

Software Engineering Übung

06

Design Prinzipien 2

Übung von Jonathan Lippert und Magnus Dierking

Tag der Einreichung: 18. Dezember 2020

Darmstadt



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Software Engineering Übung 06
Design Prinzipien 2

Übung von Jonathan Lippert und Magnus Dierking

Tag der Einreichung: 18. Dezember 2020

Darmstadt

1 Aufgabe

a

i

Siehe Abbildung 1.1

ii

LCOM = 3 von 11

b

Bei erstem Betrachten der Implementation erschien deren Kohäsion nicht optimal zu sein. Drei nicht zusammenfallende Verantwortungen sind alle in einer Klasse enthalten, die noch dazu ein sehr allgemeines Konzept beschreibt, was eine hohe Kohäsion sehr wünschenswert machen würde.

Der berechnete Wert spiegelt dies gut wieder, da er mit 3 auf jeden Fall nicht optimal ist. Angesichts der hohen Methodenanzahl in der Klasse hätte er aber dem Empfinden nach noch höher ausfallen können. Die Verantwortungen der Klasse Book sind:

1. Verwaltung der Daten eines einzelnen Buches
2. Verwaltung der Daten Bezüglich der Ausleihe eines einzelnen Buches
3. Exportieren der Daten eines einzelnen Buches als csv-Datei

Der ermittelte **LCOM**- Wert deckt sich also hier genau mit der Anzahl der Verantwortungen der Klasse.

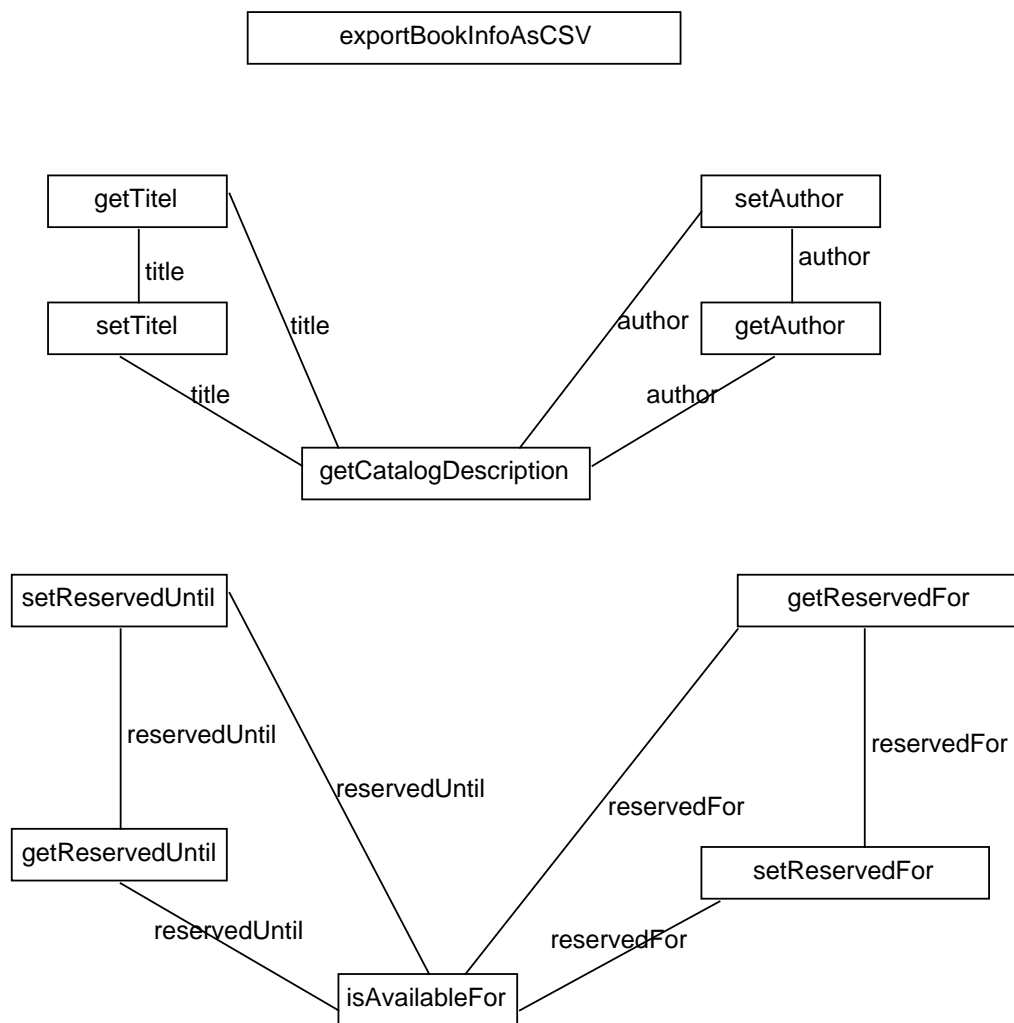


Abbildung 1.1: Zusammenhangsgraph der Klasse Book für das LCOM Verfahren.

c

Listing 1.1: Klasse Book

```

1 package org.library;
2
3 import java.util.Date;
4
5 import org.library.users.Client;
6
7 public class Book implements LibraryItem {
8
9     private String title;
10    private String author;

```

```
11
12 public Book(String title, String author){
13     this.title = title;
14     this.author=author;
15 }
16
17 // getters
18 public String getTitle() { return title; }
19
20 public String getAuthor() { return author; }
21
22 // implementiert interface LibraryItem
23 @Override
24 public String getCatalogDescription() {
25     return title + " by " + author;
26 }
27 }
```

Listing 1.2: Klasse Reservation

```
1 package org.library;
2 import java.util.Date;
3 import org.library.users.Client;
4
5 public class Reservation {
6
7     private Client reservedFor;
8     private Date reservedUntil;
9     private Book book;
10
11     public Client getReservedFor() { return reservedFor; }
12     public void setReservedFor(Client reservedFor) { this.reservedFor = reservedFor; }
13
14     public Date getReservedUntil() { return reservedUntil; }
15     public void setReservedUntil(Date until) { this.reservedUntil = until; }
16
17
18     // prueft, ob Buch vom Kunden geliehen werden kann
19     public boolean isAvailableFor(Client c, Date from) {
20         if (reservedFor != null || reservedUntil.after(from)) {
21             return false;
22         }
23         return c.numberOfBorrowedBooks() < 3;
24     }
25
26 }
```

Listing 1.3: RerservationManager

```
1 package org.library;
2 import java.util.Date;
3 import org.library.users.Client;
4
```

```
5 public class RerservationManager {
6
7
8 public static boolean isAvailableFor(Client c, Date from) {
9 if (reservedFor != null || reservedUntil.after(from)) {
10 return false;
11 }
12 return c.numberOfBorrowedBooks() < 3;
13 }
14
15 }
```

Listing 1.4: BookExporter

```
1 package org.library;
2
3 public class BookExporter{
4     public static void exportBookInfoAsCSV(org.library.formats.CSVExporter
5         exporter, Book book) throws java.io.IOException {
6         exporter.writeLine(book.getTitle(), book.getAuthor());
7     }
8 }
```

d
