**I. Project Overview**

{Program name} is a console-based application that is designed to assist small clinical institutions in managing the recipients of their services. This program allows designated personnel to register patients and visitors, view the clinic’s schedule for up to five (5) days’ time—skipping weekends of course, whence the clinic may be closed, create appointments for the schedule slots that are open, and attach records of different types to existing patients/visitors.

**II. Explanation of how OOP principles were applied**

{Program name} demonstrates the following core principles of OOP:

1. **Abstraction**

A great application of Abstraction in the code can be found in *Input.java*. Within it is the choice() function and the line() function. Both utilize logic to handle inappropriate input and allows the return of Paired values, which in turn, allows the program to do things like operation cancelling, controllable error log printing, and caching.

Abstracting this code allows the program to hide the extra bulk and complexity, letting the programmer to use it later on with a simple instance method call.

1. **Inheritance**

A great application of Inheritance in the code can be found in *Note.java*. The Note class is the parent class for the following: the Record class, the Vaccine class, and the Prescription class. The child classes have differing constructors, attributes and functionalities, but they all inherit from the parent Note class. Since the Resident class allows the binding of notes to its instances, any of the three child classes can be attached.

1. **Polymorphism**

The *Pair.java* class allows the creation of paired values. Examples are a pair of a *Boolean*: validity and an *Integer*: choice. The validity had been utilized to check if the choice should be accepted by the program. If the validity is true, then the program allows the processing of the choice. If invalid, however, the program displays an error log.

Expanding on this, since a Boolean can only hold two possible values, we can instead use a pair of an *Integer*: state and an *Integer*: choice. This allows the program to handle not only an invalid input, but an out-of-bounds input as well. (*Try entering a number that is not in the menu, or a non-numeric character in the choice field to see this particular Pair in action.*)

The Pair class is polymorphic, allowing a pair of different arguments to be passed through the same constructor.

1. **Encapsulation**

Getters and Setters are standard and can be found everywhere in the code, making sure that unwanted data changes will not accidentally occur, and that values will never take on an invalid format. An example would be the Resident class. The sex of the resident, in particular, takes on the form of an integer ranging from 0-2 (0: MALE, 1: FEMALE, 2: OTHER). Without the use of a proper setter, it can be easily updated to a value that is outside the given range.

**III. Explanation of how OOP principles were applied**

**SDG 3: Good Health and Well-being**

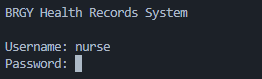
The program plays an important role in optimizing clinic operations by managing schedules, ensuring that appointments are organized efficiently. It allows the clinic staff to create and track appointments, reducing scheduling conflicts and improving patient flow. By integrating these functions, the system ensures a smoother daily operation, enabling the clinic to provide timely care to patients.

Additionally, the program's ability to create and manage resident records integrates directly into the clinic's operational workflow. By linking patient information with their appointment schedules, the system helps maintain comprehensive records that are easily accessible to medical staff. This integration not only improves administrative efficiency but also supports better patient care by keeping all necessary details in one centralized location.

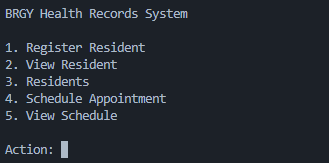
**IV. Instructions for running the program**

Note: this program is meant to be administrated by designated personnel. The administrator will receive patient information documents, records of vaccination, prescription or visit, and approved appointment documents, before they will enter the data into the program.

Upon execution, the program immediately asks for authorization.



Once the user is authorized, they are brought to the main menu:



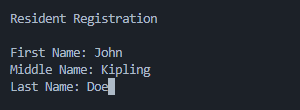
1. **Register Resident**: Initiates the process of resident registration. Make sure that the Resident’s full name, sex, and birthdate are ready.
2. **View Resident**: Shows the user an input field. Entering a Resident’s ID displays the Resident’s details in full.
3. **Residents**: Shows the entire list of Residents that are registered.
4. **Schedule Appointment**: Shows the schedule for up to five days’ time, along with how many appointment slots are open for each day. Open slots can be selected to create an appointment for a registered resident. Taken slots can be selected to mark them as completed, creating a visitation record.
5. **View Schedule**: Shows the schedule for the day.

