

THE LIBRARY PROBLEM



A Case Study for Introducing the Object Oriented Design Paradigm

APCS Readiness 2005-06 @ UCLA

Part I: *Problem Statement*

Part II: *CRC Cards*

Part III: *Testing Scenarios through Role Playing*

Part IV: *From Design to Java*

Part V: *Individual Student Project*

THE LIBRARY PROBLEM

Part I *Problem Statement*



Simple Problem Statement

This application will support the operations of a school library. This includes the searching for and lending of books. All library items have registration code (book isbn). Each borrower can borrow up to 4 books. If returned after their due date, the borrower will be charged a 5-cents-a-day fine. Materials will be lent to borrowers only if they have (1) no more than two overdue books, (2) less than a \$10 fine.

Adding Complexity

- Adding new books and borrowers.
- Expanding collection of lendables to include journals, videos, software, music, etc.
- Setting number of lendables for borrowers by membership or by collection type
- Differing late fees for different collection items
- Library catalog and list of books borrowed by user

Instructor note: It is appropriate for adding complexity for different reasons:

- (1) an in-class extension of the original problem for advanced students or groups who have finished ahead of other groups;
- (2) for students seeking extra credit;
- (3) for a follow-up homework assignment;
- (4) to revisit later in the year as students learn new concepts that will allow them to design more complex classes;

Complex Problem Statement (originally posed by Börstler)

This application will support the operations of a technical library for a university department. This includes the searching for and lending of technical library materials, including books, videos, and technical journals. All library items have registration code (research area code + running number). Each borrower can borrow up to 10 items. Each type of library item can be borrowed for a different period of time (books 6 weeks, journals 3 days, videos 1 week). If returned after their due date, the employee will be charged a fine, based on the type of item (books 5 cents a day, journals and videos 20 cents a day). Materials will be lent to employees only if they have (1) no overdue lendables, (2) fewer than 10 articles out, and (3) total fines less than \$10.

THE LIBRARY PROGRAM

Part II CRC Cards



What are CRC Cards?

CRC stands for **C**lass, **R**esponsibilities and **C**ollaborators. A CRC-card is a standard index card that has been divided into regions. Larger index cards work best.

CLASS

RESPONSIBILITIES

COLLABORATORS

Elements of CRC Cards

By Jürgen Börstler http://www.cs.umu.se/kurser/TDBA62/CRC_UMINF04.04.pdf

Class; a CRC-card corresponds to a class, i.e. a collection of objects of the same kind. Objects are things of interest in the problem/application domain. An object can be a person, place, thing, event, or concept. The class name is written across the top of the CRC-card. A class should have a single and well-defined purpose that can be described briefly and clearly. It should be named by a noun, noun phrase, or adjective that adequately describes the abstraction. A short description of the purpose of the class is written on the back of the card.

Responsibilities; a responsibility is something a class takes care of; a service the objects of a class provide for other objects. A responsibility can be either to know something or to do something. A book in a library for example should know its title and whether it is out on loan or not (among others). To do something the class usually uses the knowledge it has. If this knowledge is not sufficient for the purpose, the class can get help from collaborators (see below). The responsibilities of a class are written along the left side of the card.

Collaborators; a collaborator is another class "helping" to fulfill a specific responsibility. A collaborator can for example provide further information, or actually take over parts of the original responsibility (by means of the collaborators own responsibilities). A book can for example only know whether it is overdue or not, if it also knows the current date (in Figure 2 this information is provided by the collaborator Date). Collaborators are written along the right side of the card in the same row as the corresponding responsibility.

The back of the card can be used for the class description, comments and other details.

Why use CRC cards?

By Barbara Wells (<http://cs.colgate.edu/APCS/Java/CRCCards/CRCcards.htm>)

- They are portable... No computers are required so they can be used anywhere. Even away from the office or classroom.
- They allow the participants to experience first hand how the system will work. No computer tool can replace the interaction that happens by physically picking up the cards and playing the role of that object.
- They are a useful tool for teaching people the object-oriented paradigm.
- They can be used as a methodology them selves or as a front end to a more formal methodology such as UML.

