Yap Dian Hao - Project Portfolio

About the project

JelphaBot is a desktop application to help NUS students manage tasks. JelphaBot allows students to track and manage tasks conveniently. Users enter commands in JelphaBot through a CLI. However, a GUI is implemented with JavaFX for a smoother user experience. It is written in Java, and has about 13 kLoC.

Summary of contributions

- **Major enhancement**: added **the ability for users to add reminders to certain tasks**.
  - What it does: allows users to add or remove reminders to remind tasks that will due soon to increase users' efficiency, including:
    - Tasks that are uncompleted
    - Tasks that will due within a week
  - Justification: As our target users are students with a lot of commitments, students may miss out several commitments over time. Therefore, they will need to manage their tasks and schedule upcoming tasks accordingly. Therefore, we have decided to include a reminding feature in JelphaBot to address this need. I implemented jelphaBot's ability to link tasks and reminders, and combined them to maximize users' efficiency in fulfilling their commitments. This adds significant feature to JelphaBot, as a task manager and a task reminder.
  - Highlights:
    - This feature introduces a new class, `Reminder`, which plays the similar significance to `Task`. The `Reminder` class is leveraged on the Object-Oriented paradigm that a `Task` needs to be reminded by a `Reminder`, therefore they are closely relatable to each other. To successfully design such a class that transcends a wide range of components, a complete understanding and thorough analysis of the whole application is required.
    - Challenging as it requires real time synchronization and integration between GUI and internal logic components. For instance, a user, when deletes a `Task`, should observe the deletion of its corresponding `Reminder` from the UI as well.
    - This feature is implemented in a way that it undergoes the whole application's main components' development, in front-end and back-end. It involves the `Ui`, `Logic`, `Model`, and `Storage`, with over 3k lines of functional and test code.
  - Relevant pull requests: #88, #149, #167, #179, #207, #309, #346
- **Code contributed**: Code contributed
- **Minor enhancement**: Increased code coverage from 58% to 62% through writing new tests and improved existing tests.
  - Relevant pull requests: #33, #76

- **Other contributions**:
  - Enhancement to existing features:
    - Updated the GUI color scheme: #222
  - Documentation:
    - Updated JelphaBot's architecture diagram for Developer Guide.
    - Updated the `Storage` component's sequence diagram for Developer Guide.
    - Added screenshots and descriptions as visual aids for User Guide.
  - Community:
    - Contributed to forum discussions: requesting usage for third-party libraries (controlsfx) #81
    - Reviewed pull requests: #13, #17
    - Reported bugs and suggestions for another team in PE dry run: Reported 15 bugs

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.*

Set Reminders (Dian Hao)

JelphaBot allows you to set reminders for tasks and manage your tasks comprehensively! You can view all your existing reminders from the reminders tab.

View reminders : `remindertab`

Apart from the function to switch tabs by pressing kbd:[Ctrl] + kbd:[tab] on your keyboard, you can enter the `remindertab` command or its shortcuts `:R` or `:r` to manually switch to the reminder tab. Every `Reminder` will show the `Task` `s module code, description, due date, the days that will be reminded before the deadline, and the hours that will be reminded before the deadline.

