Assessment details for ALL students

Assessment item 2 — OOP and Data Structure

Due date:	Friday of Week 10	**ASSESSMENT**
Weighting:	20%	**2**

Objectives

- Demonstrate an understanding of Object-Oriented Programming concepts in Java.
- Gain practical skills in Graphical User Interface (GUI) programming by implementing an event-driven interface.
- Develop and test stand alone Java applications.
- Evaluate algorithm, data structure and program designs used in developing Java applications.

Assessment Task

Specification for the Nursing Home Data Manager

Introduction

For the second assignment, you will develop a windowed application to assist the staff of a nursing home in Queensland to maintain and manage a list of personal data of their residents.

Overview of the application

In a real-world software application, you would ideally use a database to store personal information. In this application however, you will use a data file to store the information. For simplicity, only 5 attributes of a resident will be stored: resident name, room number, age, gender and the care level required.

In a typical session of the application, the resident data file will be loaded from the disk and displayed. The nursing home staff can then view, sort, search, as well as update the information. The updated information can be saved back to the file. This application is an exercise in text file processing, sorting and searching a list. As well, you get to create new GUI components such as the **tabbed pane** component.

The GUI

The GUI you will create should resemble the two screens shown below.

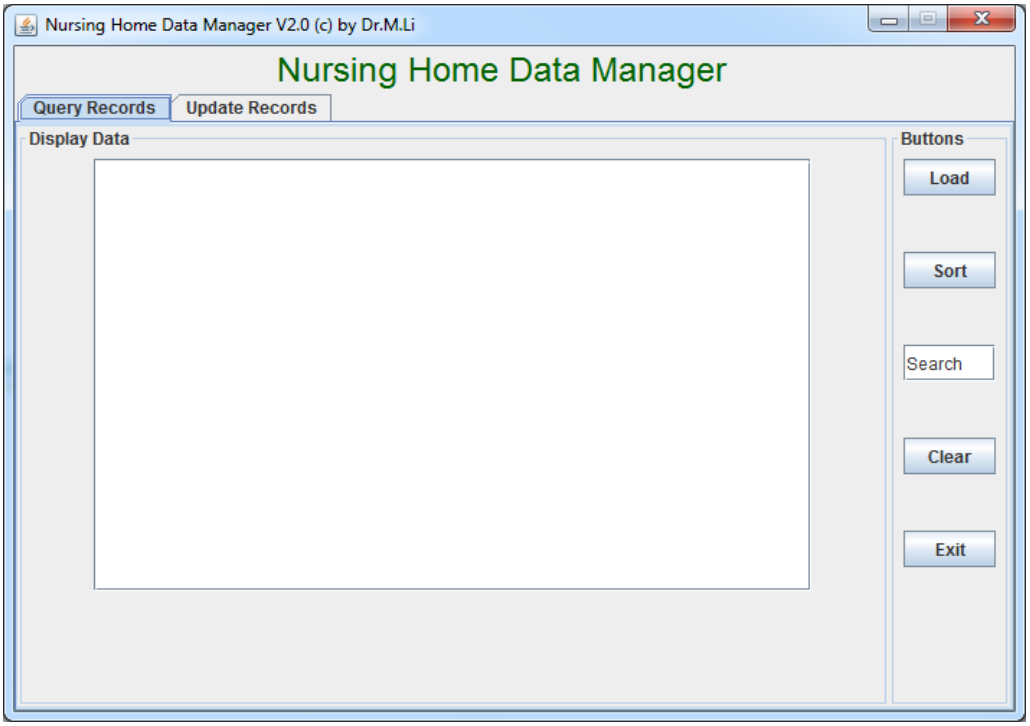


Figure 1

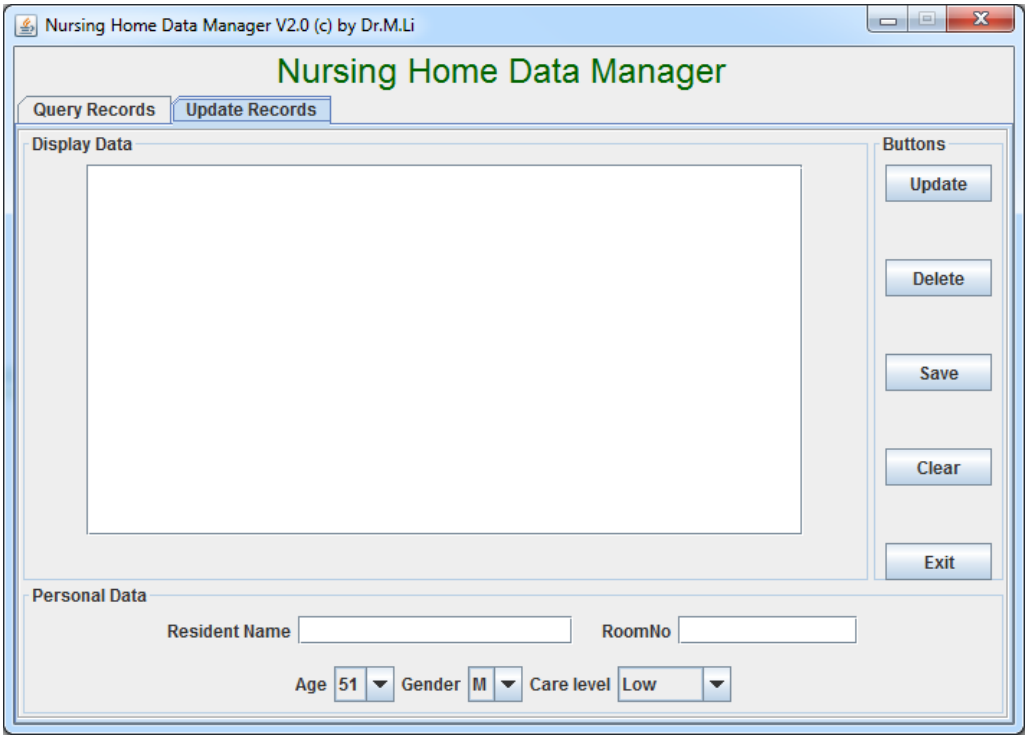


Figure 2

