

Fakultät Informatik  
Professur Softwaretechnologie

## OOSE 01

# JAVA MIT BLUEJ UND UML-BY-EXAMPLE

Dr.-Ing. Birgit Demuth  
Sommersemester 2018

## Arbeit mit BlueJ (version 3.7.1) – Demo (1)

The screenshot displays the BlueJ IDE interface for a project named 'OOSE1\_HelloLibrary'. The main workspace shows a UML class diagram with three classes: 'HelloLibrary', 'Library', and 'Book'. 'HelloLibrary' is connected to 'Library' and 'Book' by dashed lines, indicating associations. The 'Library' class is associated with 'Book'.

In the foreground, the 'HelloLibrary' class is open, showing the following source code:

```
public class HelloLibrary {  
    public static void main(String[] args) {  
        Library myLib = new Library();  
  
        Book b1 = new Book("UML");  
        Book b2 = new Book("Java7");  
  
        myLib.register(b1);  
        myLib.register(b2);  
    }  
}
```

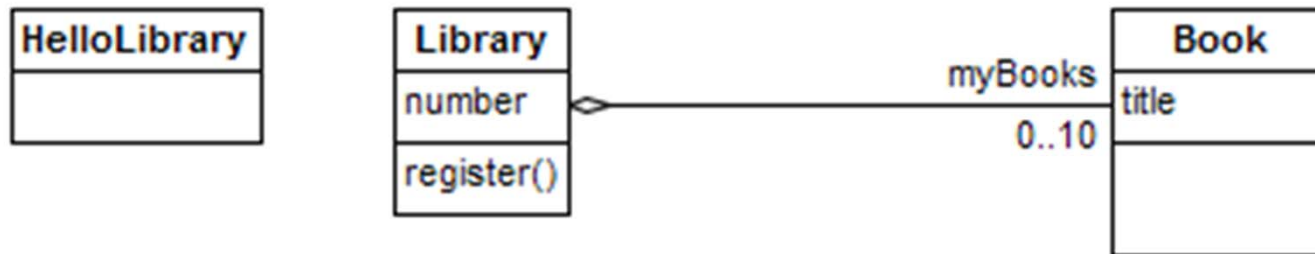
A terminal window titled 'BlueJ: Terminal Window - OOSE1\_HelloLibra...' is open, displaying the output of the program:

```
Options  
Hello, I am a small library for at most 10 books.  
A new book is registered: UML  
A new book is registered: Java7
```

