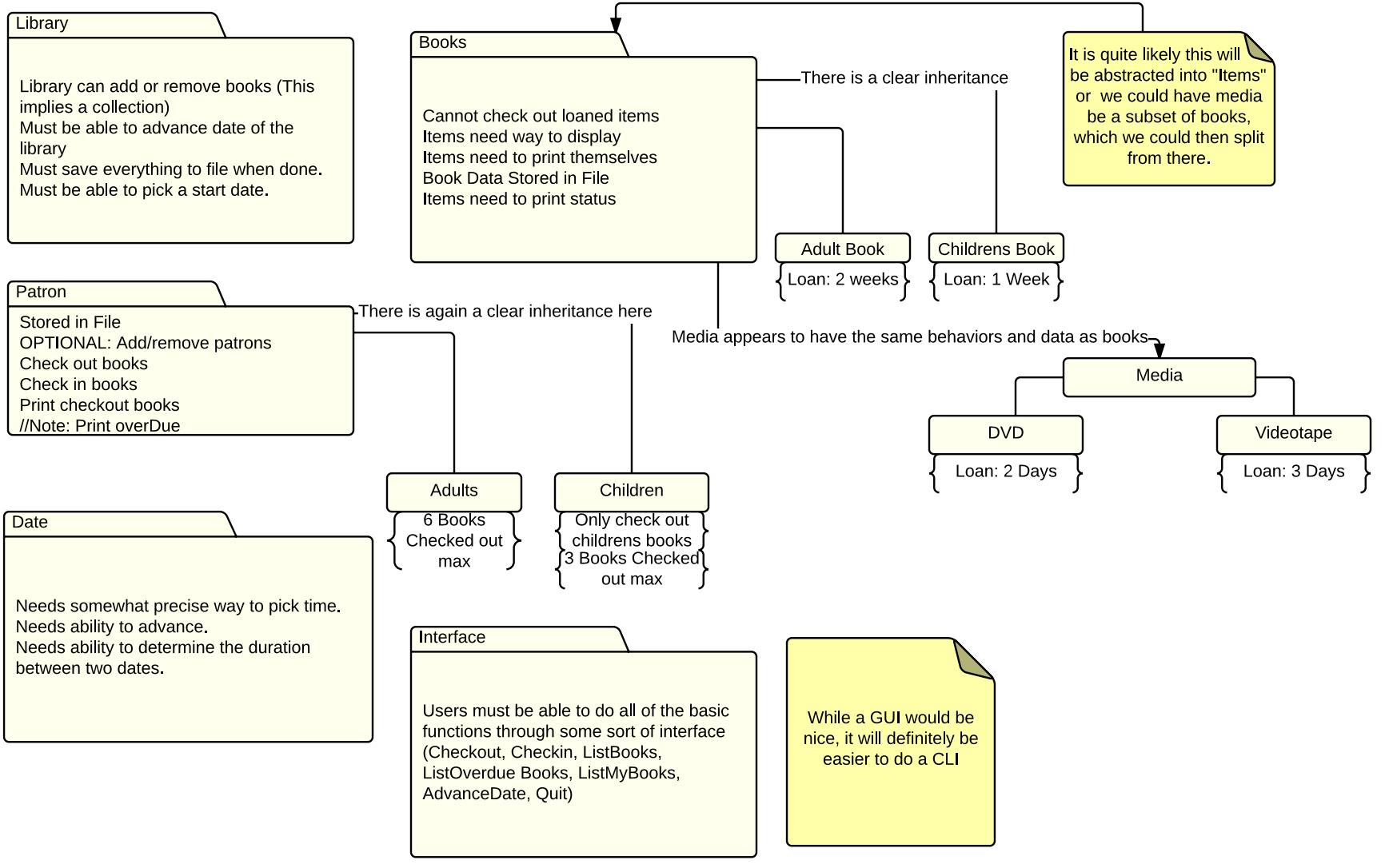

Library
Library can add or remove books (This implies a collection)
Must be able to advance date of the library
Must save everything to file when done.
Must be able to pick a start date.
Books
Cannot check out loaned items
Items need way to display
Items need to print themselves
Book Data Stored in File
Items need to print status
There is a clear inheritance
It is quite likely this will be abstracted into "Items" or we could have media be a subset of books, which we could then split from there.
Adult Book
Loan: 2 weeks
Childrens Book
Loan: 1 Week
Patron
Stored in File
OPTIONAL: Add/remove patrons
Check out books
Check in books
Print checkout books
//Note: Print overDue
There is again a clear inheritance here
Media appears to have the same behaviors and data as books
Media
DVD
Loan: 2 Days
Videotape
Loan: 3 Days
Adults
6 Books Checked out max
Children
Only check out childrens books
3 Books Checked out max
Date
Needs somewhat precise way to pick time.
Needs ability to advance.
Needs ability to determine the duration between two dates.
Interface
Users must be able to do all of the basic functions through some sort of interface (Checkout, Checkin, ListBooks, ListOverdue Books, ListMyBooks, AdvanceDate, Quit)
While a GUI would be nice, it will definitely be easier to do a CLI

Check out a book

Functional

Main Success Scenario:

1. Patron browses available titles from library
2. Patron selects book(s) that they want to check out
3. The library checks to see if the patron is eligible to check out the book
4. The library removes the book from available books and gives it to the patron

Extensions:

1a. There are no available books
    .1: The patron sees no books
3a. The patron is not eligible to check out the book
    .1: The library does not remove the book
    .2: The library notifies the patron

Check in a book

Functional

Main Success Scenario:

1. The patron browses which books are available to check in
2. The patron selects the book(s) that they wish to return
3. The patron gives the books to the library
4. The library adds this/these book(s) back to the available books.

