COMP-361 Software Engineering Project

Final

February 24th 2013: 6pm - 9pm

(10% of final grade)

February 24, 2014

Student Name:

Examiner: Jörg Kienzle

McGill ID:

Instructions:

- DO NOT TURN THIS PAGE UNTIL INSTRUCTED
- This is a closed book examination. Non-electronic translation dictionaries are permitted, but instructors and invigilators reserve the right to inspect them at any time during the examination.
- Besides the above, only writing implements (pens, pencils, erasers, pencil sharpeners, etc.) are allowed. The use of any other tools or devices is prohibited.
- Answer **all** questions **on this examination paper** and return it. If you need additional space, use pages 16-18, and clearly indicate where each question is continued.
- This examination is printed on both sides of the paper.

The exam has 18 pages with 6 questions.

From Requirements to Design: Library System

Informal Description of the Library System (same as seen in class)

The system to build is an automated library book borrowing system that is to be used in the departmental libraries of a university. The goal is to relieve the librarians from processing book loans. The system does not require users to identify themselves to search for books according to certain criteria and to check the availability of a particular book. However, to check-out books and to check their respective book loan status, users must first identify themselves to the system.

A single receipt is printed for each user check-out session; it details for each book: the title of the book and the date the book is to be returned by. At the start of each week, the system sends warning emails to all borrowers that have overdue books.

Books have physically attached barcodes, which are used for the identification of books that are checked out (a barcode scanner is to be used). If the book does not scan, it should be also possible to enter the barcode manually. Books that are on reserve are not available for loan.

Partial Use Case Model of the Library System

A partial use case model has been established for the library system. So far it only includes three use cases: RegisterNewBook, CheckAvailability and BorrowBook.

Register New Book Use Case

Use Case: RegisterNewBook
Scope: Library System
Level: User Goal
Intention in Context: The intention of the *Librarian* is to enter a new book into the system.
Multiplicity: There can be only one *Librarian* registering new books at a given time.
Primary Actor: Librarian
Facilitator Actor: AdminTerminal
Secondary Actor: BarcodePrinter
Main Success Scenario:

Librarian steps in front of the AdminTerminal.
1. Using the *AdminTerminal,* the Librarian informs *System* about title of the new book and the number of copies that were bought.

Steps 2 and 3 are repeated for each book copy.
2. *System* instructs *BarcodePrinter* to print a barcode for the copy.
3. *Librarian glues printed barcode to the book copy.*

CheckAvailability Use Case

Use Case: CheckAvailability
Scope: Library System
Level: User Goal
Intention in Context: The intention of the User is to query the system in order to determine if it is possible to take out a copy of a given book.
Multiplicity: Many Users can check availability of books at a given time.
Primary Actor: User
Main Success Scenario:

1. *User* asks *System* if a given book is available for loan.
2. *System* informs *User* of availability of book.

BorrowBook Use Case

Use Case: BorrowBook
Scope: Library System
Level: User Goal
Intention in Context: The intention of the User is to borrow a book from the library.
Multiplicity: Many Users can borrow books at a given time. A User can only borrow one book at a time.
Primary Actor: User
Facilitator Actors: BarcodeScanner, GUI, Speaker, ReceiptPrinter
Main Success Scenario:

1. *User* provides *System* with username and password.
2. *System* informs *User* of successful authentication.
   Steps 3 and 4 can be repeated as many times as desired by the user.
3. *BarcodeScanner* informs *System* of book copy that User wants to borrow.
4. *System* acknowledges successful loan to User by asking the *Speaker* to emit a beep.
5. *User* informs *System* that checkout is complete.
6. *System* instructs *ReceiptPrinter* to print the list of all current loans (including the books that were just checked out) of the user.

Extensions:

2a. Username / password pair did not match.

2a.1 *System* notifies *User* that authentication was unsuccessful. Use case continues at step 1.

3a. Barcode Scanner is out of service.

3a.1 *User* uses the GUI to inform the *System* of the book copy that he wants to borrow. Use case continues at step 4.

