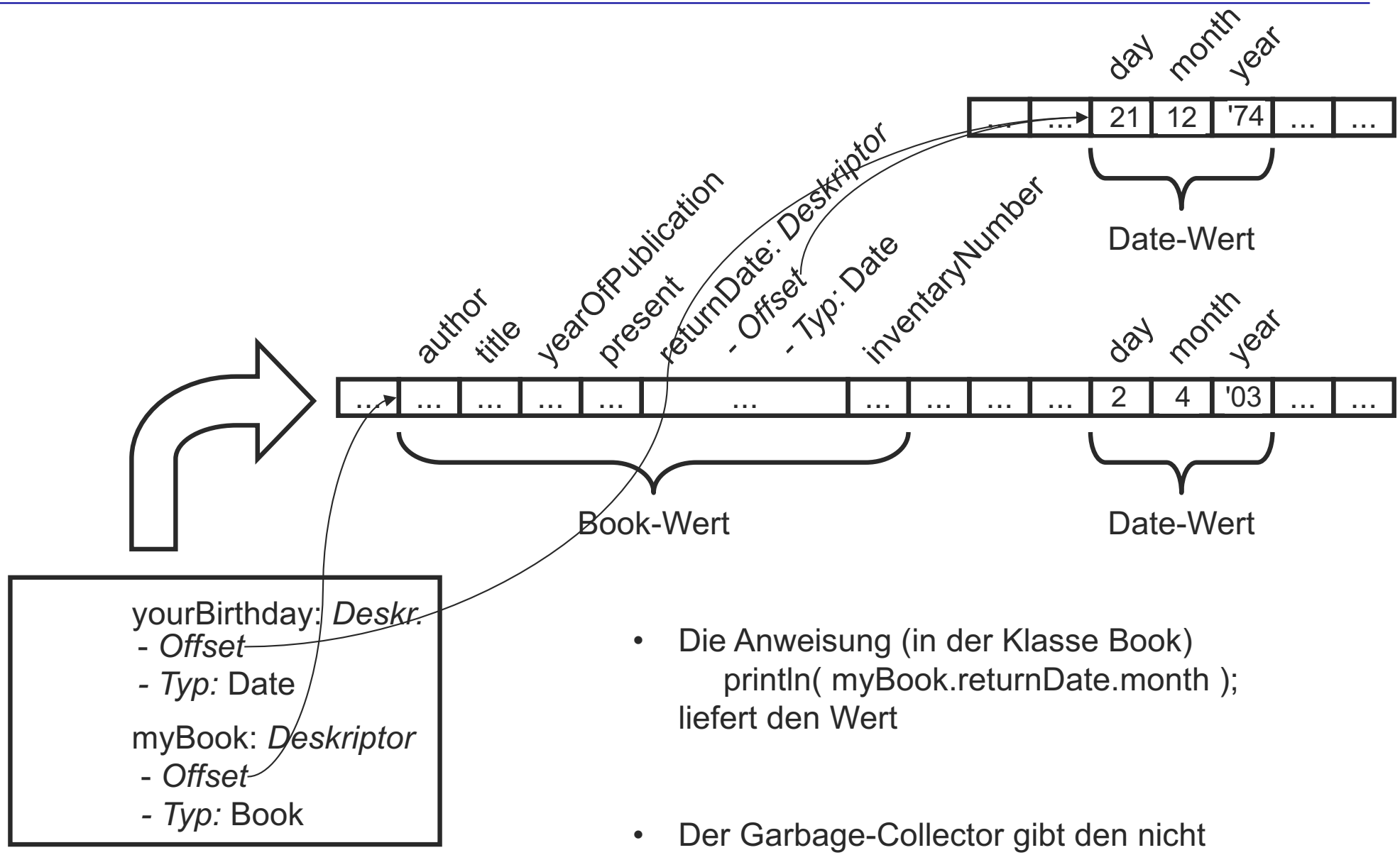


Eine Zuweisung

myBook.returnDate = yourBirthday;

bewirkt (in der *Book*-Klasse), dass wiederum ein Deskriptor und nicht etwa die im Objekt enthaltenen Werte zugewiesen werden, es kommt zu einer Aliasierung