As you will note from the screens, the GUI has two tabs: **Query Records** and **Update Records** as shown in Figure 1 and Figure 2. The first screen is displayed when the **Query Records** tab is selected and the second is displayed when the **Update Records** tab is selected. The **JTabbedPane** component is used to create the two tabs. With tabbed panes, you can have several components such as panels that share the same screen space.

The GUI components on the first screen as shown in Figure 1 are as follows:

- A label for the application title – Nursing Home Data Manager
- The Query Records tab which contains the rest of the components on the screen namely:
    - A text area for displaying a list for resident data; and
    - A panel containing 4 buttons and 1 text field.

The GUI components on the second screen as shown in Figure 2 are as follows:

- A label for the application title – which is the same one as in the first screen since it is outside of the tabbed panes.
- The Update Records tab which contains the rest of the components on the screen namely:
    - A panel containing 2 text fields and their corresponding labels, 3 JCombobox components and their corresponding labels;
    - A panel containing a text area for displaying the resident data. This text area is different from the one in the Query Records tab; and
    - A panel containing 5 buttons.

<u>Some details of how the program functions</u>

1. In a typical session of the application, the Query Records tab is selected by default. Normally the first action is to load the file by clicking the 'Load' button. The content of the loaded file is displayed on both the Query Records tab text area and the Update Records tab text area. Below is the typical screen when the file is first loaded (Figure 3). Note that if the data were not loaded but the 'Sort' button is pressed, a message like "Data to be loaded" should be displayed on the text area.

