Assignment - Equipment Loan System

[Weight: 15 marks out of the final mark of this course]

Deadline: December 03, 2021 (Friday of revision week)

For late submissions, 2% of your original marks will be deducted if you hand in 1-day late (i.e. on Dec 04), 25% for 2-days (i.e. on Dec 05), assignments handed in on or after Dec 06 will get zero mark.

Academic dishonesty is strictly prohibited. The principle concerns whether students get their deserved marks and do not intend to cause unfairness. Dishonesty also involves when one let others have a chance to copy his/her code,

Grading:

Students **must** obtain the following results in sequence:

Phase 1 (at least 80% correct in PASS) ==> Phase 2 (at least 80% correct in PASS) ==> Phase 3

- If you can finish Phase 1 with good programming styles + OO programming skills => up to B+
- If you can finish Phase 2 with good programming styles + OO programming skills => up to A
- If you can finish Phase 3 with good programming styles + OO programming skills => up to A+

Various test cases are used for each phase (e.g. Phase 1: 1a.txt, 1b.txt, etc..). If you get partial correct, your work is still considered. E.g. If you can pass 1a.txt – 1b.txt only, your grade may be up to C+.

- For "Good Programming Styles", note that proper indentations, code-layout formatting, proper, meaningful naming, well-designed classes, methods, fields are more important than writing comments.
- During manual marking (Dec 04-10), selected students will be asked to meet me for discussion of your work.

Note:

Please apply what you learn from Lab08 - Lab10. You may reuse the code that you worked for Lab08 – Lab10. Reusing these code would not be considered as plagiarism.

Please first finish your program for Lab09, then modify and add the required functionalities for this Assignment.

Assignment Description

Extend the `Lab09-Q2` program to allow adding equipment items, borrow and return items, request items (i.e., queue up for a item which is unavailable at the moment) and pick up items.

1. Revise the existing classes and add new classes for proper modelling of the system.
   - Add an `ArrayList` of `Item` objects in the `Club` class.
   - Add the `Item` class (object fields involve item ID, name, arrival date, and item status etc..)
   - Requests and queue management
     [Modelling] There are a number of ways to model the queue of requests. One approach is to maintain a queue (`ArrayList`) of requests by each item. You may use any other design.

     [Logistic] When a item is returned and the queue of requests is not empty, the first queueing member will be removed from the queue and the item is marked as on hold for this member to pick up. The on hold period will due in 3 days. If this member does not pick up the item within the period, the next queuing member will take turn after it is due.
   - The item status (available or borrowed etc.) should be implemented using the State Pattern (Learnt in Lab06):
     - public interface ItemStatus
     - public class ItemStatusAvailable implements ItemStatus
     - public class ItemStatusBorrowed implements ItemStatus
     - public class ItemStatusOnhold implements ItemStatus

     ItemStatusBorrowed etc. may contain fields like: the borrowing member, loan date
   - For each Member, the counts of borrowed items and requested items are required for quota checking, and are displayed upon `listClubMembers.`

2. New commands for the operations involve
   i. `arrive` : arrival of new items
   ii. `checkout` : a member borrows <u>at most 6 item</u> (loan period is not catered in this assignment)
   iii. `checkin`: a member returns a item.
   Note 1: See P. 1 for "[Logistic]" in point 1;
   Note 2: Undo/redo can be complicated if there is any queuing member.
   iv. `listItems`: listing of all items, ordered by item IDs.
   v. `request`: a member requests to queue up for an unavailable item, i.e., reserve the item. A member can queue for <u>at most 3 items</u> at any time. Note: a member is not allowed to request an available item, or request a item which this member is currently borrowing, or is already queuing for it.

      For each item in the output of `listItems`, the IDs of the members who request the item should be listed, according to the ordering of the requests.
   vi. `cancelRequest`: a member cancels a request. Note: undo/redo can happen!

3. The commands `startNewDay`, `listClubMembers` and `register` were started in `Lab09-Q2` already. You may need to further modify some of them.

4. For `startNewDay`:

   In Phase 3, cases of "onhold due" are to be handled upon `startNewDay`. (See P. 1 for "[Logistic]" in point 1.). The handling and output concerning these cases should be ordered by item IDs.

   For simplicity your program does not need to handle undo/redo of StartNewDay.
   i.e. although CmdStartNewDay should be undoable and redoable, you may assume that the test cases will not include the situation of undoing/redoing this operation.

