Zhang Yiping - Project Portfolio

PROJECT: FitHelper

Overview

FitHelper is a desktop address book application used for teaching Software Engineering principles. 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

- **Enhancement added-calendar display**
  - What it does: allows the application to have a monthly/weekly display of entries
  - Justification: This feature allows the user to have a clearer idea of his/her food entries/sports entries status
- **Code contributed**: [Functional code][Pull Request]
- **Other contributions**:
  - Model
    - Add duration attribute for entry, add natty date parser to parse date (removed)
  - Project management:
    - Managed releases `v1.3` - `v1.3.1` (2 releases) on GitHub
  - Enhancements to existing features:
    - Wrote additional tests for existing features to increase coverage from 0% to 17.29% (Pull requests #317, #308, #213)
  - Documentation:
    - Did cosmetic tweaks to existing contents, added screenshots with explanations, and created the calendar part: #325, #264
    - Edited existing uml diagrams, updated design of model component and created implementation of entry and calendar feature: #322, #223, #220
  - Community:
    - Contributed to forum discussions (40, 71, 80)
    - Reported bugs and suggestions for other teams in the class (https://github.com/ZhangYiping126/ped/issues)
  - Tools:
    - Set up Codacy to perform code quality check for the group fork

- Set up AppVeyor to perform Continuous Integration(CI) for the group fork
- Integrated a third party library (jfxtras icalendar) to the project (#157)

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.

Calendar View: `calendar`

Calendar view will display all the food and sports entries for each day of the referenced week/month. The referenced date is default set to be the current date, but can be changed to a user given date. Calendar can be switched between two modes, list mode or timetable mode, with default set to be in timetable mode.

Format for entering calendar view: `calendar`

Calendar list mode: `calendar m/ls [d/DATE]`

- **Monthly View**
  The lists of entries for food and sports are display by their given dates in the referenced month. Completed entries will be strikethrough. Calorie value is also shown for each date.
- **Calendar of the month**
  On the top right corner, the dates of the referenced month displayed, with red-colored date if the given date has more calorie intake than calorie burnt (considering the entries with status done of that particular date), else if calorie burnt is greater than calorie intake, the font color is green. Default color is blue. For each date, upon clicking, a popup window will show all entries of the date. The ones with strikethrough in text means the status of the entry is done. For this functionality, the corresponding command is `calendar sh/DATE`.
- **Daily calorie status**
  On the bottom right corner, calorie status for existing entries will be displayed by their dates, showing the calorie intake from food, calorie burnt from sports as well as total calorie. The calculation only considers entries which are completed.