**Register Resident**

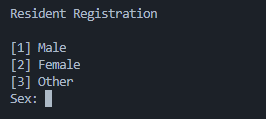
*Note: Pressing Enter while any field is empty will show this message. Subsequently entering “Y” will terminate the process. Entering anything else will allow you to continue where you left off.*



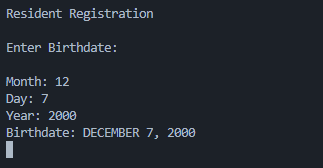
The first fields will ask for the full name of the Resident, starting from the First Name, Middle Name, and finally, Last Name.



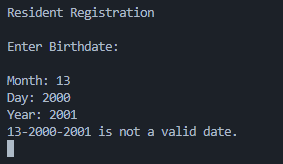
This is followed by this field, asking for a number which maps to either MALE, FEMALE, or OTHER. Entering a number outside the expected range will prompt cancellation.



Finally, the program asks for the Resident’s birthdate.

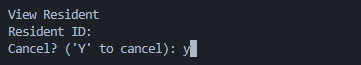


Invalid dates will not be accepted, and will prompt the user to enter the date again.



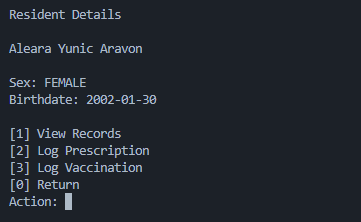
**View Resident**

*Note: To exit this menu, press Enter while the field is empty. Then Enter ‘Y’ to terminate operation.*

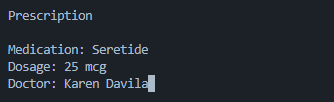


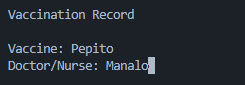
To view a specific resident, enter the Resident’s ID. You may view the Residents and their IDs in the **Residents** option in the main menu.



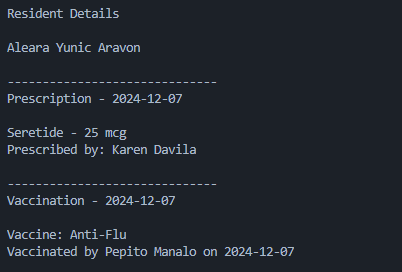


The Log options show as follows:



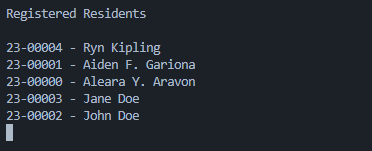


With existing records, the View Records option will have something to show.



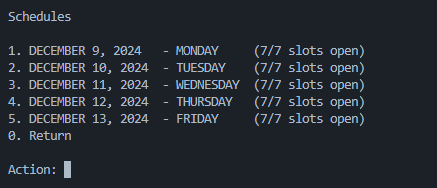
**View Resident**

This option simply shows a list of registered residents with their IDs

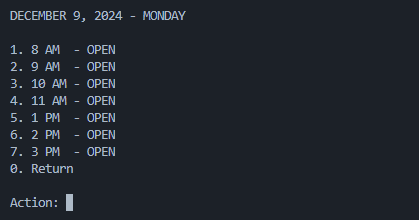


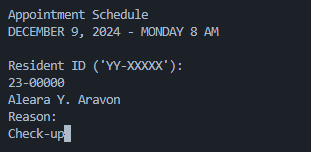
Note: These are only example data, automatically registered by the Preload() function in *App.java*. To turn it off, simply turn the invoke into a comment.

**Schedule Appointment**



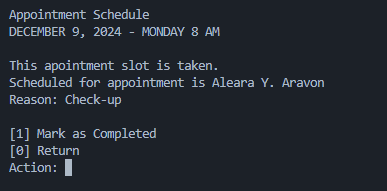
Shown are the clinic’s operating days. Since the date this documentation is written is December 7, 2024 (Saturday), the next open day is December 9 (Monday) and then the next four open days. Shown also are the open slots for each day. Selecting a day will bring the user to the Slot Appointment menu.





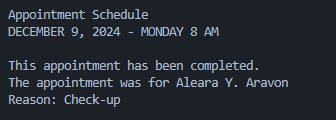
The program will ask for the Resident to be appointed the slot, and then a reason to be recorded for the Visitation log. Upon confirming, the slot will then become taken.

Selecting it again will show the appointment details instead.



By marking it as Completed, it will create a record for the Resident, which can be viewed anytime in the View Residents option.





**View Schedule**

This option simply shows a more detailed schedule for the day.