Format: `remindertab`
Shortcut: `:R` or `:r`

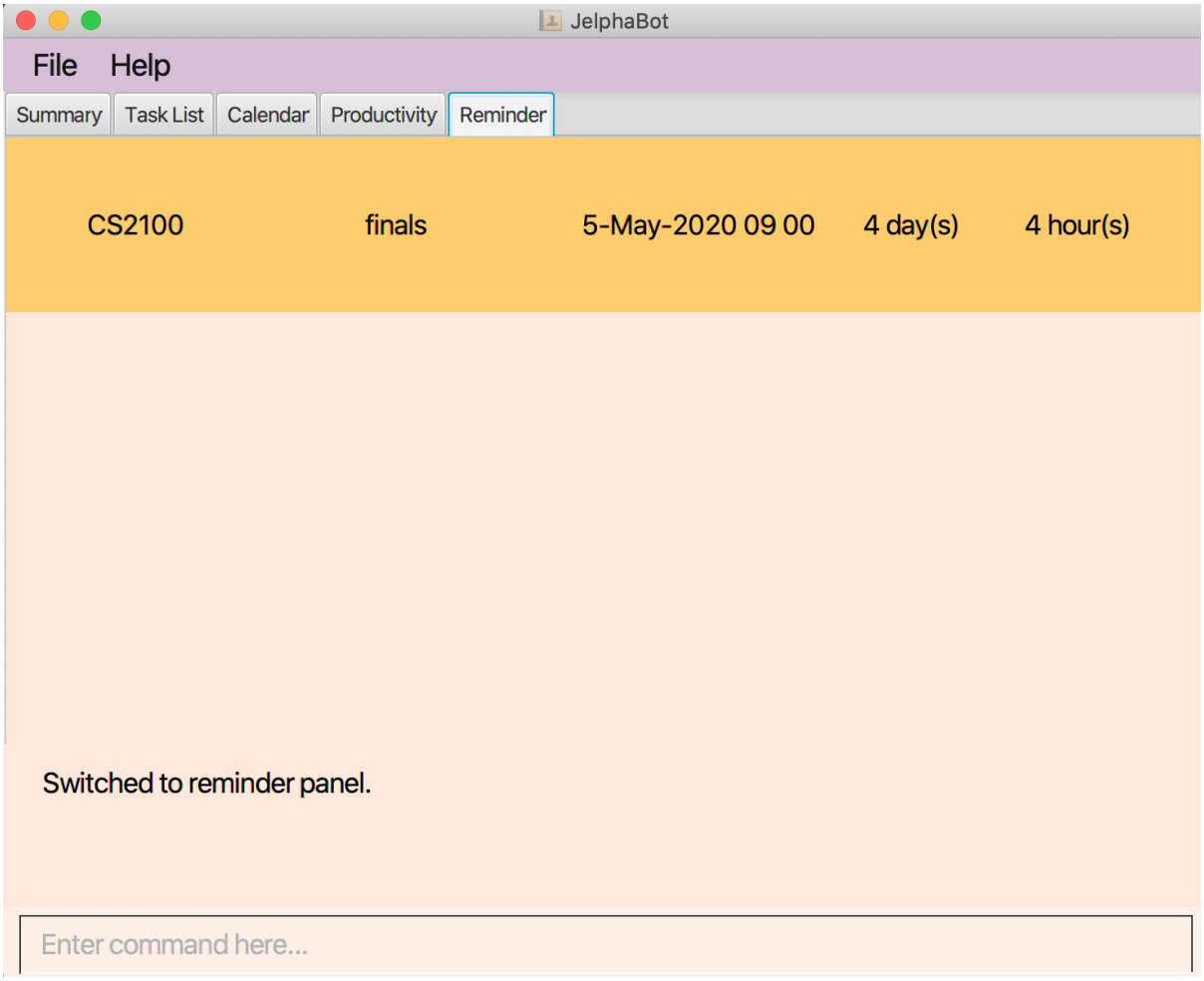

Figure 1. Example of expected result after running `remindertab`

Adding reminder : `reminder`

You can add a reminder to your specified task to remind yourself of the task if the current time is within the time-frame specified by you.

