THE UNIVERSITY OF SYDNEY

INFO1113 Assignment 1

Due: 16th October 2020, 23:59

This assignment is worth 10% of your final assessment

Task Description

In this assignment you will develop a library operating system called Librarian in the Java programming language. Use the examples below and the Javadoc found in Resources on EdStem to write the program to specification.

You are encouraged to ask questions on Ed using the assignments category. As with any assignment, make sure that your work is your own, and you do not share your code or solutions with other students.

Working on your assignment

You can work on this assignment on your own computer or the lab machines. It is important that you continually back up your assignment files onto your own machine, external drives, and in the cloud.

You are encouraged to submit your assignment on Ed while you are in the process of completing it. By submitting you will obtain some feedback of your progress on the sample test cases provided.

Implementation details

Write a program in Java that implements the Librarian application as documented in the Javadoc on EdStem. You can assume that our test cases will contain only valid input commands and not cause any integer overflows. Commands are case insensitive.

Librarian stores collections of Member accounts and Book objects. Member accounts are identified by their member number, and store their name, the books they are currently renting, and their rental history. Member accounts start at 100000 and increment up (such that the second Member account will have member number 100001).

A Book is identified by its serial number, and store the title, author, genre, member currently, and rental history. Members are able to rent and return books, with only one member able to rent a book at any time. Books are considered copies of each other if they have the same title and author (note that copies of a book all have different serial numbers).

Book collections can be read from and archived in a CSV (comma-separate values) file, stored in order of serial number in the following form:

<serial number>,<title>,<author>,<genre>

Example:

111111,Catch 22,Joseph Heller,Comedy

An example Book collection has been provided in Resources.

Archived books can be added to Librarian, either individually or as a whole collection. If any serial numbers are already present in the system, those books should not be added.

Your program should only be comprised of Library.java, Book.java and Member.java. Do not modify any of the existing method signatures found in the documentation. You are encouraged to write your own additional methods to improve style and reduce repetitive code. Your program must produce no errors when built and run on Ed. Your program will read from standard input and write to standard output.

Your program output must match the exact output format shown in the examples and on Ed. You are encouraged to submit your assignment while you are working on it, so you can obtain some feedback.

In order to obtain full marks, your program will be checked against automatic test cases, manually inspected by your tutors and you must submit a set of test cases that ensure you have implemented functionality correctly.

Commands

Your program should implement the following commands, look at the examples to see how they work. Commands with [LONG] contain optional items that modify the functionality.

EXIT ends the library process
COMMANDS outputs this help string

LIST ALL [LONG] outputs either the short or long string for all books
LIST AVAILABLE [LONG] outputs either the short of long string for all available books
NUMBER COPIES outputs the number of copies of each book
LIST GENRES outputs the name of every genre in the system
LIST AUTHORS outputs the name of every author in the system

GENRE <genre> outputs the short string of every book with the specified genre
AUTHOR <author> outputs the short string of every book by the specified author

BOOK <serialNumber> [LONG] outputs either the short or long string for the specified book
BOOK HISTORY <serialNumber> outputs the rental history of the specified book

MEMBER <memberNumber> outputs the information of the specified member
MEMBER BOOKS <memberNumber> outputs the books currently rented by the specified member
MEMBER HISTORY <memberNumber> outputs the rental history of the specified member

RENT <memberNumber> <serialNumber> loans out the specified book to the given member
RELINQUISH <memberNumber> <serialNumber> returns the specified book from the member
RELINQUISH ALL <memberNumber> returns all books rented by the specified member

ADD MEMBER <name> adds a member to the system
ADD BOOK <filename> <serialNumber> adds a book to the system

ADD COLLECTION <filename> adds a collection of books to the system
SAVE COLLECTION <filename> saves the system to a csv file

COMMON <memberNumber1> <memberNumber2> ... outputs the common books in members' history

Examples (1)

user: LIST ALL
No books in system.

user: LIST GENRES
No books in system.

user: LIST AUTHORS
No books in system.

user: MEMBER 100000
No members in system.

user: RENT 100000 111111
No members in system.

user: RELINQUISH 100000 111111
No members in system.

user: RELINQUISH ALL 100000
No members in system.

user: LIST ALL LONG
No books in system.

user: EXIT
Ending Library process.

Examples (2)

user: ADD MEMBER John Smith
Success.

user: ADD BOOK books.csv 111111
Successfully added: To Kill a Mockingbird (Harper Lee).

user: ADD COLLECTION books.csv
16 books successfully added.

user: RENT 100000 111111
Success.

user: RELINQUISH 100000 111111
Success.

user: LIST ALL
To Kill a Mockingbird (Harper Lee)
The Hobbit (J.R.R. Tolkien)
The Iliad (Homer)
The Odyssey (Homer)
The Hitchhiker's Guide to the Galaxy (Douglas Adams)
So Long and Thanks for all the Fish (Douglas Adams)
Batman Volume 1 (Bob Kane)
Harry Potter 1 (J.K. Rowling)
Harry Potter 2 (J.K. Rowling)
Harry Potter 3 (J.K. Rowling)
Harry Potter 4 (J.K. Rowling)
Harry Potter 5 (J.K. Rowling)
Harry Potter 6 (J.K. Rowling)
Wasteland (T.S. Eliot)
Biggles in the Battle (W.E. Johns)
Aeneid (Virgil)
Halloween (Jamie Lee Curtis)

user: EXIT
Ending Library process.

