Design Description for a Java-Based Library Search and Sort Application

Mathew Yamasaki

University of Maryland University College

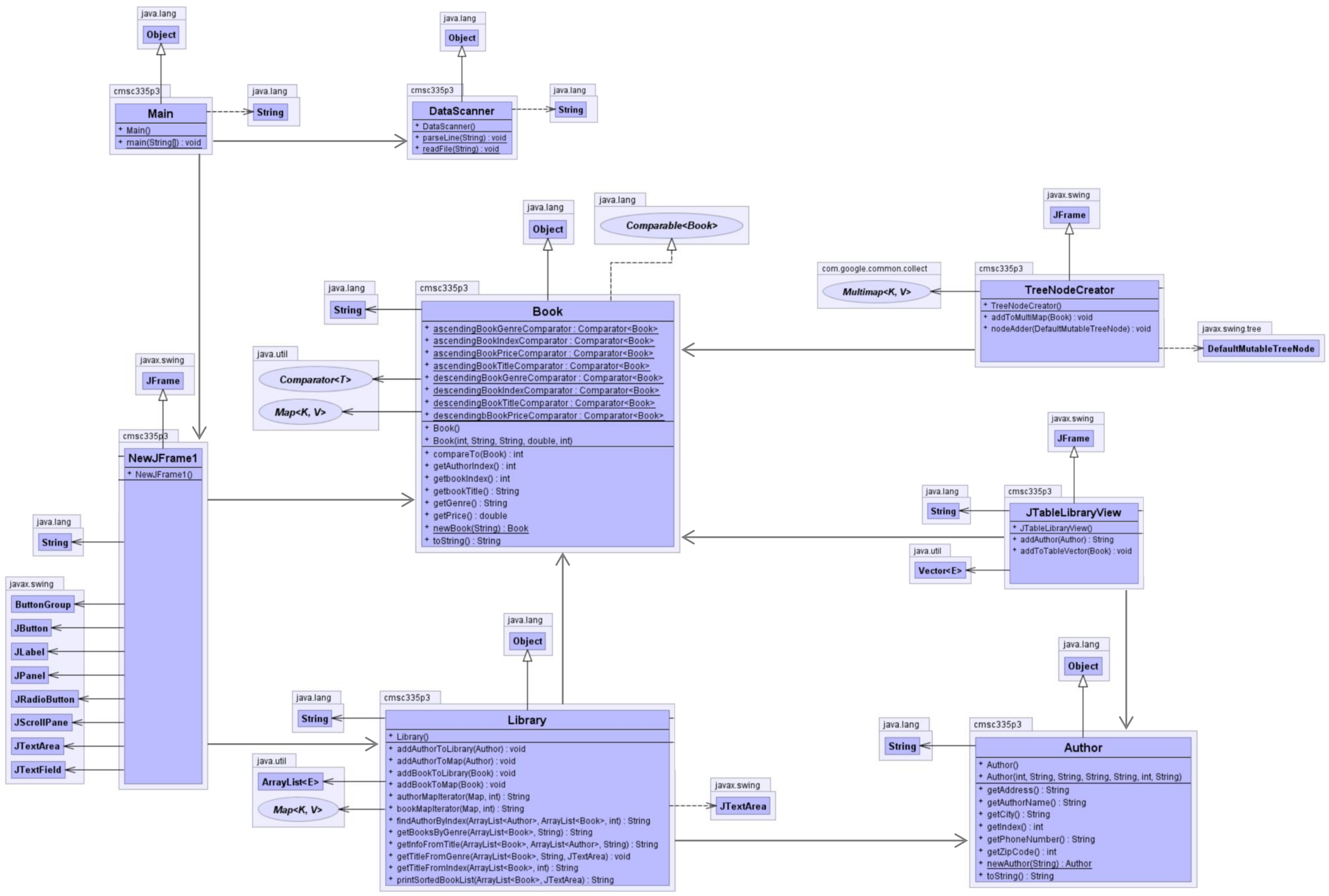

java.lang
Object
cmsc335p3
Main
+ Main()
+ main(String[]) : void
java.lang
String
java.lang
Object
cmsc335p3
DataScanner
+ DataScanner()
+ parseLine(String) : void
+ readFile(String) : void
java.lang
String
java.lang
Object
java.lang
Comparable<Book>
javax.swing
JFrame
com.google.common.collect
Multimap<K, V>
cmsc335p3
TreeNodeCreator
+ TreeNodeCreator()
+ addToMultiMap(Book) : void
+ nodeAdder(DefaultMutableTreeNode) : void
javax.swing.tree
DefaultMutableTreeNode
java.lang
String
cmsc335p3
Book
+ ascendingBookGenreComparator : Comparator<Book>
+ ascendingBookIndexComparator : Comparator<Book>
+ ascendingBookPriceComparator : Comparator<Book>
+ ascendingBookTitleComparator : Comparator<Book>
+ descendingBookGenreComparator : Comparator<Book>
+ descendingBookIndexComparator : Comparator<Book>
+ descendingBookTitleComparator : Comparator<Book>
+ descendingbBookPriceComparator : Comparator<Book>
+ Book()
+ Book(int, String, String, double, int)
+ compareTo(Book) : int
+ getAuthorIndex() : int
+ getbookIndex() : int
+ getbookTitle() : String
+ getGenre() : String
+ getPrice() : double
+ newBook(String) : Book
+ toString() : String
java.util
Comparator<T>
Map<K, V>
javax.swing
