COMP 2140 Assignment 2: Linked Lists

Helen Cameron

Due: Wednesday 26 October 2016 at 10:00 p.m.

Important Announcement

Until you agree to the honesty declaration (see the instructions at the end of this assignment under "Honesty Declaration"), you cannot hand in any assignments — that is, until you agree to the honesty declaration, you will not be able to see the assignment dropboxes in UM Learn.

How to Get Help

Your course instructor is helpful: Email me at Helen.Cameron@cs.umanitoba.ca or come to my office hours at 2:30–3:30 p.m. Mon/Wed/Fri or 4:00–4:30 p.m. Tues/Thurs in E2 477 EITC (or by appointment).

For email, please remember to put "[comp2140]" in the subject and use a meaningful subject, and to send from your UofM email account.

Course discussion groups on UM Learn: A discussion group for this assignment is available in the COMP 2140 course site on UM Learn (click on "Discussions" under "Communications"). Post questions and comments related to Assignment 2 there, and I will respond. Please do not post solutions, not even snippets of solutions there, or anywhere else. I strongly suggest that you read the assignment discussion group for helpful information.

Computer science help centre: The staff in the help centre can help you (but not give you assignment solutions!). See their website at http://www.cs.umanitoba.ca/undergraduate/computer-science-help-centre.php for location and hours. You can also email them at helpctr@cs.umanitoba.ca.

Programming Standards

When writing code for this course, follow the programming standards, available on this course's website on UMLearn. Failure to do so will result in the loss of marks.

Question

Remember: You will need to read this assignment many times to understand all the details of the program you need to write.

Goal: To maintain a ranked list of popular books for a library.

A library wants to persuade people to read more books by advertising their most popular books on their website.

Your task is to maintain a **doubly-linked list (with no dummy nodes) of books ranked by their popularity** — thus, the list is kept in order of rank, with the maximum rank (most popular) book first. A book's popularity is measured by activity at the library that references the book (a librarian adds a new copy of the book, a library patron queries the book, or borrows, or puts a hold on the book) — the more the book is queried, borrowed, etc., the higher its rank in the list.

(Note that you are NOT doing a complete library simulation. You are not maintaining a catalogue of all books in the library system, or tracking library patrons. You are only maintaining a ranked list of popular books.)

Your popular book list for the current day will begin with the previous day's top n books (see "Input" below), with each book having an initial rank of 1 (a low rank that will get higher if the book is involved in activity in the current day). Each time a book is involved in a library command — someone queries, borrows, or puts a hold on the book, or a librarian adds a new copy of the book to circulation, you should:

- If the book is already in your popular book list, then increase its rank by one, and then move (unlink and relink) the node containing the book to its proper place in the list (remember: the list is kept in order of rank, with the maximum rank first).
- If the book is not in your popular book list, then insert it into the list with rank 1.

(The exception is a return command. A library patron returning a borrowed book does not represent an increase of interest in the book. Therefore, a return command is simply ignored.)

Note that you are not allowed to change which book is stored in an existing node. Instead, you must unlink and relink nodes within the list. You must create a new node only when you are adding a book that does not already exist in the list.

Input

Your application class (the class that contains `main`) must prompt for the name of the input file.

The input file contains

- An integer n on the first line, then
- The authors and titles of the top n books (one per line) from the previous day, then
- The current day's commands (one per line).

An input line containing one of the previous day's top n books will contain the author's name (in quotes) and the book's title (in quotes). The following line is an example:

```
"Oliver Sacks" "The Man Who Mistook His Wife for a Hat"
```

(You are to insert the previous day's top n books into a doubly-linked list, giving each the rank of 1 — this is the initial popular book list for the current day.)

The current day's command are a mix of the following:

- Add (`'A'`): Add a new book to the library.
- Query (`'Q'`): Get more information about a book for a library patron.
- Hold (`'H'`): Place a hold on a book (used when somebody else has borrowed the book, or the book is at a different location than the library patron).
- Borrow (`'B'`): Borrow a book.
- Return (`'R'`): Return a book that was previously borrowed.

