Darmstadt University of Applied Sciences: Java Lab

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Library Specification

Prof. Dr. Bernhard Humm

Version: 1.0
Date: 7.09.2007

Darmstadt University of Applied Sciences
Department of Computer Science
Schöfferstr. 8b
64295 Darmstadt
Germany

1 Introduction

This specification is the basis for a Java lab at the Darmstadt University of Applied Sciences. The system to be implemented is a basic library system to be used by members of staff and customers of a library. The specification is a basic guideline for the development of the library application. However, it can and should be modified and extended meaningfully where appropriate.

This overview specification document is accompanied with a detail specification of all use cases and entities of the library system in form of JavaDoc (zip file). The JavaDoc contains prose descriptions of the interfaces, classes, attributes, and methods. Furthermore, it contains a semi-formal specification, using, e.g., pre and post conditions. The semi-formal notation is derived from the Quasar Specification Language (QSL, see J. Siedersleben: "Moderne Software Architektur" dpunkt Verlag 2004). QSL-derived elements are denoted in square brackets.

2 Library actors and use cases

The following Figure 1 gives an overview of the actors and use cases in a library as a UML use case diagram.

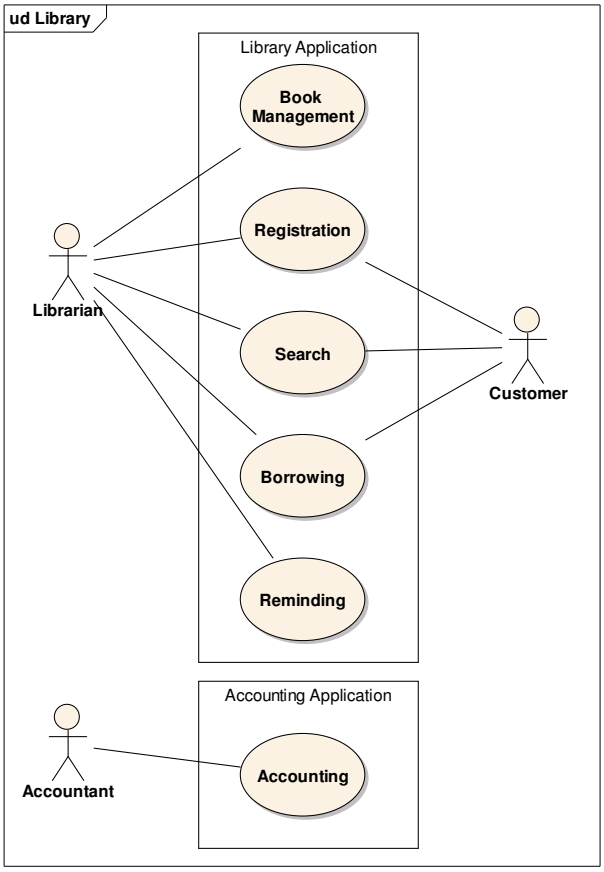

Figure 1

- `Customer`: Customers of a library can apply for a library membership (use case `Registration`). Then, they can search for books (use case `Search`) and borrow them (use case `Borrowing`). They have to return the borrowed books according to specific loan periods (use case `Borrowing`).

- `Librarian`: Librarians manage the book stock in the library (use case `Book Management`). They register new customers in the library application (use case `Registration`). They enter the loan of a book by a customer in the system (use case `Borrowing`). Books that are overdue result in a reminding /dunning process. The librarian manages this process (use case `Remind-ing`).

- `Accountant`: Registration and reminding fees are to be paid by customers. Accountants deal with those payments with a separate accounting application (use case `Accounting`).

3 Application architecture

The following Figure 2 gives an overview of the library application architecture as a UML class diagram.