4a. The book copy that the user wants to borrow is on reserve, or User has already reached the maximum of allowed books.

4a.1 *System* informs *User* that this bookcopy can not be borrowed and for what reason. Use case continues at step 3.

Partial Environment Model of the Library System

Figure 1 shows a partial environment model of the library system based on the three use cases above. The messages and parameters are:

- isAvailableForLoan(b: Book)
- authenticate(name: String, password: String)
- borrowBook(b: BookCopy)
- registerBook(title: String, nbOfCopies: Integer, c: BookCategory)
- checkoutComplete()

The output messages and parameters are:

- printBarcode(code: Integer)
- printReceipt(Set(LineItems)) (where LineItem is defined as a TupleType {booktitle: String, dueDate: Date})
- availabilityAck(yesNo: Boolean)
- authenticationResult(r: Boolean)
- borrowingNotPossible(r: ReasonKind) (assuming that ReasonKind is defined as enum {OnReserve, MaxBooksReached)
- beep()

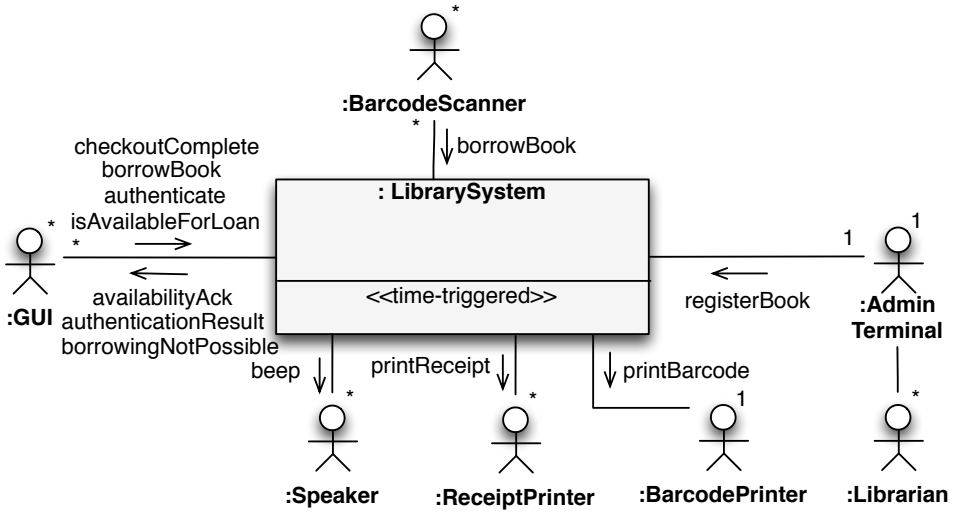

Figure 1: Partial Environment Model of the Library System

Partial Concept Model of the Library System

Figure 2 shows a partial concept model of the system.

