Amanda Lam - Project Portfolio

PROJECT: JelphaBot

Overview

JelphaBot is a desktop application integrated with various features for managing tasks efficiently, mainly focusing on NUS students as our target group. It is specially designed by my talented crew of teammates, together with myself.

JelphaBot was developed for our software engineering module, CS2103T, where we were tasked to morph an address book application that manages contact details into something else. As part of the project constraints, user interactions are primarily Command-Line Interface (CLI), assisted with a GUI created using JavaFX. It is written in Java, and has about 13 kLoC.

Summary of contributions

This section contains a summary of my coding, documentation, and other helpful contributions to my team project.

- **Major enhancement**: Implemented the Calendar feature
  - What it does: A calendar integrated to display a monthly calendar of overarching tasks due, together with a task list panel on the left for detailed tasks to be shown.
    - Allow users to navigate to other months of their choice.
    - Allow users to view specific tasks due to be listed on selected dates.
    - Allow users to navigate directly to today's date of the calendar view, as well as listing out the tasks due today.
  - Justification:
    - The use of a calendar to display a visual view of these information is neat and easy to navigate, making the feature more user-friendly.
    - Integrating a calendar allows users to be able to plan their schedules easily by being able to immediately visualise which days are busier with more tasks on hand.
  - Highlights:
    - The overarching view of the tasks displayed on the calendar (shown by the dot indicators) is done by filtering the existing main task list.
    - The implementation was challenging as it required constant updates to be accounted for after other commands edits the main task list is input.
    - Moreover, to avoid user mistakes and increase the feature's usability, the included commands implementation in carrying out various functions as listed above uses the

same command keyword : `calendar`. This involved the heavy use and interaction between the `Ui`, `Logic`, and `Model` components.

    - Relevant pull requests: #314, #299, #297, #177, #174, #165, #138, #84

- **Minor enhancement**: Added in the panel of tabs in the main Ui layout during initial stages. This was done for the subsequence integration of our individual features.
    - Pull requests: #78
- **Code contributed**: [View on RepoSense]
- **Other contributions**:
    - Project management:
        - Managed releases `v1.2.1` - `v1.3` (2 releases) on GitHub
        - Updated `SampleDataUtil` to propagate an empty instance of the app with time-sensitive test data to streamline manual testing. (Pull requests: #357)
    - Documentation:
        - Added diagrams and documentation for the Calendar feature in the Developer Guide.
        - Updated diagram for the Ui section, the delete command diagram and the proposed undo feature from AB3 in the Developer Guide.
        - Added documentation for Calendar feature in the User Guide.
    - Community:
        - Reported bugs and suggestions for other teams: Reported 11 bugs during PE Dry Run

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. The full User Guide can be found here.*

Calendar (Amanda)

JelphaBot also comes with a built-in calendar view that allows you to view your overarching tasks due on a monthly basis. Dates that have tasks due would have a dot indicator shown on the calendar. You would also be able to navigate to specific dates to view your tasks due for that day of the month!

View calender : `calendar`

Apart from the function to switch tabs by pressing kbd:[Ctrl] + kbd:[tab] on your keyboard, you can enter the `calendar` command or its shortcuts `:C` or `:c` to manually switch to the calendar tab. The calendar panel will then show you your schedule for the current month with today's date highlighted.