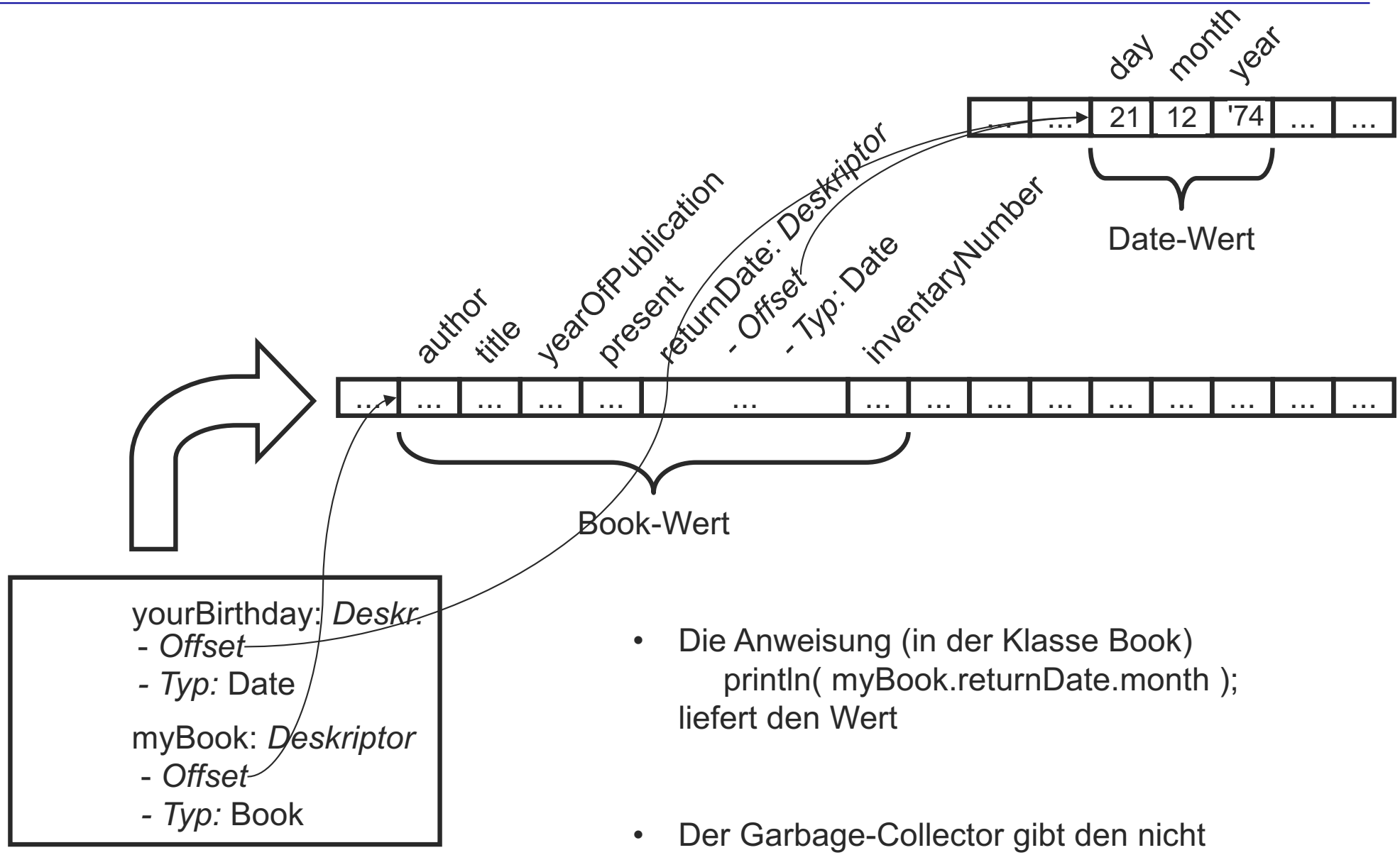
Beispiel für ein geschachteltes Objekt 2/2



- Die Anweisung (in der Klasse Book)
  println(myBook.returnDate.month);
  liefert den Wert

- Der Garbage-Collector gibt den nicht länger benötigten Bereich wieder frei

Beispiel für ein geschachteltes Objekt 2/2



- Die Anweisung (in der Klasse Book)
  println(myBook.returnDate.month);
  liefert den Wert

- Der Garbage-Collector gibt den nicht länger benötigten Bereich wieder frei

Bemerkung 1/2

- Es ist eventuell sinnvoll, das Feld *returnDate* im *Book*-Objekt bereits im Konstruktor mit einem (welchem?) Wert zu belegen

```
Book ( String author, String title, int year, int number )
{
    this.author             = author;
    this.title              = title;
    yearOfPublication       = year;
    inventaryNumber         = number;
    returnDate              = new Date( ..., ..., ... );
}
```

- So wird bei jeder Instanziierung eines *Book*-Objekts gleichzeitig ein *Date*-Objekt instanziiert

Bemerkung 2/2

- Man kann die Parameter für das *Date*-Objekt auch dem *Book*-Konstruktor als Parameter übergeben

```
Book ( String author, String title, int publicationYear, int number,
                int returnDay, int returnMonth, int returnYear )
{
    this.author             = author;
    this.title              = title;
    yearOfPublication       = publicationYear;
    inventaryNumber         = number;
    returnDate              = new Date( returnDay, returnMonth, returnYear );
}
```