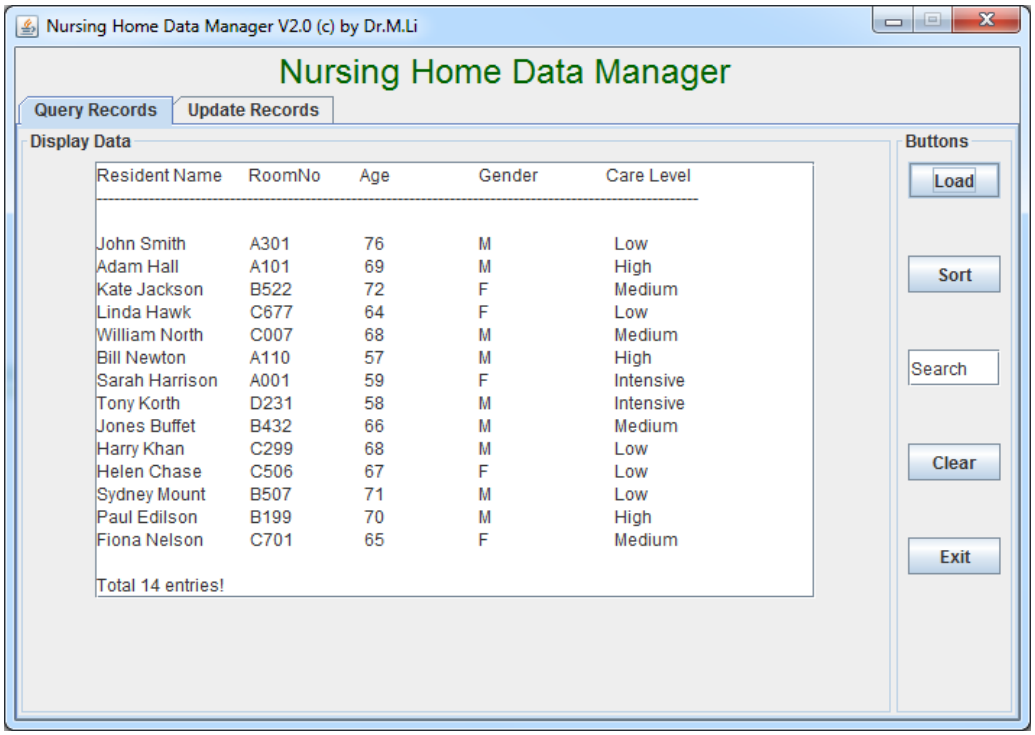


Figure 3

2. Here is the screen on which the list is sorted by name when the 'Sort' button is pressed.

   Note: Use any of the sorting algorithms covered in this course to sort the list. We would like to recommend you using the resident class with implementation of *Comparable* interface so that two resident objects can be easily compared by their names. You can use built-in sort algorithms from java package.

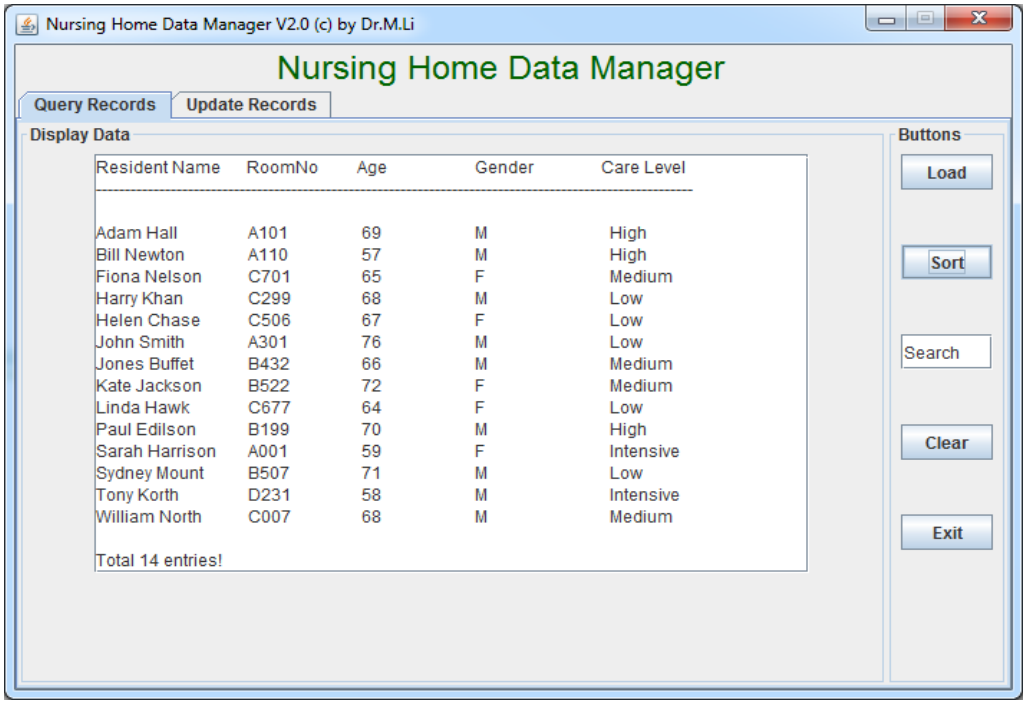


Figure 4

3. The ‘Search’ text field requires entering a resident name regardless of lower case or upper case spelling and then clicking the “Enter” key on the keyboard as shown in Figure 5a. If the resident name exists in the list, the corresponding information of this resident is display on the text area as shown in Figure 5b; otherwise the message – “No result found” is displayed on the text area.

