Assignment 4 Great Reads!

Due Date: Wednesday Nov 25, 2015
Weight: 8%
This assignment should be completed individually.

Description

Your assignment is to create a book management system called *GreatReads*. The system is inspired by the on-line book based social media site *goodreads*, http://www.goodreads.com. Your system will maintain a database of books, allow you to add reviews to books and to organize your books onto shelves.

The system will read the data, books, shelves, and reviews from a text file when the program starts. When the user exits the system the current database — books, shelves and reviews — will be saved back to the file read. This is known as data *persistence*. The data will persist from one invocation of the program to the next.

When the program is running the user will have the ability add books to, and delete books from the list of books (maintain the list). The user may also maintain a list of shelves to help organize the books. Shelves may be added and deleted and books can be assigned to shelves. The relationship between books and shelves is many-to-many, that is one particular book may be assigned to multiple shelves and each shelf may have multiple books on it. A user may add one or more reviews to a book.

Detailed Specifications

1. Books
    - Books have a title (string of length at least 1), an author (string of length at least 1) and a year published (integer, must be a valid year; the Gutenberg Bible is usually considered to be the first printed book and it was from around 1450, so valid years can be 1450-2100).
    - Books are identified by their title so you cannot have two books in your system with the same title!
2. Shelves
    - Each shelf has a name (string of length at least 1), and an optional description (string of length 0 or more).
    - Shelves are identified by their name so you cannot have two shelves in your system with the same name!

3. Review

   - A review consists of a one or more lines of text (strings) and a timestamp.
   - Reviews cannot be deleted, unless the book is deleted. As detailed below when a book is deleted all associated reviews should also be deleted.

4. Operations

   (a) List all the books in the system. For each book, list the book information (title, author, year) the number of shelves it is on and the number of reviews it has.

   (b) List the contents of a particular shelf. This should print the shelf information, plus the book information (title, author, year) for the books on that shelf.

   (c) List a summary of all shelves. This will print the number of shelves and for each shelf display its name and the number of books currently on it.

   (d) List the information for a single book. This should include the information about the book, the names of the shelves it is on, and any reviews there are of it, in ascending date/time order.

   (e) Add or remove a book. When a book is removed, it must also be removed from any shelves it is on and and reviews of it must be deleted.

   (f) Add or remove a shelf. When a shelf is removed any books must be removed from that shelf as well. Note that this does *not* mean that any books are deleted.

   (g) Add or remove a book from a shelf. A book cannot be added twice to the same shelf.

   (h) Add a review to a book.

   For all operations it may be the case that there are no books or no shelves or no reviews. The program must behave reasonably in those cases.

5. Data file format. The name of the file to read and save data should be either given on the command line, or default to `greatreads.txt`. That is, if the user does not specify a file name, or the file specified cannot be opened, use `greatreads.txt`. If the file does not exist it should be created when the system exits and all data saved to `greatreads.txt`

   (a) The file is broken down into three sections.

      i. The first contains information about all shelves.

      ii. The second section contains information about each book in the system.

iii. The third section contains the reviews.

The sections are separated by a single line that contains only the string `####`

(b) Shelves Section. Each line contains information for a single shelf, the name followed by the description, separated by a | character. Descriptions can be empty!

(c) Book Section. Each line in the file contains information for a single book. There are two parts to to each line, separated by a `#` character. The line starts with the title, author and year separated by | characters. The rest of the line starts with the `#` character, then the names of all the shelves this book is on, separated by | characters.

(d) Review Section

A review consists of a book name and a timestamp separated | characters on one line then a number of text lines followed by a line with a single `#` character.

```
<----------------- greatreads.txt --------------------------->
Read|Books I have read
Want to Read|Books I want to read
Computers|Computers, IT, Computer Science, IT and Society books
Fiction|
Non-Fiction|
####
Catch-22|Joseph Heller|1961#Fiction|Read
Clearing The Plains|James Daschuk|2013#Read|Non-Fiction
How to Create a Mind|Ray Kurweil|2012#Computers|Read
Dirk Gently's Holistic Detective Agency|Douglas Adams|1987#
####
Catch-22|2015-11-03 09:01:16
Quite possibly the best work of fiction ever.
Heller captures the aburdity of life in a way no other author
ever has.
#
How to Create a Mind|2015-11-03 09:01:16
A must read for anyone interested in AI.
#
< ----------------- greatreads.txt --------------------------->
```

6. Starting Code

(a) You must use the `UserInteraction` class for all user input and output to the program. You only need a single instance of `UserInteraction` for the program. The reference should be passed around as needed. This class cannot be modified.

(b) You must use the provided *Menu* class for all menus in the program. This class cannot be modified.

Hints and Other Considerations

- Do not simply start coding things! Think about how you will design the system first!
- Use the split() method from the String class for parsing the individual lines. In case of the books it is suggested to first split the string using the hash, #, as a delimiter. Then parse the first and second tokens separately using the bar, | as a delimiter.
- Use the trim() method from the String class on each token to remove leading and ending white space.
- Reviews. When entered reviews, or any block of text, you can read and store it as a series of lines. That is, each line read is a string!

What to Hand In

1. Rename your BlueJ project folder as: `<LastName>_<FirstName>_A4`.
2. Include a Word document in your project folder that describes your design. As in A3 it should include four sections:
   (a) Requirements. This should be a relatively short section where you detail any extra requirements or clarification to the requirements, or any assumptions that you made. No program specification, whether it is an assignment on an enterprise system is without some ambiguity.
   (b) Design. Give an object oriented breakdown for your program. Start with a short description of your program. Then include a list of all the classes, a brief description of the class, what data it will store and what functions it will perform. Show how the classes are related; you can use a diagram for this.
   (c) Testing. We will be spending a week of so on testing in the final weeks of the semester. At this stage I would just like you to describe how you tested your program. Try to think of boundary cases, cases with invalid data, etc. I am not expecting formal test cases or the like, I just want to see if you have a good grasp of what testing can entail.
   (d) Conclusions/Thoughts. Are there things you would do differently if you were to do this assignment again? Is there something in particular that you learned?

   The document should have a title block with your name, a title and the date, a page number at the bottom right. 1" margins and use a 12pt font. Times Roman is probably the best choice. Single space the text. It should be more than 1 page long and less than 4.

3. Submit the entire BlueJ project folder (with all its contents) to the COMP 1502 submit folder.

Grading

A detailed grading rubric will be provided.

Outcomes

Once you have completed this assignment you will have:

- Practiced your object-oriented design skills;
- Used both the state and behaviour of objects;
- Created relationships (dependency, aggregate) between objects;
- Used multiple instances of a class, each with its own state;
- Used command line arguments;
- Used data persistence (writing to data files);
- Used code that was written by someone else and provided to you.