Object-Oriented Software Development Using CRC Cards and UML

References:

Nancy M. Wilkinson. *Using CRC Cards - An Informal Approach to Object-Oriented Software Development*, AT&T Bell Laboratories, SIGS Books, New York, 1995

Martin Fowler with Kendall Scott. *UML Distilled, Second Edition - A Brief Guide to the Standard Object Modelling Language*, Addison-Wesley 2000

CRC Cards and UML

"Despite the buzzing about object-oriented methods and tools, the true measure of how successful someone will be with objects is the degree to which he or she intuitively grasps the underlying idea of the object-oriented approach.

Methodologies and tools cannot replace this gut-level understanding; they simply support the work done once the transition from process to object is made.

Using CRC cards is an excellent way for people to begin to make this transition."

Mary Wilkinson

"Now widely adopted as the de facto industry standard and sanctioned by the Object Management Group, the Unified Modelling Language (UML) is a notation all software developers need to know and understand.

One of the most valuable techniques for learning OO is CRC cards, which are not part of UML, although they can, and should, be used with it"

Martin Fowler

Classes, Responsibilities, Collaborators

class name	
subclasses	
superclasses	
responsibilities	collaborators

Suggested format for CRC card

CRC Cards

- **Class Definition (on back of card)**
  - A concise definition of the abstraction represented by the class.
- **Class Name**:
  - if chosen appropriately will greatly increase the understandability of the model.
- **Subclass, Superclass**
  - If they exist can be shown. Very probably these will be added late in the development cycle.
- **Responsibilities**
  - Knowledge that objects of a class maintains , and
  - Services that a class provides.
- **Collaborator**
  - Class(es) that provide services that are needed to allow fulfilment of responsibilities - listed on the same row as the relevant responsibility.

CRC Card Session

The CRC card technique is based on the use of small teams to perform analysis and design.

- Team Size
  - The ideal team size has been found to be 4 or 5.
  - A smaller group is unlikely to have the required range of problem domain and solution domain expertise.
  - A larger group tends to fragment, with greater interaction problems.

CRC Team

- Team Composition
  - **Domain Expert** — At least one participant should have "domain knowledge", ie should understand what the system should do, but may have little knowledge of how it may be accomplished.
  - **Solution Expert** — Clearly at least one member of the team should have expertise in object-oriented methodology, even if they have little knowledge of the problem domain.
  - **Facilitator** — The facilitator may also be the Solution Expert but may be a person who is not a member of the team but who uses her object-oriented expertise to keep the team on track. She may perform this role with more than one team concurrently.
  - **Group Dynamics** — Newly formed project teams sometimes find that the interactions in a team are a unifying experience, but " Don't try technological fixes for sociological problems" (Stroustrup). It is best to have a group who work well together, respect each other and are not dominated by one or a few of the members

CRC Tasks

Choosing a subset of the problem

It is essential that a CRC session focus on one fairly small and manageable portion of the system at a time.

e.g. the following example comes from Wilkinson:

The Problem

This application will support the operations of a technical library for an R&D organisation. This includes the searching for and lending of technical library materials, including books, videos, and technical journals. Users will enter their company IDs in order to use the system; and they will enter material ID numbers when checking out and returning items.

Each borrower can be lent up to five items. Each type of library item can be lent for a different period of time (books 4 weeks, journals 2 weeks, videos 1 week). If returned after their due date, the library user's organisation will be charged a fine, based on the type of item (books $1/day, journals $3/day, videos $5/day).

Materials will be lent to employees with no overdue lendables, fewer than five articles out, and total fines less than $100.

Possible Classes

A brainstorming session produces the following set of candidates for classes

- Library
- Librarian
- User
- Borrower
- Article
- Material
- Item
- Due Date
- Fine
- Lendable
- Book
- Video
- Journal

Filtering Classes

- Difference between Library and Librarian?
  - Librarian interacts with user and checks items in and out.
  - Hard to see role for Library - perhaps name of the application.
- Difference between Article, Item, Material and Lendable.
  - These all seem to refer to the same thing, so which name to use?
  - Lendable seems most expressive - discard other class names.
- Difference between User and Borrower?
  - Borrower represents information about library patrons - knows what books are out and which ones etc.
  - User represents a person using the library.
  - Retain both, at least tentatively.
- Due Date and Fine?
  - These are probably attributes of Lendable and Borrower respectively rather than independent objects.
  - But there is probably a need for a Date class since dates will need to be compared, added to etc. Due Date could be an instance of Date.