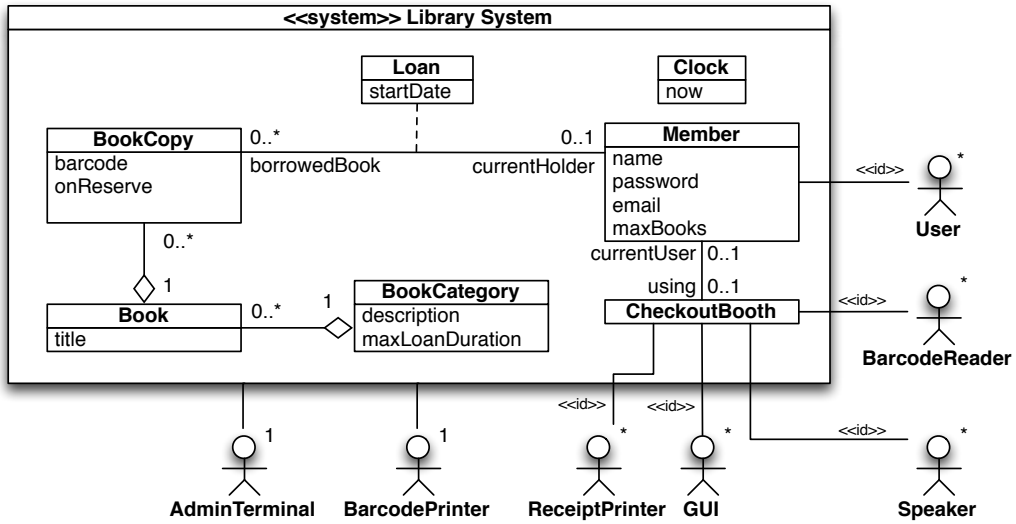

Figure 2: Partial Concept Model of the Library System

Partial Operation Model of the Library System

A partial operation model describing the effects of the five system operations *isAvailableForLoan*, *registerBook, authenticate, borrowBook,* and *checkoutComplete* is given below.

Operation: LibrarySystem::isAvailableForLoan(b: Book)
Scope: Book, BookCopy, Loan
Messages: GUI::{AvailabilityAck}
Description: If the book b is not on reserve, and there are currently copies in the library, i.e. not all copies are taken by other members, then the system sends a positive acknowledgment to the user that triggered the system operation. Otherwise it sends a negative acknowledgment.

Operation: LibrarySystem::registerBook(title: String, nbOfCopies: Integer, c: BookCategory)
Scope: Book, BookCopy, BookCategory
Messages: BarcodePrinter::{PrintBarcode}
New: newBook: Book, newCopies: Set(BookCopy)
Description: The effect of this operation is to create a new instance of Book and initialize it's attributes and associations. Also, the operation creates nbOfCopies instances of BookCopy and associates them with the new book. Each copy is assigned a unique barcode, and the BarcodePrinter is instructed to print the assigned barcodes one by one.

Operation: LibrarySystem::authenticate(name: String, password: String)
Scope: Member, CheckoutBooth
Messages: GUI::{AuthenticationResult}
Description: If the username and password match the data stored for a member in the system, then authentication is successful, which has as an effect to remember that the authenticated user is now using this particular checkout booth, and a positive authenticationResult message is sent to the corresponding GUI. If not, then a negative authenticationResult message is sent.

Operation: LibrarySystem::borrowBook(bc: Barcode)
Scope: Member, CheckoutBooth, BookCopy, Loan, Clock
New: newLoan: Loan
Messages: GUI::{BorrowingNotPossible}; Speaker:{Beep}
Description: The effect of this operation is to first check that the book copy is not on reserve, as well as to make sure that the member that is requesting to borrow the book has not reached his maxBooks capacity. If any of these checks fails, then a borrowingNotPossible message is sent to the corresponding GUI. If both checks are successful, then a new loan is created, initialized with the current time, associated with the book copy and

the member, and the speaker of the checkout booth beeps to inform the user that the transaction was successful.

Operation: LibrarySystem::checkoutComplete()
Scope: Member, CheckoutBooth, BookCopy, Book, BookCategory, Loan
Messages: ReceiptPrinter::{PrintReceipt}
Description: The effect of this operation is to complete the checkout of the member using this checkout counter by printing a receipt on the ReceiptPrinter of the counter. The receipt lists all current loans of the member, and when they expire. In order to ensure that the next member needs to authenticate again before using the checkout counter, the association between the checkout counter and the member using it is removed.

Partial Protocol Model of the Library System

Finally, the Protocol Model of the Library System is shown in Fig 3.

