TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Fakultät Informatik

Professur Softwaretechnologie

OOSE_02

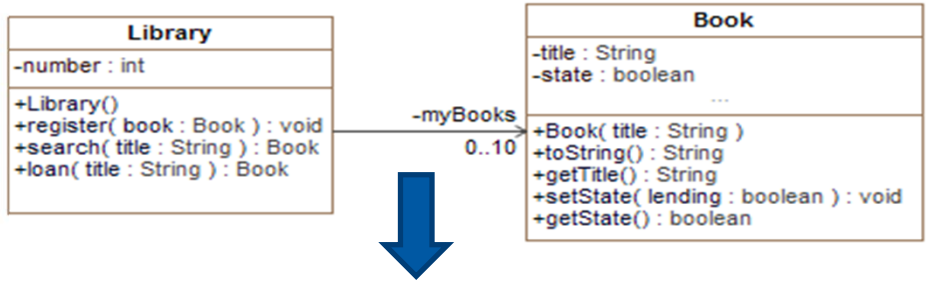
VERERBUNG UND POLYMORPHIE MIT BLUEJ

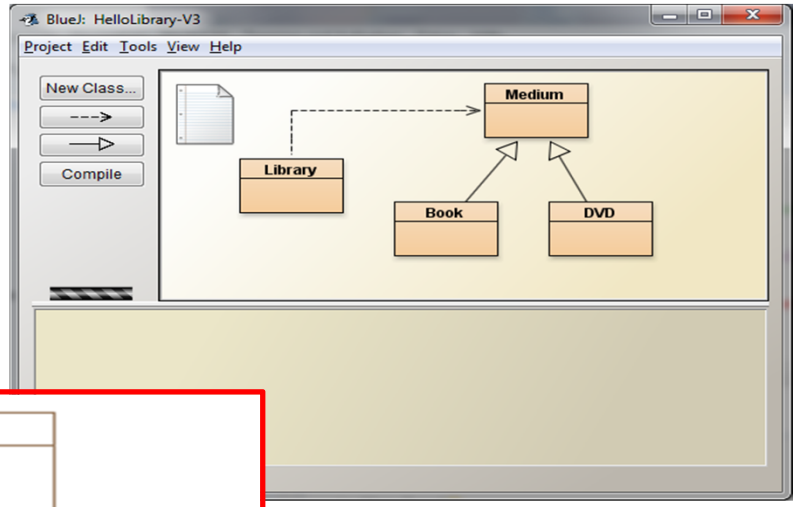
Dr.-Ing. Birgit Demuth
Sommersemester 2018