Procedure of Using CRC cards in the Library Problem

1. brainstorm possible candidates for classes;
2. filter the list of candidates;
3. create CRC-cards for the remaining candidates;
4. allocate responsibilities to CRC-cards/classes;
5. sketch a CRC Interaction diagram

Sample CRC Cards for Library Problem

CLASS: Book

RESPONSIBILITIES

Knows title
Knows isbn
Knows if on-loan
Knows return date
Knows number of days late

checkBookOut()
checkBookIn()

COLLABORATORS

Date
Date

CLASS: Borrower

RESPONSIBILITIES

Knows name
Knows max loans allowed
Knows max fines allowed
Knows overdue fines
Knows borrowed books

borrowBook()
returnBook()
payFine()

COLLABORATORS

Book
Book

CLASS: Librarian

RESPONSIBILITIES

addBorrower()
addBook()
searchCatalog()
checkOut()
checkIn()
payFine()

COLLABORATORS

Library & Borrower
Library & Book
Library
Book & Borrower
Book & Borrower
Borrower

CLASS: Library

RESPONSIBILITIES

knows catalog of books
knows catalog of borrowers

inCollection()

COLLABORATORS

Book

CLASS: Date

RESPONSIBILITIES

COLLABORATORS

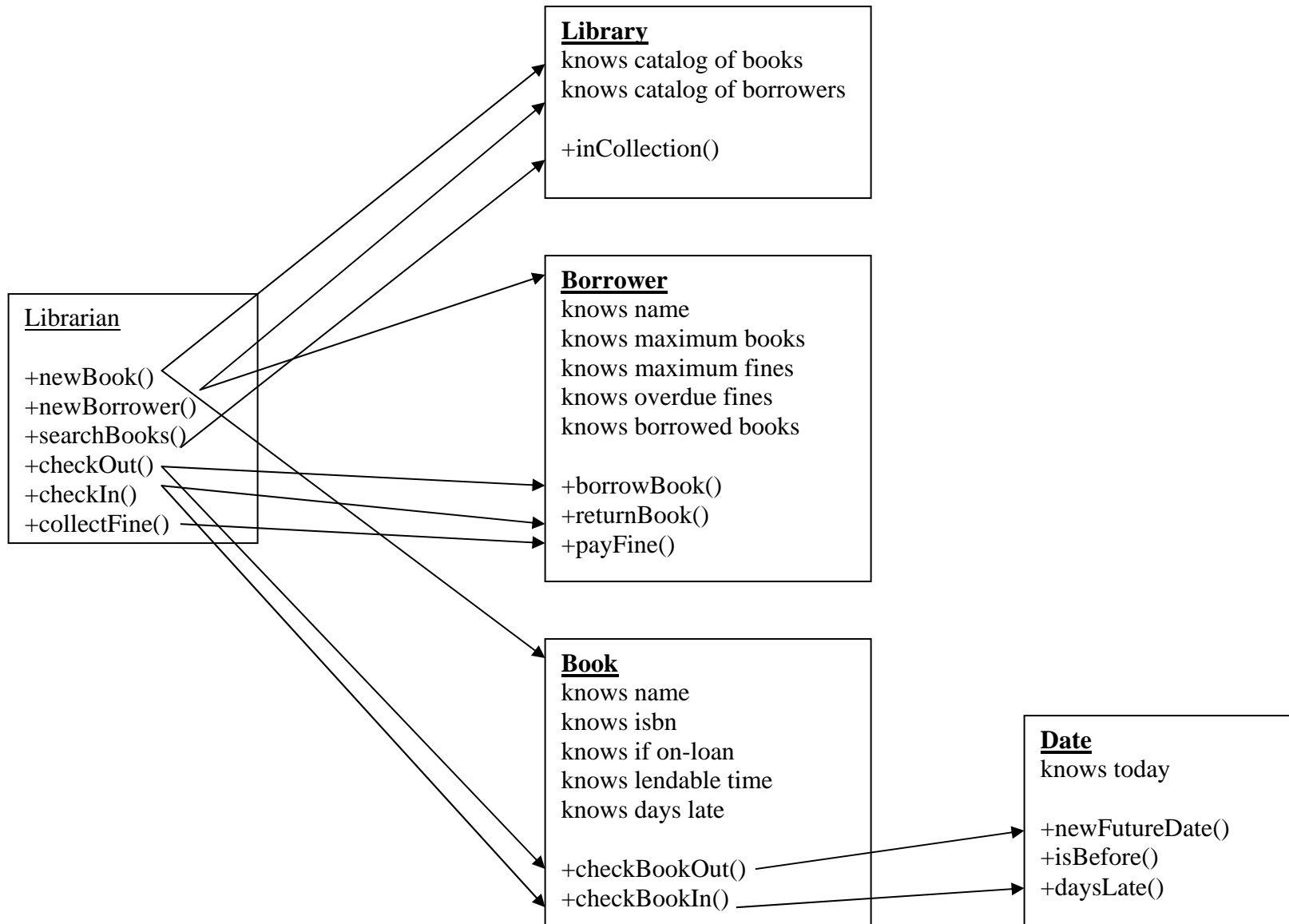
Knows today's date

isBefore()

daysLate()

newFutureDate (int numDaysFromNow)

The Library Problem CRC Interaction Diagram





THE LIBRARY PROBLEM

Part III

Testing Scenarios Through Role Playing

Why Scenarios?

