Assignment - Buffet Reservation

[Weight: 18 marks out of the final mark of this course]

Deadline: Dec 06, 2019 (Friday of revision week)

For late submissions, 20% of your original marks will be deducted if you hand in 1-day late (i.e. on Dec 07), 40% for 2-days (i.e. on Dec 08). Assignments handed on or after Dec 09 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,

Students must obtain the following results in sequence:

Phase 1 (100% correct in PASS) ==> Phase 2 (100% correct in PASS) ==> Phase 3(100% correct in PASS)

- 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 (eg. Phase 1: 1a.txt, 1b.txt, etc..). If you get partial correct, your work is still considered. Eg. If you can pass 1a.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.

Assignment Description

Note: Before you start working on the assignment, please first learn from Lab08, Lab09 and Lab10

Consider a buffet reservation system, that accepts advanced bookings. The requesting customer needs to provide his/her name, phone number, total seats needed (total persons to come), and the target dining day to come.

The restaurant has ten tables for 2 persons (Code T01, T02, ..T10), six tables for 4 persons (Code F01, F02, .. F06), and three tables for 8 persons (Code H01, H02, H03). Upon a booking request is reorded, the system will put the record in a *pending* status. The staff-in-charge can check for available tables for the target day, and then assign the table(s) for the booking request. The status of the booking will then be changed into an *accepted state.*

The system also accepts cancellation of booking, regardless of whether it is a pending or an accepted booking.

The system can generate listing for both pending and accepted bookings. For pending bookings, "pending" is shown in the listing. For accepted bookings, the codes of the assigned tables are shown.

The system also provides a handy tool, that suggests which table(s) are to be assigned for a booking, based on the number of seats needed by the booking, and the available tables.

To facilitate checking and identification, the system has a ticketing system. For each business day, one ticketing sequence (start from No. 1, then No. 2, and so on) is maintained. For example, if Mr BLACK is the first one to place a booking for Mar 22, then he gets ticket code No. 1. The next customer to place a booking for Mar 22 will get ticket code No. 2. To make sure the identification is unique, the ticket code

for each business day should not be reused. For example, even if one cancels his/her booking, the ticket code that was already given to that booking will not be reused. To maintain such a ticketing system, refer to the lecture exercise in week 11.

In addition, the system allows undo/redo (See Lab08) and handles most error cases using Exception handling (See Lab10).

Your task: Implement this system. Note: special modelling and sorting requirements are given in P. 3. Below shows a simple run-down. The input/output format follows Lab09.

1(Sample).txt

```
startNewDay|17-Mar-2019
request|TANG, Ms|90121212|3|22-Mar-2019
request|CHAN, Ms|90123456|4|22-Mar-2019
request|CHAN, Ms|90123456|10|21-Mar-2019
request|WHITE, Ms|90000001|30|22-Mar-2019

listReservations

cancel|22-Mar-2019|3
request|BLACK, Mr|93330001|30|22-Mar-2019

assignTable|22-Mar-2019|2|T01|T02
assignTable|22-Mar-2019|1|F03

listReservations
listTableAllocations|22-Mar-2019
listTableAllocations|24-Mar-2019

suggestTable|21-Mar-2019|1
suggestTable|22-Mar-2019|4
```