5. The names of command classes should start with "`Cmd`", eg. "`class CmdRegister`", "`class CmdListItems`"

6. You will need to add handling for the following error cases
   a) Member ID already in use (For register)
   b) Item ID already in use (For item arrival)
   c) Member not found (For checkout, request)
   d) Item not found (For checkout, request)
   e) Item not available -- already borrowed by or on hold for somebody (For checkout)
   f) Loan quota Exceeded (For checkout)
   g) The item is not borrowed by this member (For checkin)
   h) The item is currently available (For request -- if it is available, or on hold for the requesting member to pick up)
   i) The item is already borrowed by the same member (For request)
   j) The same member has already requested the item (For request)
   k) Item request quota exceeded (For request)
   l) Request record is not found (For cancelRequest)
   m) Insufficient command arguments (For all commands, e.g. missing member id to register)
   n) Unknown command (Checking in the main loop in `main`(). The program should continue.)

   - Most of the above should be done by Exception Handling. You should name all Exception classes with prefix: "`Ex`", eg. "`ExItemIdInUse`", "`ExMemberNotFound`"

- Please pay attention to the following:

```
Advantages of Using Exceptions (Week 09 Lecture exercise Q5)
===================================================
Advantage 1.  Separating Error-handling from "Regular" Code (** You should achieve in this assignment (most cases) !!)
Advantage 2.  Grouping and Differentiating Errors (Not needed for assignment)
Advantage 3.  Propagating Erros up the call stack (** MUST achieve in your assignment !!!)
```

7. The requirements in each phase and test cases for reference are given on Page 4.
   The given test cases and outputs are to show the functionalities that you need to implement. They also serve to specify the input and output formats.

   However, note that they do not test rigorously for the accuracy of your program.
   For example, your program should be able to handle various sequences of undo-redo of different undoable commands appropriately.
   If your program fails to work appropriately, then your grade will be affected due to incorrect solution (despite that you might have obtained 100% correct in PASS).

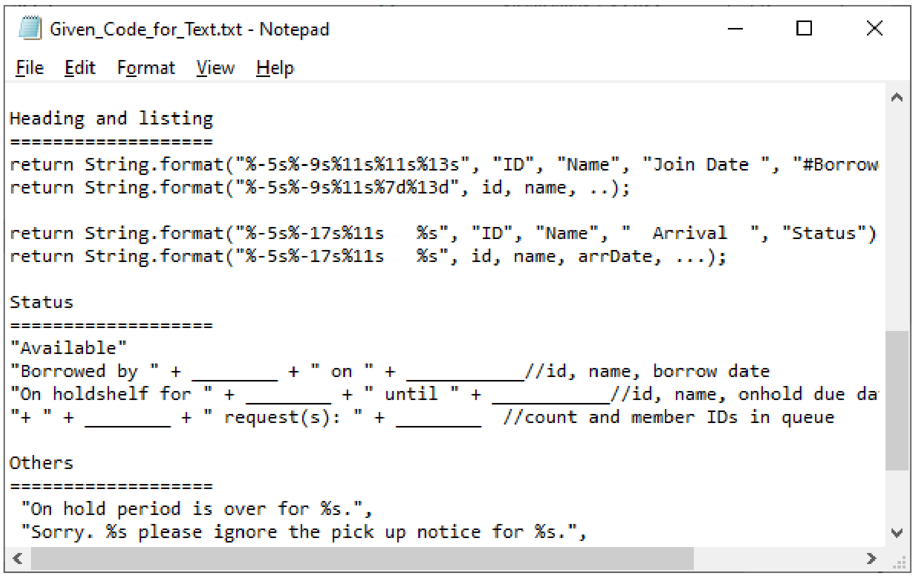
   To help verify that students have applied proper solution design, a few test cases on PASS are NOT disclosed. These test cases will be posted after manual marking.

   Q: What if my program cannot pass these test cases just because of some minor problems, e.g. spacing or string spelling problems? I don't want to lose my marks just based on these non-technical issues.

   A: Helena will manually check the cases and will restore the score for you if the discrepancy is not related to problem solving or code design.

8. Concerns about efficiency: Assume that we have up to m=5000 members and n=50000 items. For modern computers, keeping these records, doing any linear time operations, and providing sorting/searching are not a problem. Therefore, please spend more time and effort on modelling the entities involved in the case study.