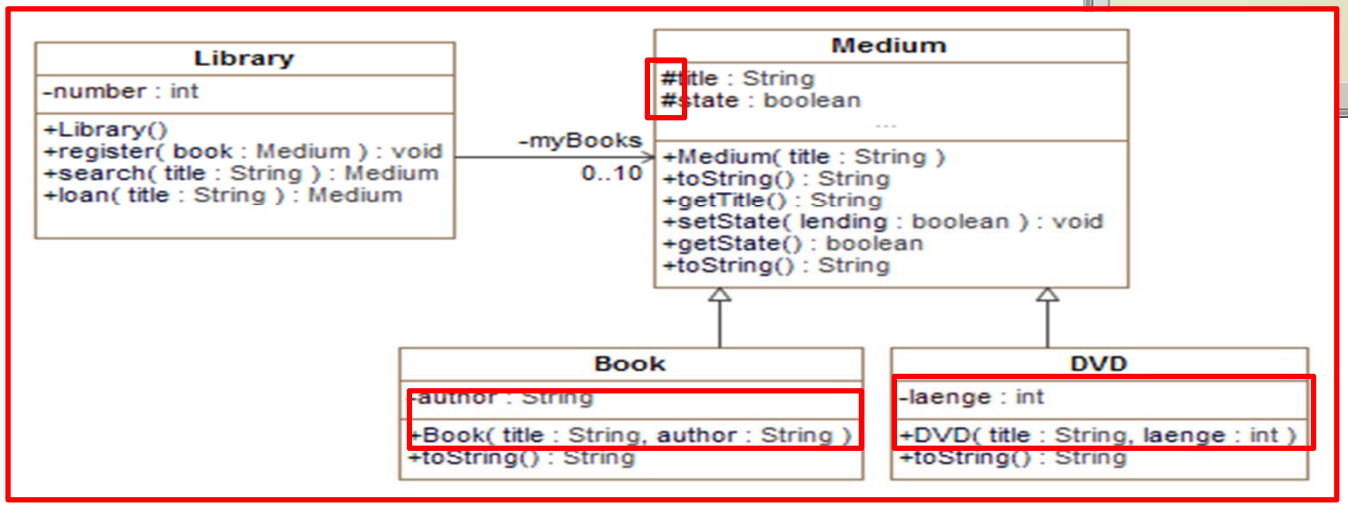
Vererbung und Polymorphie mit BlueJ

[Barnes2012]

BlueJ: Vererbung (1)
Erweiterung von HelloLibrary (U02)