- ❖ Scenarios help move students from the design element of creating CRC cards to the process of analyzing the usability of the code and making necessary modifications to design before writing code. These scenarios can be used as the main program when coding this problem to test accuracy of code.
- ❖ Valuable to have student groups or whole-class group creation of scenarios so as to include the multicultural interests and ideas of students (in this case, books, student names, etc.)
- ❖ Important step for refining design. CRC cards should be updated based on findings.

Elements of Effective Scenarios

1. Enough of each class type to run through simulation and test cases, but not more than necessary. For example, there is no need to create 10 different borrowers in the simple library problem case.
2. Situations that will make use of all methods for the different classes.
3. Situations that will test exceptions or boundary cases (i.e. return books on due date shouldn't create a fine, but the following day should).

Potential Scenario

Add three new borrowers to the library: Sara, Maria, and Jamal. All three are students and are allowed to borrow up to 4 books at a time and have a maximum fine allowance of \$10. Add following books to the library, each with a checkout time of 10 days.

Romeo and Juliet (isbn 1111)

Beloved (isbn 2222)

House Made of Dawn (isbn 3333)

A Hundred Years of Solitude (isbn 4444)

Farewell to Manzanar (isbn 5555)

Old Man and the Sea (6666)

Sara checks out *Romeo and Julie* and *Beloved* on December 2nd. Jamal checks out *A Hundred Years of Solitude* the same day. On December 9th, Sara returns *Beloved*. Jamal returns *A hundred Years of Solitude* on Demember 12th. Maria checks out *Beloved* on December 15th. On December 18th, Sara checks out *House Made of Dawn*. Jamal checks out *Farewell to Manzanar* on December 20th. On December 21st, Sara tries to check out *Old Man and the Sea*. The next day, *she returns Romeo and Juliet* and pays her current fine, then checks out *Old Man and the Sea*.

The Cast of Characters for the Library Problem Role-Play

- ❖ Main program (follows scenario)
- ❖ Library
- ❖ Librarian
- ❖ Date
- ❖ Borrower (one for each borrower defined in scenario)
- ❖ Book (one for each book defined in the scenario)
- ❖ Recorder – someone to record the interactions of the script

*If there are leftover students, extra students can serve as “understudies” for more complicated roles or for students who might need a little help.

Materials

- ❖ Nametags
- ❖ Paper for private data

Day 1: Each student or pair of students should create a script for their character, including lines for being constructed, lines for each method, and an area for listing appropriate private data. A copy should go to the teacher.

Day 2: Class role-play.

1. For each defined object in the role play (i.e., Borrower Jamal) create a name tag, labeled “Hi, My Name is Jamal, and I’m a Borrower). This helps ensure that when objects are referred to by other classes, they call the name (Jamal), rather than the object type (Borrower).
2. Each object should have a piece of paper (or chalkboard space covered by paper) with private data written on it. Though this should theoretically remain out of view for to emphasize the private nature of this data, but you should encourage students to give a peek to the class when being called on so other students can follow their procedure.
3. Each part must enact its methods out loud, even when returning no value. So, when *Beloved* is created, he should say, “My name is *Beloved* and I’m a Book. My isbn number is 2222, and I can be checked out for 10 days.”

When being checked out, the schoolLibrary should tell *Beloved*, “*Beloved*, check yourself out.” *Beloved* should say, “Hmm, I am now checked out. myNewDueDate, if I can be checked-out for 10 days, what is my new due date?”

4. Teacher should act as director and keep the role play moving, helping students or refining scripts as necessary.
5. Recorder should keep track of interactions between objects for later discussion.
6. Follow-up is critical! Ask for questions, and if necessary, create new scenario and rerun script for further clarification.



THE LIBRARY PROBLEM

Part IV *From Design to Java*

- ❖ After the role playing activity, students should now have worked out any problems that arose with their design. The final part of this design-based approach to the Library Problem is to actually create the code. This is the first time students are using the computer to solve this problem.
- ❖ This approach allows students to focus on the implementation of their design using the Java language, rather than trying to design and code simultaneously. This helps make any syntax questions more transparent.
- ❖ This activity makes for a good partner or small-group coding activity.

After groups of students create the code, the class should talk about how each groups' actual code is slightly different, despite the collaborative design process. Important to talk about how unique designing and coding style makes everyone's approach to CS a little different (like writing essays!)