Format for switching to list mode: `calendar m/ls [d/DATE]`

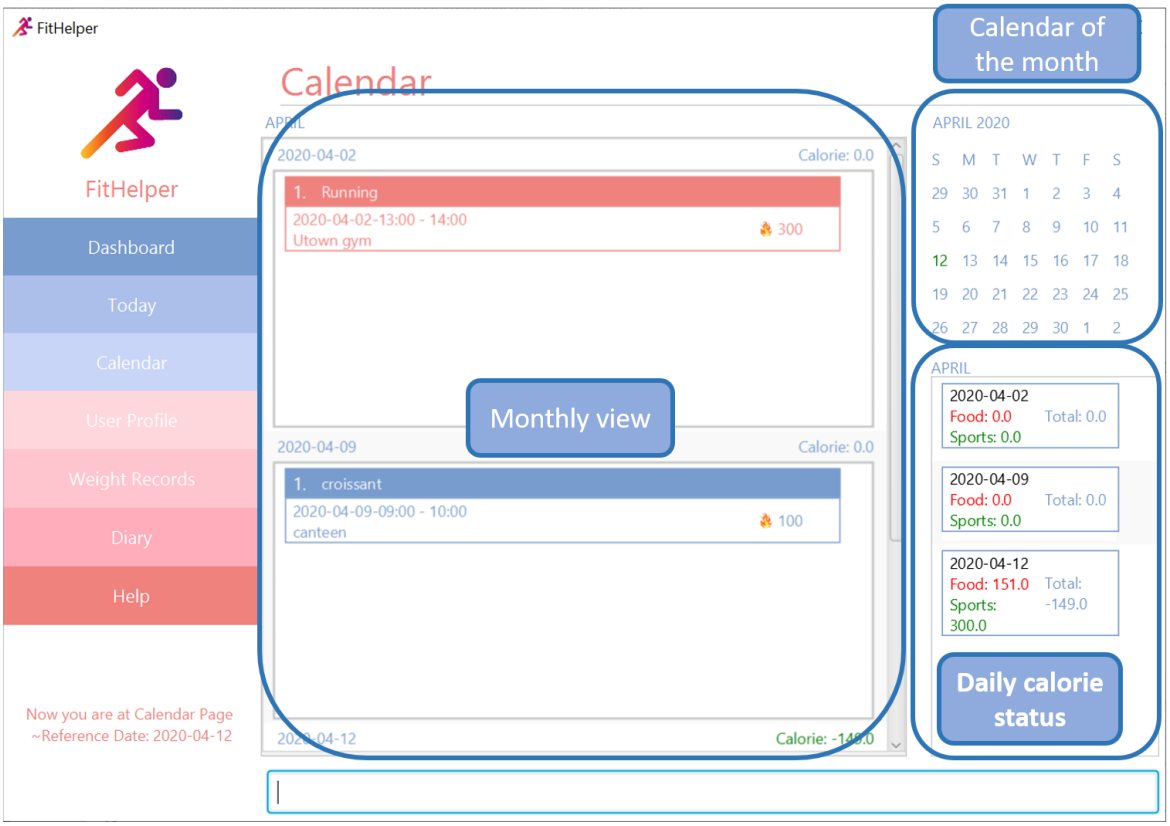

Figure 1. Calendar list mode

Calendar timetable mode: `calendar m/tb [d/DATE]`

- **Weekly View**
  The entries of food and sports of the given time period are displayed. Food entries will be in pink, sports entries will be in blue, and entries completed will be in grey regardless of their type. No entries can have time clashes. Entries with long names or locations will not be displayed fully. Only when mouse is over the entry, the details will be shown. Entries upon clicking will have console errors, caused by disabling certain functionality from third party libraries.
- **Calendar of the month**
  Same as above.
- **Upcoming list**
  It displays all entries after the current date and time and within the referenced month. Only entries with status `Undone` will be shown. So if the referenced month is in the past, no entries will be shown.
  Format for switching to timetable mode: `calendar m/tb [d/DATE]`