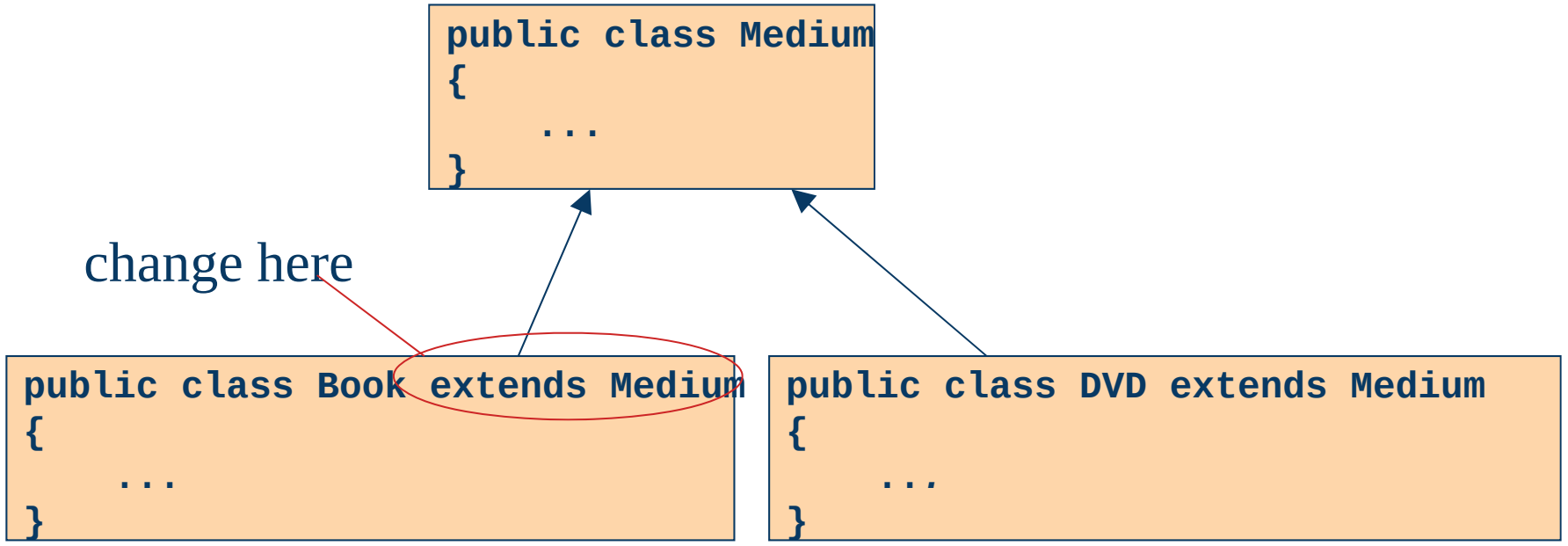




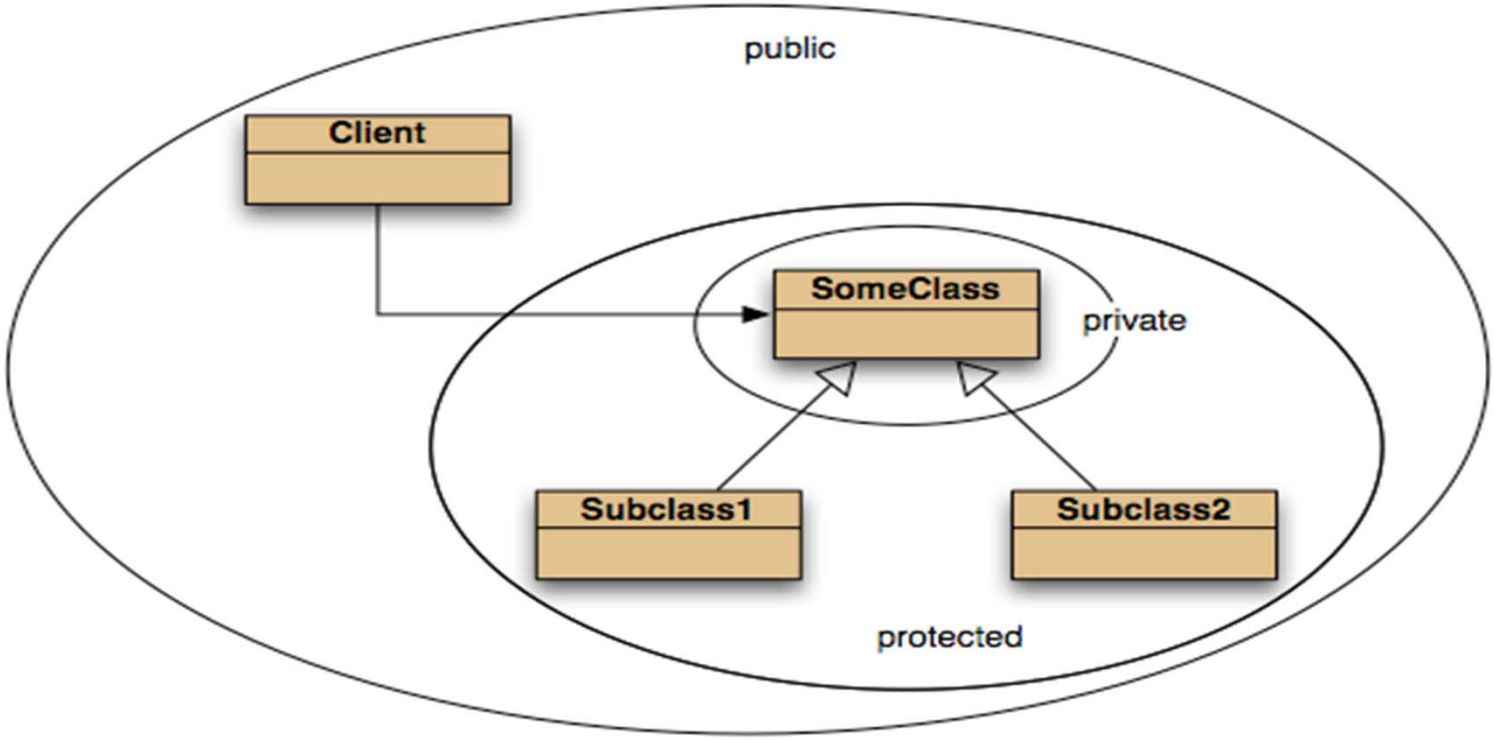
Using inheritance (Wiederholung)

define one **superclass** : **Medium**
define **subclasses** for **Book** and **DVD**
the superclass defines common attributes **title** and **state**
the subclasses **inherit** the superclass attributes **title** and **state**
the subclasses add own attributes **autor** bzw. **laenge**

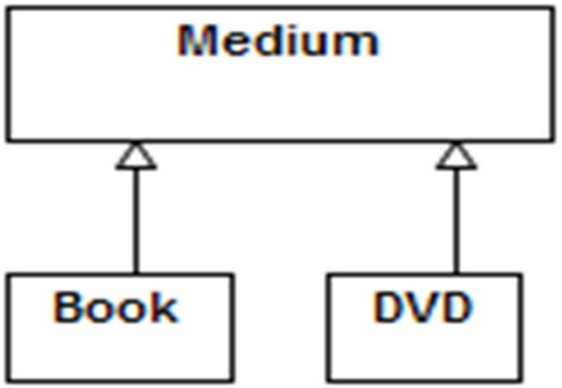
Inheritance in Java



Access levels



Subtyping and assignment



subclass objects may be assigned to superclass variables

```
Medium m1 = new Medium(…);
Medium b1 = new Book(…);
Medium d1 = new DVD(…);
```

Static and dynamic type

What is the type of b1?

```
Book b1 = new Book(…);
```

What is the type of m1?