JFrame
cmsc335p3
NewJFrame1
+ NewJFrame1()
java.lang
String
javax.swing
ButtonGroup
JButton
JLabel
JPanel
JRadioButton
JScrollPane
JTextArea
JTextField
javax.swing
JFrame
java.lang
String
cmsc335p3
JTableLibraryView
+ JTableLibraryView()
+ addAuthor(Author) : String
+ addToTableVector(Book) : void
java.util
Vector<E>
java.lang
Object
java.lang
String
cmsc335p3
Library
+ Library()
+ addAuthorToLibrary(Author) : void
+ addAuthorToMap(Author) : void
+ addBookToLibrary(Book) : void
+ addBookToMap(Book) : void
+ authorMapIterator(Map, int) : String
+ bookMapIterator(Map, int) : String
+ findAuthorByIndex(ArrayList<Author>, ArrayList<Book>, int) : String
+ getBooksByGenre(ArrayList<Book>, String) : String
+ getInfoFromTitle(ArrayList<Book>, ArrayList<Author>, String) : String
+ getTitleFromGenre(ArrayList<Book>, String, JTextArea) : void
+ getTitleFromIndex(ArrayList<Book>, int) : String
+ printSortedBookList(ArrayList<Book>, JTextArea) : String
java.util
ArrayList<E>
Map<K, V>
javax.swing
JTextArea
java.lang
Object
java.lang
String
cmsc335p3
Author
+ Author()
+ Author(int, String, String, String, String, int, String)
+ getAddress() : String
+ getAuthorName() : String
+ getCity() : String
+ getIndex() : int
+ getPhoneNumber() : String
+ getZipCode() : int
+ newAuthor(String) : Author
+ toString() : String

Figure 1: The application UML diagram.