## Unser erstes UML-Analyseklassendiagramm (HelloLibrary Anwendung)



## Klassen und Objekte

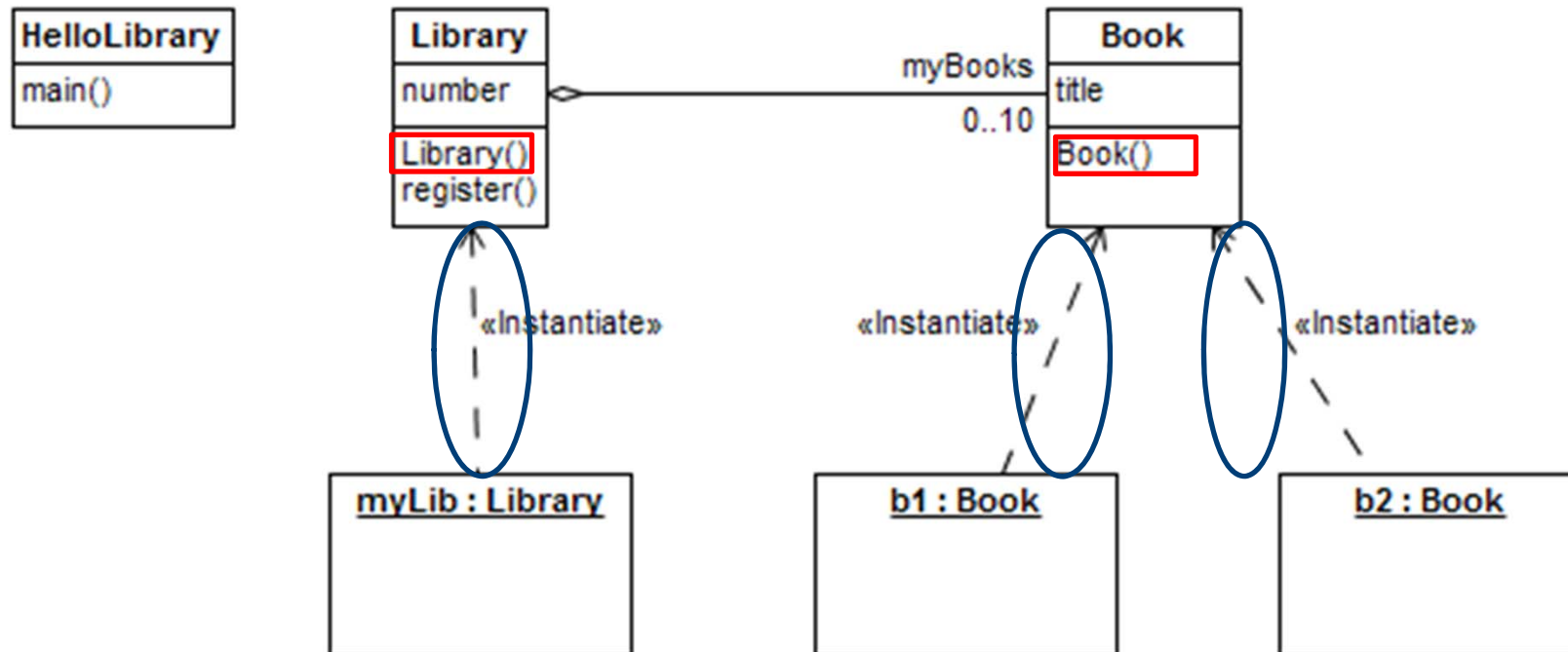


myLib : Library