Run-down (output)

```
Please input the file pathname: 1(Sample).txt

> startNewDay|17-Mar-2019

> request|TANG, Ms|90121212|3|22-Mar-2019
Done. Ticket code for 22-Mar-2019: 1

> request|CHAN, Ms|90123456|4|22-Mar-2019
Done. Ticket code for 22-Mar-2019: 2

> request|CHAN, Ms|90123456|10|21-Mar-2019
Done. Ticket code for 21-Mar-2019: 1

> request|WHITE, Ms|90000001|30|22-Mar-2019
Done. Ticket code for 22-Mar-2019: 3

> listReservations
Guest Name   Phone      Request Date  Dining Date and Ticket   #Persons   Status
CHAN, Ms     90123456   17-Mar-2019   21-Mar-2019 (Ticket 1)     10       Pending
CHAN, Ms     90123456   17-Mar-2019   22-Mar-2019 (Ticket 2)      4       Pending
TANG, Ms     90121212   17-Mar-2019   22-Mar-2019 (Ticket 1)      3       Pending
WHITE, Ms    90000001   17-Mar-2019   22-Mar-2019 (Ticket 3)     30       Pending

> cancel|22-Mar-2019|3
Done.

> request|BLACK, Mr|93330001|30|22-Mar-2019
Done. Ticket code for 22-Mar-2019: 4

> assignTable|22-Mar-2019|2|T01|T02
Done.

> assignTable|22-Mar-2019|1|F03
Done.

> listReservations
Guest Name   Phone      Request Date  Dining Date and Ticket   #Persons   Status
BLACK, Mr    93330001   17-Mar-2019   22-Mar-2019 (Ticket 4)     30       Pending
CHAN, Ms     90123456   17-Mar-2019   21-Mar-2019 (Ticket 1)     10       Pending
CHAN, Ms     90123456   17-Mar-2019   22-Mar-2019 (Ticket 2)      4       Table assigned: T01 T02
TANG, Ms     90121212   17-Mar-2019   22-Mar-2019 (Ticket 1)      3       Table assigned: F03

> listTableAllocations|22-Mar-2019
Allocated tables:
T01 (Ticket 2)
T02 (Ticket 2)
F03 (Ticket 1)

Available tables:
T03 T04 T05 T06 T07 T08 T09 T10 F01 F02 F04 F05 F06 H01 H02 H03

Total number of pending requests = 1 (Total number of persons = 30)

> listTableAllocations|24-Mar-2019
Allocated tables:
[None]

Available tables:
T01 T02 T03 T04 T05 T06 T07 T08 T09 T10 F01 F02 F03 F04 F05 F06 H01 H02 H03

Total number of pending requests = 0 (Total number of persons = 0)

> suggestTable|21-Mar-2019|1
Suggestion for 10 persons: H01 T01

> suggestTable|22-Mar-2019|4
Suggestion for 30 persons: H01 H02 H03 F01 T03
```

The above test case and all others are available at the course web.

Note:

1. When you do this assignment, follow the listing of stages shown in page 4. For example, Phase 1 (up to B-) only requires to handle booking request, startNewDay, and listing of bookings. (For Phase 1.1, the testcase contains one booking only. Thus it does not really need a full ticketing system.) In Phase 2 (up to A-) you will need to handle assignment of tables and listing of table allocations. In Phase 3 (up to A+) you will need to handle cancellation of booking and suggesting tables.

2. Additional requirements:

   (i) To view records clearly, the listing of bookings should be sorted by customer name, then by phone number, then by the target dining date.

   (ii) For each booking record in the listing, the codes of the assigned tables should not be sorted. That is, the sequence should be the same as the ordering of the input.

   (iii) For listing of table allocations, the allocated tables and available tables should be sorted by table size, and then by table code.

   (iv) For listing of suggested tables for a booking (command *suggestTable*), the tables should be sorted by table size in descending order, then by table code in ascending order.

   (v) **State Pattern** should be used for the status of booking (Lab06). There should be an interface in your source files named **RState.java**, and the implementing classes in source files named **RStatePending.java** and **RStateTableAllocated.java**. Inside RStateTableAllocated, you can have an object field that refers to an arraylist of table objects.

   (vi) Starting from Phase 2.3, you must implement a class named Table (in **Table.java**) to model the tables as objects. In phases 2.1-2.2, it is possible (though not recommended) to deal with tables easily using an arraylist of strings for the table codes.

   Concerns about efficiency: Assume that we have 100 bookings everyday. For moden computers, keeping 365 x 100 requests = 36500 records and providing sorting/searching are not a problem. Therefore, please spend more time and effort on modelling the entities involved in the case study.

   Commands to be implemented:

   - request - make a booking
   - cancel - cancel a booking
   - assignTable - assign table(s) to a booking
   - suggestTable - tell the system to suggest table(s) for a booking.