Extensions:

1a. There are no books checked out by the patron
    .1: The patron is notified that they have no checked out

List all books

Functional

Main Success Scenario:

1. The library goes through its catalogue
2. Each book is then asked to print its status

Extensions:

1a. The library has no books
    .1: The library lists nothing

List overdue books

Functional

Main Success Scenario:

1. The library goes to each patron
2. The patron lets the library know which books they have are overdue.
3. The library lists the resulting collection

Extensions:

1a. There are no patrons
    .1: There must be no books checked out
    .2: Nothing is listed
2a. The patron has no overdue books
    .1: The patron says nothing

List a patron's books

Functional

Main Success Scenario:

1. Patron wishes to list their books
2. Patron goes through the books they have checked out and overdue.
3. The patron then lists these

Extensions:

2a. The patron has no books
    .1: Nothing is listed.

Advance the date

Functional

Main Success Scenario:

1. Library tells the date to advance
2. The date advances
3. Books are notified of the change
4. Books determine if they are now overdue

Add patrons to library

Functional

Main Success Scenario:

1. User enters information
2. Library checks to see if patron already exists
3. Library adds the user

Extensions:

1a. User enters invalid information
    .1: The driver/library informs the user
2a. The patron already exists
    .1: The library informs the user
    .2: Nothing is added

Remove patrons from library

Functional

Main Success Scenario:

1. User enters information that identifies patron
2. The library checks to see if the patron exists
3. The library asks for confirmation to remove patron
4. The library removes the patron
5. The patron returns all books that it owns

Extensions:

1a. The user enters invalid information
    .1: The driver/library informs the user
2a. The patron does not exist
    .1: The library informs the user
    .2: Nothing is done
3a. The user changes their mind
    .1: Nothing is done
5a. The patron has no books checked out
    .1: Patron is deleted without a fuss

Save to file

Functional

Main Success Scenario:

1. User specifies file to save to
2. Library saves its state.
3. Objects contained within the library save their own state.
4. User is informed of success
5. User is returned to menu

Extensions:

1a. The file is invalid
    .1: User is informed
    .2: User is brought back to menu
1b. The file already exists
    .1: User is asked to proceed
    .2a: The user declines. Abandon operation.
    .2b: The user accepts. Operation continues.
2/3a. Object encounters error saving state.
    .1: The user is informed of exception
    .2: The user is returned to menu

Load from file

Functional

Main Success Scenario:

1. User specifies file to load from
2. Library loads its own state
3. All of the data items contained in the library loads its own state
4. User is informed of success
5. User is presented with menu

Extensions:

1a. File is invalid
    .1: User is informed.
    .2: User can try again
2/3a: Library/Object encounters error loading state
    .1: User is informed
    .2: Operation is abandoned

Add book to library

Functional

Main Success Scenario:

1. User inputs information for new book
2. Library adds book to available books
3. User is informed of success

Extensions:

1a. User input is invalid
    .1: User is informed of bad input.
    .2: User is brought back to menu

