
College of AI, Cyber, and Computing
Department of Computer Science

**Fundamentals of Object-Oriented Programming**
**CS 2113 (Section 3)**

# Project 3 – Patient Management Subsystem

---

Instructor: Erfan Nourbakhsh

Fall 2025

You are an employee at a software development company that has been contracted by a medical company to create a new medical system to help run their hospitals. You have been put on a team that will create the patient management subsystem. You have been tasked with building some base classes to start off the project.

The primary goal is to be able to perform a basic User login by Patients and Medical Staff, have Users be able to view their profile information, have Patients able to edit their profile information, have Medical Staff able to view and edit Patient information, and produce some reports for hospital business and oversight.

## Users

For initial testing and setup, we will only have two types of users in the system: Patients and Medical Staff. The information stored for each type of user is:

Patient - id, username, password, name, email, treatment_notes

Medical Staff - id, username, password, name, email, department

These users should inherit from a generalized class (for ease of data and behavior management). The general class should NOT be able to be instantiated as an Object in the system. Only Patients and Medical Staff should be able to exist as Objects. Both types of users will need to be able to get/return and set/modify all their information/attributes with the exception of id and username; the id and username CANNOT be changed once it has been assigned to a user. Further, both types of users will also need to be able to easily print out their information/attributes (Hint: using toString and/or declaring a method in the general class, to be defined in the specialized classes, will be ideal so it will be much easier to call the behavior without having to check the type of user later, as described below).

## Driver

There should be a Driver class that will run the main method. The Driver will instantiate/call/execute Login. Upon a successful Login, a Patient Manager object will be returned. The Driver will then utilize the Patient Manager to perform various tasks that the user wishes to perform. The user will be interacted with through menus and text prompts. Some general guidelines/requirements are detailed here about which classes handle which responsibilities/requirements, but it is up to you to decide exactly how the menus/prompts are constructed/formated, exactly what the menus/prompts say, and exactly which classes do the prompting and user input processing. (Note: it is generally a good idea to have as much user interaction/processing performed by one class (or a set/subsystem of classes) rather than spreading out the responsibility (for better code organization and ease of maintenance/debugging); e.g., you may want to contain as much prompting and input processing to the Driver class as possible).

## Login

Create a Login class that will perform a basic login function. You will be given two files, one that contains all Patients and one that contains all Medical Staff. Create a Login class that will ask a user for their username and password. The class will read the files and check each Patient and/or Medical

Staff to determine if BOTH the username and password match a record/line/Patient/Medical Staff in the files. If one is found, then a new user should be created; if no record/line in the files match, then an error message should be printed and the user should be asked to re-enter their username and password (you may also choose to throw an error message back to the main method, for Driver to handle and then recall/execute Login). It is up to you if you wish to have a single login for both types of users (which is more user-friendly, but you will need to check both files for a possible match) or separate logins by first asking if the user is trying to login as a Patient or Medical Staff (less user-friendly, but you only need to check the file that corresponds to the type of user that is attempting to login).

When a login is successful (i.e., a username and password match is found), Login will be responsible for setting up and returning the Patient Manager.

## Patient Manager

A Patient Manger will be instantiated on successful login. The manager will hold the user (created by login) that is currently logged into the system and an ArrayList of all Patients. (Note: you may have other attributes for the Patient Manager, but you will need, at least, the currently logged in user and the list of Patients). The Driver will interact with the user to ask what functions the user wishes to perform, and will then call the appropriate methods in Patient Manger.

The user that the manager will hold should be declared/typed as the general user class, though it will be instantiated/defined as the specialized type by Login, for ease of management. Regardless of the user type, the user should be able to view their profile (i.e., attribute) information. However, only Patients should be able to edit their information.

For now, we are primarily concerned with the viewing and editing of Patient information (since that will be the most commonly accessed and manipulated information in day-to-day operations). Patients should only be able to view and edit their own information. However, Medical Staff should be able to view and edit any/all Patient information. To accomplish this, when the manager is created, it should load ALL Patient information (from the patient file) into the ArrayList of Patients (i.e., instantiate a new Patient and add it to the ArrayList for EVERY Patient in the file). Then sort all Patients by id (Hint: all sorting and searching is, likely, best created in their own methods for ease of reuse). You will need to implement all search and sorting algorithms yourself (i.e., you CANNOT use any of the sorting algorithms that comes with the Java JDK or may be available in other libraries) (Note: defining some additional methods, in the user classes or the manager may make these processes easier, though is not necessary).

The manager should have three methods, one to view the currently logged in user's profile information, one to view/lookup a Patient, and one to edit the current Patient that is being viewed:

- For viewing the currently logged in user's profile, the manager should simply print/display the user object's attributes (as noted at the end of the "Users" section above).
- For the viewing/looking-up method, a user that is a Patient should throw an error back to Driver as a Patient should only be able to look up their own information. Medical Staff should be able to search for a Patient by the Patient's id using Binary Search. If the id is found, then that Patient should be set as the "currently viewed" Patient (Hint: this can be an attribute in Patient Manager if it makes it easier; if the variable used to indicate the currently viewed Patient is declared as type Patient, make sure NOT to instantiate a new Patient object, use the SAME

object that is in ArrayList; also, if a Patient is logged in, you may automatically set the "currently viewed" Patient as the logged in Patient, since that should never change and a Patient can only view themself) and that Patient's information/attributes should be printed/displayed.

- For editing the currently viewed Patient, the user should be able to indicate which attribute to change and what the new value should be (we will not be doing any validation on the values entered; e.g., if a person is changing their email, we would normally check that whatever is entered is in a valid email format, but we will not worry about that for this project) except for id and username which (as discussed earlier) should NOT be editable. (Note: make sure that the Patient's changes are reflected in the ArrayList; i.e., make sure the object you modify is the same object that is held by the ArrayList, not a separate/different object). Finally, after modifying a Patient, the changes should be saved/reflected in the Patient file. Overwrite the entire Patient file by printing out the entire contents of the ArrayList (one Patient per line with comma-separated values). (Note: this is why it is important to make sure the Patient updates are reflected in the ArrayList and not just in the currently viewed Patient variable; it will be easy to define the toString or another method in the Patient class to print out the comma-separated values so you do not have to worry about formatting it in the manager class).

## Reporting

Finally, some simple reporting will be performed for Patients. You may create this as a separate class or integrate it into the Patient Manager class. The reporting should ask the user to give a filename to print/write the report to and to choose a predefined report:

- List of Patients, sorted ascending by id (it will only print out the id, name, and email of the Patients)
- List of Patients, sorted ascending by name (it will only print out the id, name, and email of the Patients)
- List of all emails, sorted ascending alphabetically
- If the currently logged in user is a Patient or Medical Staff and all the user's information/attributes

## Submission

**Submission Instructions:** Please export your ENTIRE project from Eclipse (or whatever IDE you are using) as an archive zip file. Upload the file to the Project 3 submission in Canvas.

Create a `REPORT.txt` file to answer the following questions:

1. List all of the people that you have collaborated with on this assignment (discussing high-level ideas are OK, but never share implementation details or code). For each person indicate the level of collaboration (small, medium, large). Also write a few sentences describing what was discussed. Indicate whether you were mainly giving help or receiving help.
2. Do you think everything you did is correct? If not, give a brief description of what is

working and what progress was made on the part that is not working.

3. Comments (e.g., what were the challenges, how to make this assignment more interesting etc.).

Also, you will have a presentation for this project. The format and details of the presentation will be announced later.

**Note:** You may complete this project individually or in a group with **a maximum of 2 members**. Your code should be well-commented, as this will be part of the grading.