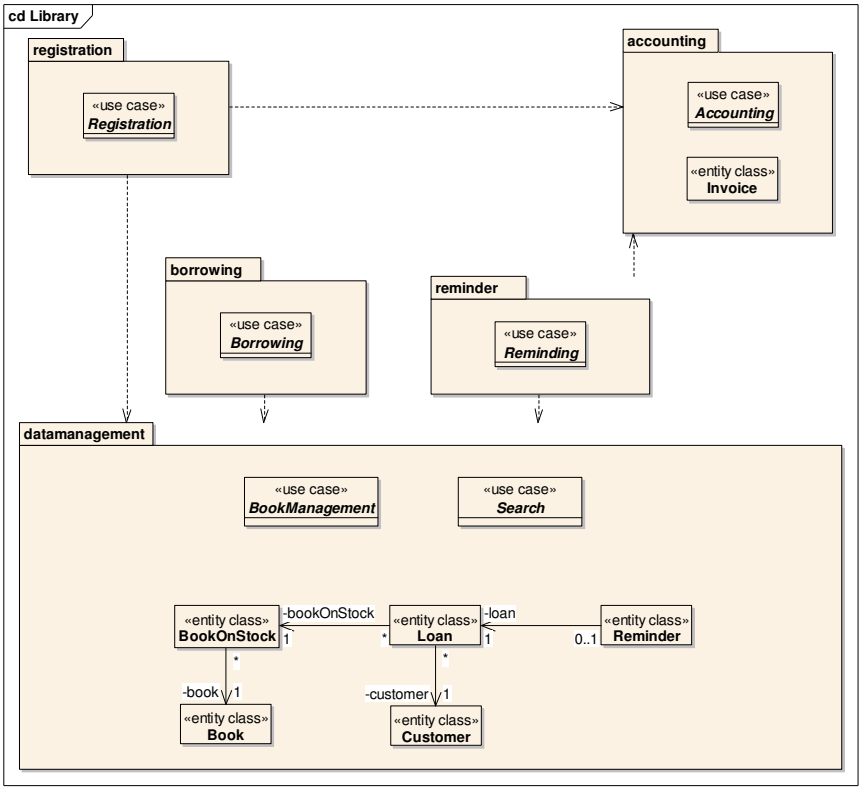

Figure 2: Library application architecture

The application architecture is component-oriented. Components contain use cases and / or entities (denoted as UML stereotypes).

The `datamanagement` component comprises most entities of the library application (`Book`, `BookOnStock`, `Loan`, `Customer`, and `Reminder`) and the use cases `BookManagement` and `Search`.

The `registration` component contains the use case `Registration` only. It depends on the `datamanagement` component (see UML dependency relationship) since it uses its services for reading and writing `Customer` entities. Furthermore, it uses the `accounting` component for invoicing registration fees.

The `borrowing` component contains the use case `Borrowing` only. It depends on the `datamangement` component for reading `Book`, `BookOnStock`, and `Customer` entities and for writing `Loan` entities.

The `reminder` component contains the use case `Reminding` only. It depends on the `datamanagement` component since it must access overdue `Loan` entities and corresponding `Customer` data. Furthermore, it generates `Reminder` entities. It also depends on the `accounting` component for invoicing reminder fees.

Finally, the `accounting` component comprises the use case `Accounting` and the entity `Invoice`. It is regarded as a separate application which is used by the library application and, potentially, by other applications.

In the following sections, we give an overview of all components.

4 Component datamanagement

4.1 Overview

The following Figure 3 gives an overview of the `datamanagement` component.