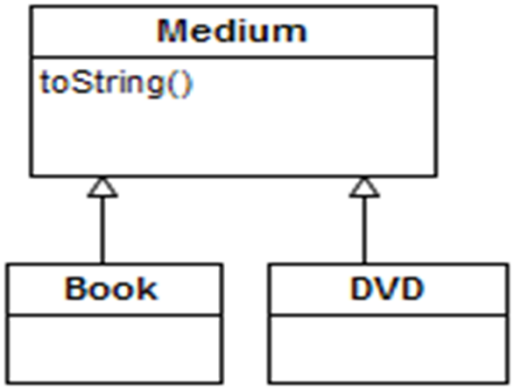
```
Medium m1 = new Book(…);
```

- The declared type of a variable is its **static type**.
- The type of the object a variable refers to is its **dynamic type**.

The Problem

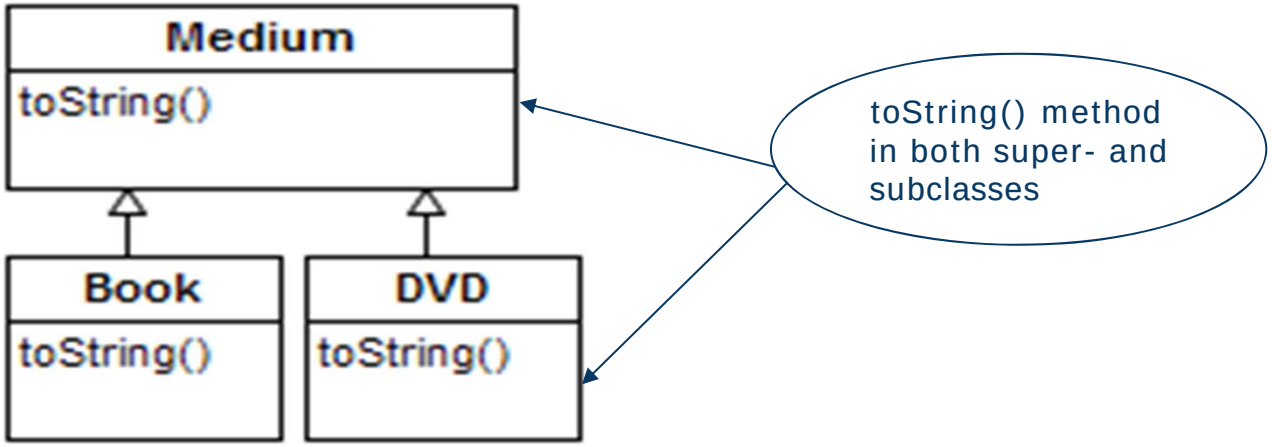
The toString() method in Medium only prints the common fields (title).
Inheritance is a one-way street:

- A subclass inherits the superclass fields.
- The superclass knows nothing about its subclass's fields.