Each command contains the letter (`'A'` or `'Q'` or `'H'` or `'B'` or `'R'`)), the author's name (in quotes) and the book's title (in quotes). The following line is an example command:

```
Q "Robert Zurbin" "The Case for Mars"
```

I will provide a small input file (`tinyInput.txt`) and a larger input file (`largerInput.txt`). Your program should work correctly on both of them.

Output

As well as the usual title and trailer messages, you should print out:

- The initial popular book list (each book and its rank);
- Each command as you process it.
- After every time you have processed a group of n of the current day's commands, you should print out the highest-ranked n books in your popular book list (the first n books in your popular book list (with their ranks)).
- Finally, after you have processed all of the current day's commands, you should print out the entire popular book list (each book with its rank).

(See "Input" above for more on what n is.)

Sample Output

```
Comp 2140  Assignment 2   Fall 2016
Keeping track of popular books in a doubly-linked list.
Model Solution

Enter the book lists file name (.txt files only):
tinyInput.txt

*********************************************

Initial top-5 book list:
------------------------
"Neanderthal Man: In Search of Lost Genomes" by Svante Paabo
"Hill of Bones" by The Medieval Murderers
"Moby-Dick; or, The Whale" by Herman Melville
"A Large Harmonium" by Sue Sorensen
"Tales of a Starving Actress" by Karen Kucera Bate

The Day's Commands
------------------
Command: Q "Robert Zurbin" "The Case for Mars"
Command: B "Sue Sorensen" "A Large Harmonium"
Command: R "James S. A. Covey" "Abaddon's Gate"
Command: A "Jane Yolen" "Tales of Wonder"
Command: B "Svante Paabo" "Neanderthal Man: In Search of Lost Genomes"

        Top 5 Books
        -----------
        Rank = 2: "A Large Harmonium" by Sue Sorensen
        Rank = 2: "Neanderthal Man: In Search of Lost Genomes" by Svante Paabo
        Rank = 1: "Hill of Bones" by The Medieval Murderers
        Rank = 1: "Moby-Dick; or, The Whale" by Herman Melville
        Rank = 1: "Tales of a Starving Actress" by Karen Kucera Bate

Command: Q "Sue Sorensen" "A Large Harmonium"
Command: Q "James S. A. Covey" "Abaddon's Gate"
```

```
Command: H "Sue Sorensen" "A Large Harmonium"
Command: Q "Sue Sorensen" "A Large Harmonium"
Command: H "James S. A. Covey" "Abaddon's Gate"

        Top 5 Books
        -----------
        Rank = 5: "A Large Harmonium" by Sue Sorensen
        Rank = 2: "Neanderthal Man: In Search of Lost Genomes" by Svante Paabo
        Rank = 2: "Abaddon's Gate" by James S. A. Covey
        Rank = 1: "Hill of Bones" by The Medieval Murderers
        Rank = 1: "Moby-Dick; or, The Whale" by Herman Melville

Command: B "Herman Melville" "Moby-Dick; or, The Whale"
Command: R "Herman Melville" "Moby-Dick; or, The Whale"
Command: Q "Herman Melville" "Moby-Dick; or, The Whale"
Command: Q "Herman Melville" "Moby-Dick; or, The Whale"
Command: R "Svante Paabo" "Neanderthal Man: In Search of Lost Genomes"

        Top 5 Books
        -----------
        Rank = 5: "A Large Harmonium" by Sue Sorensen
        Rank = 4: "Moby-Dick; or, The Whale" by Herman Melville
        Rank = 2: "Neanderthal Man: In Search of Lost Genomes" by Svante Paabo
        Rank = 2: "Abaddon's Gate" by James S. A. Covey
        Rank = 1: "Hill of Bones" by The Medieval Murderers

Complete Book List (End of Day)
------------------------------
Rank = 5: "A Large Harmonium" by Sue Sorensen
Rank = 4: "Moby-Dick; or, The Whale" by Herman Melville
Rank = 2: "Neanderthal Man: In Search of Lost Genomes" by Svante Paabo
Rank = 2: "Abaddon's Gate" by James S. A. Covey
Rank = 1: "Hill of Bones" by The Medieval Murderers
Rank = 1: "Tales of a Starving Actress" by Karen Kucera Bate
Rank = 1: "The Case for Mars" by Robert Zurbin
Rank = 1: "Tales of Wonder" by Jane Yolen

Program ends.
```

Suggestions for Processing the Input

I used a `Scanner`, so I used `nextInt()` and `nextLine()` and `hasNextLine()`.