9. You may copy given code for text contents from the following file (available on the course web):



```
Heading and listing
===================
return String.format("%-5s%-9s%11s%11s%13s", "ID", "Name", "Join Date ", "#Borrow
return String.format("%-5s%-9s%11s%7d%13d", id, name, ..);

return String.format("%-5s%-17s%11s   %s", "ID", "Name", "  Arrival  ", "Status")
return String.format("%-5s%-17s%11s   %s", id, name, arrDate, ...);

Status
===================
"Available"
"Borrowed by " + ________ + " on " + ___________//id, name, borrow date
"On holdshelf for " + ________ + " until " + ___________//id, name, onhold due da
"+ " + ________ + " request(s): " + ________  //count and member IDs in queue

Others
===================
 "On hold period is over for %s.",
 "Sorry. %s please ignore the pick up notice for %s.",
```

Grading and requirements by phases:
(See next page)

Grading and requirements by phases:

The table below lists the main requirements and test cases of each phase. For command formats and required outputs, please refer to the styles in Lab08 Q2 to Lab10 Q1, and the contents in the given test cases and outputs on the course web.

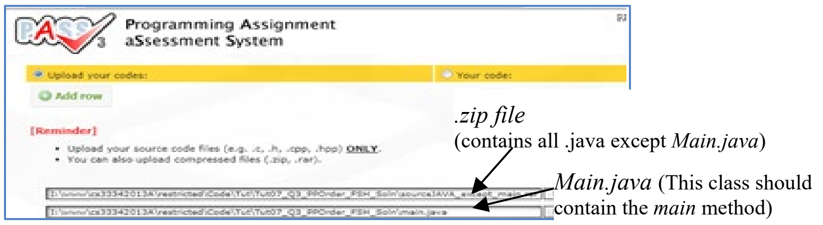
Also required:
good programming styles +
OO programming skills
(Please refer to P. 1)

Phases	Execution of commands	Requirements of proper undo/redo	Requirements of Exception handling	Test case for reference (most can be found on courseweb *)	Maximum Grade
Phase 1					
Phase 1.1	Register member, List members Start new day	--	Member ID already in use (For register)	1a.txt	~C
Phase 1.2	Arrive new item List items Start new day	--	Item ID already in use (For arrival of new item)	1b.txt	~C+
Phase 1.3	1.1 + 1.2 + checkout (borrow item)	--	--	1c.txt	~B-
Phase 1.4	Same as above	undo/redo of Register member Arrive new item Checkout	--	1d.txt	~B
Phase 1.5	Same as above	--	Member ID already in use (For register) Item ID already in use (For arrival of new item) Member not found (For checkout) Item not found (For checkout) Item borrowed by others (For checkout) Quota checking (Each can borrow 6 items only)	1e.txt	~B+
Phase 1.6	Assorted (1.1-1.5)			1f.txt	~B+
Phase 2					
Phase 2.1	Phase 1 + checkin	undo/redo of checkin	Item not borrowed by this member	2a.txt	~B+
Phase 2.2	Phase 1 + request	--	--	2b.txt	~A-
Phase 2.3	Phase 1 + request	undo/redo of request	checking related to request	2c.txt	~A-
Phase 2.4	Phase 1 + request + checkin + checkout checkin : let the first queuing member to pick up in 3 days checkout: pick-up the onhold book	--	--	2d.txt	~A-
Phase 2.5	Phase 2.4 + + cancelRequest	undo/redo of cancelRequest	checking related to cancelRequest	2e.txt	~A
Phase 2.6	Assorted (1.1-2.5)			2f.txt	~A
Phase 3					
Phase 3.1	StartNewDay: check onhold items which are due	(For simplicity, assume no undo/redo of StartNewDay)	--	3a.txt	~A
Phase 3.2	request + checkin	undo/redo of checkin !!	--	3b.txt	~A+
Phase 3.3	Assorted			3c.txt, 3c1.txt, 3c2.txt, 3c3.txt	~A+

* 1f.txt, 2f.txt, 3c1.txt, 3c2.txt, 3c3.txt are not disclosed before manual marking.

Submission:

Please submit them to PASS as shown below:


Programming Assignment aSsessment System
Upload your codes:
Your code:
Add row
.zip file
(contains all .java except *Main.java*)
Main.java (This class should contain the *main* method)

[Reminder]
- Upload your source code files (e.g. .c, .h, .cpp, .hpp) **ONLY**.
- You can also upload compressed files (.zip, .rar).

-- end --