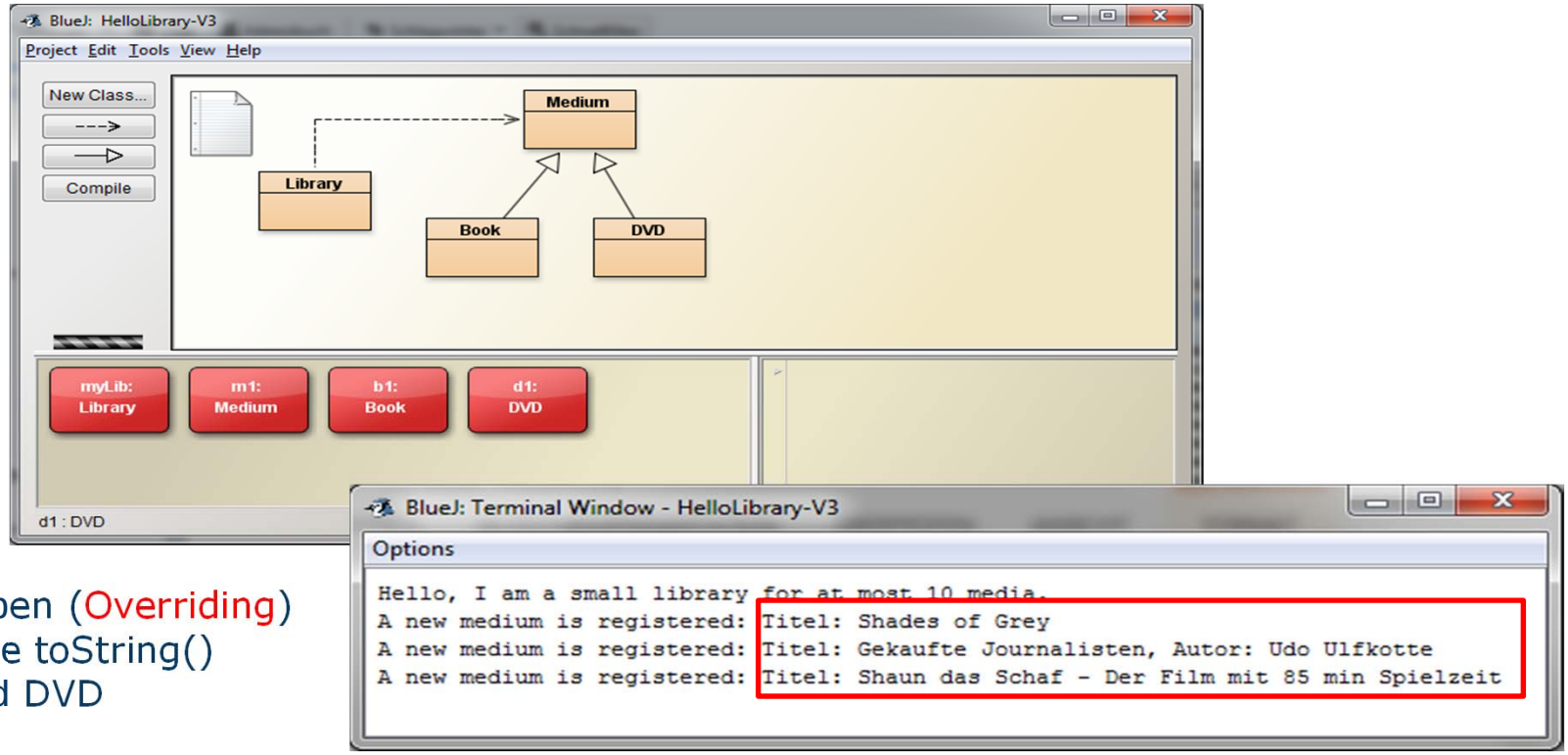


Overriding: the Solution

- Superclass and subclass define methods with the same signature.
- Each has access to the fields of its class.
- Superclass satisfies static type check.
- Subclass method is called at runtime – it *overrides* the superclass version.



BlueJ: Vererbung (2) - Demo
Erweiterung von HelloLibrary (U02)



Überschreiben (Overriding)
der Methode toString()
in Book und DVD

Method lookup / Polymorphic method dispatch

Method calls are polymorphic.
The actual method called depends on the dynamic object type:

- The variable is accessed.
- The object stored in the variable is found.
- The class of the object is found.
- The class is searched for a method match.
- If no match is found, the superclass is searched.
- This is repeated until a match is found.
- Overriding methods take precedence.
- The matched method is dispatched.

Super call in methods

Overridden methods are hidden ...

... but we often still want to be able to call them.

An overridden method *can* be called from the method that overrides it.

