| CSC 207 | Grinnell College | Fall, 2014 |
| --- | --- | --- |
|  |
| Algorithms and Object-Oriented Design | | |
|  |

# Example with Class and Interface Inheritance

## Summary

This laboratory provides practice in extending one class while implementing an interface.

## Acknowledgment

This lab has been motivated by Exercise 4.51 in the book *Data Structures and Problem solving Using* [*Java*](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/lab-library-books.shtml#)*, Fourth Edition* by Mark Allen Weiss. However, elements of this lab involving Java's Comparable interface and sorting by call number are not part of Weiss' exercise, and other parts of Weiss' original questions have been changed.

## Introduction

This lab considers a class hierarchy that models books in a library. Although the model is streamlined to aid implementation for this lab, the approach could be expanded and refined to yield an extensive [application](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/lab-library-books.shtml#).

The main elements of the framework for books is shown in the following.



The following comments provide a high-level overview of these classes.

* The Book class models information common to all books.
* The LibraryBook class is a Book, but library books also have a call number, and the call number allows books to be ordered on the shelves. (The ordering is given by a compareTo specified in the Comparable interface. In addition, because many library books can circulate (be checked out and returned), each library book has a circulation status (e.g., on shelf, checked out, permanently in reference collection).
* A ReferenceBook is a type of Library Book that is housed in a specified collection (e.g., a reference area in Burling, in Science, or in the Iowa Room archive), and reference books do not circulate.
* A CirculatingBook is located on the shelves until it is checked out, and the book is placed back on the shelves when it is returned.

## The Book class and its subclasses

Documentation for the functionality of these classes is given in [the javadoc documentation](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/). A more detailed description of these classes follows.

### Class Book

[Class Book](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/Book.html) models information common to all books. Any book has

* fields
  + an author
  + a title
  + an ISBN or International Standard Book Number that provides a unique number used by publishers and book stores.
* constructors and methods
  + a null constructor
  + a constructor using 3 parameters for an author, title, and ISBN number
  + getters and setters
  + toString should provide a string of the field data in a format suitable for printing

Within a Book class and its subclasses, these fields might be used directly, but processing by other classes and objects should be done via getters and setters.

### Class LibraryBook

Class LibraryBook models information common for library books. In addition to author, title, and ISBN, library books have call numbers, and library books are stored on the shelves in [order](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/lab-library-books.shtml#) by call number. Further, many library books may be able to circulate. In addition to fields and methods of books, any library book has

* fields
  + call number
* constructors and methods
  + a constructor using 4 parameters for an author, title, ISBN number, and call number
  + getters and setters
  + checkout handles processing for a patron to check out a book. A due date also is recorded.
  + returned handles processing for when a book is returned after having been checked out.
  + circulationStatus indicates whether the book is on the shelves, checked, or non-circulating in the reference collection.
  + compareTo allows comparison/ordering of library books, following the format of  [Java's Comparable interface](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html)
  + toString from Book is augmented with a circulation status and call number

Since checkout, returned, and circulationStatus depend upon subclasses (i.e., whether book is circulating or in the reference collection), these methods are *abstract* � identified but not implemented.

Although library books can be ordered by call number, these call numbers are not present in the Book. Thus, LibraryBook can implement  [Java's Comparable interface](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html), but Book cannot.

Since some methods are defined but not implemented, **LibraryBook is an abstract class**.

### Class ReferenceBook

[Class ReferenceBook](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/ReferenceBook.html) models a reference book. Reference books are housed in a specific collection (e.g., the Iowa Room, Burling, or Science), and reference books do not circulate.

In addition to fields and methods of library books, a reference book has

* fields
  + collection
* constructors and methods
  + a constructor using 5 parameters for an author, title, ISBN number, call number, and collection
  + getters and setters
  + checkout is not allowed for reference books, so the method should print that the patron "cannot check out a reference book".
  + returned should print "reference book could not have been checked out -- return impossible".
  + circulationStatus should return "non-circulating reference book".
  + toString from LibraryBook should be augmented with the collection information

With checkout, returned, and circulationStatus implemented for LibraryBook, this is a fully-implemented class.

### Class CirculatingBook

[Class CirculatingBook](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/CirculatingBook.html) models books that can circulate. When checked out, information is stored about the patron who borrowed the book and when it is due. In addition to fields and methodds of library books, a circulating book has

* fields
  + currentHolder (patron, if any, who checked out the book)
  + dueDate (when book should be returned)
* constructors and methods
  + a constructor using 4 parameters for an author, title, ISBN number, and call number. When constructed, currentHolder and dueDate should be null.
  + getters and setters
  + checkout handles processing when a book is checked out, storing the patron's name and the date the book is due to be returned.
  + returned handles process when a book is returned (currentHolder and dueDate are set to null).
  + circulationStatus should return the currentHolder's name and due date, if the book has been checked out; or "book available on shelves" if the book is available.
  + toString from LibraryBook should be augmented with current holder and due date, if applicable.