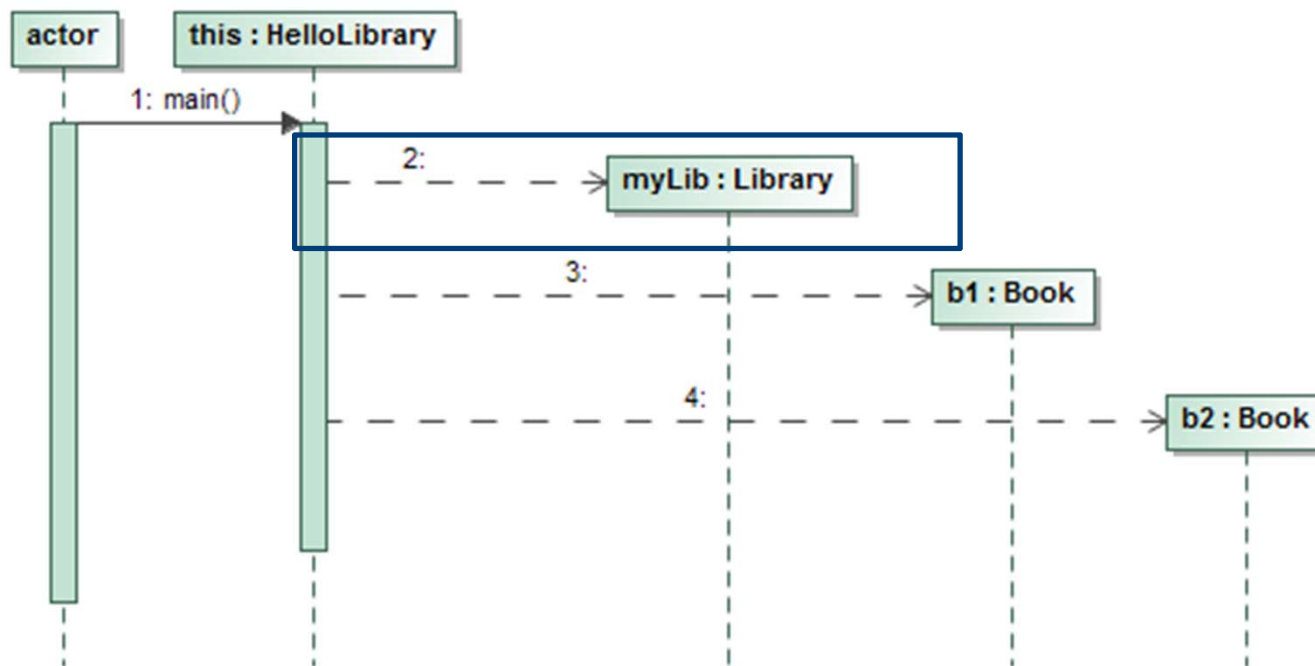
b1 : Book

b2 : Book

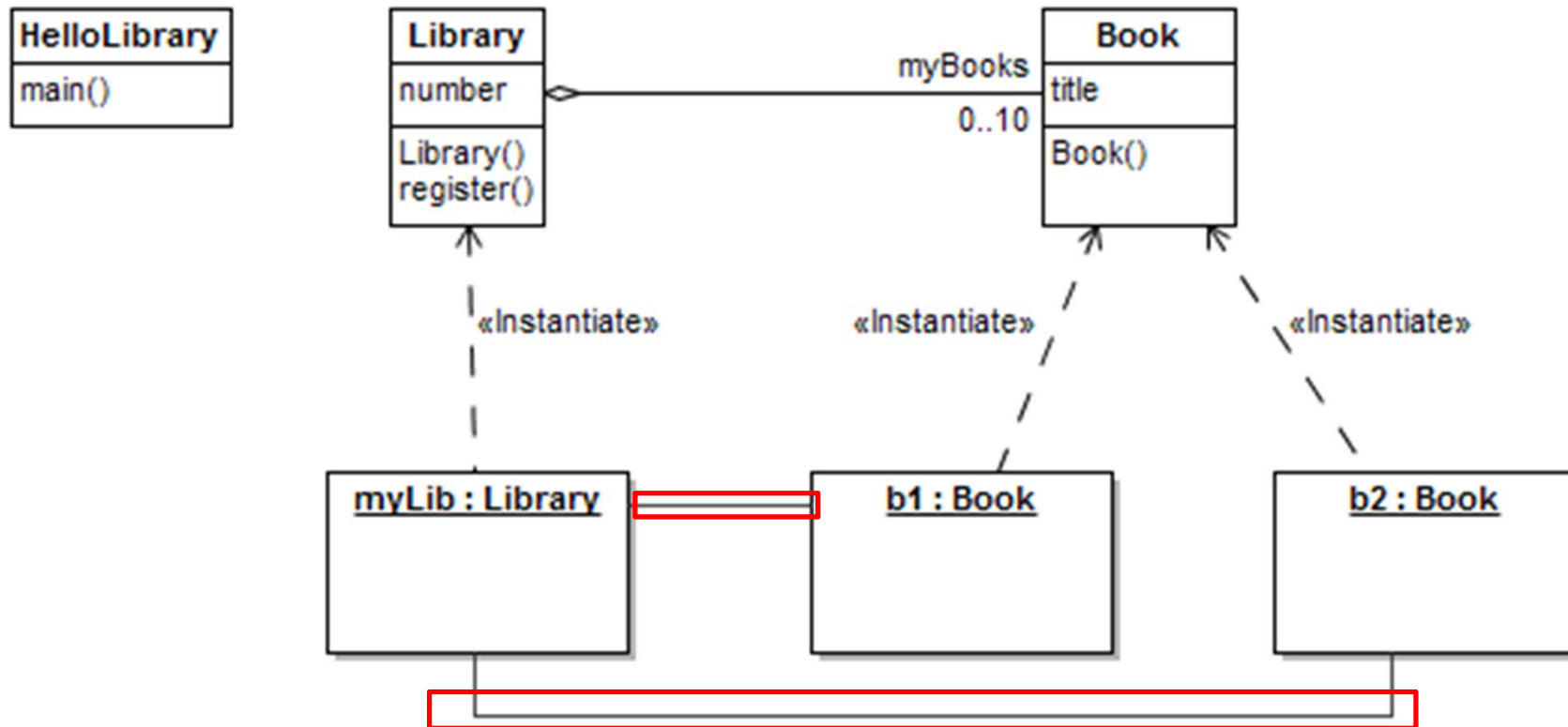
# Konstruktor und Instanziierung von Objekten