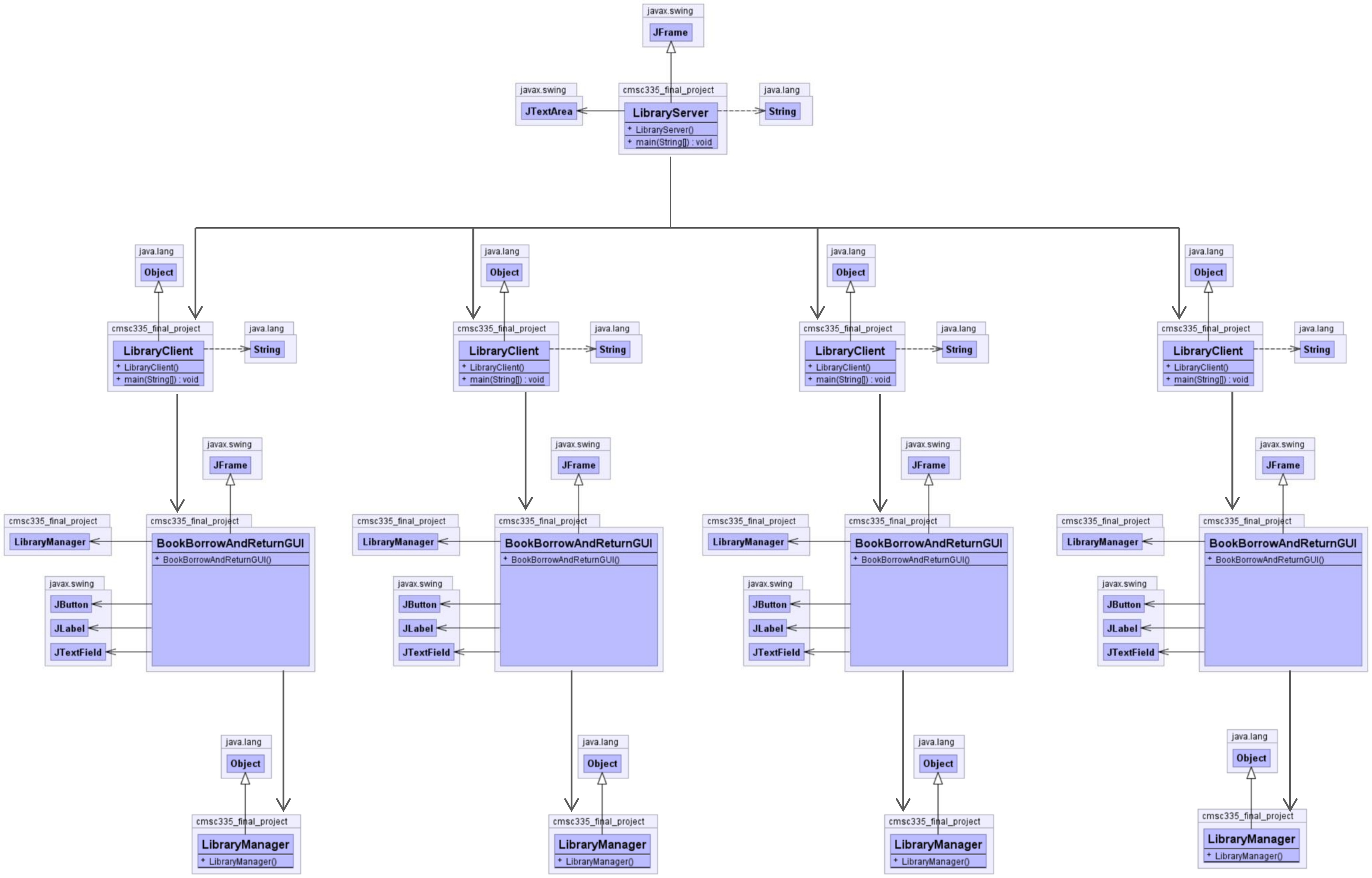


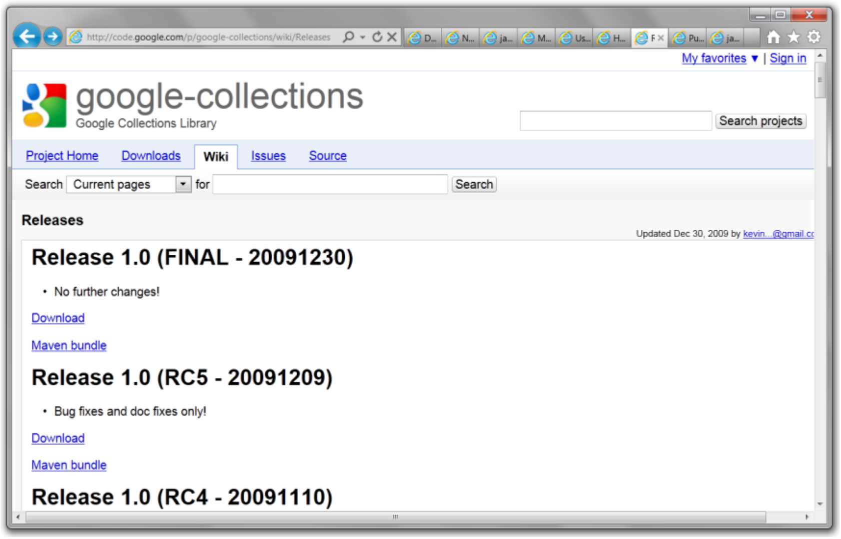
Figure 2: The application UML diagram running multiple threads.

Design Description

As shown in the application unified modeling language (UML) diagram in Figures 1 and 2, there are four major class additions to this version of the Library Search and Sort (LSS) program, dubbed version 1.4.1.

Application Setup

The Google Collection Library (GCL) `Multimap` requires downloading and adding the GCL Java archive (JAR) file to the project resources library. The following instructions use NetBeans 7.0.1 to illustrate this process. Download the GCL from the website as shown in Figure 3.