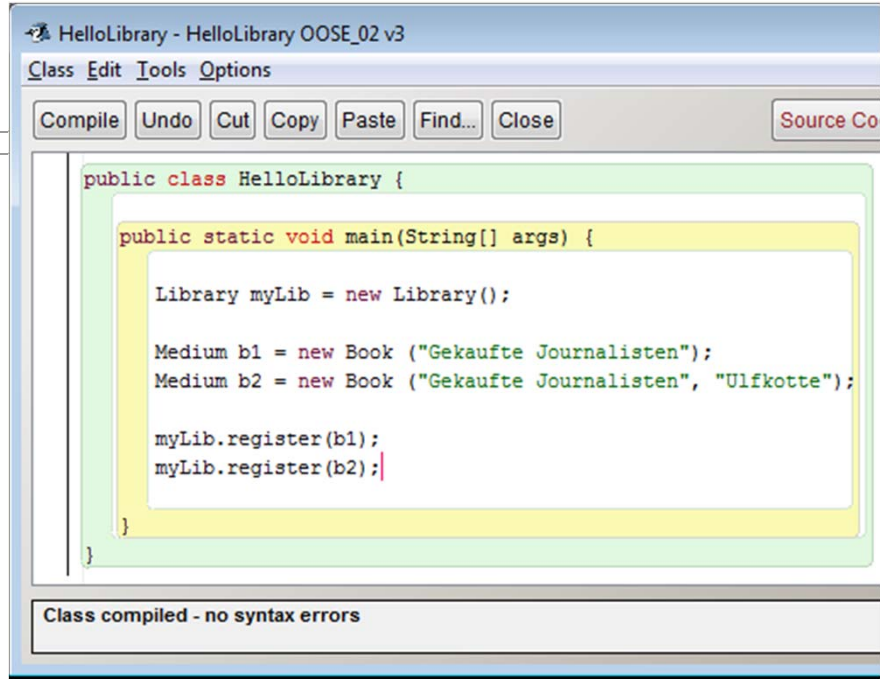
- **`super.method(...)`**
- use of **`super()`** in constructors.