Format: `calendar`
Shorcut: `:C` or `:c`

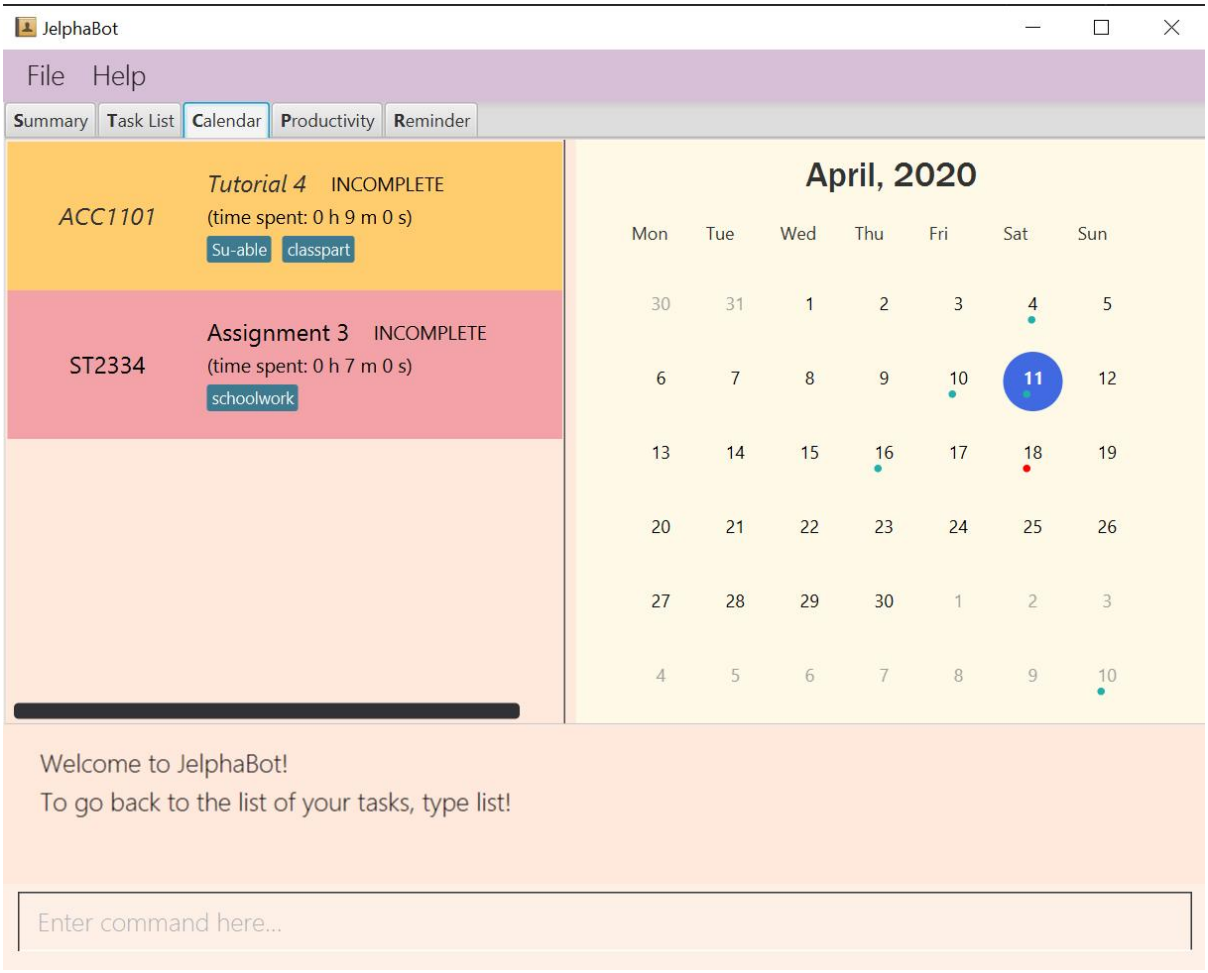

Figure 1. Example of expected result after running `calendar`

NOTE	Highlighting of Dates: Today's date would be highlighted in dark blue, while other dates would be in light blue.

Change month and year view of Calendar : `calendar`

You can navigate the calendar panel to another month and year by specifying it. The calendar panel would be updated accordingly while highlighting the first day of the month. The task list panel on the left will display the tasks due on the first day of the month.
Format: `calendar MONTHYEAR`

- For MONTHYEAR format, it should be MMM-YYYY, but it also allows some other formats shown when your format is invalid.

Examples:

