Wong Shun Min - Project Portfolio

PROJECT: ClerkPro

Overview

ClerkPro is a desktop application used for managing clinic's appointments, queue and scheduling. The user interacts with it using a CLI, and it has a GUI created with JavaFX. It is written in Java, and has about 10 kLoC.

Summary of contributions

- **Major enhancement 1**: added **the ability to enqueue/dequeue patients into the queue**
  - What it does: allows the user to add patients into the queue based on the `referenceId` of the patient. It also allows the user to remove patients from the queue based on the index number.
  - Justification: This feature has high priority as the user (clerk/receptionist) would need to add or remove patients from the queue.
  - Highlights: This enhancement affects existing commands and commands to be added in future. It required an in-depth analysis of design alternatives. The implementation too was challenging as it required changes to existing commands.
- **Major enhancement 2**: added **the ability to serve patients and assign them to a doctor**
  - What it does: serves the first patient in the queue and assigns them to a doctor based on the index number.
  - Justification: This feature is important as the user (clerk/receptionist) would need to assign patients in the queue to doctors.
- **Major enhancement 3**: added **the ability to keep track of doctors on duty**
  - What it does: allows the user to be able to know how many doctors are on duty. Also, allows the user to know if any doctors went for a break.
  - Justification: This feature is useful as it allows the user (clerk/receptionist) to know how many doctors are on duty and is able to assign patients to them.
- **Minor enhancement**: added compatibility with the undo/redo commands
- **Code contributed**: [Functional code] [Test code] *{give links to collated code files}*
- **Other contributions**:
  - Project management:
    - Managed releases `v1.3` amd `v1.2.1` (2 releases) on GitHub
  - Enhancements to existing features:

        - Updated the UI for doctors on duty and queue (Pull requests #203, #134, #119)
    - Documentation:
        - Did cosmetic tweaks to existing contents of the User Guide.
        - Created various UML diagrams for Developer Guide. #87
    - Community:
        - Reported bugs and suggestions for other teams

Contributions to the User Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

Marks the doctor as on-duty: `onduty`

Marks the doctor, based on the index given, as on-duty and adds him/her to a list of on-duty doctors.

Format: `onduty <ON_DUTY_DOCTOR_ENTRY_ID>`

- e.g. `onduty 1`

Marks the doctor as off-duty: `offduty`

Marks the doctor, based on the index given, as off-duty and removes him/her from the list.

Format: `offduty <ON_DUTY_DOCTOR_ENTRY_ID>`

- e.g. `offduty 1`

Queue Management

Adds a patient to the queue: `enqueue`

Adds a patient to the queue based based on the patient's Id. The enqueued patient must be a registered. Staff members cannot be enqueued.

Format: `enqueue <PATIENT_REFERENCE_ID>`

- e.g. `enqueue E0000001A`

Removes a patient from the queue: `dequeue`

Removes a patient from the queue based on their queue position.

Format: `dequeue <QUEUE_INDEX>`

- e.g. `dequeue E0000001A`

Assigns next patient to an available doctor : `next`

Assigns the next patient in the queue to a doctor.

Format: `next <ENTRY_ID>`

- e.g. `next 1`

Doctor takes a break: `break`

Avoids directing patients to a doctor. e.g. Doctor is on a lunch break

Format: `break <ENTRY_ID>`

- e.g. `break 1`

Doctor resumes his/her duty: `resume`

Allows patients to be directed to a doctor. e.g. Doctor is back from his/her break.

Format: `resume <ENTRY_ID>`

- e.g. `resume 1`

Commands Summary

- **Patient Management**
  - Search for patient using reference Id, name or phone number: `patient [<SEARCH_KEYWORD>]`
  - Register new patient: `newpatient -id <PATIENT_REFERENCE_ID> -name <PATIENT_NAME> [-phone <PHONE_NUM>] [-email <EMAIL>] [-address <ADDRESS>] -num [-tag <Tags>]…`
  - Edits patient details: `editpatient -entry <ENTRY_ID>[-id <PATIENT_REFERENCE_ID>] [-name <NAME>] [-phone <PHONE_NUM>] [-email <EMAIL>] [-address <ADDRESS>] -num [-tag <Tags>]…`
- **On-Duty Doctors Management**
  - Search for doctors using reference Id, name or phone number: `doctor [<SEARCH_KEYWORD>]`
  - Register new doctor: `newdoctor -id <STAFF_REFERENCE_ID> -name <NAME> [-phone <PHONE_NUM>] [-email <EMAIL>] [-address <ADDRESS>] [-tag <TAGS>]…`
  - Edit doctor details: `editdoctor -entry <ENTRY_ID> [-id <STAFF_REFERENCE_ID>] [-name <NAME>] [-phone <PHONE_NUM>] [-email <EMAIL>] [-address <ADDRESS>]-num`
  - Mark doctor as on-duty: `onduty <ENTRY_ID>`
  - Mark doctor as off-duty: `offduty <ENTRY_ID>`
- **Queue Management**
  - enqueue: `enqueue <PATIENT_REFERENCE_ID>`
  - dequeue: `dequeue <QUEUE_INDEX>`
  - Assigns next Patient in queue to doctor: `next <DOCTOR_ENTRY_ID>`
  - Marks doctor on break: `break <DOCTOR_ENTRY_ID>`