Tentative Classes

Librarian: the object that fufils user requests to check out check in and search for items

User: The human being that comes to use the library

Borrower: The set of objects that represent users who borrow from the library

Date: The set of objects that represent dates in the system

Lendable: The set of objects that represent items to be borrowed from the library

Book: The set of objects that represent books to be borrowed from the library

Video: The set of objects that represent DVDs and VideoTapes to be borrowed from the library

Journal: The set of objects that represent Technical journals to be borrowed

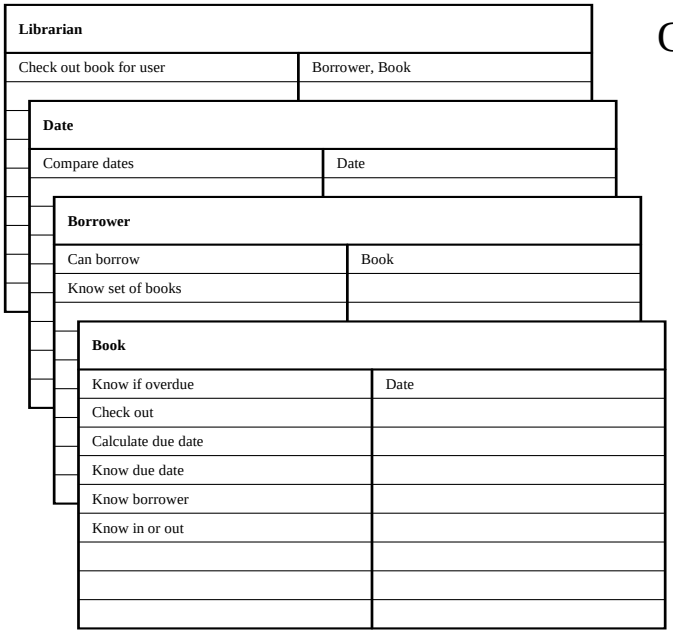
Check-out Scenario

"What happens when Barbara Stewart, who has no accrued fines
and one outstanding book, not overdue,
checks out a book entitled *Effective* C++ *Strategies*?"

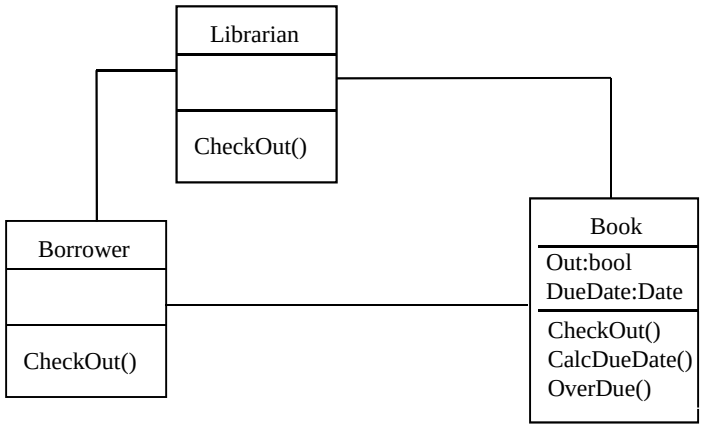
Check-out Scenario

- Librarian
  - needs to add “check out lendable”to list of responsibilities.
    - needs to collaborate with Book and Borrower to do this.
- Book
  - needs to add “know if overdue” to list of responsibilities.
  - needs to add “check out” to list of responsibilities. This requires updating:
    - in-or-out status, borrower, due date.
  - needs to add “calculate due date” to list of responsibilities.
    - needs to clllaborate with Date to do this.
- Borrower
  - needs to add “know set of lendables” as a responsibility.
- Date
  - needs to add “compare dates” to list of responsibilities.
  - needs to add “add days to date” to list of responsibilities.

