<WEN SHUFA> - 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

This section's purpose is to summarise my contributions for the duration of this project. For more details about the contributions, click the link to the corresponding pull request.

- **Major enhancement 1**: added **Appointment Management** feature
  - What it does: The Appointment feature enables users to manage appointments for patients by providing basic Create, Read, Update, Delete (CRUD) of appointments. User is also able to find missed appointments and settle each missed appointment efficiently.
  - Justification: This feature greatly improves the product as it meets the necessary demands a receptionist faces on a day-to-day basis, i.e. Add appointments.
  - Highlights: Appointments, in reality, rely heavily on different components such as patients and onduty doctors to be booked. The challenge in implementing this feature was checking appointments time with doctors' duty shifts. It requires an in-depth understanding of clerkPro's architecture.
- **Major enhancement 2**: added **Duty Shift Schedule** feature
  - What it does: The duty shift scheduling provides users the ability to schedule duty shifts for doctors. It can help doctors to check, add, cancel and edit duty shifts efficiently.
  - Justification: This feature is important as the user would need to schedule duty shift for doctor. i.e. change duty shift.
- **Code contributed**
  - The following link contains all the code contributed throughout the entire project: Functional code Test code
- **Other contributions**:
  - Community:
    - Contributed to forum discussions (example: 1]
    - Reported bugs and suggestions. i.e. I mentioned that user should not edit or delete data file in UG.

- Documentation:
  - Added command information regarding the appointment feature into the User Guide:(examples: 1, 2)
  - Added examples to different commands:(example: 1)
  - Added UML diagrams for appointment feature into the Development Guide:(examples: 1)
- Database:
  - Edited pre-load database(Pull request #139)
- Enhancements to existing features:
  - Wrote additional tests for existing features to increase coverage from 66.25% to 71.55% (Pull requests #112)

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.

Appointment Management

NOTE

Before you process any commands in appointment management, please ensure you have a patient with referenceid of E0000001A by using `patient E0000001A`.
You can refer to `newpatient` command to register this patient if the patient is not registered.

Displays appointments: `appointments`

Displays a sorted list of upcoming appointments for the patient that is associated to the given `REFERENCE_ID`. If no keyword is given, all upcoming appointments will be displayed.

Format: `appointments [<REFERENCE_ID>]`

Schedules an appointment: `newappt`

Schedules a new appointments for a patient.

NOTE

The appointment(s) will be rejected by the system, if there are insufficient staff doctors on duty at the time of the appointment.
e.g. Cannot schedule more appointments than available doctors on duty.

If both `-reoccur <INTERVALS>` and `-num <REOCCURRING_TIMES>` fields are present, appointments of these `<REOCCURRING_TIMES>` will be added to the patient appointment schedule in `<INTERVALS>`. Otherwise, only one event will be added.

If the optional field `[-end <END_TIMING>]` is absent, default end time is 30 mins after start time of the appointment. Otherwise, end time will be `[-end <END_TIMING>]`.

The optional field `[-reoccur <INTERVALS>]` can be `-reoccur w`, `-reoccur m`, or `-reoccur y`.They represent to add weekly, monthly, yearly repeat appointment respectively.

Format: `newappt -id <PATIENT_REFERENCE_ID> -start <START_TIMING> [-end <END_TIMING>] [-reoccur <INTERVALS> -num <REOCCURRING_TIMES>]`

Acknowledges a appointment: `ackappt`

Acknowledges the most upcoming appointment only if patient arrives on the same day and the arriving time is before the appointment's end time.
Format: `ackappt <REFERENCE_ID>`

Cancels an appointment: `cancelappt`

Cancels the specified appointment.

NOTE	To avoid accidental cancellation of another patient's appointments, the user must first narrow down the search to a single patient by using the `appointments [<REFERENCE_ID>]` command.

Format: `cancelappt <ENTRY_ID>`

Changes the appointment date: `editappt`

Changes the timing for an existing appointment.
If optional field `[-end <END_TIMING>]` is not present, default endTiming is 30 mins after startTiming. Otherwise new endTiming will be `[-end <END_TIMING>]`.