Although production-level [software](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/lab-library-books.shtml#) would include substantial error checking, the implementation for this lab may involved minimal [error](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/lab-library-books.shtml#) checking. That is, it is not necessary to check that the book has not been checked out when calling checkout or that the book has been checked out when calling returned.

### Consequences of Interface Comparable

Many Java classes rely upon  [Java's Comparable interface](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html). For example, the  [Collections class](http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html) contains methods to sort and search an ArrayList of Comparables. Thus, by defining a compareTo method, applications involving the LibraryBook class can use many operations with *no* additional work!

## Work for this Lab

Implement the classes identified in this narrative, following the above notes and the relevant [javadoc](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/) docuentation.

* [Class Book](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/Book.html)
* abstract class LibraryBook, including
  + String field callNumber
  + getters and setters
  + abstract methods with these signatures:  
     abstract String circulationStatus();  
     abstract void checkout (String patron, String dueDate);  
     abstract void returned ();
  + implemented constructor  
    LibraryBook (String au, String ti, String num, String callNum)
  + implemented methods  
     /\*\*   
     \* implementation of Comparable's compareTo method  
     \* @param lib: Library object being compared  
     \* @return 0 if call numbers of this and lib match  
     \* < 0 if call number of this comes before call number of lib  
     \* > 0 otherwise  
     \*/  
     public int compareTo (LibraryBook lib)   
       
     /\*\*  
     \* @return title, author, ISBN, call number as a String for printing  
     \*/  
     public String toString ()   
    Note that toString can call abstract method circulationStatus, although the details of circulationStatus will be supplied in subclasses.
* [Class ReferenceBook](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/ReferenceBook.html)
* [Class CirculatingBook](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/CirculatingBook.html)

Although this lab requires the implementation of four classes, each class can be reasonably short. For example, in preparing this lab, I implemented and documented these four classes, and my notes are in the following table.

| Class/Interface | Time to Create Code | Time to Document with javadoc,  other comments | Comments |
| --- | --- | --- | --- |
| Class Book | 4 minutes | 3 minutes | create package, class, fields, constructor, toString, all getters and setters  to create getters and setters in Eclipse, go to Source tab and click "Generate Getters and Setters" |
| Interface Comparable | 0 minutes | 0 minutes | Predefined at  [Java's Comparable interface](http://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html) |
| abstract Class LibraryBook | 5 minutes | 3 minutes | create class with call number field, constructor, getters and setters, circulationStatus, compareTo, toString  the abstract method circulationStatus will be set in subclasses, but it can be used in the toString method |
| Class ReferenceBook | 10 minutes | 3 minutes | create class with collection field, constructor (mostly calls super, getters and setters, specify circulation status is always "non-circulating reference book", add collection string to toString |
| Class CirculatingBook | 8 minutes | 3 minutes | create class with currentHolder and dueDate fields, getters and setters, provide for book checkout and bookReturned, clarify circulationStatus based on whether or not book is checked out |
| Class Library | 60 minutes | 15 minutes | create class, test data, check searching and sorting  most development time devoted to writing tests  [shell for this class supplied](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/Library.java), so your testing can begin with some test cases already identified |

## Testing

As the table shows, the book-related classes can be implemented in a reasonable amount of time. However, I got carried away in putting together test cases, so my implementation of a [Library](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/Library.html) testing class took over an hour. For reference, I also include [output from running Library.java.](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/library-output)

For the Library testing class, work on main took a substantial amount of time. The other capabilities for a library, however, went quite quickly.

Although [the full Library class](http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/library-javadoc/Library.java) is given to allow the start of testing for this lab, be sure you understand how this code works � tests will assume knowledge of this material.

## Work to Turn in

* Code for Book, LibraryBook, ReferenceBook, CirculatingBook
* Enhancements made to expanding testing for the Library class
* A printout of the output produced by Library

This document is available on the World Wide Web as

http://www.cs.grinnell.edu/~walker/courses/207.fa14/labs/lab-libary-books.shtml

| *created 15 February 2012*  *revised 23 February 2012*  *revised for Java 7 30 September 2014* |  |
| --- | --- |
| For more information, please contact [Henry M. Walker](http://www.cs.grinnell.edu/~walker/) at [walker@cs.grinnell.edu](mailto:walker@cs.grinnell.edu). |