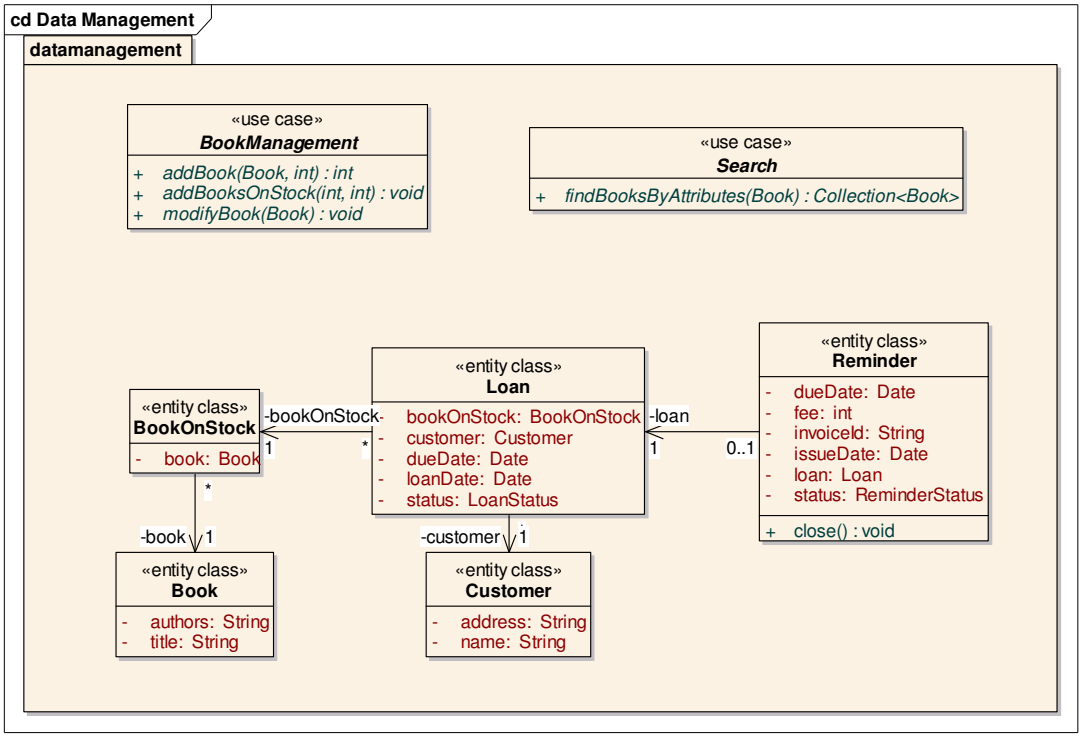

Figure 3: Component `datamangement`

See the detail specification of the use cases and entities (accompanied zip file with JavaDoc).

5 Component borrowing

The following Figure 4 gives an overview of the `borrowing` component.

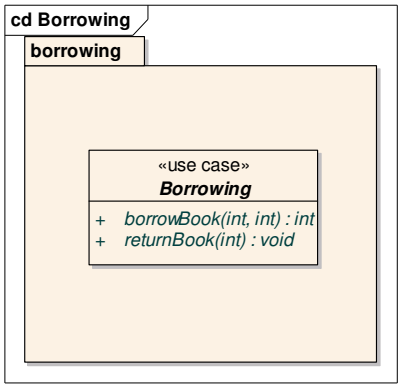

Figure 4: Component `borrowing`

See the detail specification of the use case (accompanied zip file with JavaDoc).

6 Component registration

The following Figure 5 gives an overview of the `registration` component.

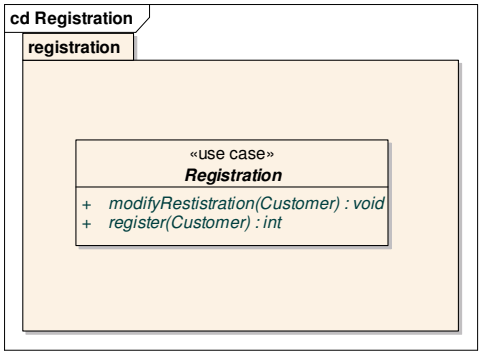

Figure 5: Component `registration`

See the detail specification of the use case (accompanied zip file with JavaDoc).

7 Component reminder

The following Figure 6 gives an overview of the `reminder` component.

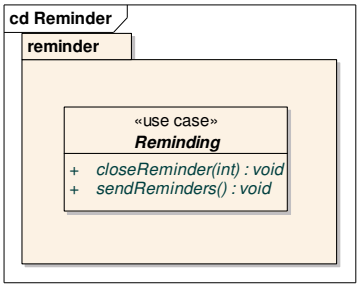

Figure 6: Component `reminder`

See the detail specification of the use case (accompanied zip file with JavaDoc).

8 Component accounting

The following Figure 7 gives an overview of the `accounting` component.

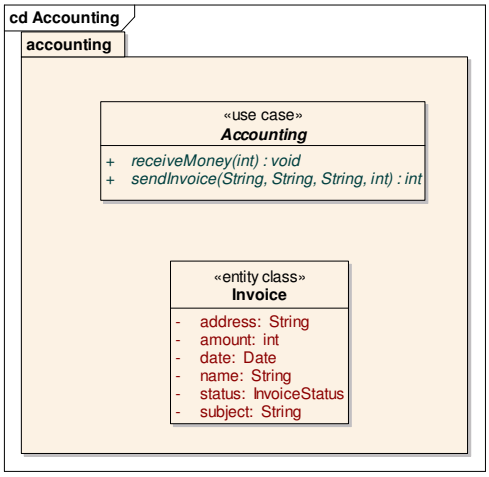

Figure 7: Component `accounting`

See the detail specification of the use case and the entitiy (accompanied zip file with JavaDoc).