NOTE	To avoid accidental rescheduling of another patient's appointments, the user must first narrow down the search to a single patient using the `appointments [<REFERENCE_ID>]` command.

NOTE	The operation is rejected if there are insufficient staff doctors on duty at the time of the new appointment.

Format: `editappt -entry <ENTRY_ID> -start <START_TIMING> [-end <END_TIMING>]`

Lists patients who have missed their appointments: `missappt`

Lists all appointments that are missed.

NOTE	An appointment is considered missed if the appointment was not acknowledged and the current time has passed the appointment's end time.

Format: `missappt`

Sets missed appointments as settled/notified: `settleappt`

Settles and removes the missed appointment based on the given index.
Settling refers to the user following up on contacting the patient who has missed his/her appointment.

NOTE	Only missed appointments can be settled. The user must first display the missed appointment listing by using `missappt`, before using this command.

Format: `settleappt <ENTRY_ID>`

Shows the empty slots: `slot` (v2.0)

List all the available empty slots for patients to make appointments

Format: `slot -start <START_DATE>`

Duty-shift Management

NOTE	Before you process any commands in Duty-shift Management, please ensure you have a doctor with referenceId of W0000001A by using `doctor W0000001A`. You can refer to `newdoctor` command to register this doctor if the doctor is not registered.

Displays duty shifts: `shifts`

Displays a sorted list of upcoming duty shifts for the staff doctors that is associated to the given `REFERENCE_ID`. If no keyword is given, all upcoming shifts will be displayed.

Format: `shifts [<REFERENCE_ID>]`

Adds a duty shift for a doctor: `newshift`

If both `-reoccur <INTERVALS>` and `-num <REOCCURRING_TIMES>` fields are present, duty shifts of these `<REOCCURRING_TIMES>` will be added to the doctor duty roster in `<INTERVALS>`. Otherwise, only one shift will be added.

Format: `newshift -id <STAFF_REFERENCE_ID> -start <START_TIMING> -end <END_TIMING> [-reoccur INTERVALS -num REOCCURRING_TIMES]`

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.

Appointment feature

The Appointment feature enables users to manage appointments for patients by providing basic Create, Read, Update, Delete (CRUD) of appointments. User is also able to find missed appointments and settle each missed appointment efficiently.

- e.g. `newappt -id E0000001A -start 01/12/19 0900 -end 01/12/19 0940 -reoccur m -num 2` - creates two monthly reoccurring appointments to patient whose `referenceId` is E0000001A.
- e.g. `editappt -entry 1 -start 01/12/19 1000 -end 01/12/19 1040` - edits a patient's first appointment timing to be the input timing if there is no conflict with doctor's duty shift.

The number of scheduled appointments cannot be more than the number of on-duty staff doctors at any given time.

Current Implementation

The Appointment feature contains multiple operations to indirectly manipulate the `UniqueEventList`. The implemented operations include:

`newappt` Command - Creates an new appointment or reoccurring appointments for a patient.

`ackappt` Command - Acknowledges an appointment whenever the patient checks in with the clerk.

`appointments` Command - Lists all upcoming appointments or appointments which involves the patient whose referenceId contains a certain keyword.

`editappt` Command - Edits an appointment timing.

`cancelappt` Command - Cancels an appointment.

`missappt` Command - Lists all missed appointments.

`settleappt` Command - Removes a missed appointment.

The appointment class diagram below illustrates the interactions between the Appointment class and associated classes.

