[Lab 6: Final Project]

About the project

With the final project, we are now pulling it all together into a user-friendly care facility management application. The scenario described below is similar to the previous labs, however, the implementation approach is left opened to you. You will pick any class/package from your previous labs that you see fit to design the solution. Additionally, any data structure and algorithm provided in the standard java library can be used. However, at the end of your implementation, an explanation will need to be provided to justify your choice of the different algorithm (search , sort) and data structure (list, array, stack, queue, tree…), focusing on the efficiency of your program.
Finally, user friendliness will need to be taken into consideration when designing your solution. Either an intuitive command based interface or a GUI (Graphical User Interface) should be provided to allow the end-user to easily navigate through the system.

Scenario

A Care facility provides bed and care for long-term care patients. When patients are registered, they are put on a waitlist and later assigned beds, based on their criticality as well as bed availability. Patients that have the same level of critically, will be assigned beds on first come first served basis.

Additionally, the care facility employs hourly employees and casual employees. You have been tasked with creating a solution to help manage the facility.

System Requirement

The system should be able to provide/meet at least the requirements below. Additional requirements can be added at your own discretion for a bonus mark

1. **Patient:** ability to manage patient
    a. Registration (first name, last name and priority)
    b. Modification/correction of patient name or priority
    c. Admission: being assigned a bed and occasionally a casual employee
    d. Discharge: when they no longer need the bed/casual employee. The bed/casual employee is reassigned to a different patient and patient is taken off the list of patients
2. Bed
    a. Ability to add bed: bed details include name (eg: bed1, bed2,---) and location (eg: floor1, floor2,…)
    b. Ability to remove a bed, if it is not already assigned to a patient
    c. Ability to change the status of the bed:
        i. Available: can be assigned to a patient

ii. unavailable: there is no patient in the bed, but cannot be assigned to a patient either. This may happen when the bed needs some repairs or replacement

iii. assigned: already assigned to a patient and cannot be assigned to another patient. However, when the patient is discharged, the bed becomes available for the next patient

d. Employees

i. Ability to manage both fulltime and casual employees

1. Add employee : name and salary (for fulltime employees) or hourly pay (for casual employees)
2. Update employee
3. Remove employee
4. Change the status for casual employees
    a. Available : can be assigned to a patient
    b. Assigned: already assigned to a patient and cannot be assigned to a second patient
    c. Unavailable: cannot be assigned to a patient

e. Reports

i. Ability to run multiple reports

1. List of patient on the waitlist: registered, but not yet admitted
2. List of patients already admitted
    a. Those that have been assigned both a bed and a casual employee
    b. Those that have only been assigned a bed
3. List of bed with corresponding status
4. List of fulltime employees
5. List of casual employees with corresponding status

[Lab 6: Final Project]

Grading Scheme

Item	Grade
User friendliness (GUI, Command Based)	10
Data structure and algorithm selection	15
Class implementation	10
Reports	5
Documentation	5
wow factor	5
	_____
Total	**50**

Please note that the focus of this lab is on giving you an opportunity to learn and put in practice everything you learned in class, and "possibly" have fun while doing it.

To answer the question posed earlier: " ….. at the end of your implementation, an explanation will need to be provided to justify your choice of the different algorithm (search , sort) and data structure (list, array, arraylist, stack, queue, tree…), focusing on the efficiency of your program.
please use the template below

Data Structure		type	Reason
	Patient	eg: queue	
	Bed		
	Casual Employee		
Search & sort			
	patient search	eg: binary search	
	…….		
	……		
	patient sort	eg: insertion sort	
	…….		
	……		