Releases

Updated Dec 30, 2009 by kevin...@gmail.c

Release 1.0 (FINAL - 20091230)

- No further changes!

Download

Maven bundle

Release 1.0 (RC5 - 20091209)

- Bug fixes and doc fixes only!

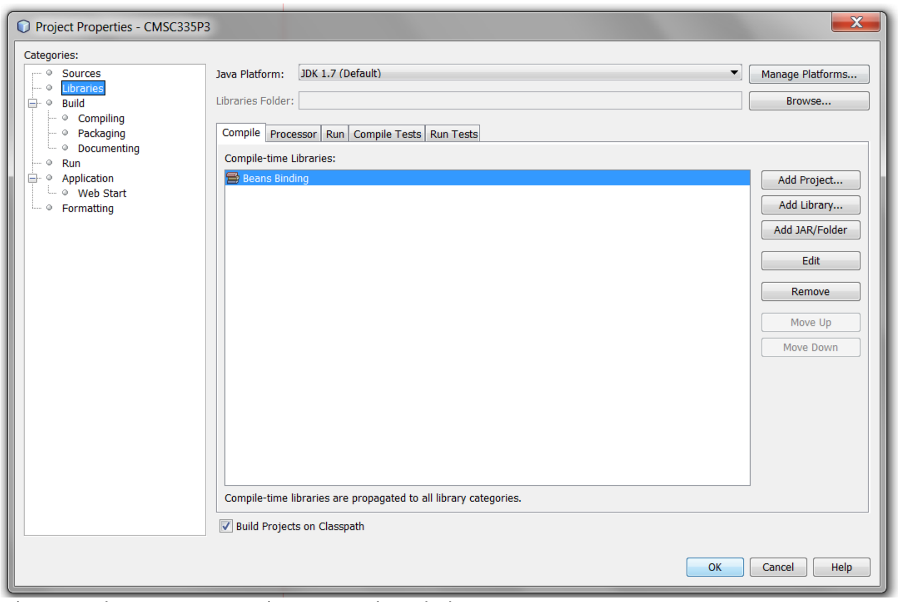
Download

Maven bundle

Release 1.0 (RC4 - 20091110)

Figure 3: Location of the Google Collections Library.

The download and extract the .zip file to a location of your choosing. In the NetBeans project viewer window, right-click on the project name folder where the source files have been saved to. Selecting “Properties” from the menu will open the “Project Properties” window. Select the “Library” node as shown in Figure 4.



Java Platform: JDK 1.7 (Default)

Libraries Folder:

Compile-time Libraries:

Beans Binding

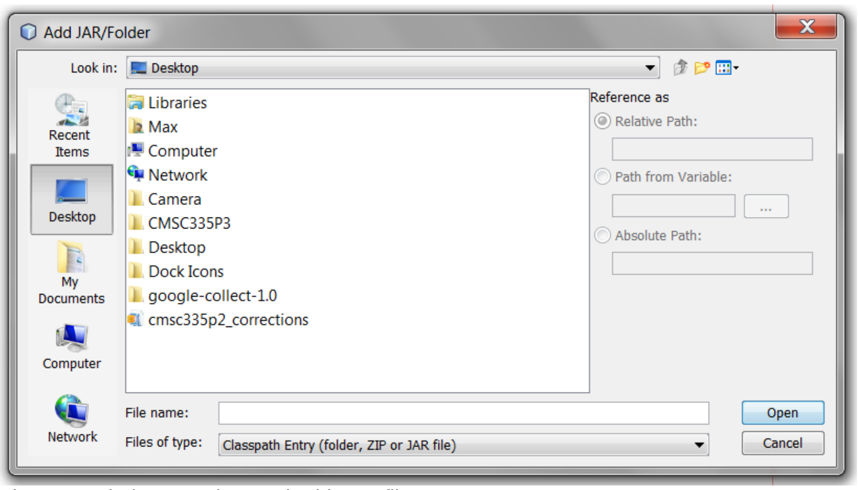
Compile-time libraries are propagated to all library categories.

[✓] Build Projects on Classpath

Figure 4: The NetBeans Project Properties window.

On the right-side menu, select “Add JAR/Folder”. The window in Figure 5 will appear.

Navigate to the location of the extracted folder named “google-collect-1.0”.