I also used the `String` method `trim()` to get rid of leading and trailing whitespace.

I used `String` method `charAt(0)` to see what command an input line contains.

To find the author(s) and title of a book from a line of input, I used `String` methods such as:

- `int posn=inputLine.indexOf('\"')` (find the first quote mark in `String inputLine`).
- `int nextPosn=inputLine.indexOf('\"',posn+1)` (find the first quote mark starting from position `posn+1` in `String inputLine`).
- `String author=inputLine.substring(authorStart,authorEnd)` (to retrieve the author's name from between the two quote marks). Note that `authorStart` must be the position after the opening quote

mark and that `authorEnd` must be the position of the closing quote mark.

Program Organization

You should write a book class. Each book instance must contain the title and author of the book.

You should write a node class. Each node will contain a rank (an `int`) and a book (or you could use generics instead of book), as well as back (to the previous node) and forward (to the next node) pointers. You can either use an object-oriented node class (outside your linked list class, similar to my Version 2) or something similar to my non-object-oriented node class (inside your linked list class, like my Version 1) — except that the nodes must be doubly-linked.

You should write a linked list class, which needs (at least) methods to:

- Insert a book with a given rank if it is not already in the list,
- Increment the ranks of a book: find the book, and — if it is in the list — increase its rank by 1 and move it to its new correct position in the ordered list (you must unlink and relink the node),
- Print the n highest rank books, and
- Print all of your popular book list.

Of course, you need to write your application class that contains `main` and that does all the input and organizes all the output, using your linked list and book classes to keep track of popular books.

Hand-in Instructions

Go to COMP2140 in UM Learn, then click "Dropbox" under "Assessments" at the top. You will find a dropbox folder called "Assignment 2". (If you cannot see the assignment dropbox, follow the directions under "Honesty Declaration" below.) Click the link and follow the instructions. Please note the following:

- Submit ONE .java file only. The .java file must contain all the source code. The .java file must be named
  `A2<your last name><your student id>.java` (e.g., `A2Cameron1234567.java`).
- Please do not submit anything else.
- You can submit as many times as you like, but only the most recent submission is kept.
- We only accept homework submissions via UM Learn. Please DO NOT try to email your homework to the instructor or TA or markers — it will not be accepted.
- We reserve the right to refuse to grade the homework or to deduct marks if you do not follow instructions.
- **Assignments become late immediately after the posted due date and time.** Late assignments will be accepted up to 49 hours after that time, at a penalty of 2% per hour or portion thereof. After 49 hours, an assignment is worth 0 marks and will no longer be accepted.

Honesty Declaration

To be able to hand in any assignment (i.e., to make the dropbox visible to you), you must first agree to the blanket honesty declaration. This declaration covers ALL your work in the course. To agree to the honesty declaration:

- Go to the COMP 2140 homepage on UM Learn, look at the main header (the navigation bar) at the top. Along with "Resources", "Communication", and "Assessments", you will see "Checklist" — click on "Checklist".

- If you can see a blue "Honesty Declaration" above the red "Note" near the top (and you canNOT see the "Save" option at the bottom), click on the blue "Honesty Declaration" at the top.
- Read the honesty declaration, check off "Agreement" (verifying that you have read the honest declaration and agree to it), and finally, click "Save". Only after this task is completed will you be able to see any dropboxes.

You only have to agree to the honesty declaration once, because it covers all your work in the course. (If you have handed in a previous COMP 2140 assignment to a dropbox, then you have already agreed to the honesty declaration.)