    - Marks doctor on resuming work: `resume <DOCTOR_ENTRY_ID>`

- **Appointment Management**
    - Search for appointments: `appointments [<REFERENCE_ID>]`
    - Add new appointment: `newappt -id <REFERENCE_ID> -start <START_TIMING> [-end <END_TIMING>] [-reoccur <INTERVALS> -num <NUMBER_OF_TIMES>]`
    - Edit appointment: `editappt -entry <ENTRY_ID> -start <START_TIMING> [-end <END_TIMING>]`
    - Cancel appointment: `cancelappt <ENTRY_ID>`
    - Acknowledge arrival of patient for appointment: `ackappt <REFERENCE_ID>`
    - List all missed appointments: `missappt`
    - Mark missed appointment as settled: `settleappt <ENTRY_ID>`
- **Duty-shift Management**
    - Search for shift: `shifts [<REFERENCE_ID>]`
    - Add new shift: `newshift -id STAFF_REFERENCE_ID -start <START_TIMING> -end <END_TIMING> [-reoccur <INTERVALS> -num <NUMBER_OF_TIMES>]`
    - Change shift: `editshift -entry <ENTRY_ID> -start <START_TIMING> -end <END_TIMING>`
    - Cancel shift: `cancelshift <ENTRY_ID>`
- **Inventory commands (v2.0)**
    - inventory: `inventory`
    - prescription: `prescription <PRESCRIPTION_ID | PRESCRIPTION_NAME>`
- **User Accounts (v2.0)**
    - login: `login <USER_NAME>`
    - logout: `logout`
- **General Commands**
    - help: `help`
    - exit: `exit`
    - undo: `undo`
    - redo: `redo`

Contributions to the Developer Guide

Given below are sections I contributed to the Developer Guide. They showcase my ability to write technical documentation and the technical depth of my contributions to the project.

Queue feature

The queue feature allows the user to enqueue and dequeue a patient from the queue.

- e.g. `enqueue 003A` - Enqueues the patient with `referenceId` 003A.

- e.g. `next 1` Serves the next patient in queue and allocates him/her to room 1.

Queue supports a few basic commands:

- Enqueue — Enqueues a patient into the queue.
  Format: `enqueue <PATIENT_REFERENCE_ID>`
- Dequeue — Dequeues a patient from the queue.
  Format: `dequeue <QUEUE_INDEX>`
- Next — Assigns the next patient in the queue to a doctor.
  Format: `next <ENTRY_ID>`
- Break — Avoids directing patients to a doctor. e.g. Doctor is on a lunch break
  Format: `break <ENTRY_ID>`
- Resume — Allows patients to be directed to a doctor. e.g. Doctor is back from his/her break.
  Format: `resume <ENTRY_ID>`

Current Implementation

The queue will be displayed in a list.

The following activity diagram summarizes what happens when a user executes an enqueue command:

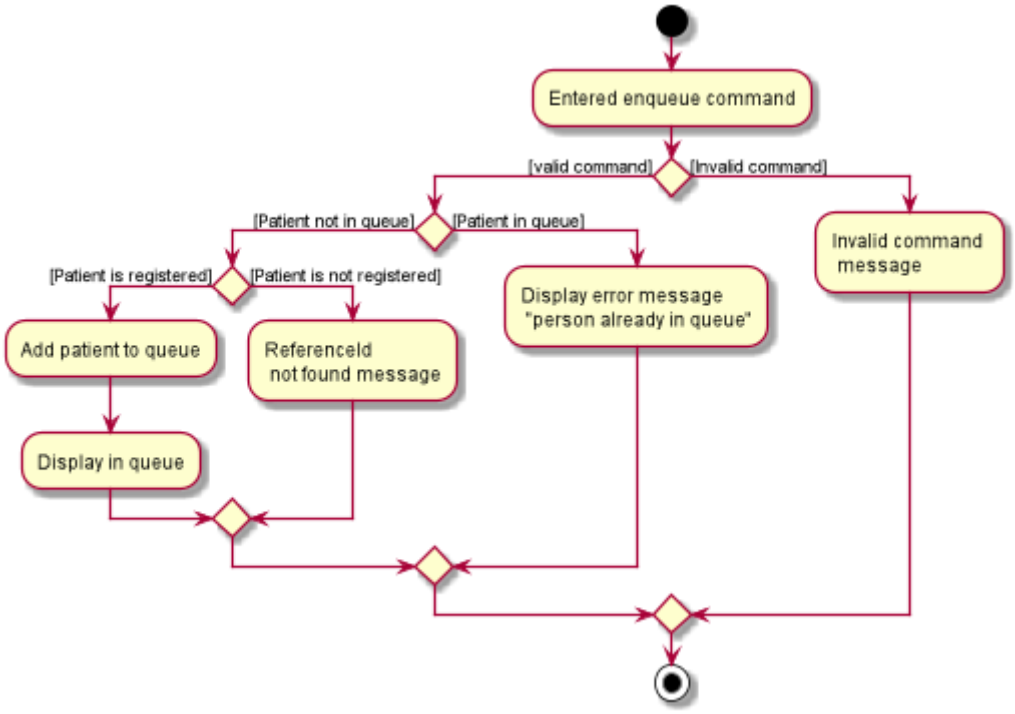

Figure 1. Enqueue Activity Diagram

The following activity diagram summarizes what happens when a user executes a next command:

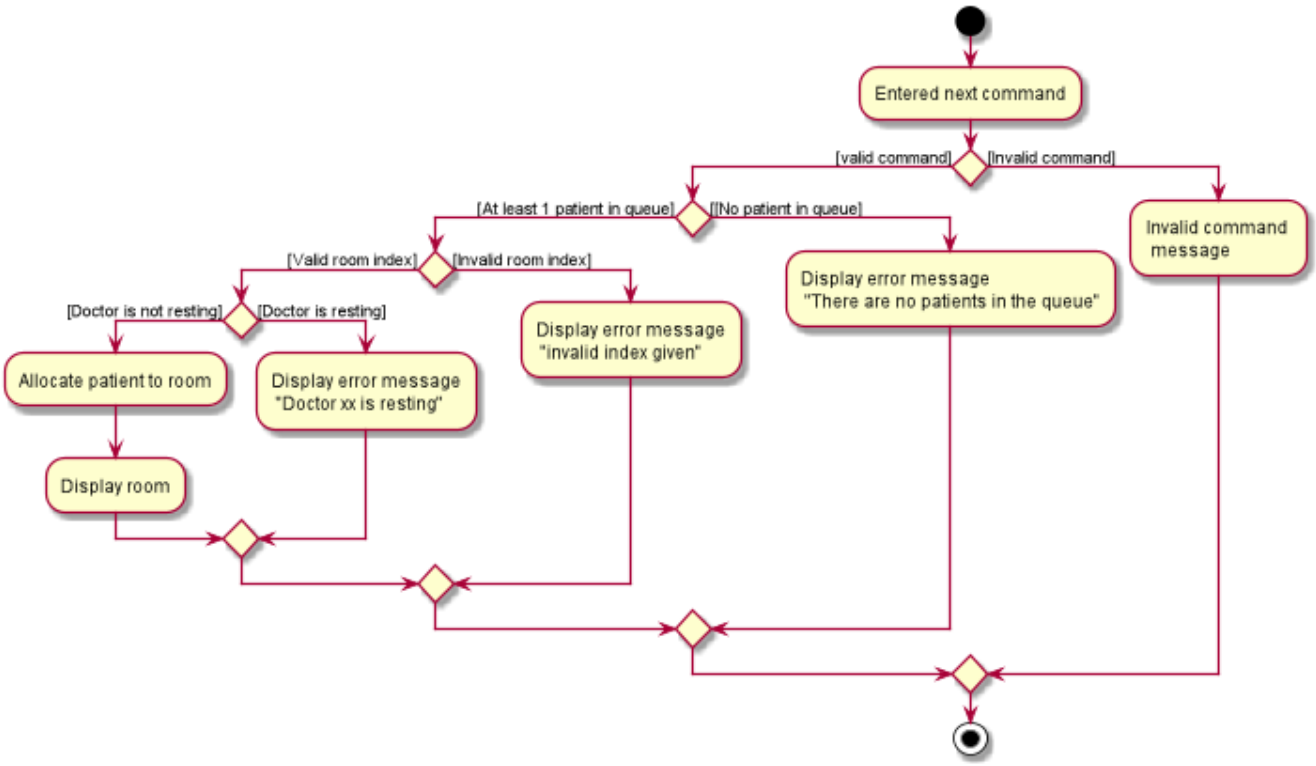

Figure 2. Next Activity Diagram

Below is an example usage of the queue feature.