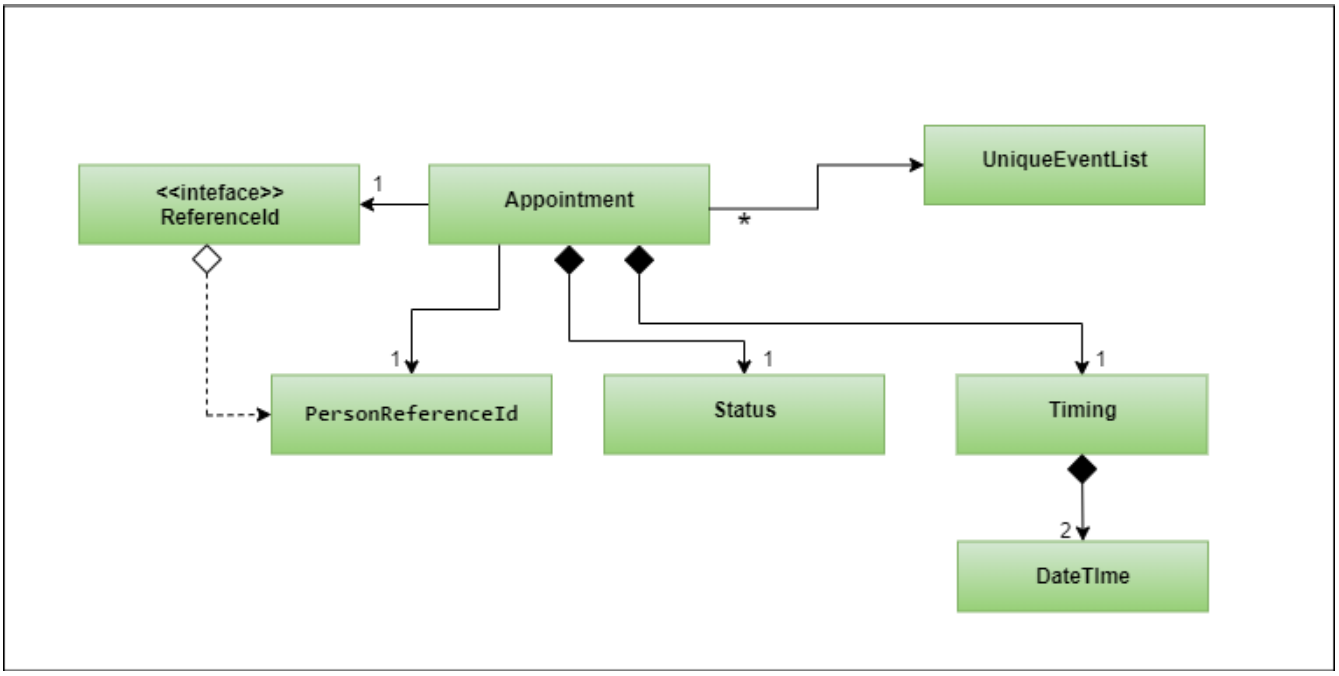

Figure 1. Appointment class diagram

he appointment will be rejected by the system, if there are insufficient staff doctors on duty at the time of the appointment.
Each `Appointment` object consists of a `PersonReferenceId`, `Timing`, `Status`. `Timing` class has 2 `DateTime` objects which indicates the start and end time of the `appointment`. The `UniqueEventList` contains 0 or more appointments.

The current implementation of `Appointment` checks with `patient` object by the unique `referenceId` and also checks the timing with doctors' dutyRoster. If the `referenceId` exists within the Model#UniquePersonList and the timing is valid, then the `Appointment` object is constructed. This ensures that the patient is registered before making any appointments and the appointment's timing is valid.

The appointment will be rejected by the system, if there are insufficient staff doctors on duty at the time of the appointment.

Before an appointment can be scheduled, the system first checks the total number of staff doctors on duty in that timeslot. Next, the system checks the existing appointments in that timeslot. If the number of appointments in that timeslot is less than the number of doctors, the appointment will be scheduled. Otherwise, the appointment will not be scheduled.

`newappt` Command

The `newappt` command is similar to the `new` command of patient and doctor. The command takes in the parameters required to construct `ReferenceId`, `DateTime` and `Status`. The image below shows how the `Appointment` object is constructed.