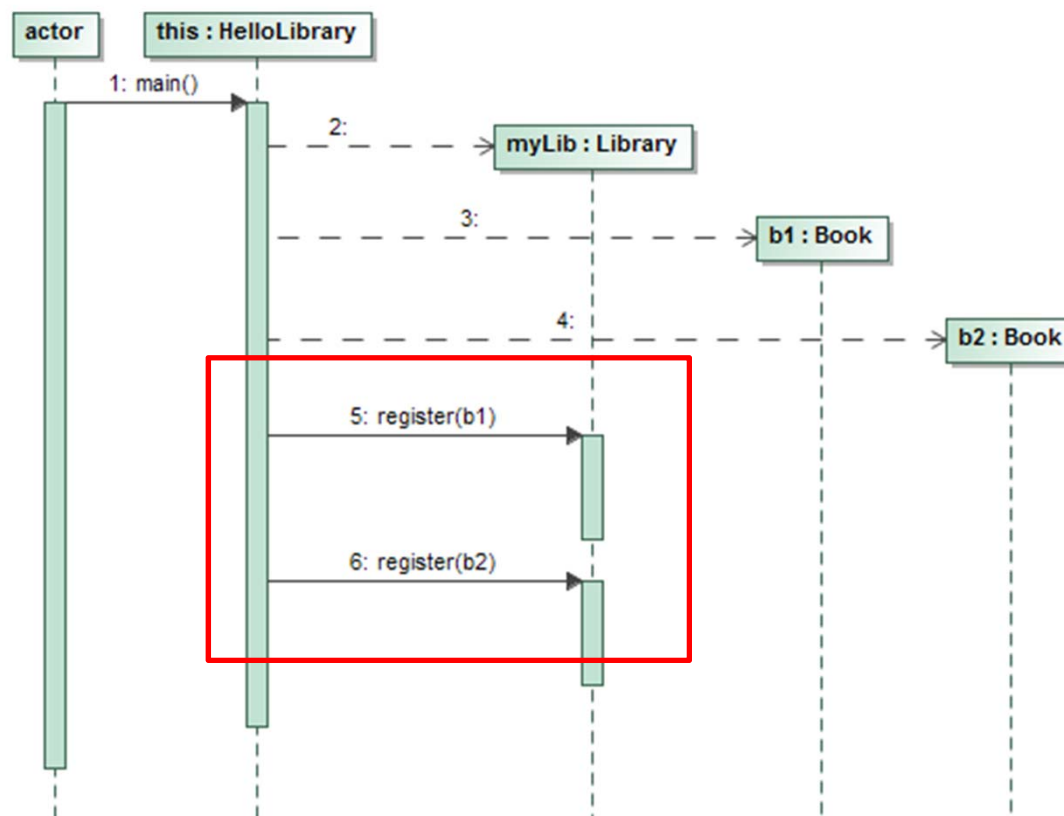
# Instanziierung von Objekten Darstellung im Sequenzdiagramm



## Registrierung von Objekten/Darstellung im Objektdiagramm (Erzeugung von **Links** durch Aufruf der Methode register())