Format: `reminder INDEX days/DAYS hours/HOURS`

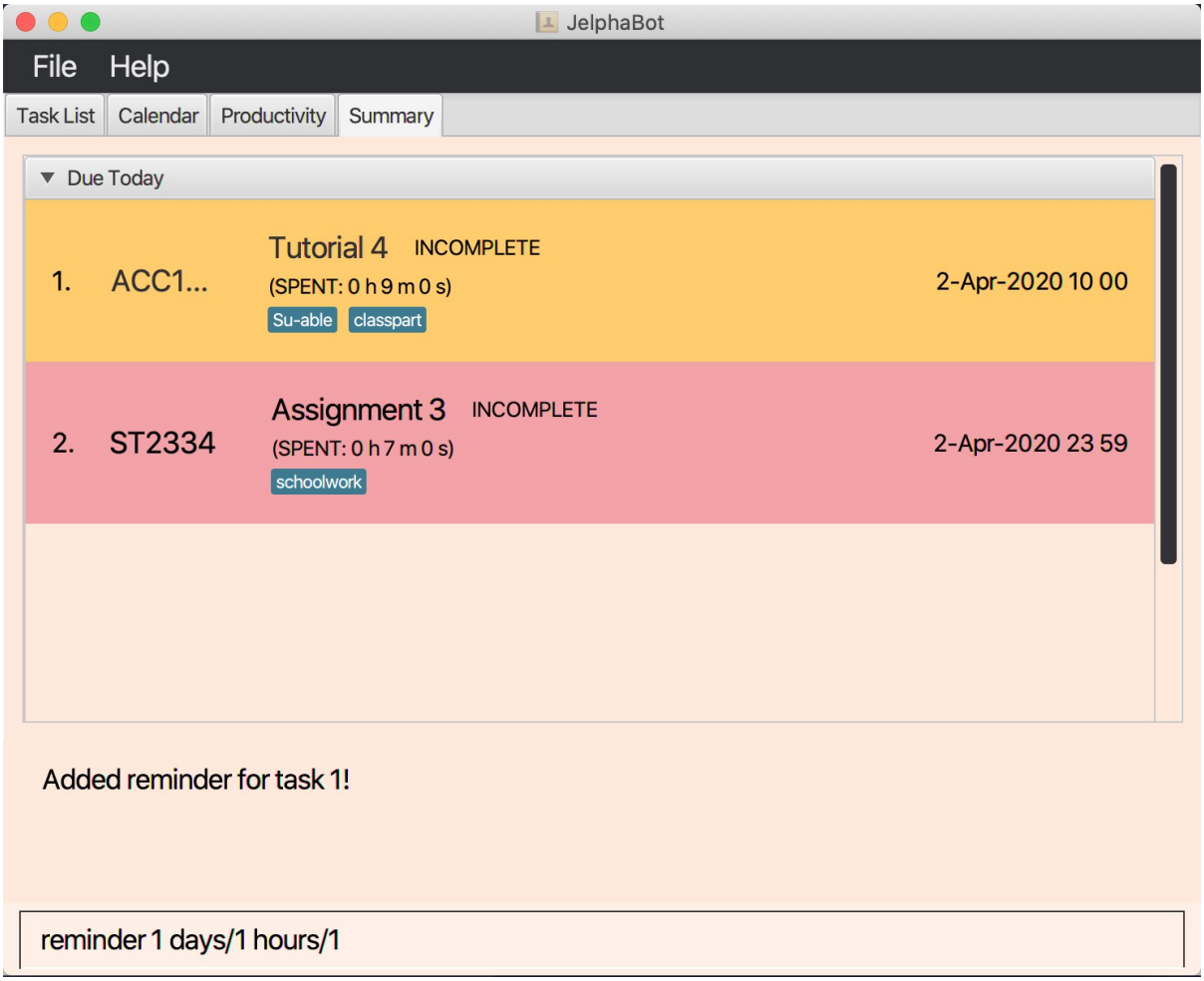

Figure 2. Example of expected result after running `reminder 1 days/1 hours/1`

- Adds a reminder to the task which is at the specified `INDEX`.
- The index refers to the index number shown in the displayed task list.
- The index **must be a positive integer** 1, 2, 3, ...
- `DAYS` refers to the number of days before the due date of the task when you want to be reminded of it.
- `HOURS` refers to the number of hours before the due date of the task when you want to be reminded of it.
- You can only specify `DAYS` to be in the range 0 - 7 inclusive.
- You can only specify `HOURS` to be in the range 0 - 24 inclusive.
- Every `Task` can only have one `Reminder` .
- A `Task` that is completed cannot have a `Reminder` .
- A `Task` that is reminded but are not completed after the deadline will still be reminded for.
- A `Task` will not be reminded if the task is completed but it has a `Reminder` .

Removing reminder : `delrem`

If you would like for a task's reminder to be deleted, you can enter the `delrem` command to remove

the reminder of that task.