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

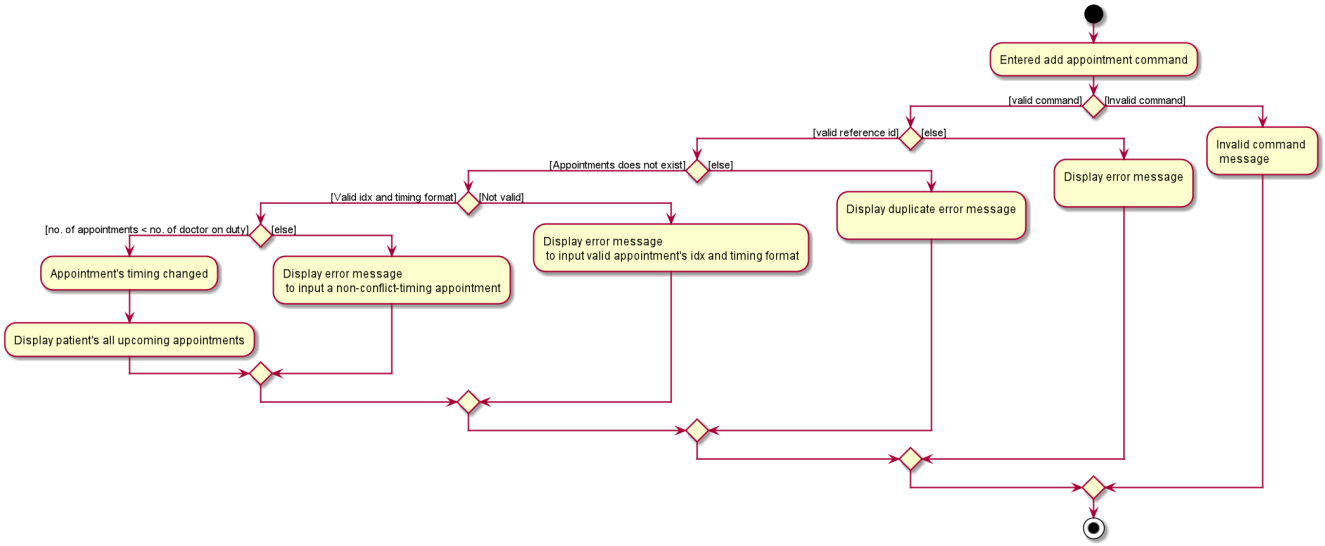

Figure 2. Interactions Inside the Logic Component when a user executes a `newappt` command

`ackappt` command

The `ackappt` command marks the patient's the most upcoming appointment as acknowledged only if it is on the same day and it is before the appointment's end time and also updates `UniqueEventList` to display the rest appointments belonging to the patient.

`appointments` Command

The `appointments` command works in two different ways.

Case 1: `appointments referenceId`

The `appointments` command searches for appointments that belong to the patient based on the given `referenceId`. The filtered appointments are found in `ModelManager`. The list is instantiated by filtering the `UniqueEventList` using `EventContainsKeywordPredicate` which is created from the `ReferenceId` argument supplied by the user.

Case 2: `appointments`

If the `appointments` command is executed without any other arguments, it executes with the predicate `EventContainsApprovedStatusPredicate`. updateFilteredAppointmentList() is called and the entire list of upcoming appointments is shown to the user.