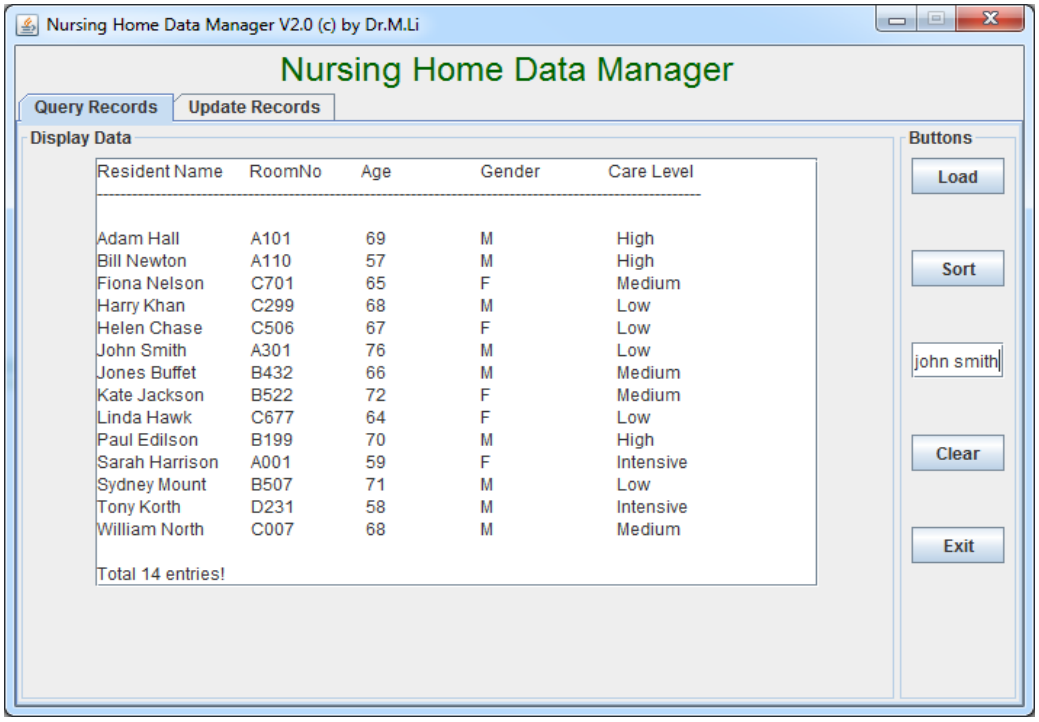


Figure 5a

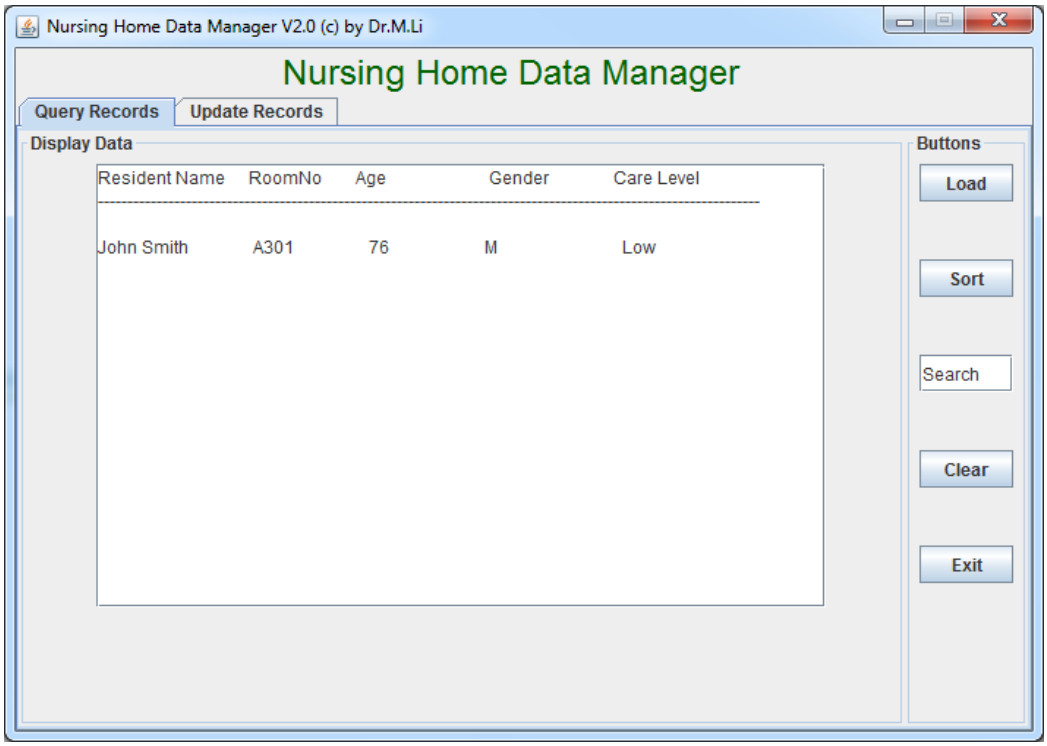


Figure 5b

4. Now switching to the Update Records tab, here is the screen when the information for a new resident has been added to the list and displayed as shown in Figure 6a (before the 'Update' button is pressed.) and Figure 6b (After the 'Update' button is clicked, where the text fields and JCombobox are required to return to the initial default status.).