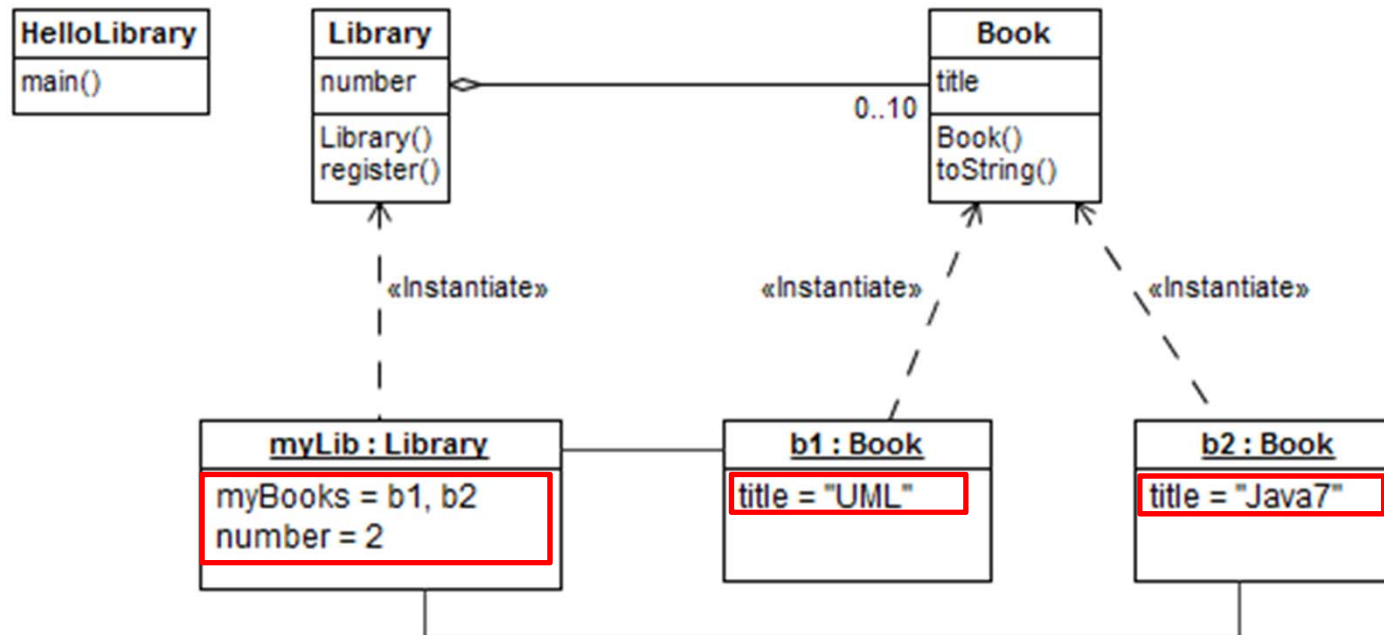


## Registrierung von Objekten/ Darstellung im Sequenzdiagramm

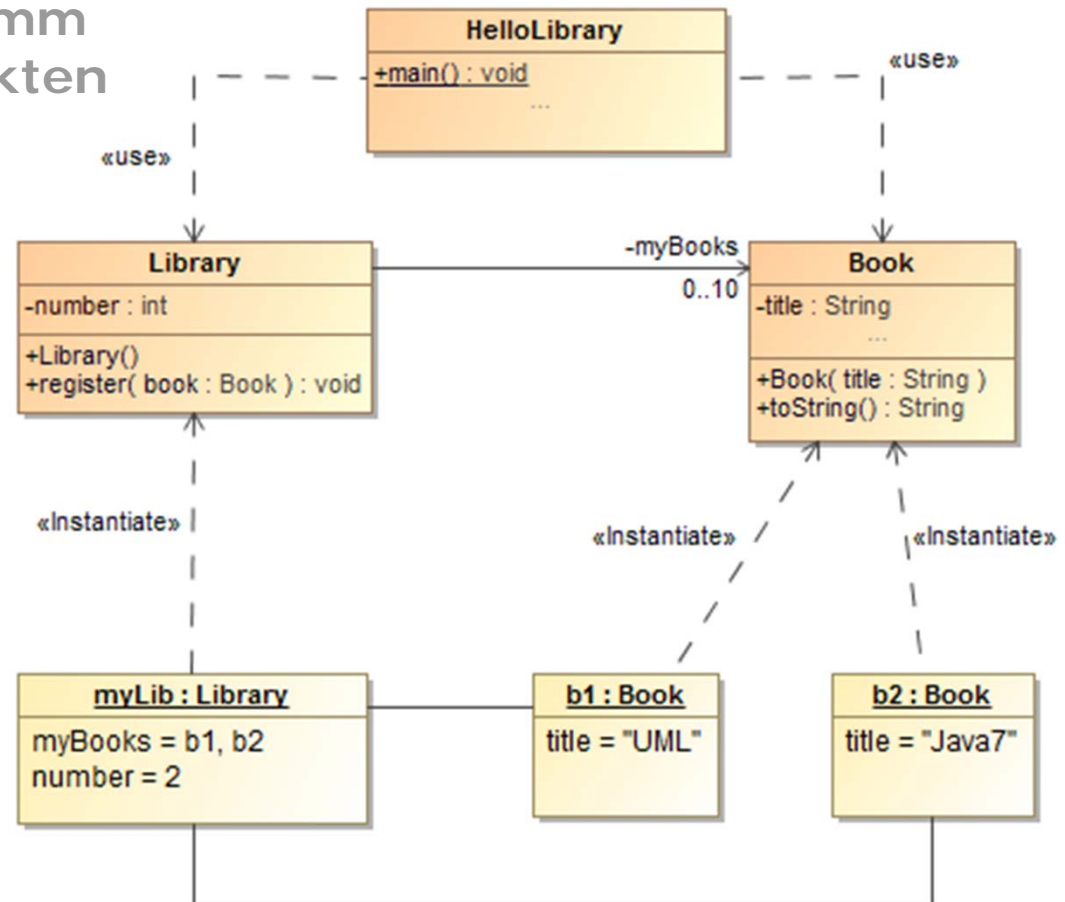




## Objektdiagramm mit Slots (Attribut = Wert)

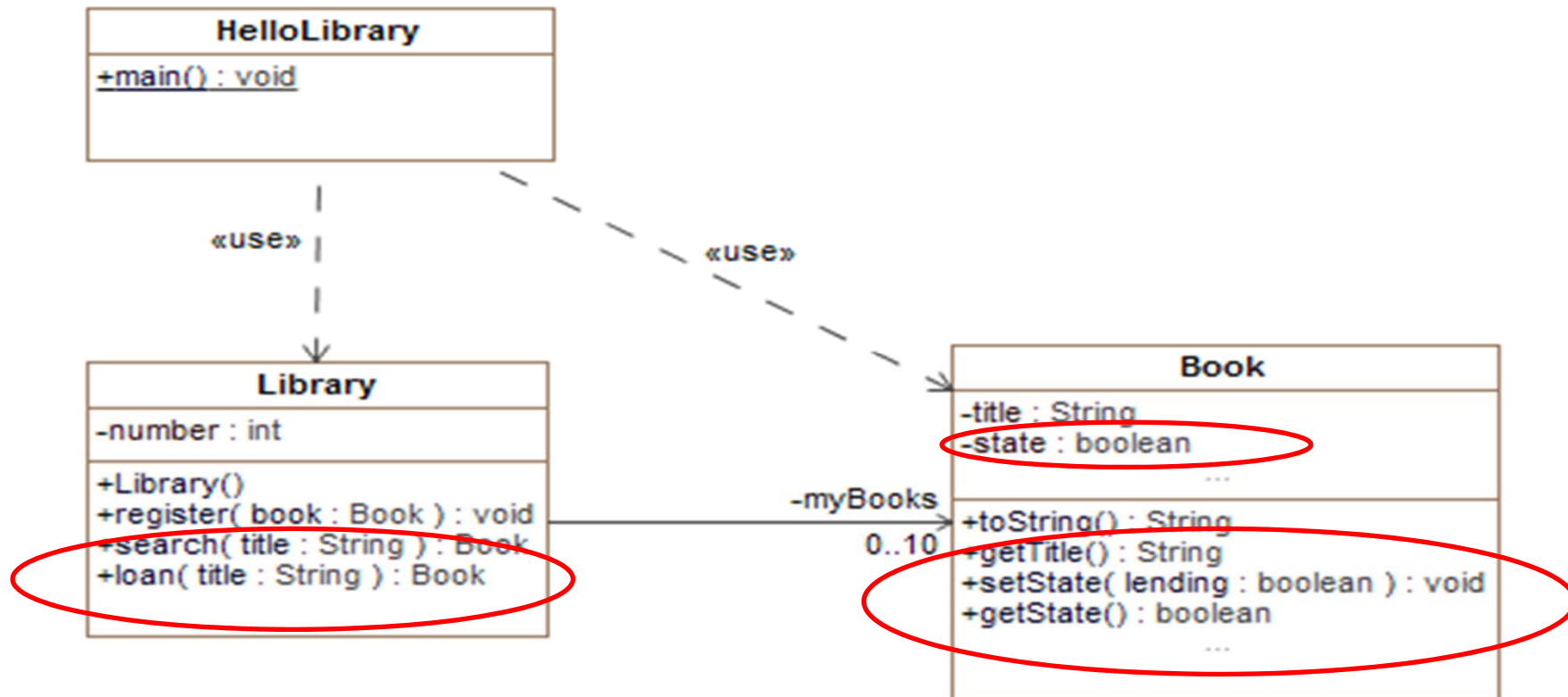


## Entwurfsklassendiagramm mit instanziierten Objekten

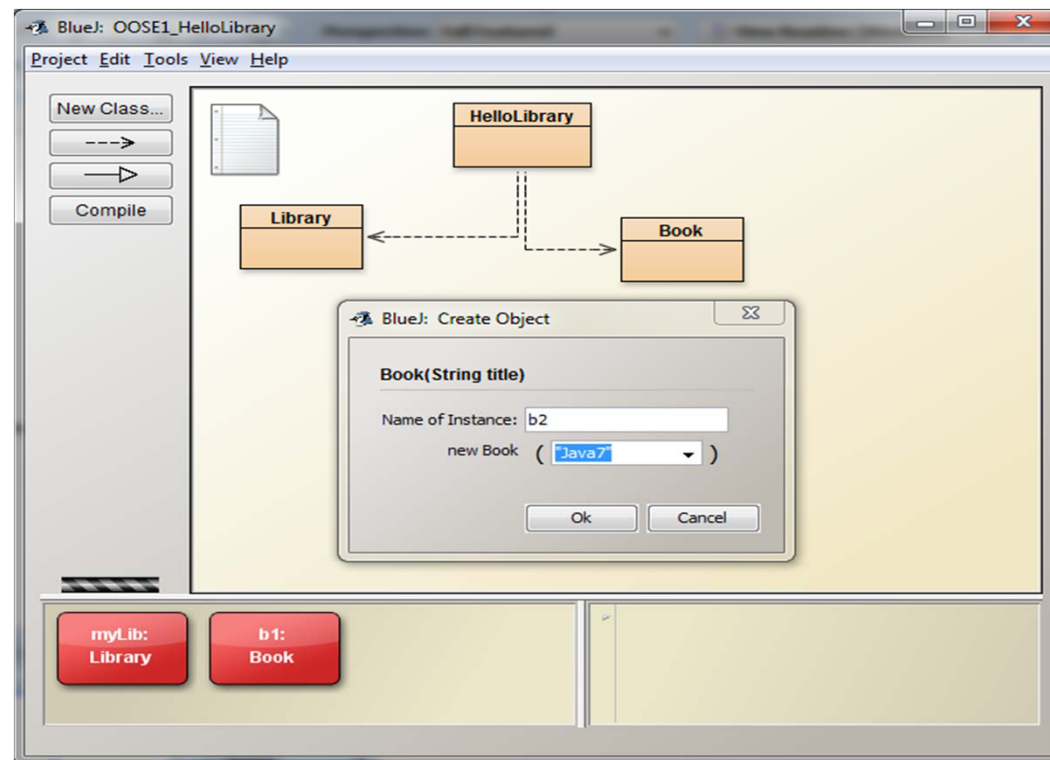


# Entwurfsklassendiagramm

## HelloLibrary extended (Vorschau Übung U02)

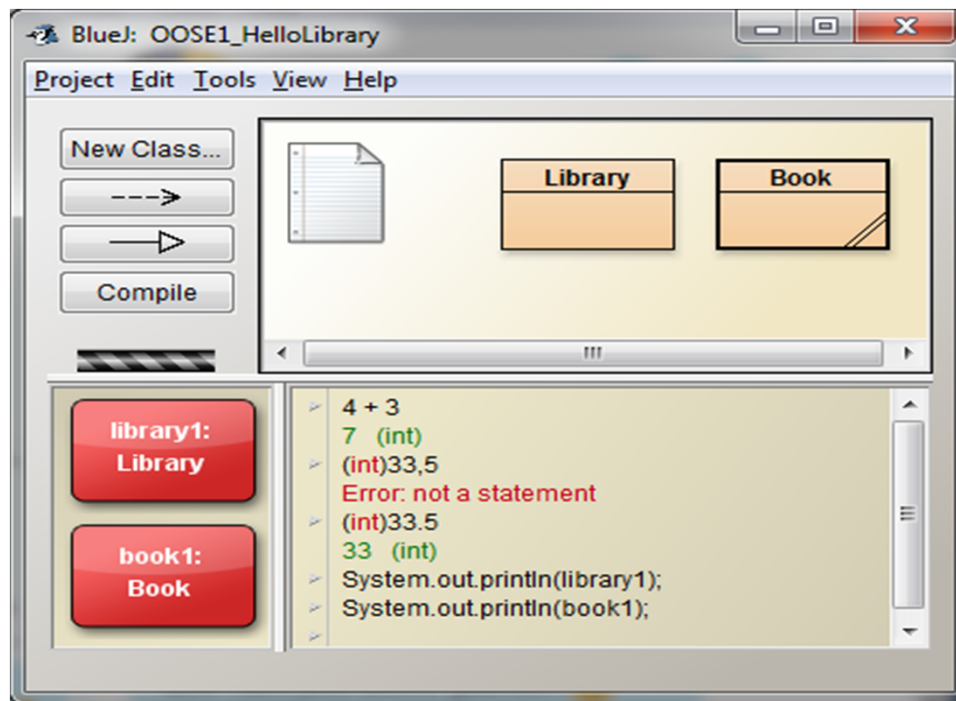


## Arbeit mit BlueJ (version 3.7.1) – Demo (Object Bench)



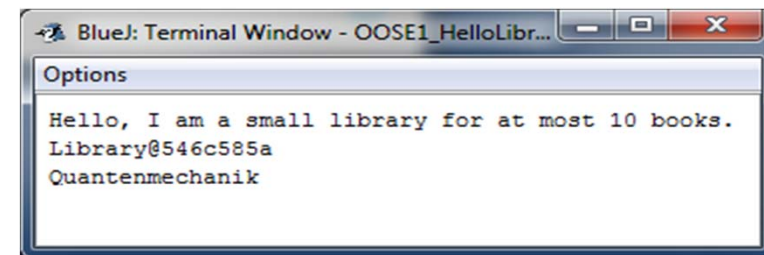
Object Bench:  
„Spielen“ mit Objekten