`editappt` Command

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

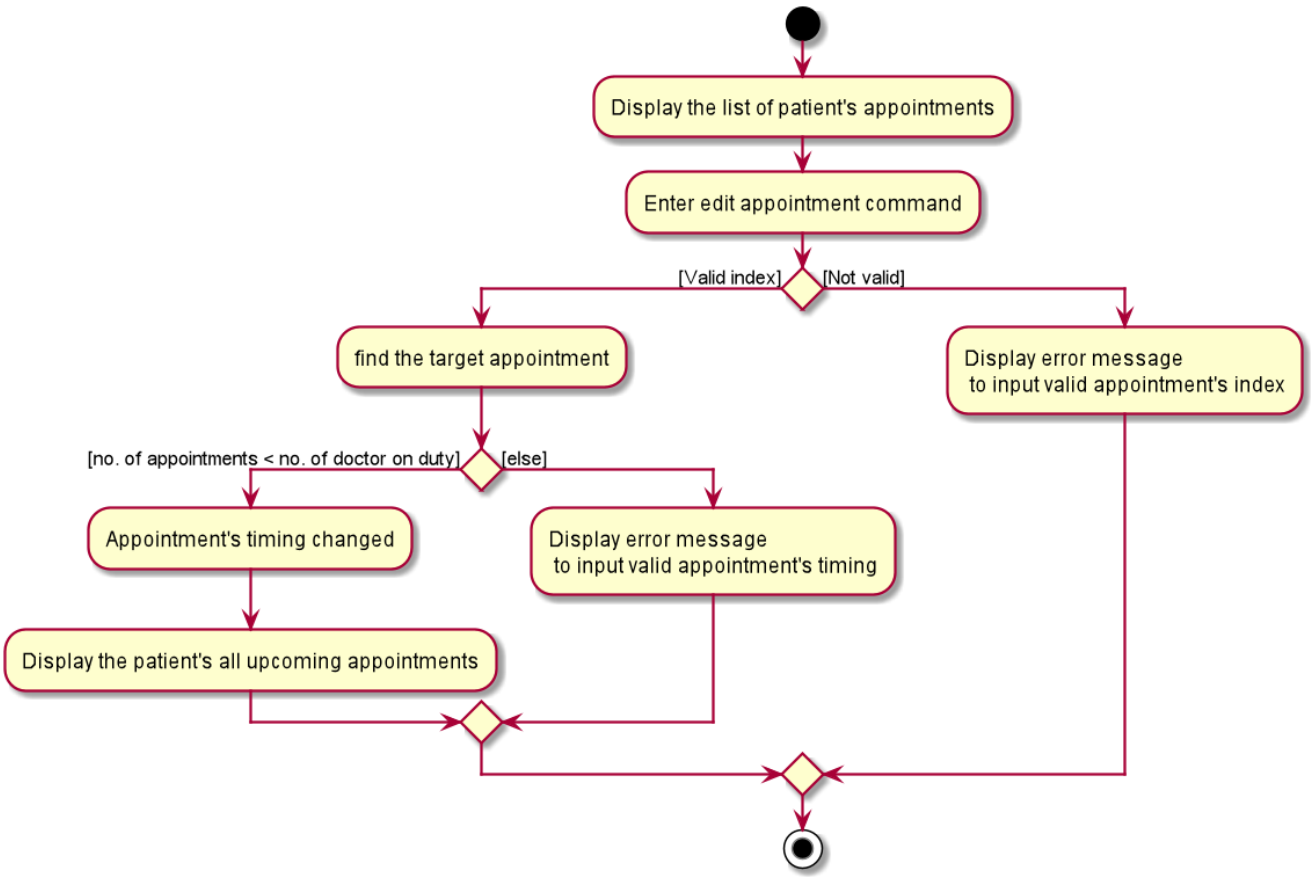

Figure 3. Interactions Inside the Logic Component when a user executes a `editappt` command

`cancelappt` Command

`cancelappt` simply takes in the index of the target appointment to cancel according to the displayed appointment list.

Given below is the sequence diagram for interactions within the `Logic` component for the `execute("cancelappt 1")` API call.