Sample Java Code

```
public class Book
{
    private String myName ;
    private int myIsbn;
    private int myRentalTime;
    private Boolean myOnLoan;
    private int myDaysLate;
    private Date myDueDate;

    public Book(String name, int isbn, int rentalTime)
    {
        myName = name;
        myIsbn = isbn;
        myOnLoan = 0;
        myRentalTime = rentalTime;
        myDaysLate = 0;
    }

    public String getName()
    {
        return myName;
    }

    public int getISBN()
    {
        return myISBN;
    }

    public Boolean isOnLoan()
    {
        return myOnLoan;
    }

    public int getRentalTime()
    {
        return myRentalTime;
    }

    public int daysLate()
    {
        if (myOnLoan == 1) && (myDueDate.isBefore() == 0)
            return myDueDate.daysLate();
        else return 0;
    }

    public void checkBookOut()
    {
        myOnLoan = 1;
        myDueDate = newFutureDate(int myRentalTime)
    }

    public void checkBookIn()
    {
        myOnLoan = 0;
        myDaysLate = 0;
    }
}
```

```

public class Borrower
{
    private String myName ;
    private int myMaxLoans;
    private int myMaxFines;
    private int myFines;
    private int myLoans;

    private ArrayList myBorrowedBooks;

    public Borrower(String name, int maxFines, int maxLoans)
    {
        myName = name;
        myMaxFines = maxFines;
        myMaxLoans = maxLoans;
        myFines = 0;
        myLoans = 0;
        private ArrayList myBorrowedBooks = new ArrayList();
    }

    public String getName()
    {
        return myName;
    }

    public int getFines()
    {
        return myFines;
    }

    public int getNumLoans()
    {
        return myLoans;
    }

    public void addFines(int daysLate)
    {
        myFines += daysLate*.05;
    }

    public int isAllowedtoBorrow()
    {
        if (myFines < myMaxFines) && (myBorrowedBooks.size() < myMaxLoans)
            return 1;
        else return 0;
    }

    public void borrowItem (Book newBook)
    {
        myBorrowedBooks.add(newBook);
    }

    public void returnItem(Book returnBook)
    {
        for (index=0; index<myBorrowedBooks.size(); index++)
        {
            if ((myBorrowedBooks.get(index)).getISBN() == returnBook.getISBN())
                myBorrowedBooks.remove(index);
        }
    }

    public void payFines (int amountPaid)
    {
        myFines -= amountPaid;
    }
}

```

```

public class Librarian
{
    private Library myLibrary;

    public Librarian(){ }

    public void newBook (string name, int isbn)
    {
        myNewestBook = Book(name, isbn, 4);
        myLibrary.addBook(myNewestBook);
    }

    public void newBorrower (string name, int maxFines, int maxLoans)
    {
        myNewestBorrower = Borrower(name, maxFines, maxLoans);
        myLibrary.addBorrower(myNewestBorrower);
    }

    public Book searchCollection (Book searchBook)
    {
        return myLibrary.inCollection (searchBook);
    }

    public void checkOut (Book outBook, Borrower outBorrower)
    (
        if (outBorrowerer.isAllowedToBorrow() == 1) && (outBook.isOnLoan() == 0)
        (
            outBook.checkOut();
            outBorrower.borrowItem(Book);
        }
    )
}

public void checkIn (Book inBook, Borrower inBorrower)
{
    if (inBook.daysLate() > 0)
        inBorrower.addFine(inBook.myDaysLate());
    inBorrower.returnItem(inBook);
    inBook.bookIn();
}

public void collectFine (Borrower finePayer, int amountPaid)
{
    finePayer.payFines(amountPaid);
}
}

```

```

public class Library
{
    private ArrayList myCollection;
    private ArrayList myBorrowers;

    public Library ()
    {
        private ArrayList myCollection = new ArrayList();
        private ArrayList myBorrowers = new ArrayList();
    }

    public void addBook(Book newBook)
    {
        myCollection.add(newBook);
    }

    public void addBorrower(Book newBorrower)
    {
        myBorrowers.add(newBorrower);
    }

    public int collectionSize()
    {
        return myCollection.size;
    }

    public int borrowerSize()
    {
        return myBorrowers.size;
    }

    public Boolean inCollection (Book requestedBook)
    {
        int index = 0;
        do
        {
            if ((myCollection.get(index).getISBN() == requestedBook.getISBN())
                return 1;
            index++;
        }
        while (index < myCollection.size());
        return 0;
    }
}

```



THE LIBRARY PROBLEM

Part V *Individual Student Project*

After the *Library Problem* example, you should be more familiar with the paradigm of designing applications to simulate small systems we encounter in our daily lives. This assignment requires you to identify a small system of interacting objects, pose a problem statement, create CRC cards, draw a CRC interaction diagram, create a scenario, and write the actual code to simulate the system. A page of written explanation should summarize your design methodology.

Ideas to start thinking about...

Board games
An elevator
Sports
Social interactions

Project Approval

Before a week from today, you must turn in a few sentences describing your project idea. Once I approve your idea, you may begin your project. If you have any questions, this is a good time for you to ask for help!

Due date: One month from now (ideal winter break assignment)

This assignment will have a large impact on your course grade, as it counts as **3** homework assignments.