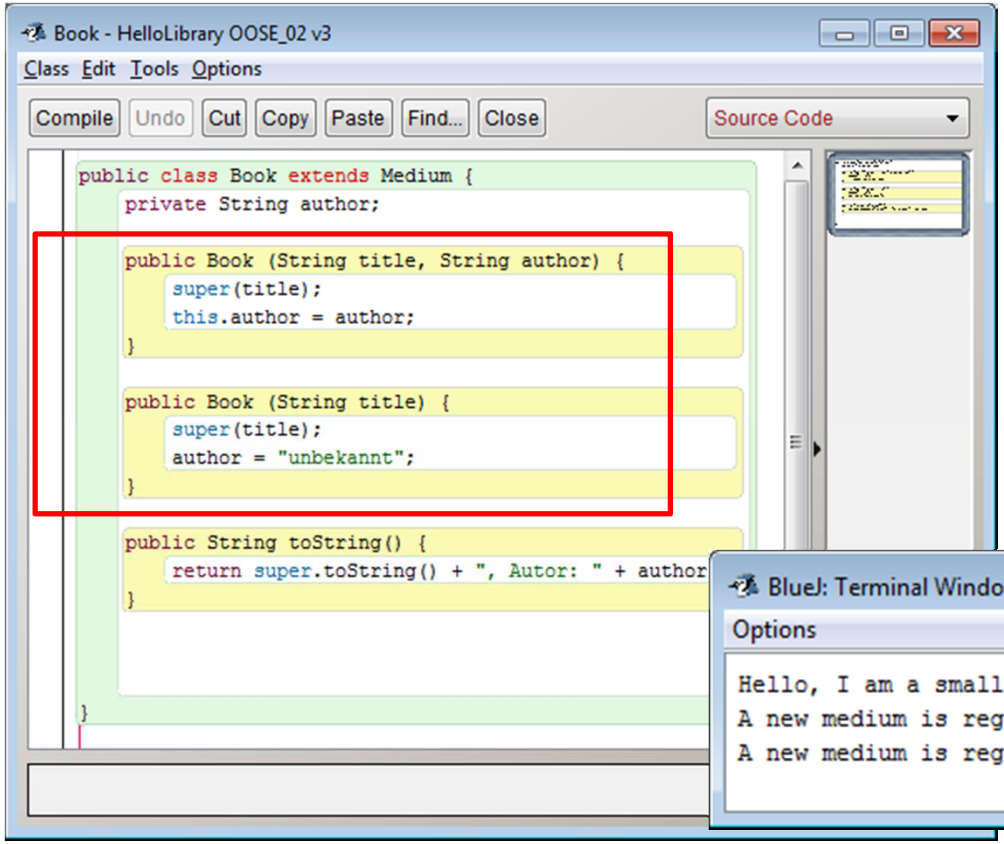
Overridden method

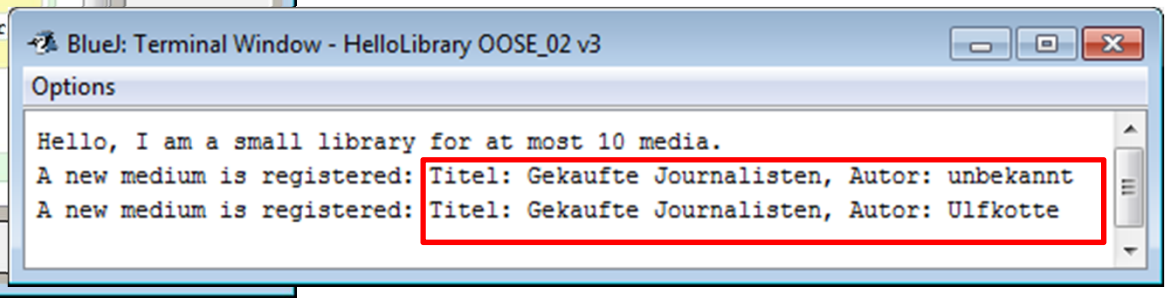
```
public class DVD
{
    ...
    public void toString(){
       return  super.toString() +
               " mit " + laenge + " min Spielzeit";

    }
}
```

TECHNISCHE UNIVERSITÄT DRESDEN

BlueJ: Overloading of Constructors







Lessons learned (1)

- Inheritance allows the definition of classes as extensions of other classes.

Inheritance
- avoids code duplication
- allows code reuse
- simplifies the code
- simplifies maintenance and extending

- Variables can hold subtype objects.
- Subtypes can be used wherever supertype objects are expected (substitution).

Lessons learned (2)

- The declared type of a variable is its static type.
- Compilers check static types.
- The type of an object is its dynamic type.
- Dynamic types are used at runtime.
- Methods may be overridden in a subclass.
- Method lookup starts with the dynamic type.
- Protected access supports inheritance.

- Overriding means runtime (dynamic) polymorphism.
- Overloading means compile time (static) polymorphism.

Polymorphism (Polymorphie) - Two Types in Java

[http://beginnersbook.com/2013/03/polymorphism-in-java/]

Method Overriding

- Applies only to inherited methods
- Object type (NOT reference variable type) determines which overridden method will be used at runtime
- Overriding method can have different return type
- Overriding method must not have more restrictive access modifier
- Abstract methods must be overridden
- Static and final methods cannot be overridden
- Constructors cannot be overridden
- It is also known as **runtime polymorphism**

Method Overloading

- Overloading can take place in the same class or in its sub-class.
- Constructor in Java can be overloaded
- Overloaded methods must have a different argument list.
- Overloaded method should always be the part of the same class (can also take place in sub class), with same name but different parameters.
- The parameters may differ in their type or number, or in both.
- They may have the same or different return types.
- It is also known as **compile time polymorphism**.

Literatur

[Barnes2012] David J. Barnes and Michael Kölling: Objects First with Java - A Practical Introduction using BlueJ. Fifth edition, Prentice Hall / Pearson Education, 2012

[Ratz2006] Dietmar Ratz, Jens Scheffler, Detlef Seese, Jan Wiesenberger: Grundkurs Programmieren in Java. Carl Hanser Verlag, 3. Auflage, 2006

Ende