Format: `delrem INDEX`

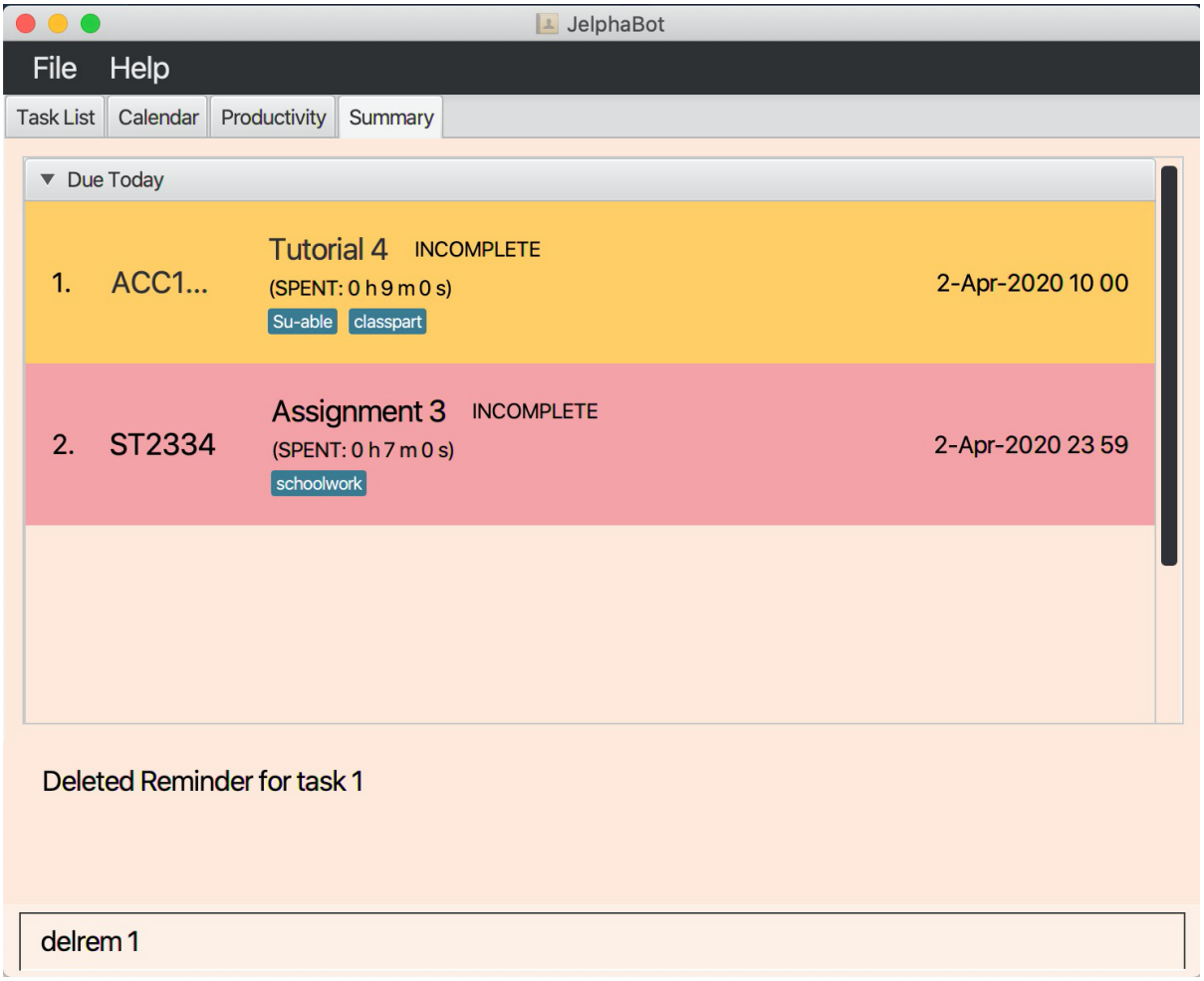

Figure 3. Example of expected result after running `delrem 1`

- Removes a reminder associated to the task at the specified `INDEX`.
- The index refers to the index number shown in the displayed task list.
- The index **must be a positive integer** 1, 2, 3, ...
- Whenever a task is deleted, the corresponding reminder will also be removed.

Reminder popup notification

Every time you run JelphaBot after adding your reminders, JelphaBot will show a list of tasks that will be overdue soon, and tasks that are past their deadline but have not been completed.

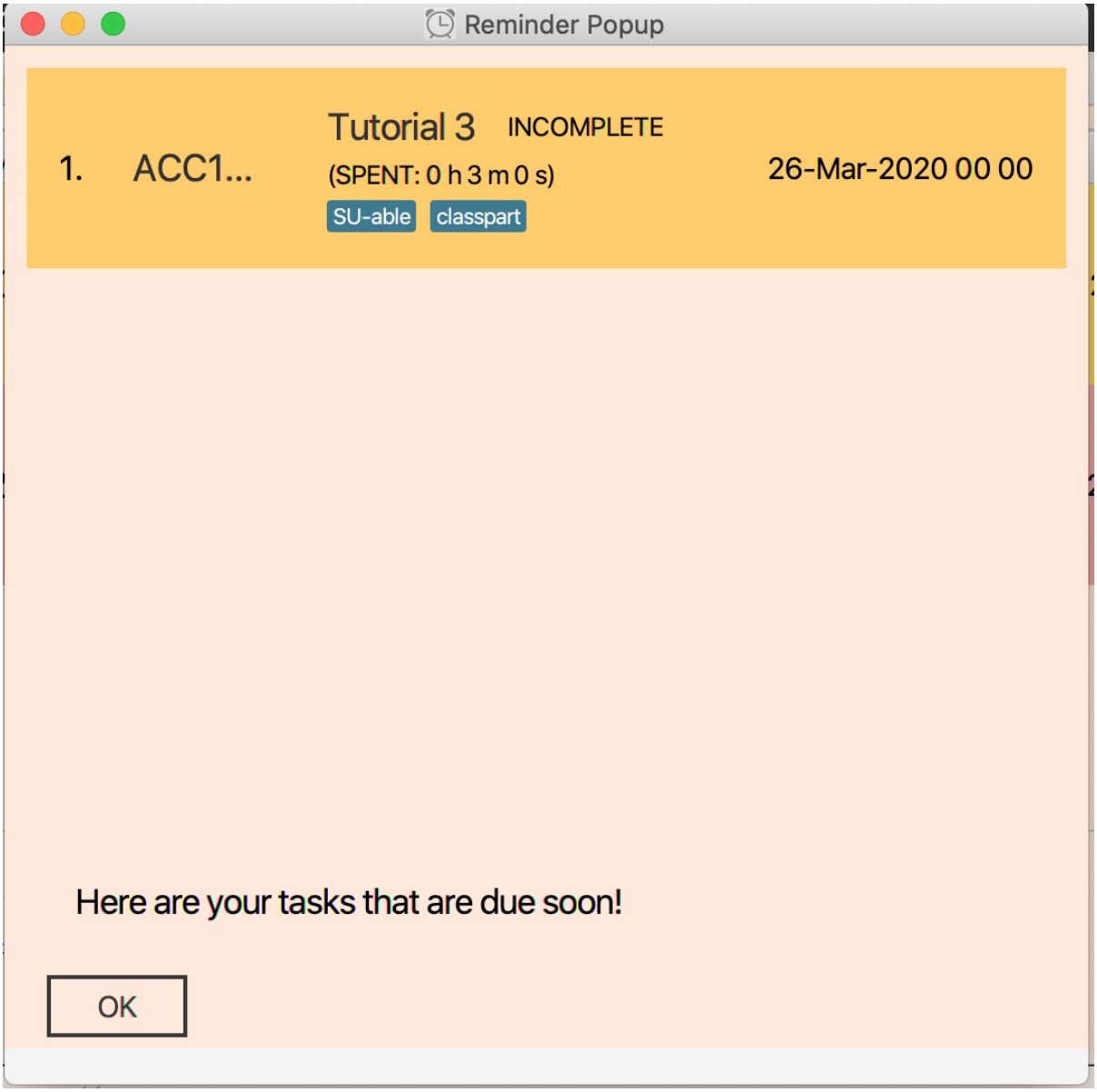

Figure 4. Example of expected result after tasks are being reminded.