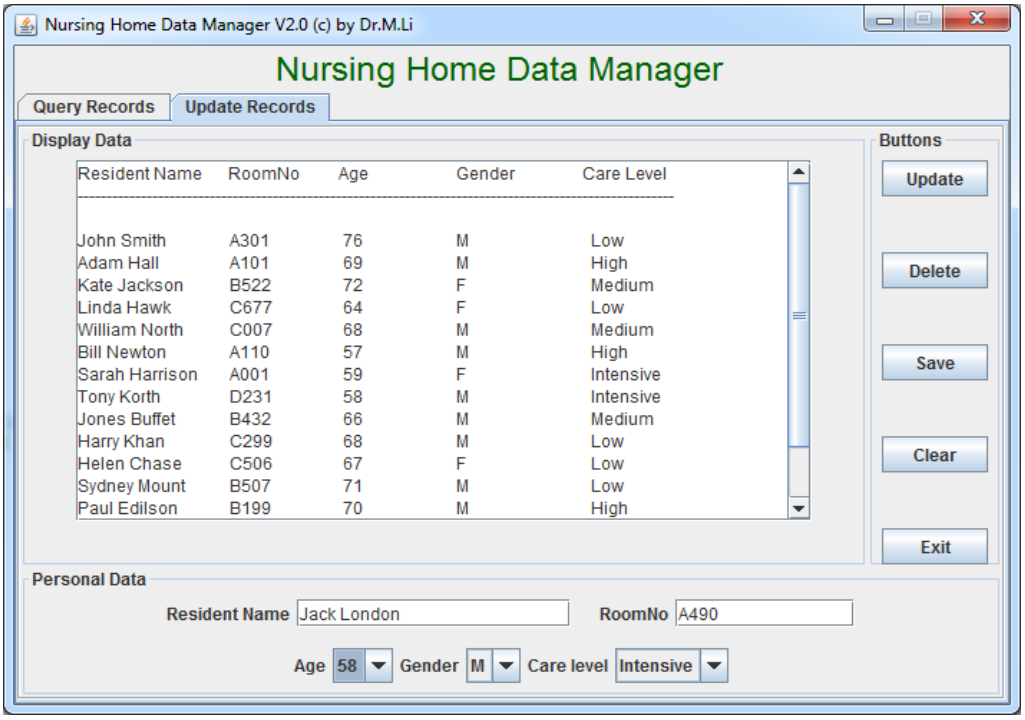


Figure 6a

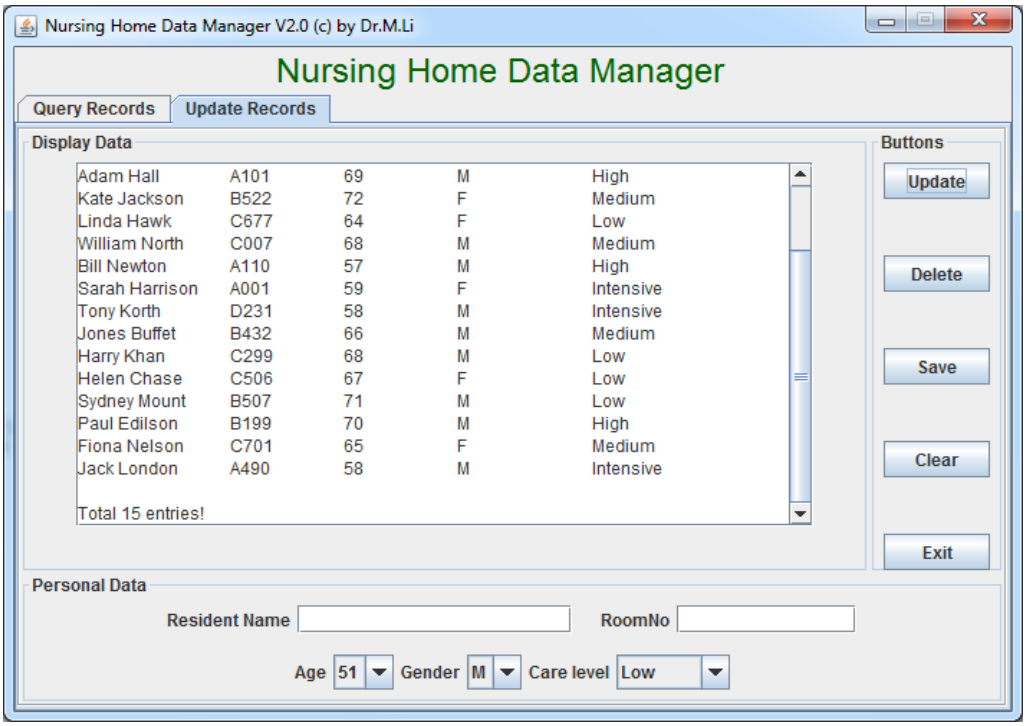


Figure 6b

5. When the ‘Update’ button is clicked, the program must first validate the resident information entered – the name input can be any string except blank but the room number must be a combination of a letter (at the beginning) plus 3 digits (like A380). If the resident is already in the list, then only his/her room number, and care level can be updated. Otherwise a new resident entry with all the information is created, added to the list and then displayed. The program needs to search by the resident name to determine whether or not a resident entry exists.

6. The ‘Delete’ button requires entering a resident name to be deleted from the list via the name text field as shown in Figure 7a. After clicking the ‘Delete’ button, the patient data (if existed) is removed with the confirmation message as Figure 7b. If the entered resident name is not existed in the list, a message box with the content “Resident name not found” pops out.