Examples (3)

user: ADD MEMBER Ben James Smith
Success.

user: MEMBER 100000
100000: Ben James Smith

user: MEMBER BOOKS 100000
Member not currently renting.

user: MEMBER HISTORY 100000
No rental history for member.

user: ADD BOOK books.csv 111111
Successfully added: To Kill a Mockingbird (Harper Lee).

user: RENT 100000 111111
Success.

user: MEMBER BOOKS 100000
To Kill a Mockingbird (Harper Lee)

user: RELINQUISH 100000 111111
Success.

user: MEMBER BOOKS 100000
Member not currently renting.

user: MEMBER HISTORY 100000
To Kill a Mockingbird (Harper Lee)

user: EXIT
Ending Library process.

Examples (4)

user: ADD MEMBER Amelia Earhart
Success.

user: ADD MEMBER Tom Clancy
Success.

user: ADD COLLECTION books.csv
17 books successfully added.

user: RENT 100000 111111
Success.

user: RENT 100000 111112
Success.

user: RENT 100000 111113
Success.

user: RENT 100001 111111
Book is currently unavailable.

user: RELINQUISH ALL 100000
Success.

user: RENT 100001 111112
Success.

user: RENT 100001 111113
Success.

user: RELINQUISH ALL 100001
Success.

user: COMMON 100000 100001
The Hobbit (J.R.R. Tolkien)
The Iliad (Homer)

user: EXIT
Ending Library process.

Examples (5)

user: ADD MEMBER Dev Patel
Success.

user: ADD MEMBER Robert Pattinson
Success.

user: ADD MEMBER Daniel Craig
Success.

user: ADD BOOK books.csv 111111
Successfully added: To Kill a Mockingbird (Harper Lee).

user: BOOK 111111
To Kill a Mockingbird (Harper Lee)

user: BOOK 111111 LONG
111111: To Kill a Mockingbird (Harper Lee, Historical Fiction)
Currently available.

user: RENT 100000 111111
Success.

user: BOOK 111111 LONG
111111: To Kill a Mockingbird (Harper Lee, Historical Fiction)
Rented by: 100000.

user: RELINQUISH 100000 111111
Success.

user: RENT 100001 111111
Success.

user: RELINQUISH ALL 100001
Success.

user: RENT 100002 111111
Success.

user: BOOK HISTORY 111111
100000
100001

user: EXIT
Ending Library process.

Examples (6)

user: ADD COLLECTION books.csv
17 books successfully added.

user: LIST GENRES
Comedy
Comic
Epic
Fantasy
Historical Fiction
Horror
Poetry
War

user: LIST AUTHORS
Bob Kane
Douglas Adams
Harper Lee
Homer
J.K. Rowling
J.R.R. Tolkien
Jamie Lee Curtis
T.S. Eliot
Virgil
W.E. Johns

user: GENRE Fantasy
The Hobbit (J.R.R. Tolkien)
Harry Potter 1 (J.K. Rowling)
Harry Potter 2 (J.K. Rowling)
Harry Potter 3 (J.K. Rowling)
Harry Potter 4 (J.K. Rowling)
Harry Potter 5 (J.K. Rowling)
Harry Potter 6 (J.K. Rowling)

user: AUTHOR Bob Kane
Batman Volume 1 (Bob Kane)

user: EXIT
Ending Library process.

Writing your own testcases

We have provided you with some test cases but these do not test all the functionality described in the assignment. It is important that you thoroughly test your code by writing your own test cases.

You should place all of your test cases in a tests/ directory. Ensure that each test case has a .in input file along with a corresponding .out output file. We require that the names of your test cases are descriptive so that you know what each is testing, e.g. genre.in & genre.out and we can accurately and quickly assess your test cases. **Note: If you do not format your test case files as explained (where each test case has <name>.in and <name>.out files for input and output), you shall receive 0 for this component.**

Submission Details

Final deliverable for the correctness and manual inspection will be due on the 16th of October at midnight.

You must submit your code and tests using the assignment page on Ed. To submit, simply place your files and folders into the workspace, click run to check your program works and then click submit.

You are encouraged to submit multiple times, but only your last submission will be considered.

Marking

You will only be given valid inputs as part of the automatic test suite. Your program will be checked for errors that a user can possibly make. In addition, we will mark your program against a substantial collection of hidden test cases.

5 marks are assigned based on automatic tests for the correctness of your program. This component will use hidden test cases that cover every aspect of the specification. Your program must match the exact output in the examples and the test cases on Ed.

4 marks are assigned based on the code coverage of your own test cases. Make sure that you carefully follow the assignment specifications regarding the structuring of your test cases.

1 mark is assigned based on a manual inspection of the style. Style will be assessed based on the conventions set out in the Google Java Style Guide (https://google.github.io/styleguide/javaguide.html)

Additional Notes

To sort a list of Strings lexicographically, use *list.sort((x, y) -> x.compareTo(y));*

Academic declaration

By submitting this assignment you declare the following:

I declare that I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure, and except where specifically acknowledged, the work contained in this assignment/project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgment from other sources, including published works, the Internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

I acknowledge that the School of Computer Science, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and/or communicate a copy of this assignment to a plagiarism checking service or in-house computer program, and that a copy of the assignment may be maintained by the service or the School of Computer Science for the purpose of future plagiarism checking.