- `calendar May-2020`

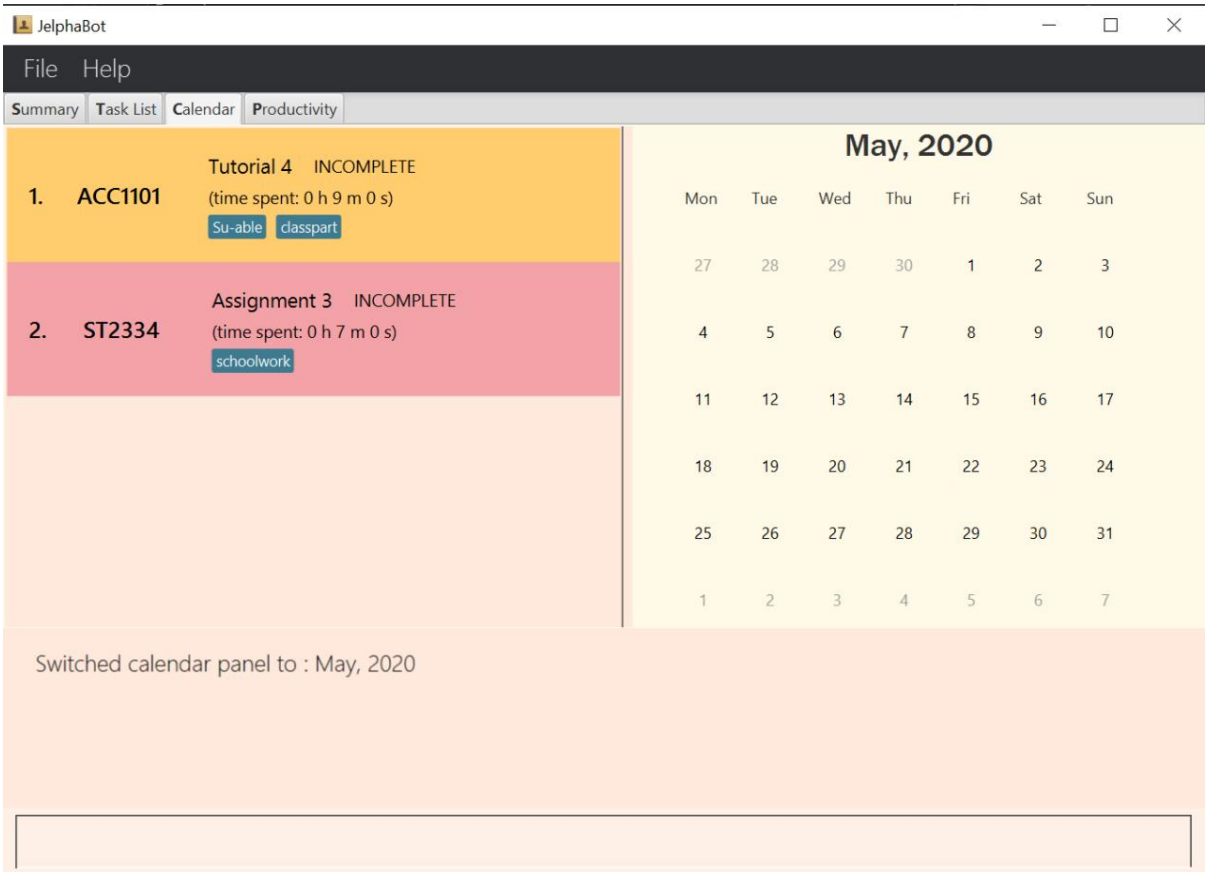

Figure 2. Example of expected result after running `calendar May-2020`

Displays month of May in the year 2020 in the calendar panel on the right.

IMPORTANT	Format of the month in MONTHYEAR (MMM) input has to have the first letter in upper-case. E.g `Mar` instead of `mar` when specifying the month of March.

Show tasks due on specific date : `calendar`

Displays the tasks due on specified date, while highlighting that day on the calendar
Format: `calendar DATE`

- The date specified **must be for the month and year of the shown Ui** for that corresponding date to be highlighted
- For DATE formats, we recommend the format to be MMM-dd-YYYY, but it also allows some other formats shown when you type in the command word.

Examples:

- `calendar Apr-1-2020`
- `calendar Apr/1/2020`

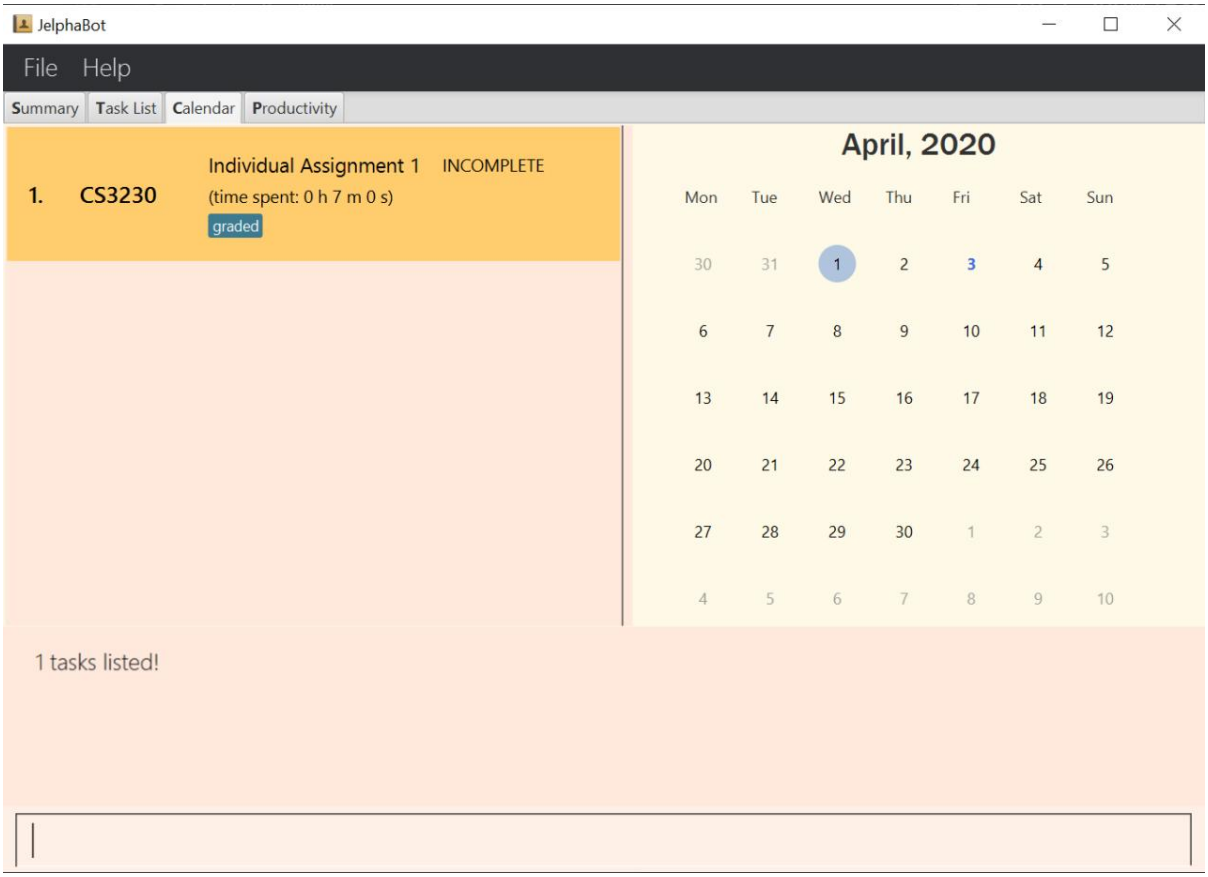

Figure 3. Example of expected result after running `calendar Apr-1-2020`

Highlights 1st of April in the calendar panel on the right and displays the corresponding tasks due on the left.

NOTE	Dot indicator showing tasks: Dates that have more than 3 tasks due would have a red dot indicator, while dates with at least 1 task but less than 4 tasks due would be represented with a green dot indicator.

Navigate directly to today's date on Calendar : `calendar`

Immediately displays the calendar view for this month and highlights today's date. The task list panel on the left will display the tasks due today as well.
Format: `calendar today`

Examples:

- `calendar today`

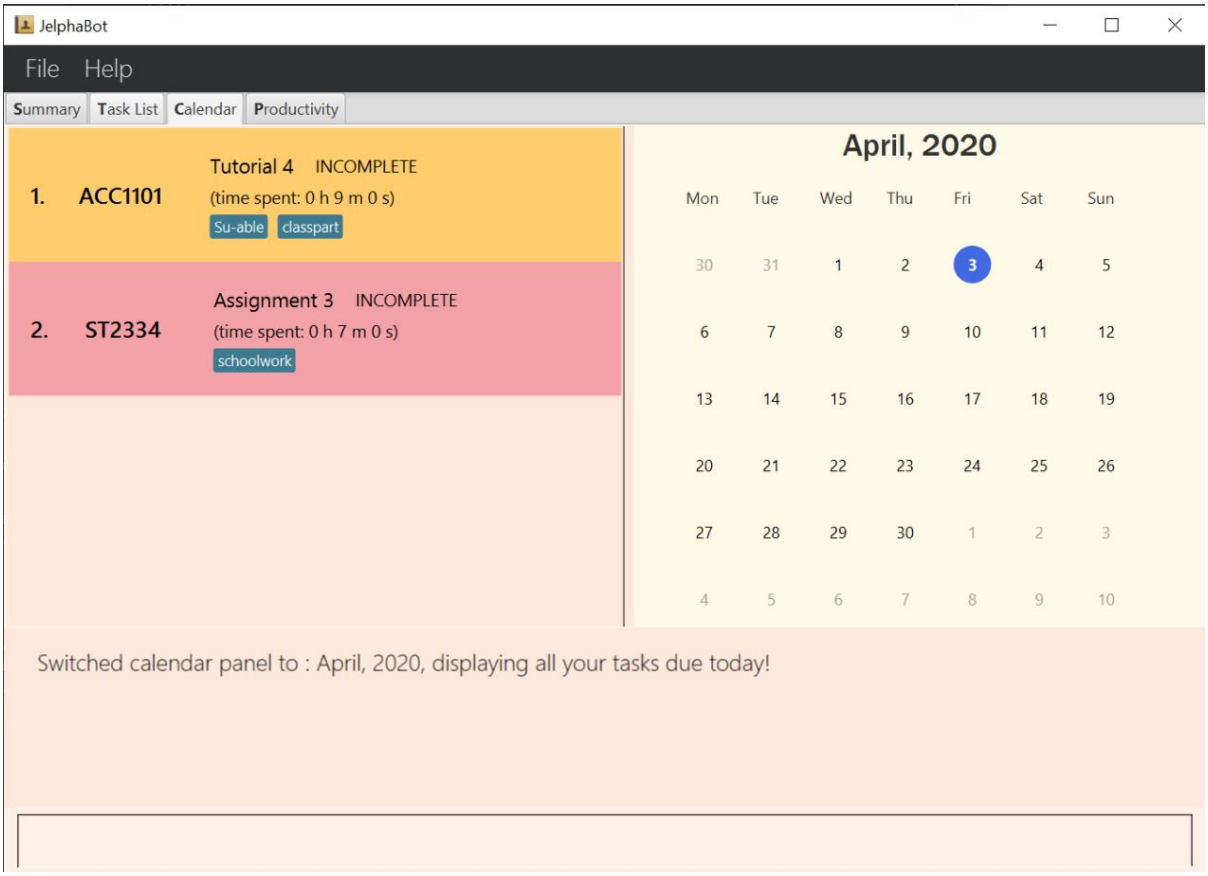

Figure 4. Example of expected result after running `calendar today`

Displays month of April in the year 2020 in the calendar panel on the right, with today's date highlighted and displays the corresponding tasks due today on the left.