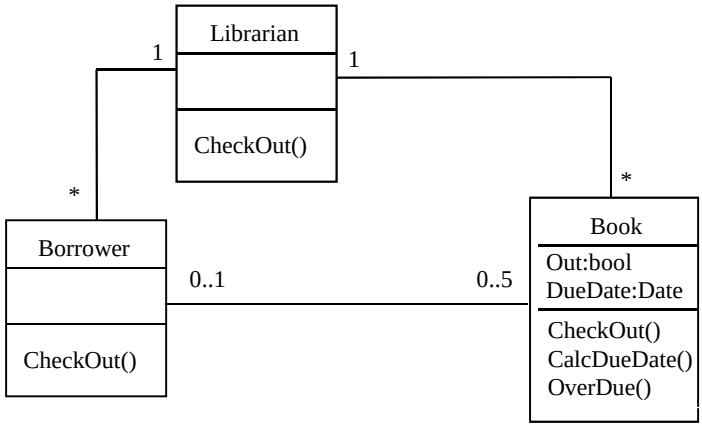
Cards after First Scenario



UML Class Diagram

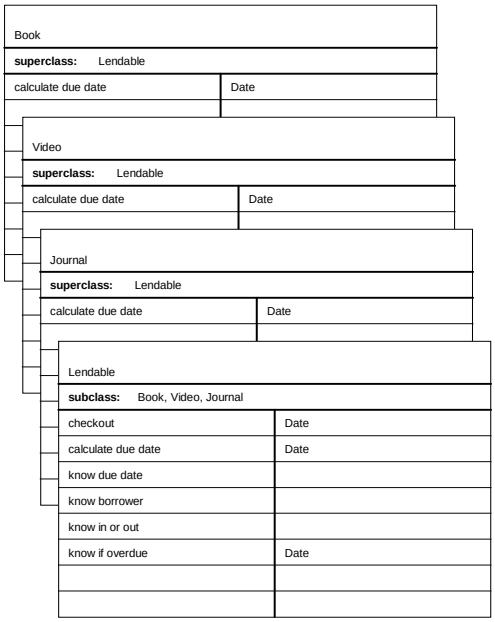


Adding Multiplicity



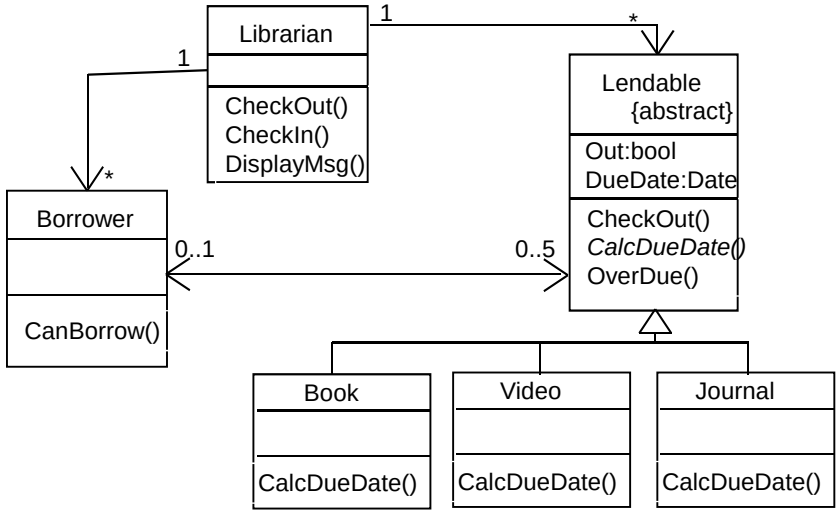
Superclass Discovery

- Checking out a Video or a Journal gives a very similar scenario to checking out a book.
- Lendable should be a superclass of Book, Video and Journal.
  - The difference between these lies in the way that due date is calculated
- Librarian should replace “check out book” by “check out lendable” in list of responsibilities
- Borrower should replace “know set of books” by “know set of lendables” in list of responsibilities



Cards after Superclass Discovery

Adding Inheritance and Navigability



Check-in Scenario

"What happens when Liz Flanagan,
returns the book, *Barely Managing O-O Projects*,
on time?"

Check-in Scenario

- **Librarian**
  - needs to add “check in lendable”to list of responsibilities.
  - needs to collaborate with Book and Borrower to do this
- **Book**
  - needs to add “check in” to list of
  - responsibilities of base class. This requires updating:
    - in-or-out status
    - borrower
- **Borrower**
  - already has “know set of lendables” as a responsibility

Exceptional Scenarios

“What happens when Gregg Vosander tries to check out a journal,
Genetic Programming in your spare time
when he already has 5 volumes of the same journal at home

- Borrower has “know set of lendables” responsibility, so knows that at lendable limit and can report this to Librarian.
- Librarian, since borrower is at lendable limit, needs to alert User that no more can be checked out, and end the session.
- Librarian needs to add “display message” to set of responsibilities.

Another Exceptional Scenario

"What happens when Garrett Ziegler returns a video, *Introduction to Interactive Television*, 3 days overdue

- Book has "know if overdue" as a responsibility so can report this to Librarian.
- Librarian needs to know fine to charge.
  - Book is obvious choice for this since each type of lendable does this differently. Book adds "calculate fine" to list of responsibilities.
- Librarian asks Borrower to record the amount of the fine.
  - Borrower adds "know fine amount" to list of responsibilities.
- Librarian needs to inform User of amount of fine.
  - Uses "display message" responsibility to do this.