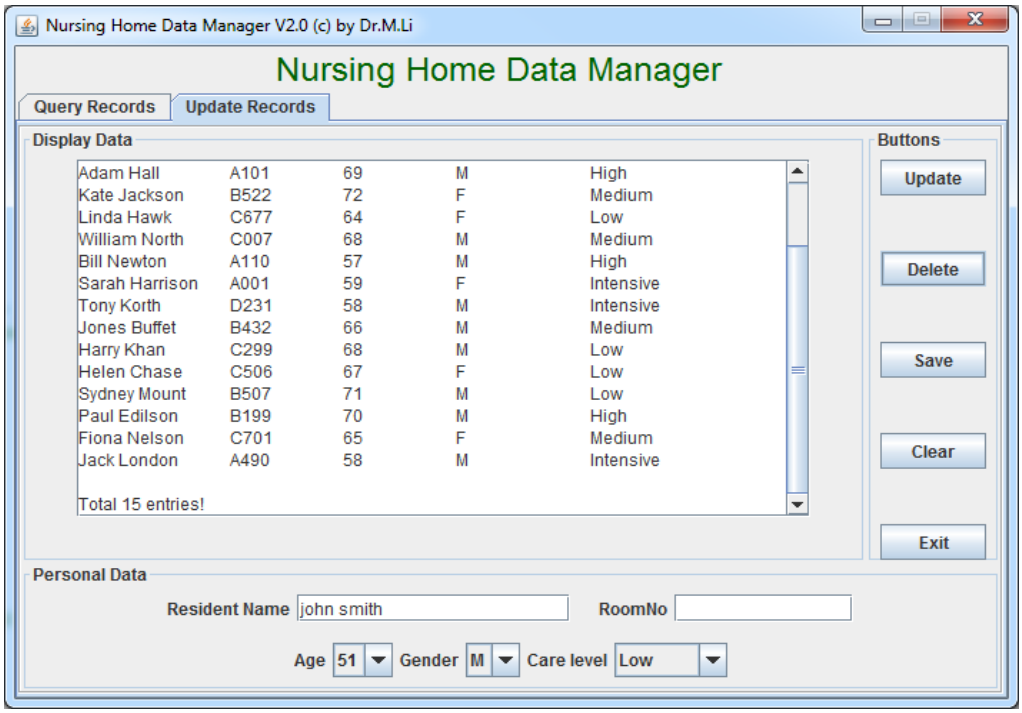


Figure 7a

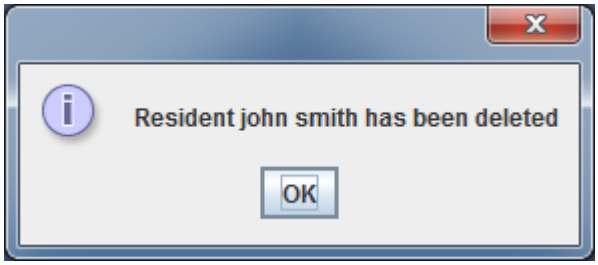
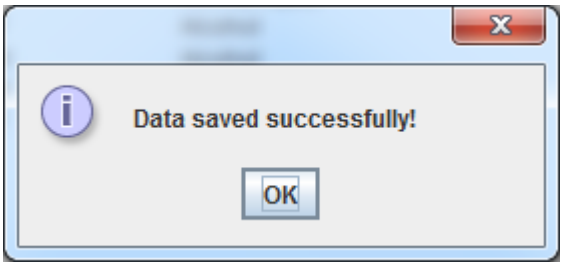


Figure 7b

7. The ‘Save’ button saves the updated list to the file. A confirmation message should be given to the user if the old file is successfully overwritten with the updated one.



8. The ‘Clear’ button in each tabbed pane clears the text area in that pane. And the Exit button in both panes terminates the program.

9. Important: Structure your program by using helper methods. The main technique is to identify the main tasks and then break these down to small manageable tasks that can be translated easily into suitable named methods.

Data storage structures

1. A text file called Resident.txt will be used to store the resident information. This file will be made available to you on the course website.
2. The resident list is loaded from file into a Linkedlist or Arraylist of Resident objects. For this you will need the Resident class file - Resident.java. For convenient comparison of Resident objects, we suggest you to implement *Comparable* interface based on comparison of the resident name for this class.
3. The Resident class should declare suitable variables to store resident name, age, gender, room number and the care level and then only defines relevant get, set, toString( ) methods. ResidentManager class contains the implementation of GUI, event-handling, data file reading and writing, and other necessary methods.