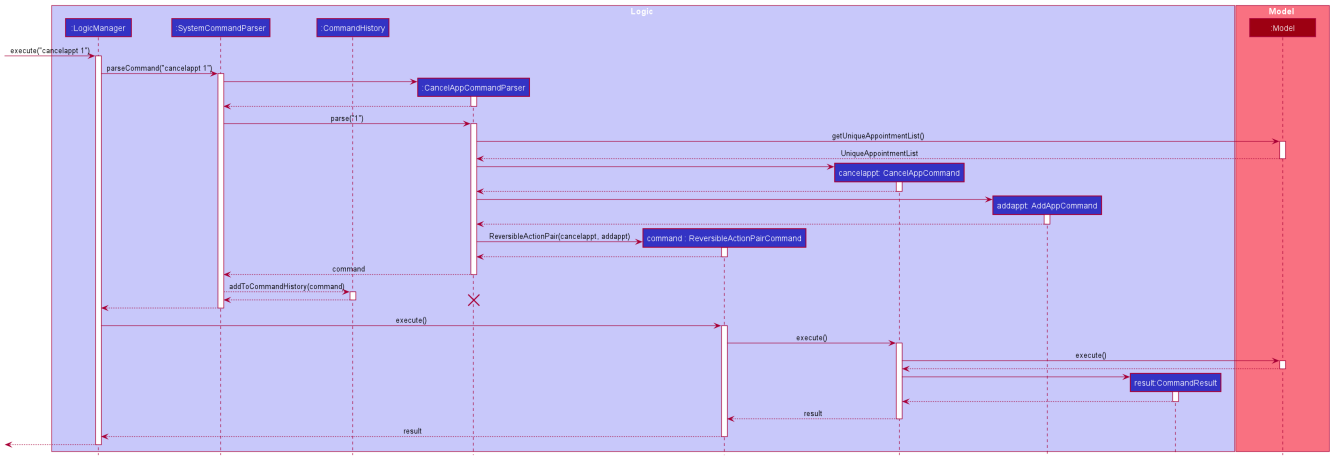

Figure 4. Interactions Inside the Logic Component for the `cancelappt 1` Command

Duty Shift Scheduling

The duty shift scheduling provides users the ability to schedule duty shifts for doctors. It can help

doctors to check, add, edit duty shifts efficiently.

- e.g. `newshift -id W0000001A -start 01/11/19 0900 -end 01/12/19 2100 -reoccur m -num 2` - allows the user to create two monthly reoccurring duty shifts to doctor whose `referenceId` is W0000001A.
- e.g. `editshift -entry 1 -start 02/12/19 0900 -end 02/12/19 2100` - allows the user to change a doctor's first duty shift to be the input timing if there is no conflict with appointments.

Current Implementation

The duty shift scheduling contains multiple operations to indirectly manipulate the `UniqueEventList`. The implemented operations include:

`newshift` Command - Adds a duty shift or reoccurring duty shifts to a doctor.

`shifts` Command - Lists all duty shifts involving the doctor's referenceId which contains the keyword.

`editshift` Command - Change a current duty shift's timing.

`cancelshift` Command - Cancels duty shift.

Each Duty Shift is an `Event` object consists of a `PersonReferenceId`, `Timing`, `Status`. `Timing` class has 2 `DateTime` object as they indicate the start and end times of the duty shift.

The current implementation of duty shift checks with doctor object by the unique `referenceId` and also checks the timing with appointments. If the `referenceId` exists within the Model#UniquePersonList and the timing is valid, then the duty shift is constructed. This ensures that the doctor is registered and the duty shift's timing is valid before making any duty shifts.

The duty shift will be rejected by the system, if there are insufficient staff doctors on duty at the given time.

Before a duty shift's time can be edited, the system first checks the total number of staff doctors on duty in that timeslot. Next, the system checks the existing appointments in that timeslot. If the number of appointments in that timeslot is less than the number of doctors, the duty shift's time will be changed. Otherwise, the duty shift will not be allowed to edit.

`newshift` Command

The `newshift` command behaves similarly to the `new` command used for patient and doctor. The command takes in the parameters required to construct `ReferenceId`, `DateTime` and `Status`.

`shifts` Command

The `shifts` command works in two different ways.

Case 1: `shifts ReferenceId`

The shifts command searches for duty shifts that belong to the doctor based on the given

`ReferenceId`. The filtered shifts are found in `ModelManager`. The list is instantiated by filtering the `UniqueEventList` using `EventContainsKeywordPredicate` which is created from the `referenceId` argument supplied by the user.

Case 2: `shifts`

The shifts behaves similarly to `shifts ReferenceId` when it does not take in any other arguments. Instead, it automatically executes with the predicate `EventContainsApprovedStatusPredicate`. updateFilteredEventList() is called and the entire list of the upcoming duty shifts is shown to the user.