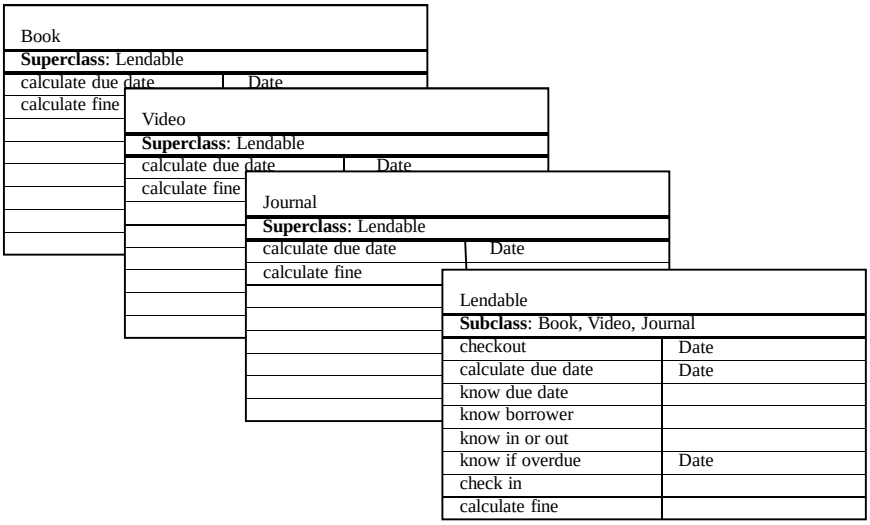
Cards after Exceptional Scenarios (1)

Date	
compare dates	Date
add days to date	

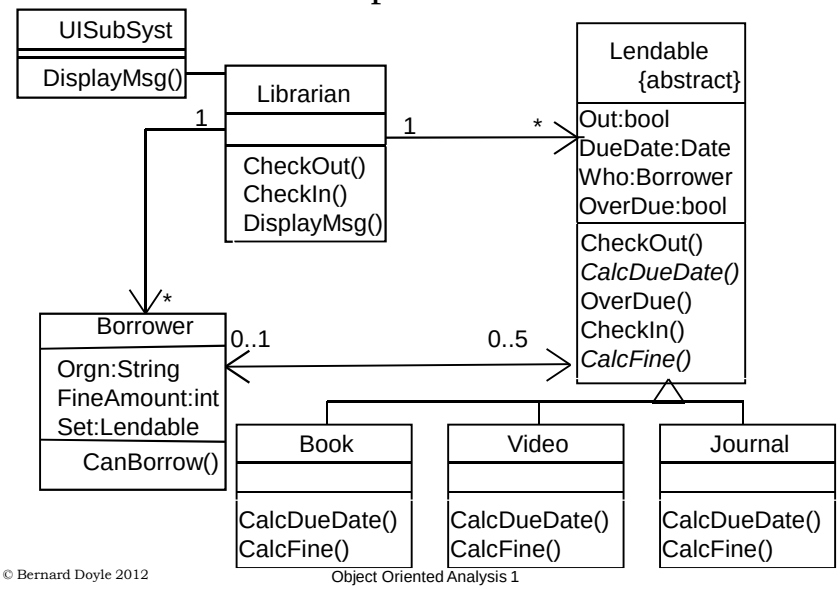
Borrower	
can borrow	Lendable
know set of lendables	
know fine amount	
know organisation	

Librarian	
check out lendable	Lendable, Borrower
check in lendable	Borrower, Lendable
display message	*UI Subsystem*

Cards after Exceptional Scenarios (2)



After Exceptional Scenarios



Discovering New Classes

"What happens when Ned Brooks comes to the Library
in search of a book entitled *The Mythical Mammoth*?"

- Librarian takes responsibility "search for lendable", but where to look?
- Need a Collection class that has responsibility "know set of lendables". Will need as collaborators, DB Subsystem and Book to get the information requested.
- This leads to a discussion of what Books and other lendables will need to support searching. Rather than have a "know" responsibility for each of these, represent this information a *attributes* on the back of eack card.

CRC Cards with Attributes on Back

Lendable: the set of objects that represent items to be borrowed

Attributes:

in/out status	due date
borrower	Dewey decimal #

Book: the set of objects that represent books to be borrowed

Attributes:

title	publication date
author	publisher

Journal: the set of objects that represent journals to be borrowed

Attributes:

name	volume
date	issue #

Video: the set of objects that represent videos to be borrowed

Attributes:

title	date
DVD or Tape	producer