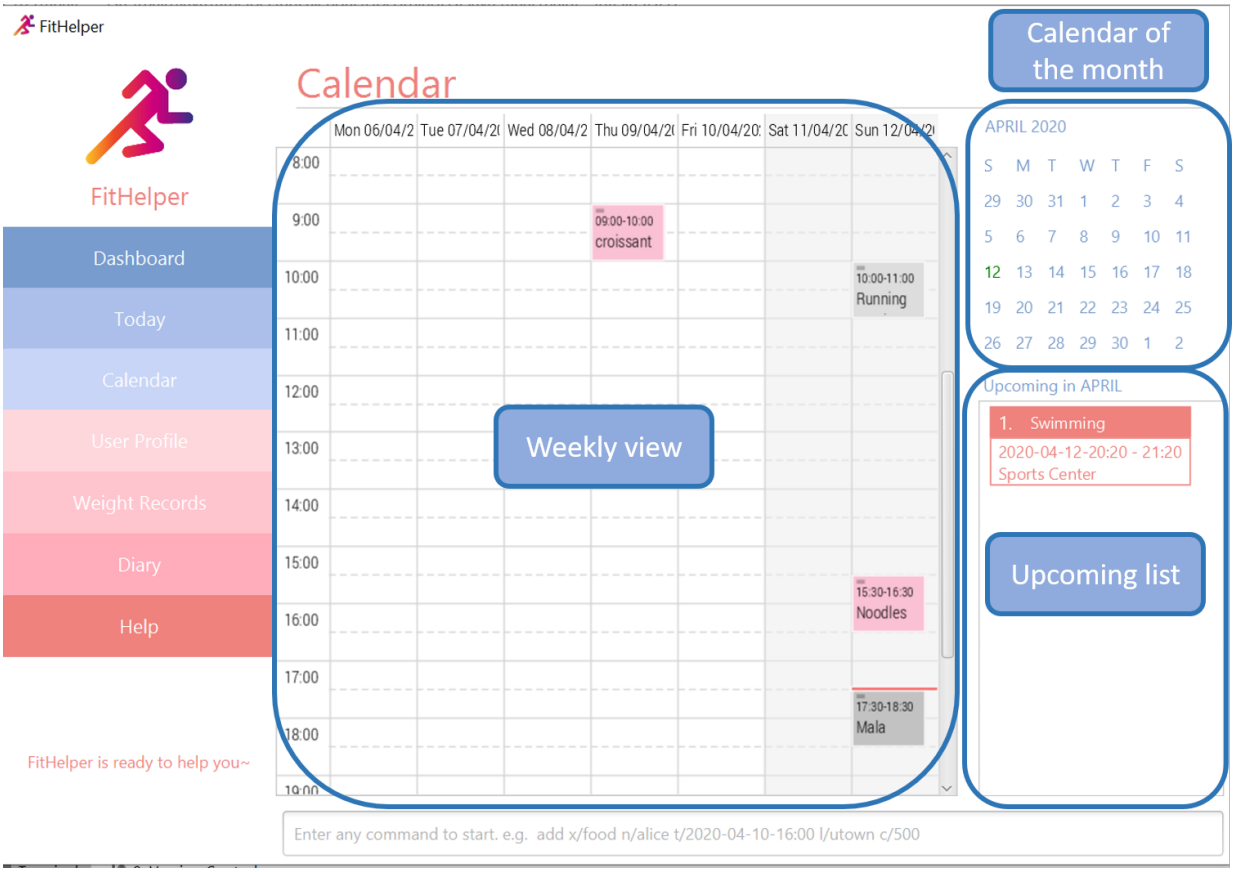

Figure 2. Calendar timetable mode

Calendar display entries from a day: `calendar sh/DATE`

A pop up window will display all entries from the chosen date.

Format for showing entries of a particular date: `calendar sh/DATE`

Examples:

- `calendar d/2020-04-15`
- `calendar m/ls d/2020-04-20`
- `calendar sh/2020-05-01`

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.

Model component

The `Model`,

- stores a `UserPref` object that represents the user's preferences.
- stores `UserProfile` and `WeightRecords` objects for user's personal information.

- stores the FitHelper data.
- stores `FitHelperCommit` and `VersionedFitHelper` objects for execution of `redo` and `undo` instructions.
- exposes multiple unmodifiable `ObservableList<Entry>` and one unmodifiable `ObservableList<Diary>` that can be 'observed' e.g. the UI can be bound to this list so that the UI automatically updates when the data in the list change.
- does not depend on any of the other three components.