- Whenever JelphaBot is booted, if you have any tasks that is not completed but has a reminder, JelphaBot's reminder window will popup.
- This will persist until you either complete the `Task`, or delete the `Reminder`.

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.

Reminder feature (Dian Hao)

JelphaBot has a reminder feature that reminds users whenever they have tasks that will be overdue soon. This feature offers two main functions:

- Adds a reminder to a task.
- Delete a reminder that is associated to a task.

Classes for Reminder feature in Model

The `Reminder` feature was implemented by a new set of classes to `Model`. A new `Reminder` class is stored in Jelphabot's `UniqueReminderList`, which consists of a list of `Reminder` s. Each `Reminder` consists of 3 objects:
`Index`: the `Task` 's index of which the user wants to be reminded for.
`ReminderDay`: the number of days before the `Task` 's deadline that the user wants to be reminded for.
`ReminderHour`: the number of hours before the `Tasks` 's deadline that the user wants to be reminded for.

The following class diagram summarizes shows the relationship between the classes.

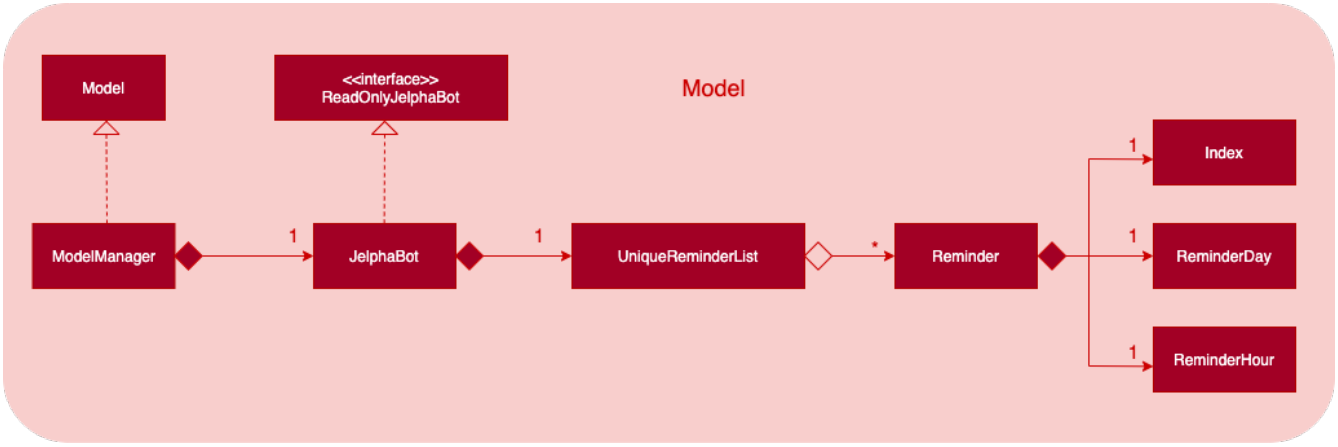

Figure 5. Reminder Class Diagram in the Model component

Implementation

Function 1: Creates a reminder for a specified task
To add a reminder to a certain task, the user enters the *reminder INDEX days/DAYS hours/HOURS* command. (e.g, reminder 2 days/2 hours/1)

The `Logic execute()` method creates a `ReminderCommand` from the input string by parsing the input according to the command word and several other attributes. Next, the input string is converted into `Index`, `ReminderDay`, `ReminderHour`, and a `Reminder` object with these properties are forwarded to `Model`.

The `Model` first checks the validity of the attributes respectively. The valid `Reminder` is then added to the `UniqueReminderList` after checking that there are no other `Reminder` with the same `Index`.

After the above actions are correctly performed, the `Logic` fires the `Storage` to save the `Reminder`.

Upon successful execution of the command, the user adds a reminder associated to the task at `INDEX`. Upon exiting JelphaBot, the reminder will be saved. By the next time the users starts JelphaBot, it will remind the user should the task's due date fall within the period set by the user from the current date.

The sequence diagram for interactions between the `Logic`, `Model`, and `Storage` is shown below.