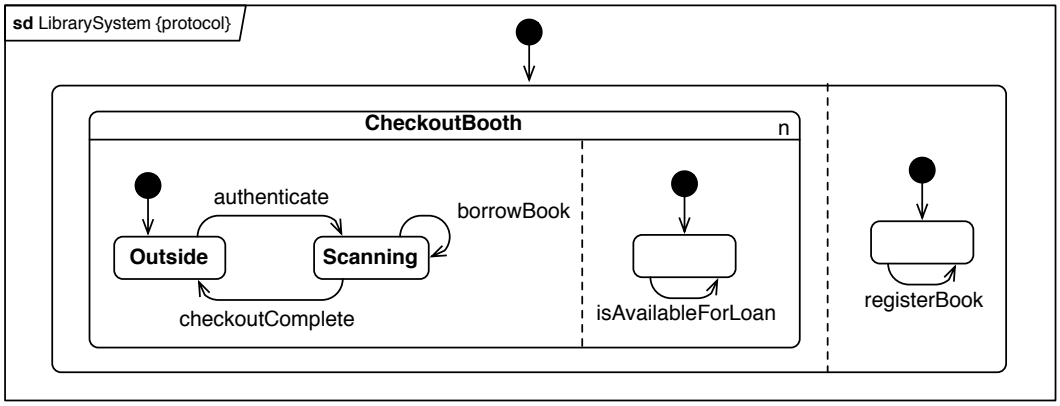

Figure 3: Partial Protocol Model of the Library System

Partial Interaction Model of the Library System

isAvailableForLoan Design

The Graphical User Interface (GUI) that allows a user to trigger the operation was designed by other developers and is not shown here. The following design decisions were made:

- The conceptual parameter b: Book of the system operation is implemented as a string containing the title of the book. It is the responsiblity of the GUI to not pass a string value that does not match a title of an existing book as a parameter. Book titles are assumed to be unique.
- The design class CheckoutBoothInterface can display messages in forms of strings to the user standing in front of the booth (operation display(message: String)). When the GUI invokes the isAvailableForLoan operation, it passes a reference to the instance of the CheckoutBoothInterface class that represents the booth that is currently being used.

The sequence diagram describing the design of isAvailableForLoan is shown in Fig. 4.

registerBook Design

The sequence diagram describing the design of registerBook is shown in Fig. 4.