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. The full Developer Guide can be found here.

Calendar feature (Amanda)

JelphaBot has a calendar feature which provides an overarching view of their schedules and to allow users to view their tasks due.

This feature offers two main functions:

- Displays an overview of tasks in calendar for a selected month and year
- Displays a list of tasks due for a specified date

Implementation

The implementation of the main calendar panel is facilitated by the `CalendarMainPanel` class, which serves as the main container for this feature. This main container consists of a `SplitPane` comprising of a `CalendarPanel` on the right, which displays the calendar view in a month, and a

`CalendarTaskListPanel` on the left to display specific tasks.

The diagram below describes the class structure of the calendar class structure.

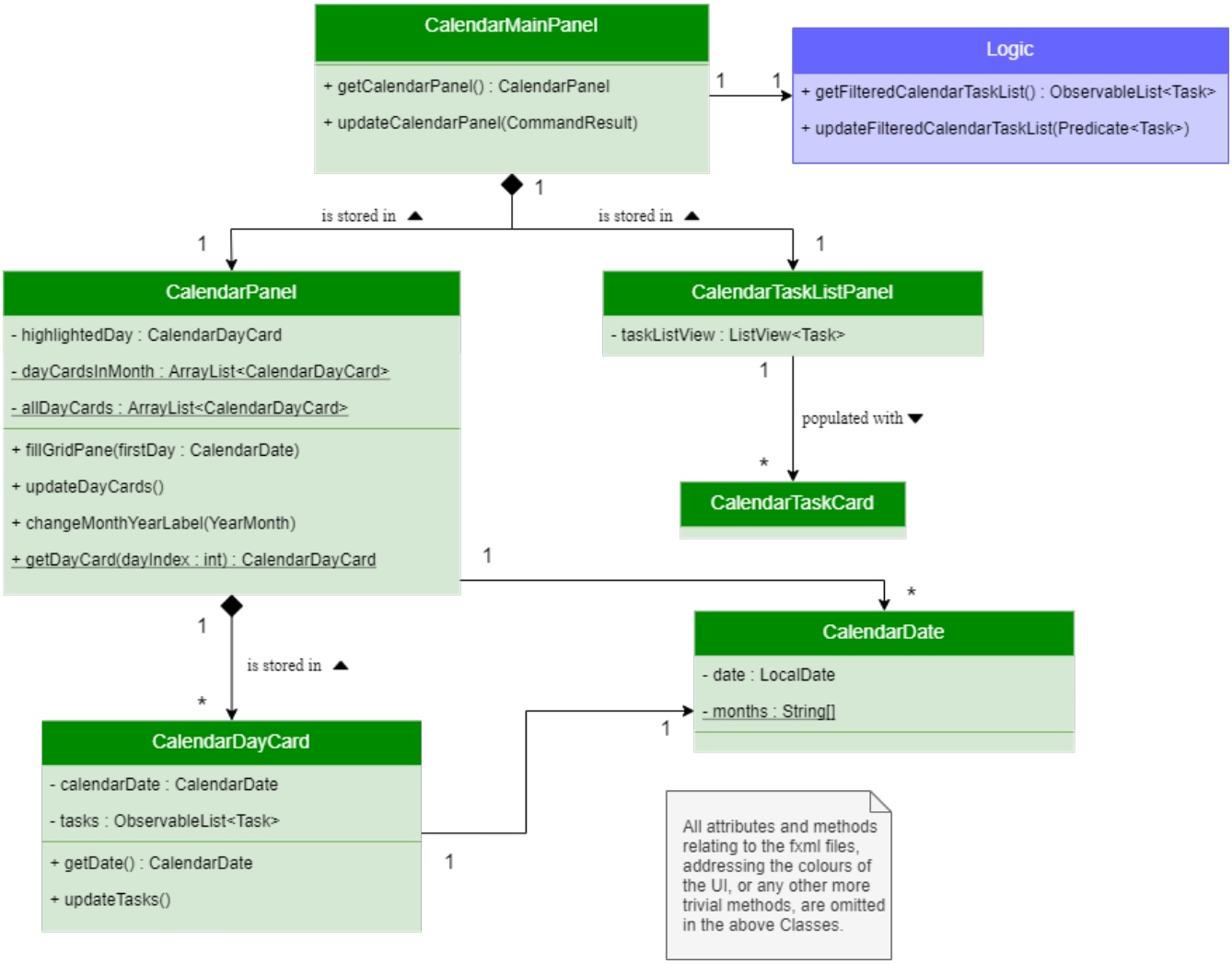

Figure 5. Class Diagram for Calendar classes

Upon initialisation of the `CalendarMainPanel`, the `CalendarPanel` would be set to display the current month and year calendar, with the dates filled up by `CalendarDayCards` by CalendarPanel#fillGridPane() with a `CalendarDate` starting from the first day of the current month. Today's date would also be highlighted, with `CalendarTaskListPanel` set to display the tasks due today by running Logic#getFilteredCalendarTaskList() and then Logic#updateFilteredCalendarTaskList() with a predicate to filter by today's date.