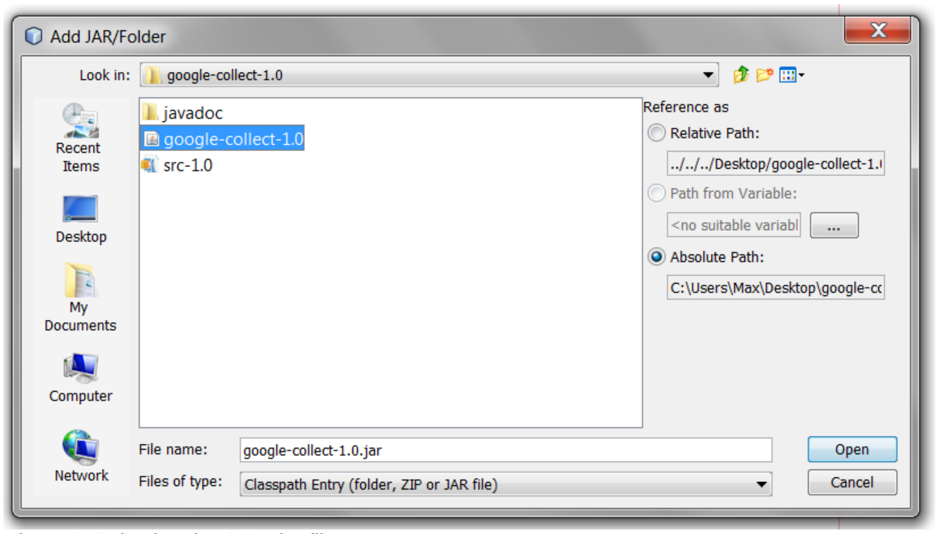
Look in: Desktop

File name:

Files of type: Classpath Entry (folder, ZIP or JAR file)

Figure 5: Window to select and add JAR files.

Open the folder and select the “google-collect-1.0.jar” file as shown in Figure 6.



Look in: google-collect-1.0

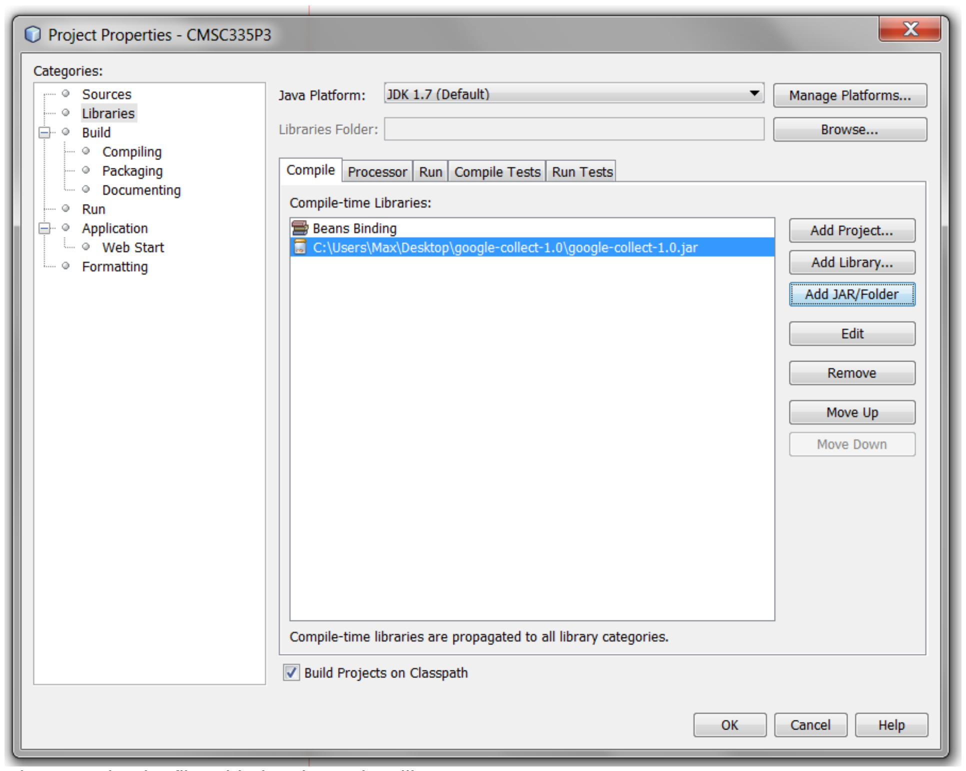
javadoc
google-collect-1.0
src-1.0

Reference as
Relative Path: ../../../Desktop/google-collect-1.
Path from Variable: <no suitable variabl
[✓ Absolute Path:] C:\Users\Max\Desktop\google-cc

File name: google-collect-1.0.jar
Files of type: Classpath Entry (folder, ZIP or JAR file)

Figure 6: Selecting the GCL .jar file.