## Arbeit mit BlueJ (version 3.7.1) – Demo (Code Pad)



### Code Pad:

- Berechnung von Java-Ausdrücken
- Ausführen von Java-Statements
- Eingabe von mehrzeiligen Anweisungen (durch Drücken von Shift Enter) und Ausführen (Enter)



# Debugger

The image shows a sequence of three screenshots from the BlueJ IDE illustrating the debugging process:

- 1. Set a breakpoint:** The first screenshot shows the 'demo1: Demo' class in the project view. A red arrow labeled '1' points to the 'demo1: Demo' button, with the text 'Setzen eines Breakpoints' (Setting a breakpoint) below it.
- 2. Method Call:** The second screenshot shows the 'Method Call' dialog box. It displays the method signature 'int loop(int count)' and the call 'demo1.loop ( 4 )'. A red arrow labeled '2' points from the 'demo1: Demo' button to this dialog.
- 3. Debugger:** The third screenshot shows the 'BlueJ: Debugger' window. The 'Threads' panel shows 'main (at breakpoint)'. The 'Call Sequence' panel shows 'Demo.loop'. The 'Instance variables' panel shows 'private String name = "Marvin"' and 'private int answer = 42'. The 'Local variables' panel shows 'int count = 4', 'int sum = 17', and 'int i = 0'. A red arrow labeled '3' points from the 'Method Call' dialog to the debugger window.

The source code in the second screenshot is as follows:

```
private String name;
private int answer;

/**
 * Constructor for objects of class Demo
 */
public Demo()
{
    name = "Marvin";
    answer = 42;
}

/**
 * Loop for a while and do some meaningless computations.
 */
public int loop(int count)
{
    int sum = 17;
    for (int i=0; i<count; i++) {
        sum = sum + 1;
        sum = sum - 2;
    }
    return sum;
}

/**
 * Method for demonstrating single stepping with nested method call.
 */
public int carTest()
{
    int places;
    Car myCar = new Car(2, 3);
}
```

Ich höre und ich vergesse.  
Ich sehe und ich erinnere mich.  
Ich handle und ich verstehe.

*Xun Zi (chinesischer Philosoph)*

*Vor mehr als 2000 Jahren*