`editshift` Command

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

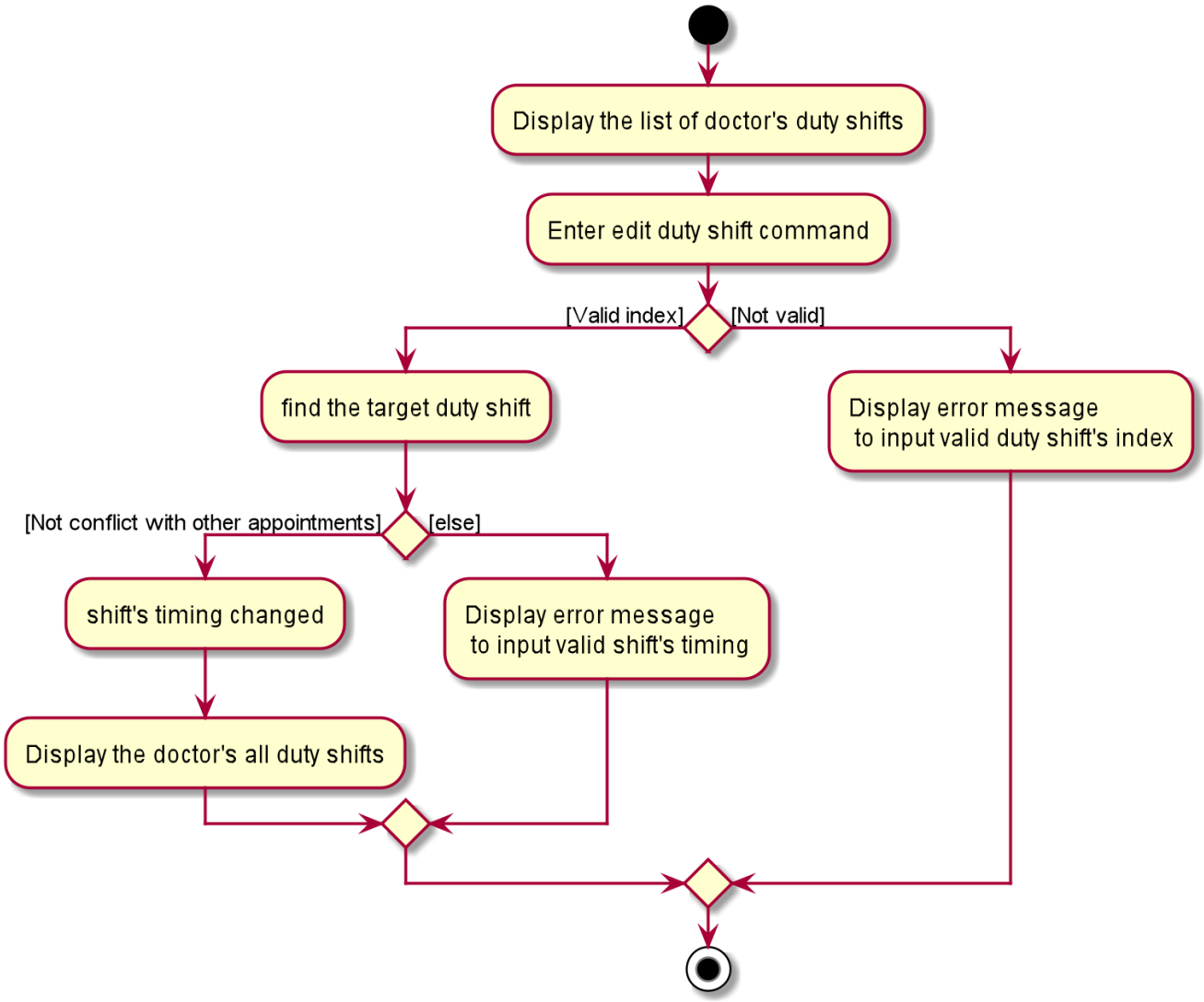

Figure 5. Interactions Inside the Logic Component when a user executes a `editshift` command

`cancelshift` Command

`cancelshift` simply takes in the index of the target duty shift to cancel according to the displayed shift list.