The following diagram depicts how each individual day cell of the calendar will look like:

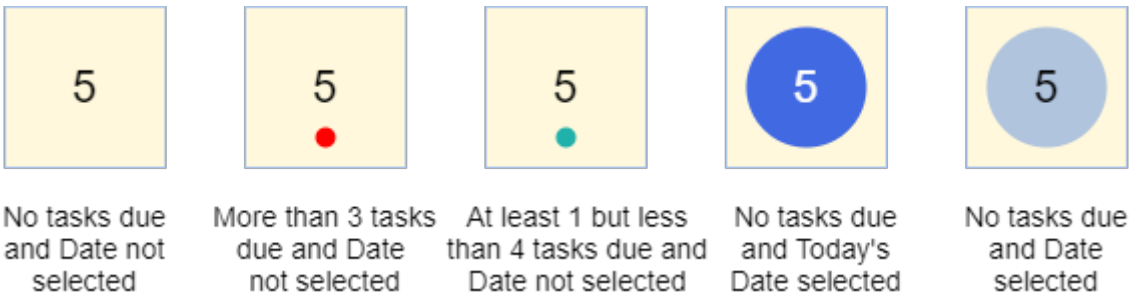

Figure 6. Expected display of dates on calendar

After every execution of command, MainWindow#updateTasksInCalendarDayCards() will be run such that any commands that updates the JelphaBot task list (e.g `DoneCommand`, `DeleteCommand`, `EditCommand`) would be updated by the dot indicators in the calendar.

Function 1: Displays an overview of tasks in calendar for a selected month and year

There are 2 commands that users can issue to perform function 1:

1. `calendar today`: Displays calendar for the current month with today's date highlighted, and its corresponding tasks due listed.
2. `calendar MONTHYEAR`: Displays calendar for the month and year specified, with the first day of the month highlighted, and its corresponding tasks due listed (e.g. calendar Apr-2020). Refer here, for the diagram describing this process.

Upon execution of the `calendar MONTHYEAR` or the `calendar today` command, CalendarCommand#execute() will run `updateFilteredCalendarTaskList()` to filter the task list to display the tasks on the `CalendarTaskListPanel` according to the first day of the `MONTHYEAR` or the tasks due `today` respectively. The filtering of the tasks according to date is done using the `TaskDueWithinDayPredicate`. A distinct `CommandResult` would then be generated according to the input command and is returned to the `LogicManager`. Finally, the `CommandResult` is passed to the `MainWindow` in UI. Now, the updates can be done for the respective components:

UI Component: Using the CommandResult, MainWindow calls MainWindow#updateCalendarMainPanel(), which is then passed to call CalendarMainPanel#updateCalendarPanel(). For the `calendar MONTHYEAR` command, this updates the `CalendarPanel` display with the respective `MONTHYEAR` view, and highlights the first day of the month. For the `calendar today` command, this updates the `CalendarPanel` display to the current month and year, with today's date highlighted.

The following example sequence diagram shows you how the `calendar MONTHYEAR` (e.g. `calendar Apr-2020`) command works.