Step 1: User enters the `enqueue E0000001A` command.

Step 2: The command then calls Model#enqueuePatient to enqueue the patient into the queue.

Step 3: Patient will then displayed in the queue.

Appendix A: User Stories

Priorities: High (must have) - * * *, Medium (nice to have) - * *, Low (unlikely to have) - *

Priority	As a …	I want to …	So that I can…
* * *	new user	see usage instructions	refer to instructions when I forget how to use the App
* * *	clerk	find out the upcoming appointments for given patients	

Priority	As a …	I want to …	So that I can…
* * *	clerk	update the doctors' details by typing commands and user details	
* * *	clerk	add new doctors into system	
* * *	clerk	edit patients' details	keep their particulars up to date
* * *	clerk	register new patients with optional fields	
* * *	clerk	add ad-hoc patients to the queue	
* * *	clerk	search for patients using their name or phone number	
* * *	clerk	look up how many patients are in the queue, on a side panel	recommend estimated time that the patient will be attended to
* * *	clerk	look up patient using a reference id	
* * *	clerk	reschedule appointments of patients	

Priority	As a …	I want to …	So that I can…
* * *	clerk	search for appointment slots easily	schedule appointments for patients easily
* * *	clerk	assign a queue number to each patient in the queue	
* * *	clerk	use the appointment scheduler	schedule appointments for my patients
* * *	clerk	add reoccurring appointments	schedule new reoccurring appointments for my patients
* * *	clerk	save time managing the queue	have more time to do my own work
* * *	clerk	take note of the doctors that are on-shift	effectively direct patients to available doctors
* *	clerk	remove a patient from the queue if they leave.	
* *	clerk	view the number of patients who visited the clinic today	
* *	clerk	schedule patient's follow up appointments	

Priority	As a …	I want to …	So that I can…
* *	clerk	find all patients who have missed their appointments	keep track of the list of patients whom I need to inform
* *	clerk	see relevant information only	so that my focus is not lost
* *	clerk	use auto-complete to predict my commands	save time on verifying its existence and correctness
* *	clerk	quick-fill the command box with the suggestions of Auto-Complete	so that it reduces typing of the entire command
* *	clerk	refer to command history	review entered commands that maybe incorrect
* *	clerk	quick-fill the command box with history commands	inputting last few commands is easier
* *	receptionist	use the undo and redo feature	to remedy any mistakes
* *	clerk	acknowledge appointments if patients are present for their appointments	keep track of patients who came for their appointments
* *	clerk	tag patient with known allergies	keep track of their allegies

Priority	As a …	I want to …	So that I can…
* *	clerk	cancel appointments for patients	free up appointment time slots

Appendix B: Use Cases

(For all use cases below, the **System** is the `ClerkPro` and the **Actor** is the `user`, unless specified otherwise)

Use case: Add patient into queue (UC1)

MSS

1. New patient arrives at the clinic
2. User wants to add new patient into the queue
3. System adds the patient into the queue

Use case ends.

Extensions

- 2a. User inputs invalid format
    - 2a1. System requests for correct input format.

      Use case resumes at step 2.

Use case: Remove person from queue (UC2)

MSS

1. Patient wants to leave
2. User requests to remove patient from the queue
3. System removes the patient from queue

Use case ends.

Extensions

2a. Person is not in queue

Use case ends.

3a. The given index is invalid.

3a1. System shows an error message.

Use case resumes at step 2.

Use case: Serve next patient (UC3)

MSS

1. Patient exits from room 1
2. User requests to allocate patient into room 1
3. System removes the patient from queue and allocates him/her to room 1

Use case ends.

Extensions

2a. Doctor is resting

Use case ends.

3a. The given index is invalid.

3a1. System shows an error message.

Use case resumes at step 2.

Use case: Doctor takes a break (UC4)

MSS

1. User requests to avoid directing patients to the doctor in room 1
2. System sets the doctor to be on break

Use case ends.

Extensions

1a. Doctor is already on break

- 1a1. System shows an error message.

Use case ends.

2a. The given index is invalid.

2a1. System shows an error message.

Use case resumes at step 1.

Use case: Doctor resumes his/her duty (UC5)

Pre-condition: Doctor is on break

MSS

1. User requests to start directing patients to the doctor in room 1
2. System sets the doctor to be on duty

Use case ends.

Extensions

1a. Doctor is already on duty

- 1a1. System shows an error message.

  Use case ends.

2a. The given index is invalid.

  2a1. System shows an error message.

  Use case resumes at step 1.