Click the "Open" button, and verify that the file has been added to the project library (Figure 7).



Java Platform: JDK 1.7 (Default)
Libraries Folder:

Compile-time Libraries:
Beans Binding
C:\Users\Max\Desktop\google-collect-1.0\google-collect-1.0.jar

Compile-time libraries are propagated to all library categories.

☑ Build Projects on Classpath

Figure 7: The .jar file added to the project library.

Test Plan

The LSS v.1.4.1 graphical user interface (GUI) is shown in Figure 8. The "Show Inventory" and "Tree View" buttons provide the user with access to the new program features.

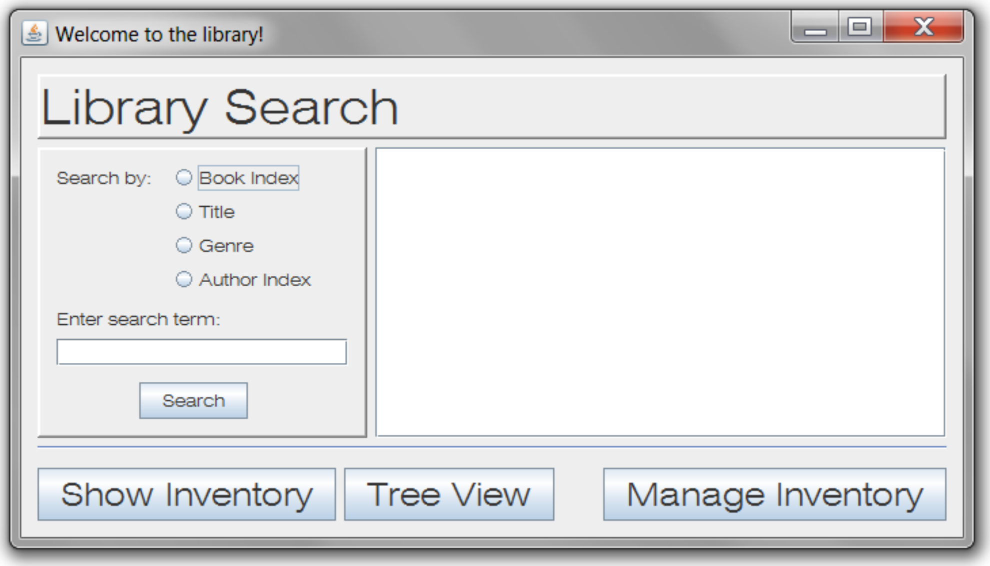
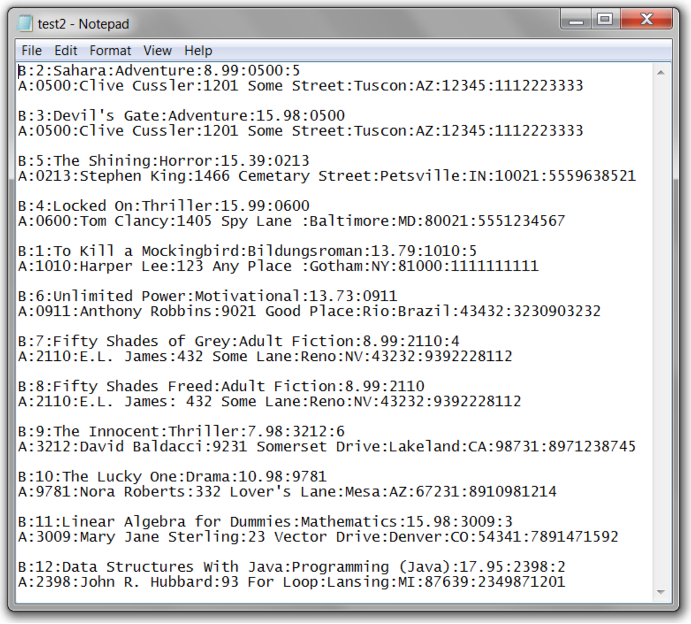


Figure 8: LSS v.1.4.1 GUI.