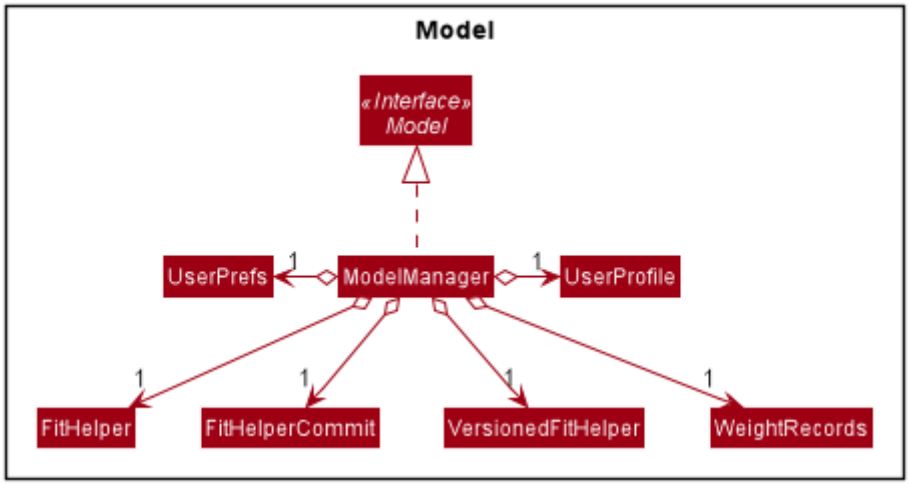

Figure 3. Structure of the Model Component

API : `Model.java`

Below are the class diagrams for different components of model

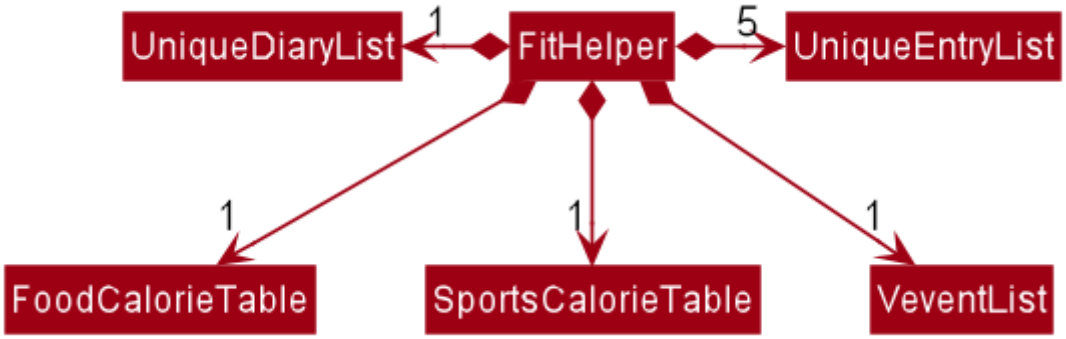

Figure 4. Class Diagram for FitHelper

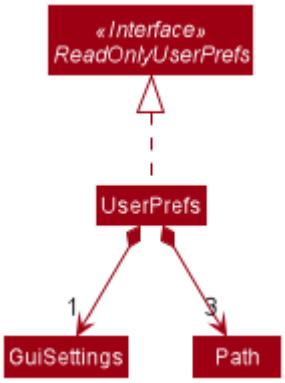

Figure 5. Class Diagram for UserPrefs

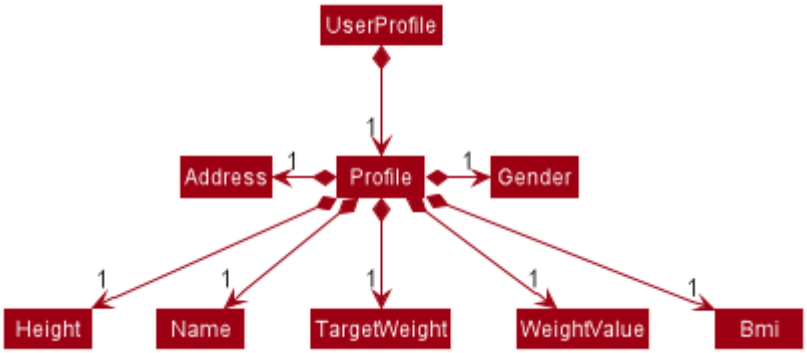

Figure 6. Class Diagram for UserProfile

Figure 7. Structure of VersionedFitHelper and FitHelperCommit

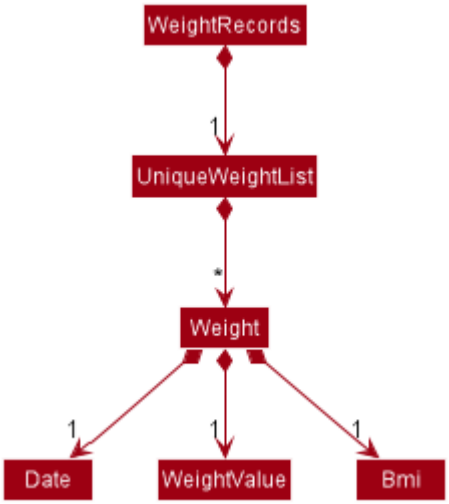

Figure 8. Class Diagram for WeightRecords

Entry Feature

The Entry consists of the following:

- Each `Entry` consists of a unique combination of `Name`, `Calorie`, `Location`, `Duration`, `Type`, `Remark`, `Status` and `Time`
- Each `Entry` consists of a `Duration` in hours, default set to 1, smallest accuracy is 0.02 (1 min).
- Each `Entry` consists of a `Type`, either food or sports
- Each `Entry` consists of a `Remark`, default set to be empty
- Each `Entry` consists of a `Status`, either `Done` or `Undone`
- Each class has their respective getter methods

The class diagram below is an overview of the `Entry` class.

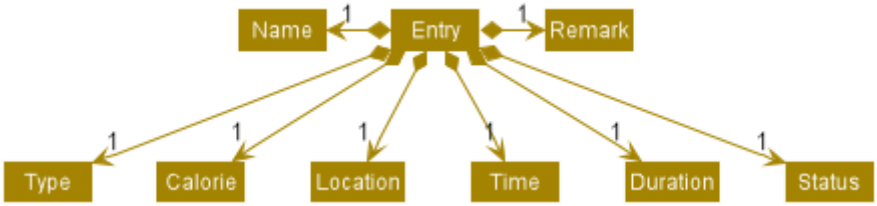

Figure 9. Entry Class Diagram

Implementation of Entry Commands

`Entry` class supports multiple commands. It includes:

- `AddCommand` - Adds a `Entry` into `FitHelper`
- `DeleteCommand` - Deletes a `Entry` from `FitHelper`
- `EditCommand` - Edits a `Entry` from `FitHelper`
- `FindCommand` - Finds all `Entry` whose `name` contains the keywords user entered
- `ListCommand` - Lists all `Entry`

All the above entry commands will be parsed in `FitHelperParser` and based on their types (i.e Add, Delete, Edit etc), the corresponding parsers will be invoked (i.e `AddCommandParser`, `EditCommandParser` etc). After which, the corresponding command will be executed (i.e `AddCommand`, `EditCommand` etc).

The figure below shows the execution of an `EditCommand`.