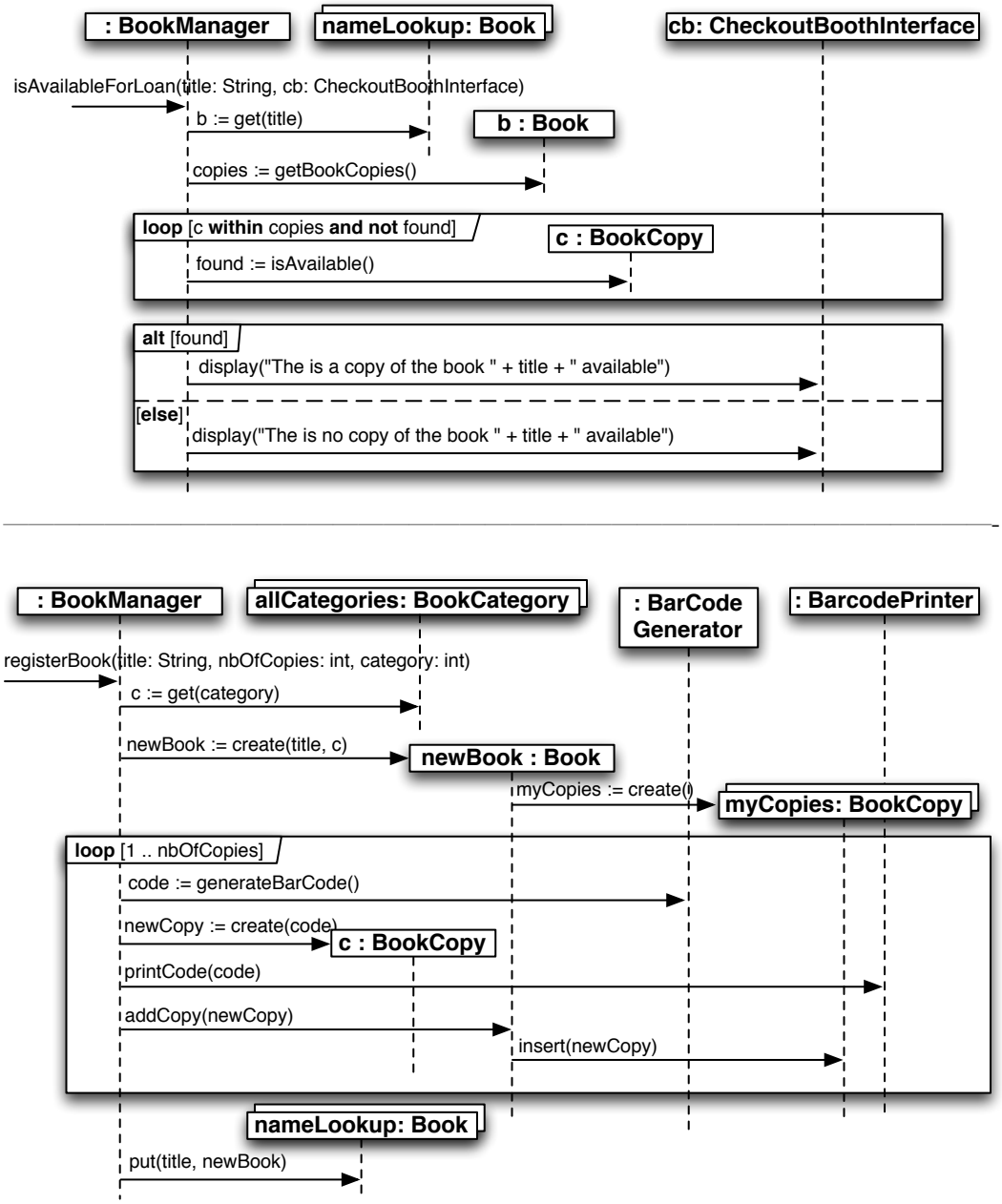

Figure 4: Interaction Model for *isAvailableForLoan* and *registerBook*

Partial Design Model of the Library System

Based on the *isAvailableForLoan* and *registerBook* interaction models shown above, a partial design class model for the Library System was established. It is shown in Fig. 5.

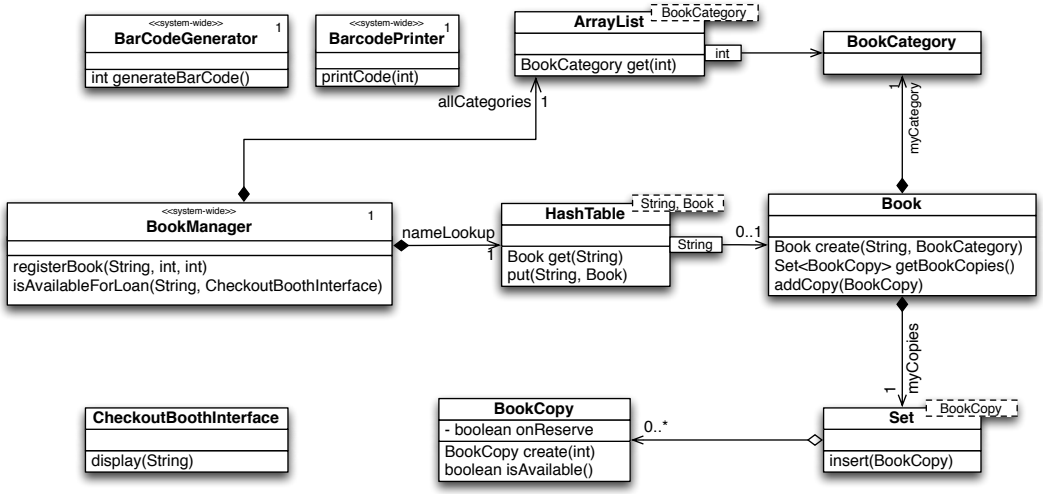

Figure 5: Partial Design Class Model for the Library System

Question 1 - Authenticate

Elaborate a design for the system operation *authenticate*. The Graphical User Interface (GUI) that allows a user to trigger the operation was designed by other developers. They have already made certain design decisions:

- They designed a class CheckoutBoothInterface that can display messages in forms of strings to the user (operation display(message: String)). When the GUI invokes the *authenticate* operation that you are designing, it passes a reference to an instance of the CheckoutBoothInterface class as a parameter. This will allow your design to display the acknowledgment on the correct user screen.