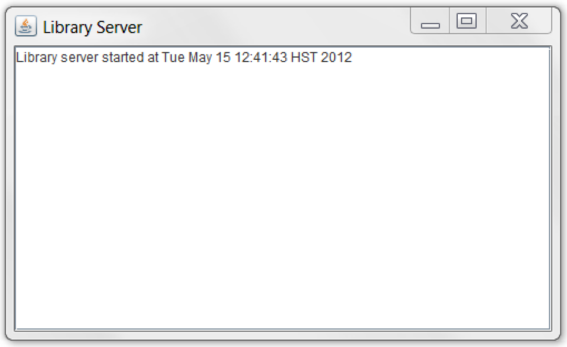
Figure 9 shows the data file that was used for initial testing.



```
B:2:Sahara:Adventure:8.99:0500:5
A:0500:Clive Cussler:1201 Some Street:Tuscon:AZ:12345:1112223333

B:3:Devil's Gate:Adventure:15.98:0500
A:0500:Clive Cussler:1201 Some Street:Tuscon:AZ:12345:1112223333

B:5:The Shining:Horror:15.39:0213
A:0213:Stephen King:1466 Cemetary Street:Petsville:IN:10021:5559638521

B:4:Locked On:Thriller:15.99:0600
A:0600:Tom Clancy:1405 Spy Lane :Baltimore:MD:80021:5551234567

B:1:To Kill a Mockingbird:Bildungsroman:13.79:1010:5
A:1010:Harper Lee:123 Any Place :Gotham:NY:81000:1111111111

B:6:Unlimited Power:Motivational:13.73:0911
A:0911:Anthony Robbins:9021 Good Place:Rio:Brazil:43432:3230903232

B:7:Fifty Shades of Grey:Adult Fiction:8.99:2110:4
A:2110:E.L. James:432 Some Lane:Reno:NV:43232:9392228112

B:8:Fifty Shades Freed:Adult Fiction:8.99:2110
A:2110:E.L. James: 432 Some Lane:Reno:NV:43232:9392228112

B:9:The Innocent:Thriller:7.98:3212:6
A:3212:David Baldacci:9231 Somerset Drive:Lakeland:CA:98731:8971238745

B:10:The Lucky One:Drama:10.98:9781
A:9781:Nora Roberts:332 Lover's Lane:Mesa:AZ:67231:8910981214

B:11:Linear Algebra for Dummies:Mathematics:15.98:3009:3
A:3009:Mary Jane Sterling:23 Vector Drive:Denver:CO:54341:7891471592

B:12:Data Structures With Java:Programming (Java):17.95:2398:2
A:2398:John R. Hubbard:93 For Loop:Lansing:MI:87639:2349871201
```

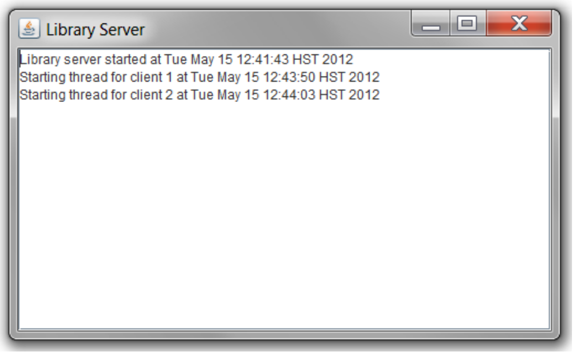
Figure 9: The file, `test2.txt`, used for initial testing.

When the `LibraryServer` class is run, the window in Figure 10 opens and indicates the date and time the server was started. Figure 11 shows multiple clients accessing the server.



```
Library server started at Tue May 15 12:41:43 HST 2012
```

Figure 10: Running the `LibraryServer` class file.



```
Library server started at Tue May 15 12:41:43 HST 2012
Starting thread for client 1 at Tue May 15 12:43:50 HST 2012
Starting thread for client 2 at Tue May 15 12:44:03 HST 2012
```

Figure 11: Multiple clients running on the server.

After the `LibraryServer` is started, multiple instances of the `LibraryClient` class can be run as shown in Figure 12. As shown, the Library GUI opens allowing users access to the `LibraryManager` class which contains methods to borrow and return books.