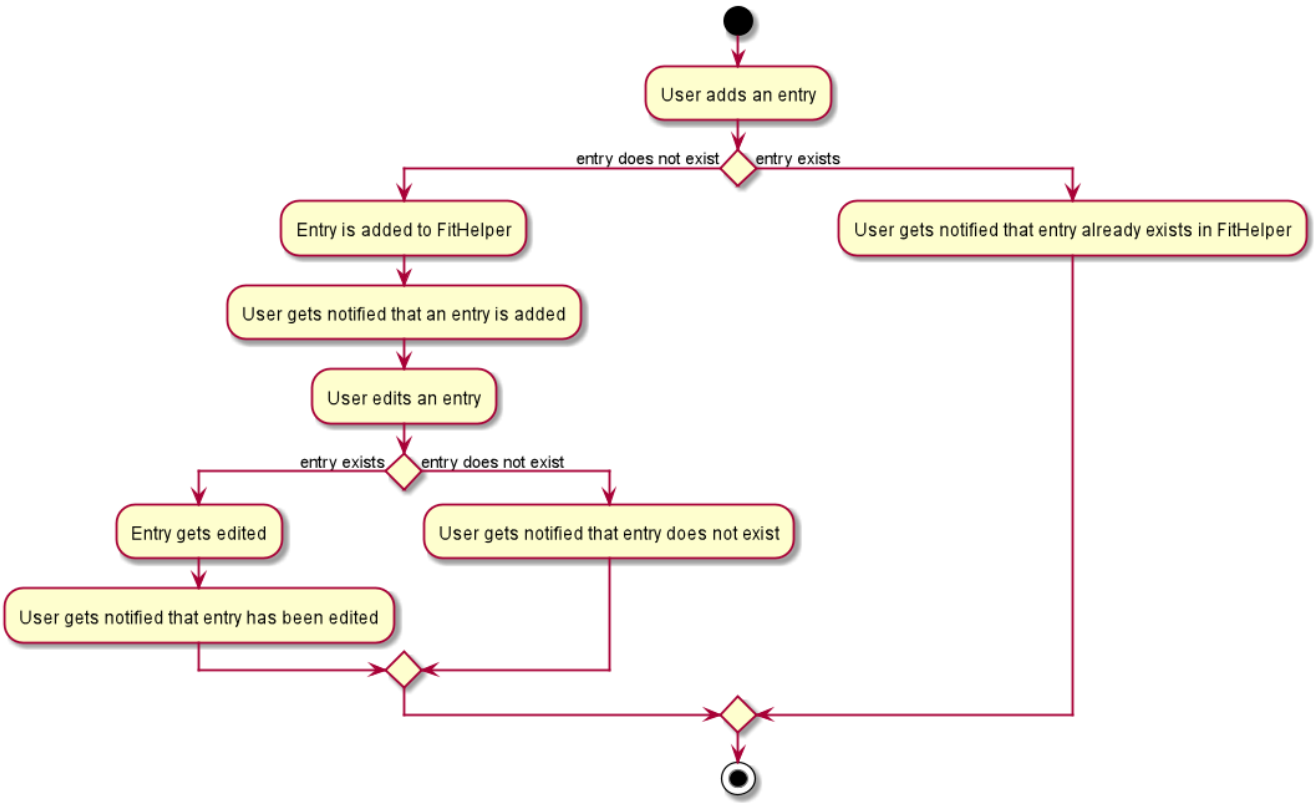

Figure 10. EditCommand Activity Diagram

After a successful execution, the entry with the given index will be edited from FitHelper.

Design Considerations

Aspect: `Type` for the entry

- **Option 1:** As a string attribute in `Entry`
  - Pros: Easy to implement, less code required
  - Cons: Provides a lower level of abstraction
- **Option 2:** Use two different classes to represent types, such as `FoodEntry` and `SportsEntry`
  - Pros: Higher level of abstraction
  - Cons: More code, generic types are required for implementation of common functionality

In the end, we chose Option 1 as it reduces the amount of duplicated code required, given that all parameters of food entries and sports entries are the same. However, Option 2 is still an viable option.

Aspect: `Time` for the entry

- **Option 1:** Fix the format of `Time` to be `yyyy-MM-dd-hh:mm`
  - Pros: Easy to implement, less bug prone
  - Cons: Adds inconvenience to the user
- **Option 2:** Use natty, the natural language date parser
  - Pros: Brings more convenience for CLI users

    - Cons: More bug prone due to the inaccuracy of the date parser. Moreover, only date can be parsed, not time.

Consequently, we chose Option 1 as it standardized the format of date and time across this application.

Aspect: `Duration` **for the entry**

- **Option 1:** As an optional attribute
    - Pros: More user friendly, since duration for food entry is less meaningful
    - Cons: Calendar display will not able to display food entries
- **Option 2:** As an optional attribute, with default set to 1
    - Pros: Calendar display will not able to display food entries with no duration provided
    - Cons: The duration does not reflect the true value when user chooses not to enter

We chose Option 2 for better display of entries on the calendar === Calendar

Implementation

1. The user enters a view command in the `calendar d/2020-04-13`.
2. `LogicManager` parses the user input, constructs and executes the `CalendarCommand`.
3. The `CalendarCommand` reaches `setCalendarDate`, `setCalendarMode`, `setCalendarShow` in the `Model` and returns the `CommandResult` to the `LogicManager`.
    - `Model#setCalendarDate()` — Set the referenced date for calendar, default set to current date.
    - `Model#setCalendarMode()` — Set the calendar display mode, can be either `list` or `calendar` mode.
    - `Model#setCalendarShow()` — Set the display of entries of a particular date, default set to `null`.
4. The `LogicManager` returns the `CommandResult` to the `Ui`.
5. The `Ui` gets the `CommandResult` from `LogicManager` and updates the Ui to display the module. The following sequence diagram shows how the update operation works to change calendar page:

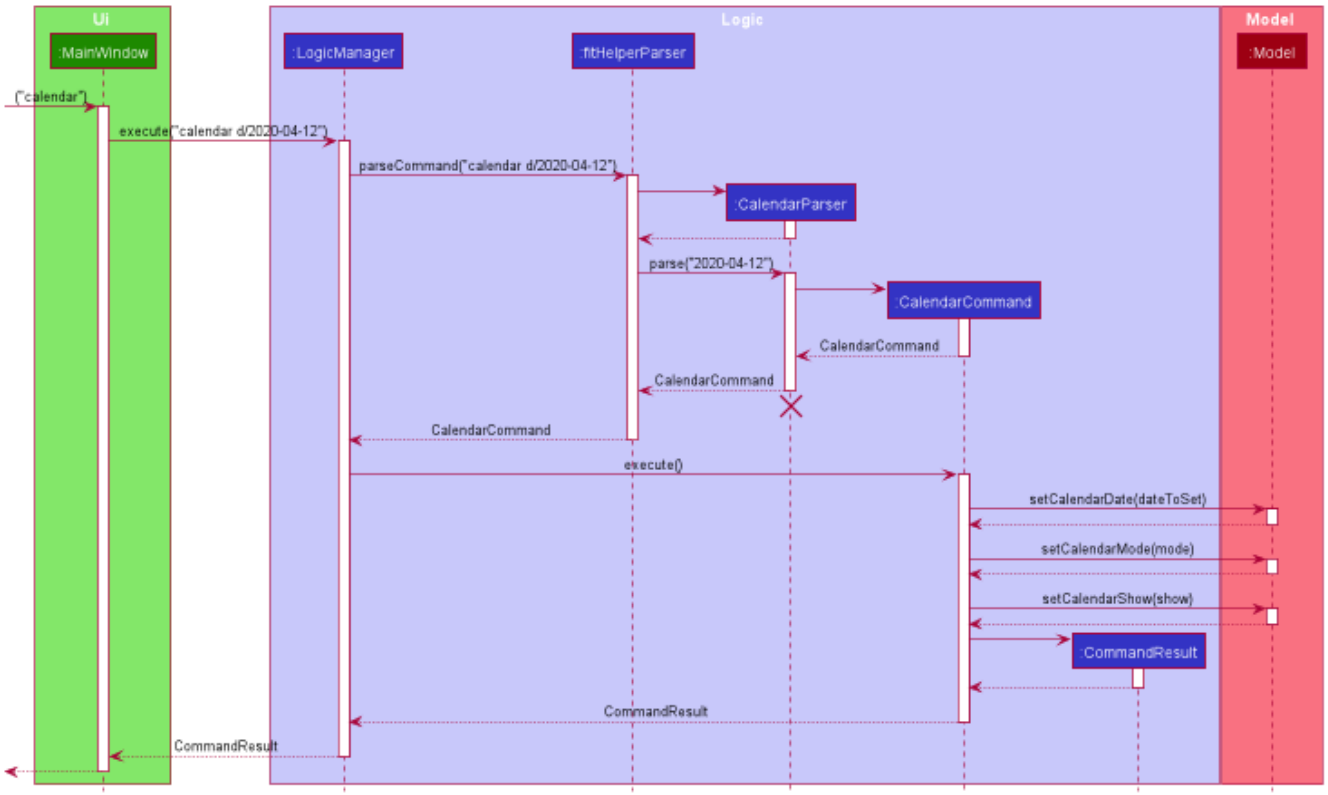

Figure 11. CalendarCommand Sequence Diagram

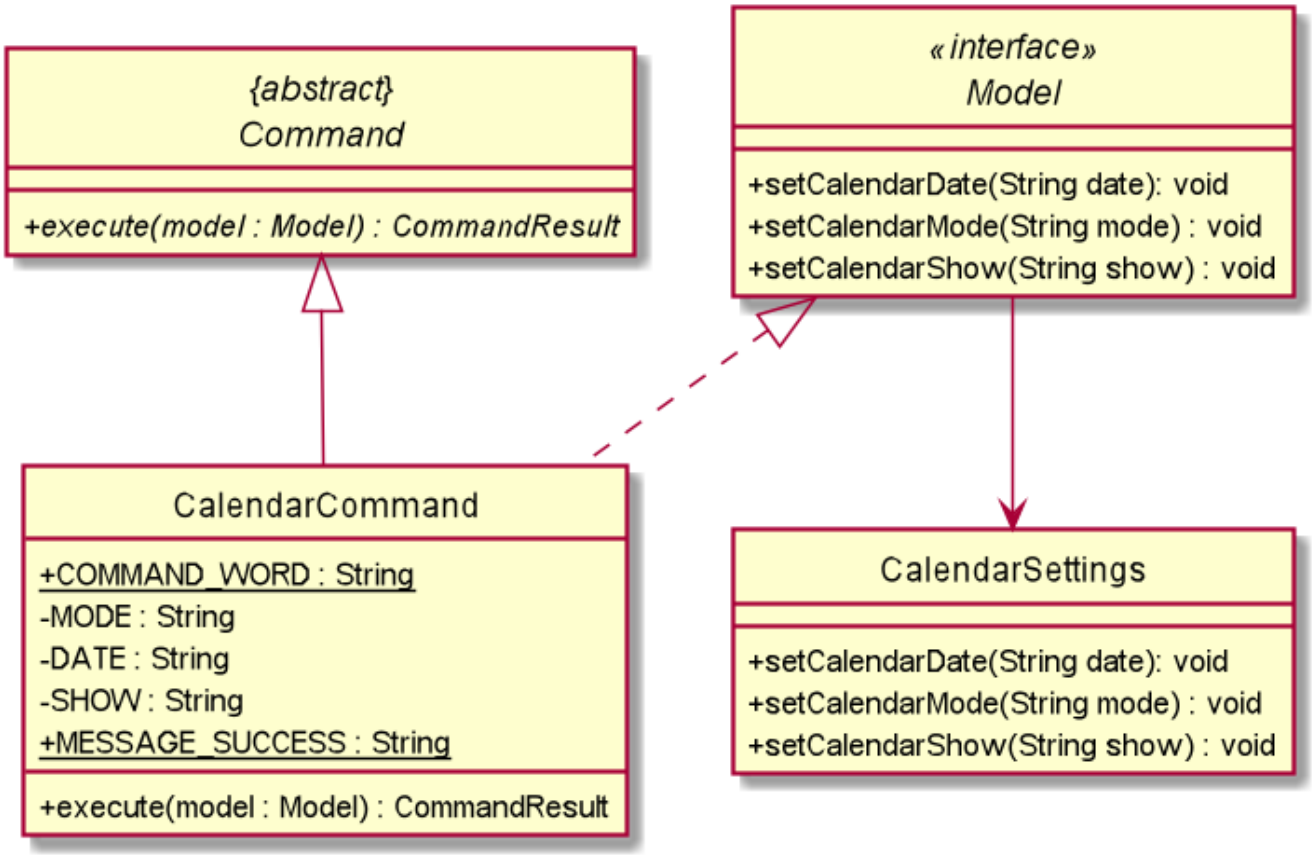

Figure 12. CalendarCommand Class Diagram

Design consideration

Aspect: Allowing no time clashes for all entries

- **Option 1:** Allow multiple entries to exist over the same time period
  - Pros: More user friendly, since users might be doing multiple things for a given time period
  - Cons: Calendar display will not able to display food entries
- **Option 2:** No time clashes allowed
  - Pros: Calendar display becomes clearer
  - Cons: Users are not given the freedom to add multiple entries with the same time period

We chose Option 1 for better display of entries on the calendar