Remove book from library

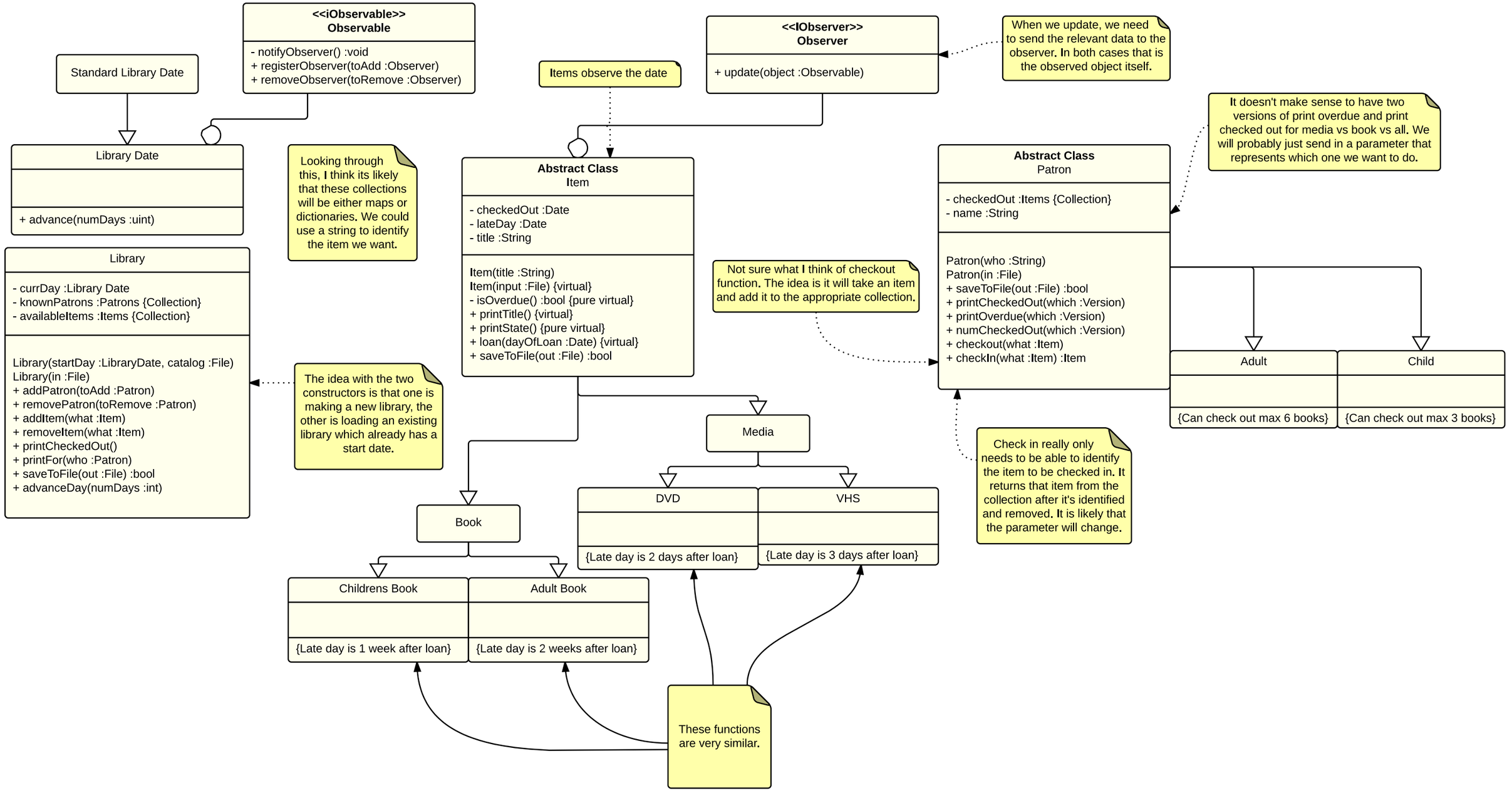
Functional

Main Success Scenario:

1. User browses books to remove
2. User selects book to remove
3. Library checks if the book is currently checked out
4. Library removes book from books
5. User is informed of success

Extensions:

1a. There are no books
    .1: Nothing is printed
3a. Book is currently checked out
    .1: Patron which owns book returns it


Standard Library Date
<<iObservable>>
Observable
- notifyObserver() :void
+ registerObserver(toAdd :Observer)
+ removeObserver(toRemove :Observer)
Items observe the date
<<IObserver>>
Observer
+ update(object :Observable)
When we update, we need to send the relevant data to the observer. In both cases that is the observed object itself.
Library Date
+ advance(numDays :uint)
Looking through this, I think its likely that these collections will be either maps or dictionaries. We could use a string to identify the item we want.
Abstract Class
Item
- checkedOut :Date
- lateDay :Date
- title :String
Item(title :String)
Item(input :File) {virtual}
- isOverdue() :bool {pure virtual}
+ printTitle() {virtual}
+ printState() {pure virtual}
+ loan(dayOfLoan :Date) {virtual}
+ saveToFile(out :File) :bool
It doesn't make sense to have two versions of print overdue and print checked out for media vs book vs all. We will probably just send in a parameter that represents which one we want to do.
Abstract Class
Patron
- checkedOut :Items {Collection}
- name :String
Patron(who :String)
Patron(in :File)
+ saveToFile(out :File) :bool
+ printCheckedOut(which :Version)
+ printOverdue(which :Version)
+ numCheckedOut(which :Version)
+ checkout(what :Item)
+ checkIn(what :Item) :Item
Library
- currDay :Library Date
- knownPatrons :Patrons {Collection}
- availableItems :Items {Collection}
Library(startDay :LibraryDate, catalog :File)
Library(in :File)
+ addPatron(toAdd :Patron)
+ removePatron(toRemove :Patron)
+ addItem(what :Item)
+ removeItem(what :Item)
+ printCheckedOut()
+ printFor(who :Patron)
+ saveToFile(out :File) :bool
+ advanceDay(numDays :int)
The idea with the two constructors is that one is making a new library, the other is loading an existing library which already has a start date.
Not sure what I think of checkout function. The idea is it will take an item and add it to the appropriate collection.
Adult
{Can check out max 6 books}
Child
{Can check out max 3 books}
Media
DVD
{Late day is 2 days after loan}
VHS
{Late day is 3 days after loan}
Book
Childrens Book
{Late day is 1 week after loan}
Adult Book
{Late day is 2 weeks after loan}
Check in really only needs to be able to identify the item to be checked in. It returns that item from the collection after it's identified and removed. It is likely that the parameter will change.
These functions are very similar.