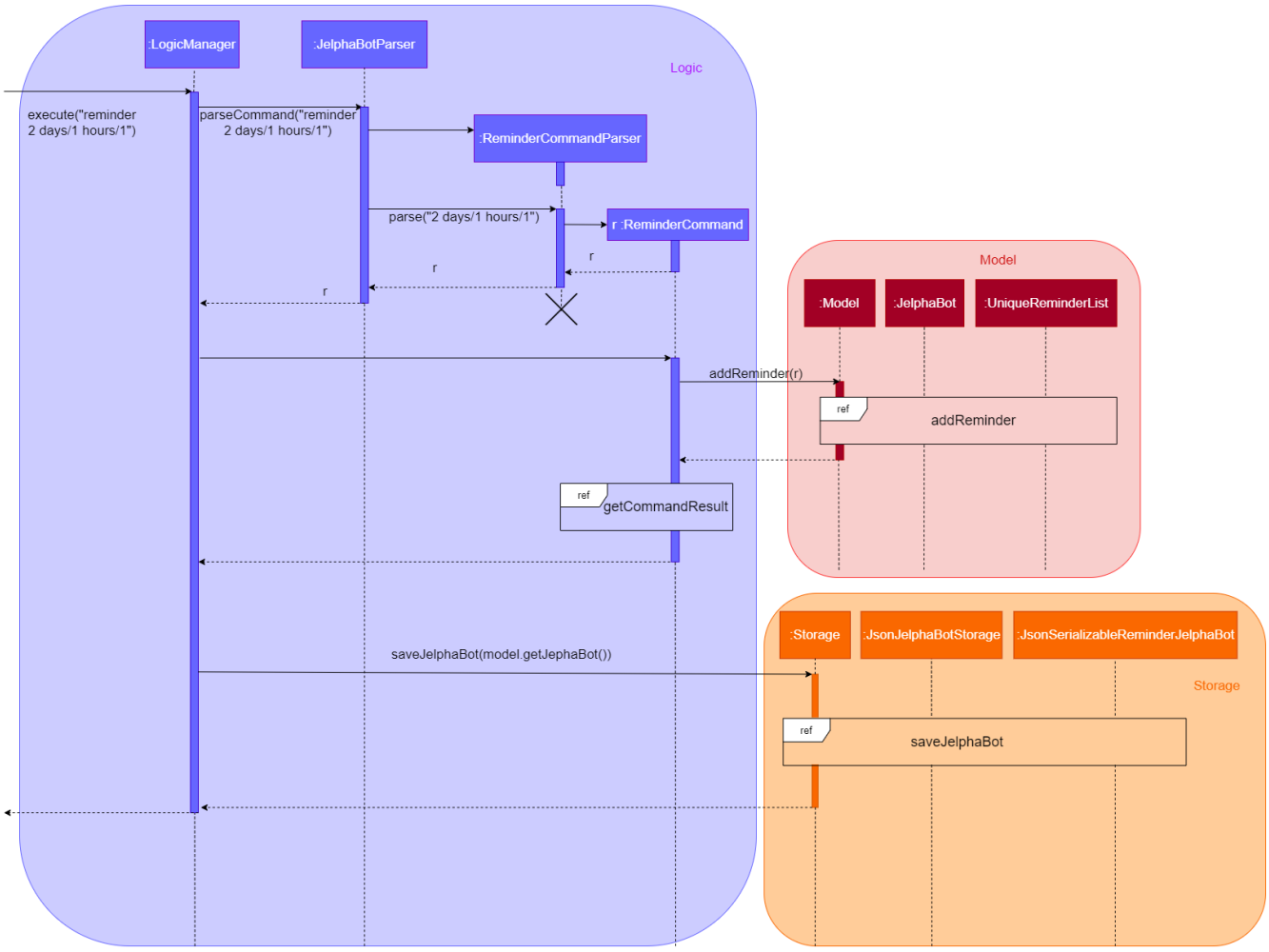

Figure 6. Sequence Diagram after running reminder 2 days/2 hours/1

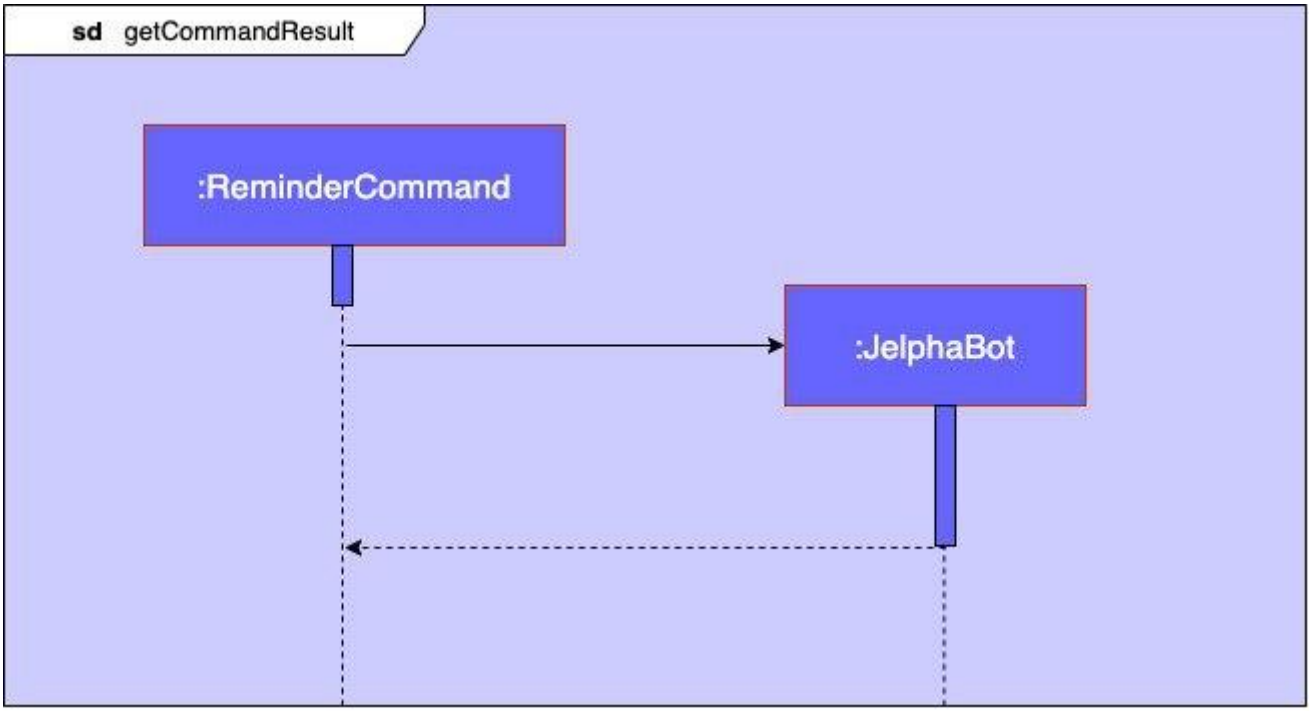

Figure 7. The reference frame