     [Hint]

     Basically the system could first look for the biggest available table(s) best-fit for the booking (but not larger than neeed total seats), then make up with smaller-size tables.
     For some cases this basic approach doesn't produce the best result (e.g. suppose T01, T02, and F01 are available. For 3 persons, the approach would generate "T01 + T02". However "F01" should be better). Please refer to the given test cases and expected outputs for reference.

     If your program comes up with a different output that is reasonable, please notify Helena by email (cshwong@cityu.edu.hk). You'll not loss any mark if your approach and design are appropriate.
   - listReservations - listing of bookings
   - listTableAllocations - listing of allocated/available tables, and total pending cases on a business day.
   - startNewDay - advance the system date.

3. All command classes should start with prefix: "`Cmd`", eg. "`class CmdRequest`", "`class CmdCancel`"

4. You will need to add handling for the following error cases:
   a) Booking by the same person for the dining date already exists (For request booking)
   b) Booking not found (For assignTable, suggestTable, cancel)
   c) Date has already passed (For request booking, assignTable, cancel)
   d) Not enough seats for the booking (For assignTable)
   e) Table already assigned for another booking (For assignTable)

f) Table already assigned for this booking (For assignTable)
g) Table code not found (For assignTable)
h) Insufficient command arguments (For all commands, eg. missing ticket code in the cancel command)
i) Wrong number format (For where numeric value is needed, eg. the ticket code)
j) Unknown command (Checking in the main loop in `main`())

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

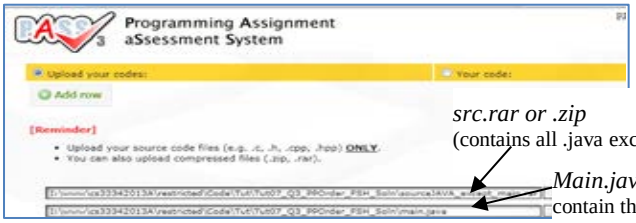
Refer to the following and the given files at the course web for the requirements in each phase of your work and the exact outputs. You may also copy some code contents from given_code.txt (Check course web!)

Also required: good programming styles + OO programming skills

Requirements of Each Phase:

Phases	Execution of commands	Requirements of proper undo/redo	Requirements of Exception handling / checking	Test case for reference	Maximum Grade
Phase 1					
Phase 1.1	Request booking (One booking only, ticket code = 1) List bookings Start new day	--	--	1a.txt	~C
Phase 1.2	Same as 1.1, but several bookings	--	--	1b.txt	~C+
Phase 1.3	Same as 1.2	undo/redo of Request and Start new day	--	1c.txt	~B-
Phase 1.4	Same as 1.2	Same as 1.3	Wrong number format Insufficient command arguments! Booking by the same person for the dining date already exists Date to book has already passed	1d.txt	~B
Phase 1.5	Assorted (1.1-1.4)			1e.txt	~B
Phase 2					
Phase 2.1	Phase 1 items + assign table	--	--	2a.txt	~B+
Phase 2.2	Same as 2.1	undo/redo	Booking not found Table(s) already assigned for this booking	2b.txt	~B+
Phase 2.3	Same as 2.1 + list table allocations	--	--	2c.txt	~A-
Phase 2.4	Same as 2.3	undo/redo	--	2d.txt	~A-
Phase 2.5	Same as 2.3	undo/redo	Checking related to assign table and listing	2e.txt	~A-
Phase 3					
Phase 3.1	Same as 2.5 + Cancel booking	--	--	3a.txt	~A
Phase 3.2	Same as 3.1	undo/redo	Checking related to cancel booking	3b.txt	~A
Phase 3.3	Same as 3.1 + Suggest tables (simple cases: 2,4,8 persons)	--	Checking related to suggest tables	3c.txt	~A
Phase 3.4	Same as 3.1 + Suggest tables (more cases)	--	Checking related to suggest tables	3d.txt	~A+
Phase 3.5	Assorted			3e.txt	~A+

Submission - Please submit them to PASS as shown below:



src.rar or .zip
(contains all .java except *Main.java*)

Main.java (This class should contain the *main* method)

-- end --