4. Use meaningful variable names to store all values required in the application.

Implementation platform

You will implement your program in Java using either the TextPad Editor (strongly recommended) or NetBeans. NetBeans version 6.9 or later is available in University computing labs but may also be downloaded from the following site: http://netbeans.org/downloads/index.html

TextPad is also provided in the University computing labs. It can also be downloaded from

http://textpad.com/download/index.html

Assignment submission

1. You must submit your assignment using the Moodle online submission system.
2. You will submit three (3) files:

   a) The source code of your program (file name)-ResidentManager.java

   b) the Resident.java file

   c) the Resident.txt file.

You must zip up your files and name the zip file Assignment2.zip before submitting. Do NOT submit an executable (.class) version of your program, it will not be marked.

3. IMPORTANT: You must make a back up of your assignment before submitting in case your submitted copy becomes corrupted.

Assignment 2 Marking criteria

Item	Marks
GUI overall presentation	3
Query Records tab	
Load	3
Sort	2.5
Search	1.5
Update Records tab	
Update	3
Delete	2.5
Save	1.5
Clear & Exit function	0.5
`Resident` class implementation	1
Input validation	0.5
Quality of code (comments, indentation, naming, and readability etc)	1
Total	20
Deduction (Plagiarism, late etc)	
Final total	

Note: • If your program doesn't compile or run, partial marks will be allocated by inspection of the source code.