of getting the `CommandResult` in the `Logic` component.

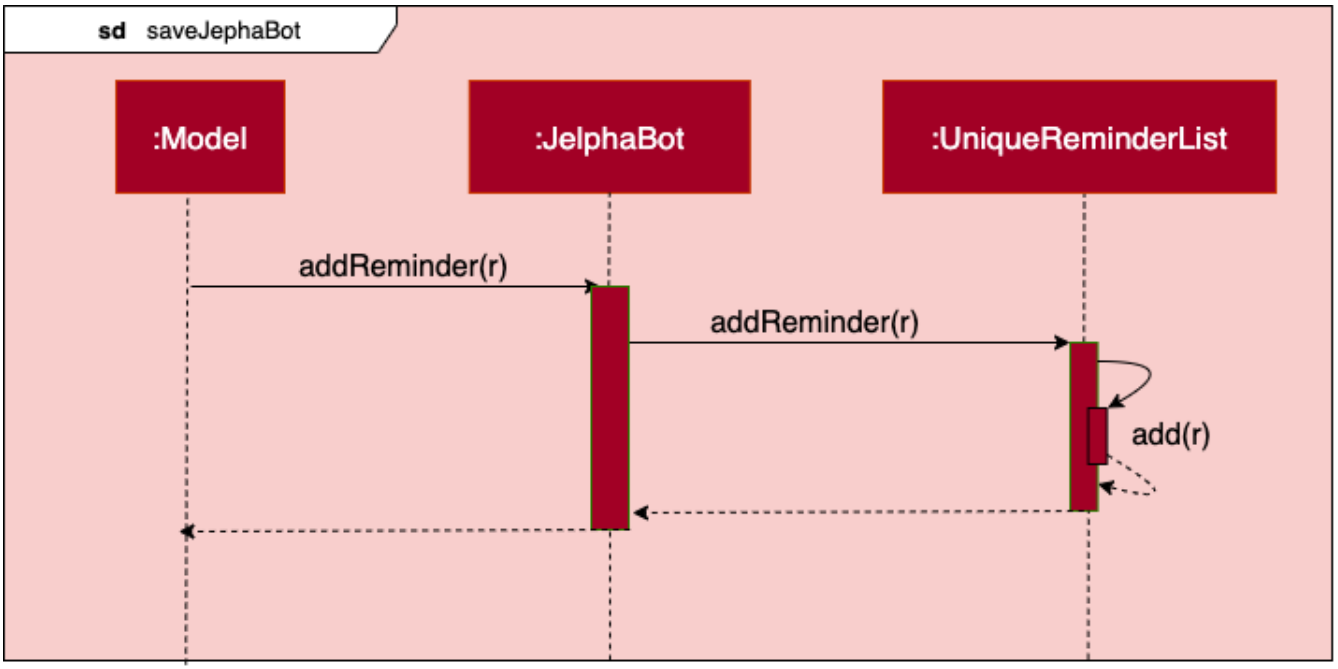

Figure 8. The reference frame of adding the `Reminder` in the `Model` component.