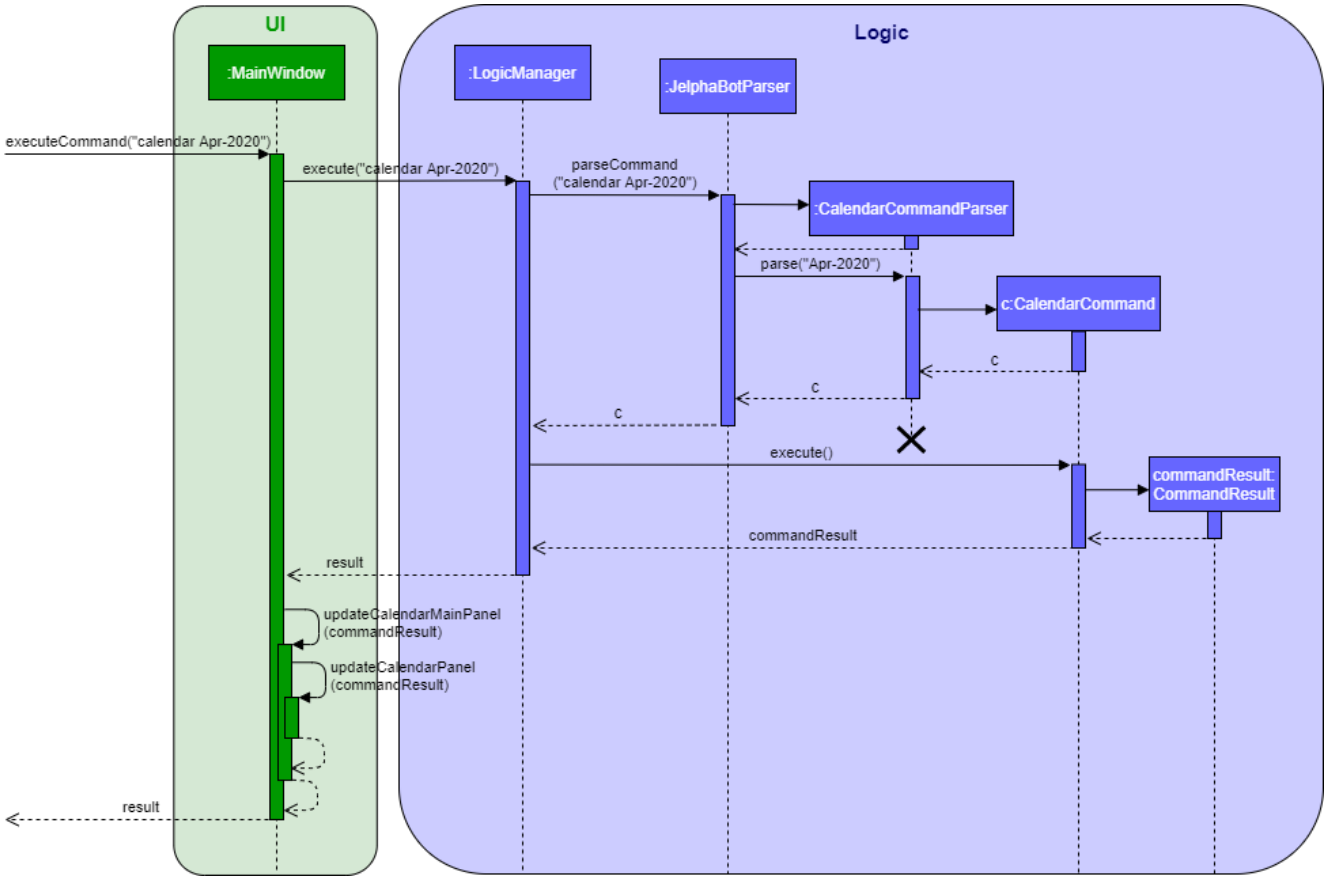

Figure 7. Sequence diagram after running `calendar Apr-2020`

Function 2: Display a list of tasks due for a selected date in the month

In order to display the task list for specific input dates, the user enters the `calendar DATE` command (e.g. calendar Jan-1-2020).

NOTE	Only a date belonging in the current displayed month on the `CalendarPanel` would be highlighted after processing the `calendar DATE` command. A date that falls in other month and years would just display its corresponding tasks due on the `CalendarTaskListPanel`.

The implementation of the previous two calendar commands (`calendar DATE` and `calendar today`) are largely similar and run in the same process. The only exception is regarding the `calendar DATE` command which fulfills **Function 2** listed above, where the `GridPane` in `CalendarPanel` is not altered by running CalendarPanel#fillGridPane() unlike the other two commands fulfilling **Function 1**. Only `CalendarTaskListPanel` is updated.

The following diagram shows the sequence flow for variants of these three calendar commands which modifies the `CalendarMainPanel`:

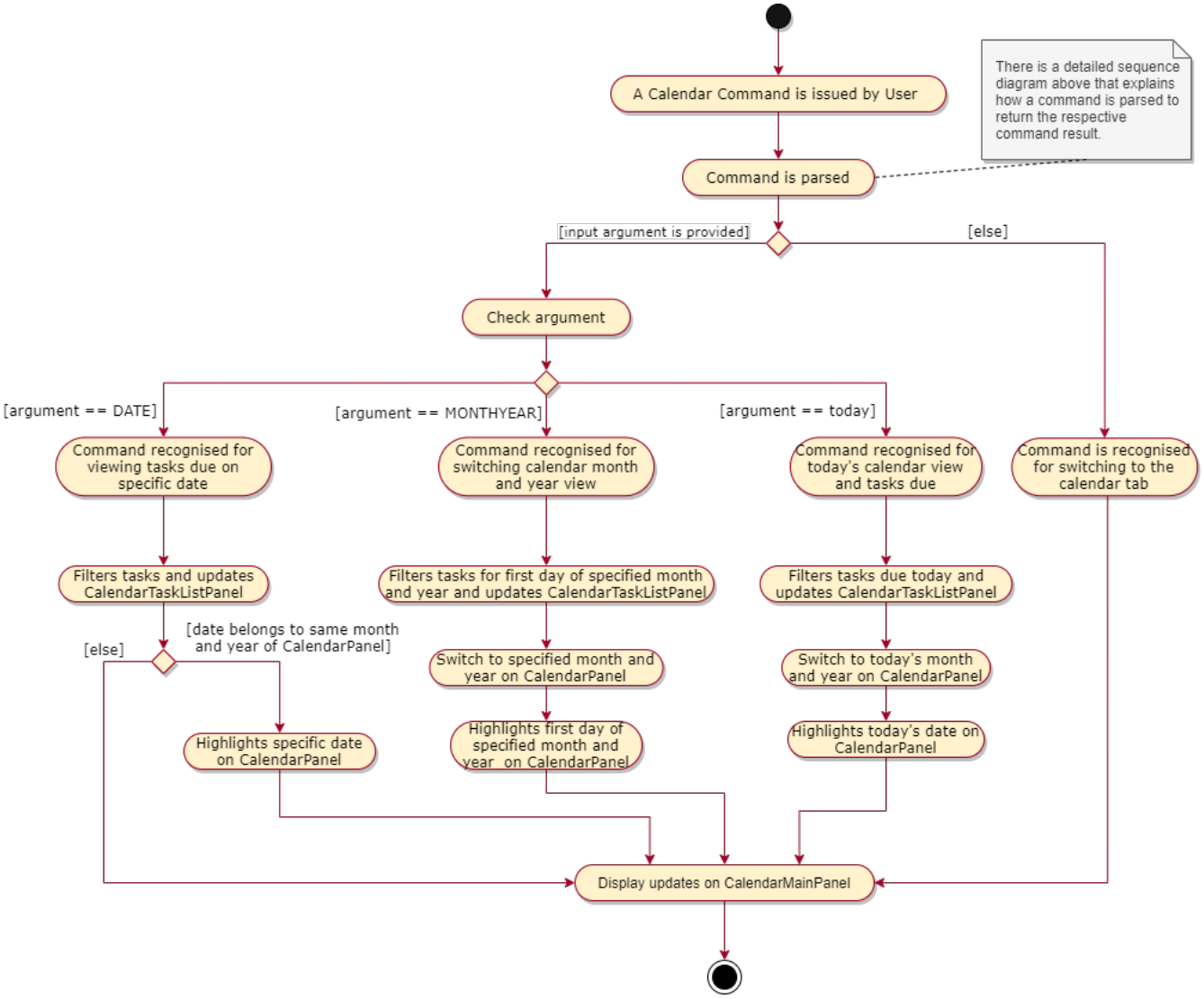

Figure 8. Activity Diagram showing the updating of `CalendarMainPanel`

Design Considerations

Aspect 1: How the user can navigate between specific dates and change the calendar month view

- **Current Solution:** Use the same `calendar` command word for both viewing tasks in specific dates, and changing the calendar view. The next input following the command word (`DATE`, `MONTHYEAR`, `today`) is then parsed separately to give different command results.
  - Pros: Easier and more understandable for user interactions.
  - Pros: More open and accessible to future implementations regarding the calendar feature.
  - Cons: Implementation in the `CalendarCommand` class might seem a bit bulky.
- **Alternative 1:** Use completely separate commands for viewing tasks in specific dates and changing the calendar view.
  - Pros: Less chance of a parse exception, with more precise error messages when invalid command formats are input by the user.
  - Cons: Certain areas of the code might be repetitive.
  - Cons: Less intuitive for users to use.

Reason for chosen implementation:
The current solution is more user-friendly as it reduces the number of varying commands that users have to remember in order to access the respective information. Additionally, upcoming changes and future implementations can be easily integrated into the existing code base as well.

Aspect 2: Method of storing `ObservableList<Task>` of tasks for each day card (Implementation of the Dot Indicator)

- **Current Solution:** Each `CalendarDayCard` stores a filtered list of tasks due on its specific date. This is done by obtaining all the tasks in the task list from Logic#getFilteredTaskList() and applying a filter function with the `TaskDueWithinDayPredicate`, specifically with the date of the day card. The list of tasks stored for each day card in the calendar panel would be re-filtered after the execution of each command.
  - Pros: Do not have to manually update the tasks stored in each `CalendarDayCard` (e.g add and remove manually in the separately stored copy)
  - Cons: Completely reliant on the main task list, possible errors might be carried over.
- **Alternative 1:** Use a static HashMap of Dates as keys and a list of tasks due in that date as values.
  - Pros: Retrieving the tasks in a specific date and storing in the day card is fast - can be done in O(1) time.
  - Cons: Implementation would be much more complex.
  - Cons: Updating of this HashMap of the tasks as the main task list is being edited constantly can be very tedious.

Reason for chosen implementation:
The current solution is easier to implement since we are filtering the tasks we want to see directly from the main task list. This reduces the amount of methods to implement over various class and components as constant updates of the tasks in each day card of the calendar is done. The ease of implementation is crucial given the tight deadlines we have to meet for the project.