You can model your design using a communication diagram or a sequence diagram, whichever you prefer. Please justify the choice of controller. Make sure that this design is "compatible" with the design of *isAvailableForLoan* and *registerBook*.

Question 2 - BorrowBook

Elaborate a design for the system operation *borrowBook*. You are allowed to make changes to the design of *isAvailableForLoan* and *registerBook* to make the 3 interaction models "compatible" (report on these changes, if any, in questions 2.1, 2.2 and 2.3). You can model your design using a communication diagram or a sequence diagram, whichever you prefer. Please justify the choice of controller.

Question 2.1 - Updating the behaviour of isAvailableForLoan

Now that you designed the behaviour of *borrowBook*, do you need to change the interaction model of *isAvailableForLoan*? Please explain briefly, and provide a (partial) sequence diagram describing the changes, if any.

Question 2.2 - Updating the behaviour of registerBook

Now that you designed the behaviour of *borrowBook*, do you need to change the interaction model of *registerBook*? Please explain briefly, and provide a (partial) sequence diagram describing the changes, if any.

Question 2.3 - Updating the structure of already existing design classes

Now that you designed the behaviour of *borrowBook*, do you need to add new attributes or references to design classes that were already present in the existing design shown in Fig. 5? Please explain briefly, and show the updated design class diagram. Note that it is ok to only show the additional properties (attributes / references) of classes, i.e., it is not necessary to repeat properties that were already depicted in Fig. 5.

Question 3 - CheckoutComplete

Elaborate a design for the system operation *CheckoutComplete*. You can model your design using a communication diagram or a sequence diagram, whichever you prefer. Please justify the choice of controller. Make sure that this design is "compatible" with the design you did for question 2.

Question 4 - Design Class Model

Elaborate a design class model for your design, i.e., for the classes you introduced in questions 1, 2, 2.1, 2.2, 2.3, and 3. Remember that the design class model must show attributes and methods for every class, as well as navigable associations with role names between them, if any. You can assume that template collection classes like the ones that Java provides are available in order to implement any multi-objects you used. Also, remember that the design class model does not need to have design-equivalents for all the classes shown in the concept model. You only need to show the ones that you are actually using in the design.

Note that you only have to show the *new design classes* that YOU introduced, *as well as classes* that were already depicted in Fig. 5 *to which you added additional properties* (attributes / references) that are needed within the interactions of authenticate, borrowBook and checkoutComplete.

Question 5 - Analyzing Design Dependencies

Answer the following questions for the complete design class model (i.e. what you have shown in question 4 combined with what is shown in Fig. 5):

- Which objects in your design are system-wide objects (also known as singletons)?

- Please list all the classes that are responsible for instantiating objects of other classes. For example, if an instance of A instantiates objects of class B, then write: "As instantiate Bs"

- Do any of your business classes (i.e., the classes that represent classes in the concept model) depend on classes that implement communication with the outside world? If yes, list them, together with the kind of dependency (permanent association, parameter dependency, instantiation dependency, call dependency).

Question 6 - Enforcing the Protocol Model

The protocol model of the Library System clearly states that users must first authenticate before they can borrow books. The design team that elaborated the GUI took care of ensuring that the GUI sends the input messages to the system in the right order. You have to now make sure that in case any of the other input messages (i.e., messages not sent by the GUI) arrive out-of-order, the user is notified of his mistake by displaying an appropriate error message.

Please explain which system operation(s) are affected, and show how the corresponding interaction model(s) need to be changed.

Additional Page:

Please specify which problem you are answering.

Additional Page:

Please specify which problem you are answering.

Additional Page:

Please specify which problem you are answering.