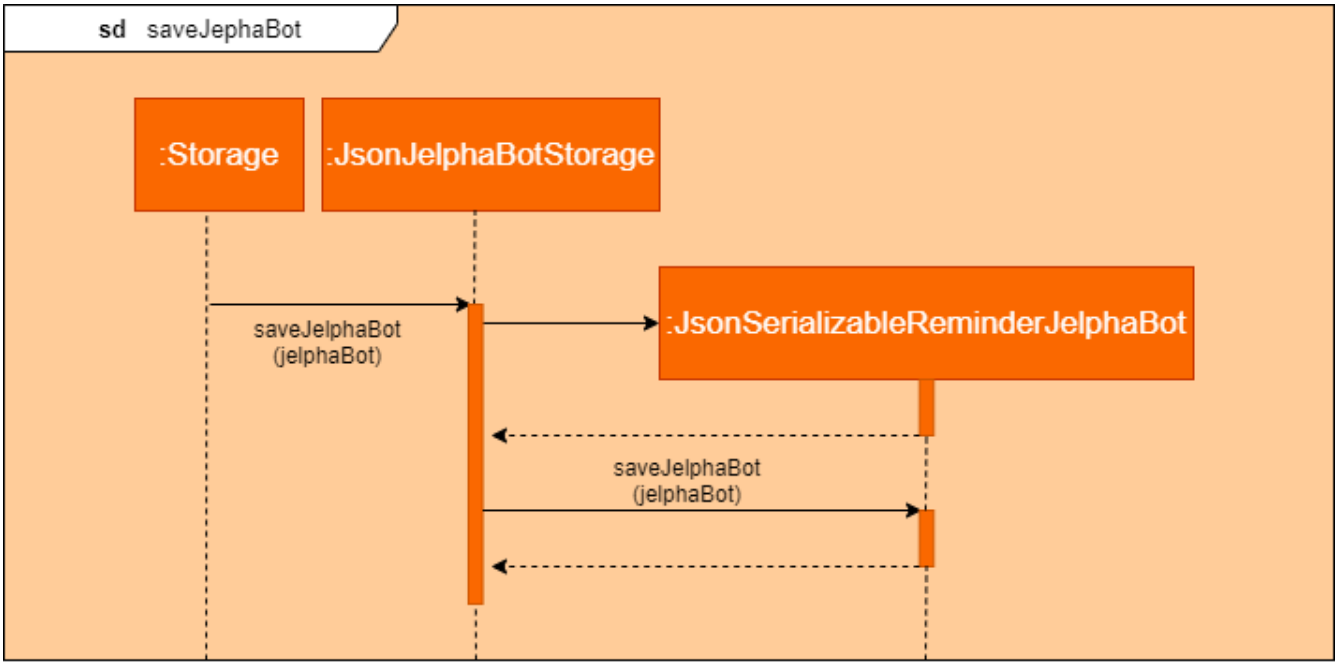

Figure 9. The reference frame of saving a `Reminder` by the `Storage` component.

NOTE	If the user attempts to add a reminder to tasks that have reminders, the command will fail to execute. The user also will not need to set reminders to tasks that are complete. However, if tasks that has reminders are not completed, JelphaBot will still warn the user.

Function 2: Deletes a reminder for a specified task
To delete a reminder associated to a certain task, the user enters the `delrem INDEX` command. (e.g. delrem 2)

The interaction between components is similar to adding a `Reminder`. A key difference that this command removes the `Reminder` that reminds the `Task` at `INDEX` from the `UniqueReminderList`. Moreover, `delrem` command requires that the `Reminder` with `INDEX` is in the list.

Upon successful execution of the command, the reminder of the task at `INDEX` is removed.

Design Considerations

Aspect 1: Implementing `Reminder` object

- **Current solution:** Implement `Reminder` as a standalone class
  - Rationale: A `Reminder` is an object, with the same hierarchy as the `Task` class, with similar attributes.
  - Pros: Fully capture the idea of an object-oriented design and robust in handling future changes.
  - Cons: An additional storage is required to store the `Reminder` objects, which causes overhead while reading from and writing to json files.
- **Alternative 1:** Design `Reminder` as one of the attributes of a `Task`
  - Rationale: A `Reminder` can also be seen as one of `Task` 's properties, analogous to `Description` and other properties.
  - Pros: Easy to implement. Concurrent fetching and storing from the json files while reading and writing `Task`.
  - Cons: A `Reminder` has to remind users the moment when Jelphabot is booted. At that instance, `Storage` has not started to read `Task` from the json files yet, therefore the `Reminder` could not be read beforehand.

Reason for chosen implementation:
We decided to choose the current solution due to the dynamic nature of tasks and users' needs. For upcoming changes in the future, it is easier to implement by adding similar classes or attributes to the existing design.

Aspect 2: Rendering `Reminder` on `ReminderListPanel`

- **Current solution:** Shows the `ModuleCode`, `Description`, and `DateTime` of the `Task` that is being reminded, the respective `ReminderDay` and `ReminderHour`.
  - Pros: convenient and simple to understand. Users only need to refer to the `TaskListPanel` to look at the details of the `Task`.
  - Cons: FXML styling will be squeezy.
- **Alternative 1:** Shows the `Reminder` similar to how the `Task` is displayed.
  - Pros: Simple, as it only shows the details of the `Reminder`.
  - Cons: Users need to constantly refer to the `TaskListPanel` for details. Both panels have an `Index` listed, which may cause confusion.

Reason for chosen implementation:
We decided to choose the current solution after considering users' needs and convenience of fully using the `Reminder` s. Users do not need to switch back and forth between tabs to refer between `Task` s and `Reminder` s, which saves time that can be better spent by completing the `Task` s.