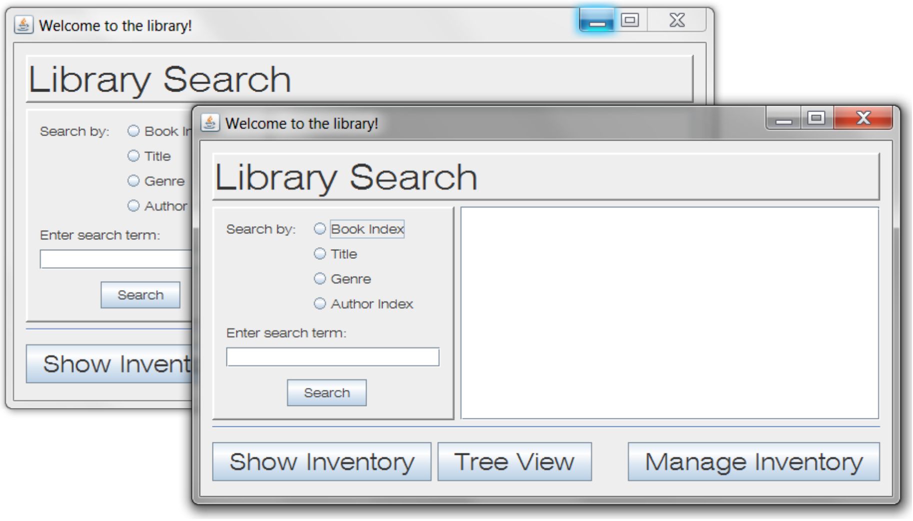


Figure 12: Multiple client GUIs running simultaneously.

Figure 13 shows the result of selecting the “Manage Inventory” button.

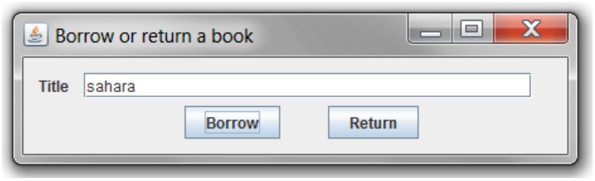
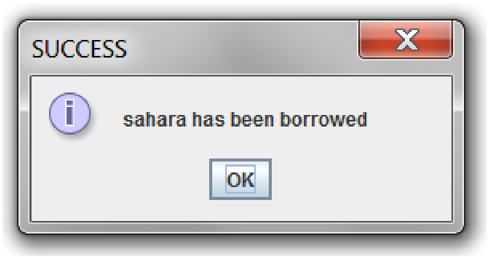


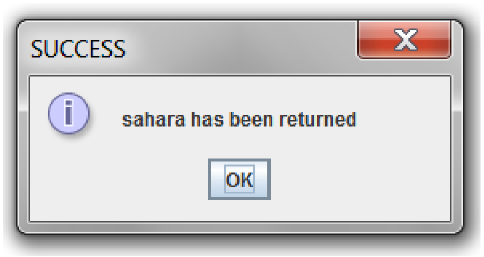
Figure 13: The book borrow and return GUI.

When a book title is entered and either the "Borrow" or "Return" button is selected, the informational windows as shown in Figures 14 and 15 appear.


SUCCESS

sahara has been borrowed

Figure 14: After a book has been borrowed.


SUCCESS

sahara has been returned

Figure 15: After a book has been returned.

Known Issues

The `Book` objects created by the test text file (`test2.txt`) are loaded into a `LinkedList` which is accessed by borrow and return methods when the appropriate button is selected from the book borrow and return GUI. Difficulty in determining the proper methods and data structures to maintain the current number of items in the LinkedList before and after books have been borrowed or returned causes the program to periodically ignore the default number of a particular title that the library can hold. Appropriate thread `stop()` methods were created but were not implemented or tested.

Lessons Learned

The use of threads and object synchronization should have been researched more thoroughly. Although some aspects of the project specification were not fulfilled, a tremendous amount about multithreading, servers, and clients was learned.

Future Improvements

I would like to create a fully functional and correct application that exploits greater thread manipulation while ensuring program safety.

References

Bourrillion, K., Levy, J., Bostock, M., & Wilson, J. (2010). Google Collections Library – Release 1.0 (FINAL – 20091230) [Software]. Available from http://code.google.com/p/google-collections/wiki/Releases

Oracle Corporation. (2011). NetBeans IDE 7.0.1 [Software]. Available from http://netbeans.org/community/releases/70/