Liao Meng - Project Portfolio

PROJECT: FitHelper

Overview

FitHelper is a desktop diet-and-exercise-recording application. It is built based on AddressBook (Level 3), 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 21 kLoC.

Summary of contributions

- **Major enhancement 1**: added **the ability to search for calorie data of common food/sports**
  - What it does: allows the user to search for calorie intake/consumption of food/sports by keywords.
  - Justification: This feature makes it easier for the user to estimate how much calorie he/she will gain/burn by eating/exercising, so that he/she can create food/sports entry without the need to search online for relevant information.
  - Highlights: The application will perform up to 3 rounds of search, each round with different criteria (from most strict matching to less strict matching), to find matching data. This ensures that more relevant data are found and added first, and at the same time loosely-matching data can be shown when strictly-matching data are not found.
  - Credits: The calorie data are collected from the following websites: https://www.calories.info/ and https://www.nutristrategy.com/caloriesburned.htm
- **Major enhancement 2**: added **the ability to sort entry and reminder list**
  - What it does: allows the application to sort and display the entries in food and/or sports entry list based on time/name/calorie value in ascending or descending order. The user can specify sort order and criterion. Note that the entries in the reminder list will also be sorted in the same manner.
  - Justification: This feature allows the user to view entries in a particular order. It also helps the user to view entry that he/she may deem more important first (for example entry with a higher calorie value, as it has a larger impact on the user’s fitness), which facilitates editing.
  - Highlights: the user can specify which list to sort and in what order.
- **Minor enhancement**: added and updated the Help Page, which shows the list of valid user commands and their usage, and the link to the user guide.
- **Code contributed**: [Functional and Test code]
- **Other contributions**:
  - Project management:

        - Managed release `v1.2.1` on GitHub
    - Team-Based tasks:
        - Frequently performed manual testing of the application, and gave feedback about bugs/feature flaws to team members promptly.
    - Testing:
        - Added test cases (Pull requests #323, #332).
    - Documentation:
        - Reviewed and edited sections of the User Guide according to the change in application design and functionality (refer to **Contributions to the User Guide** for details)
    - Community:
        - Contributed to forum discussion #118.

Contributions to the User Guide

Given below are sections I wrote in the User Guide.

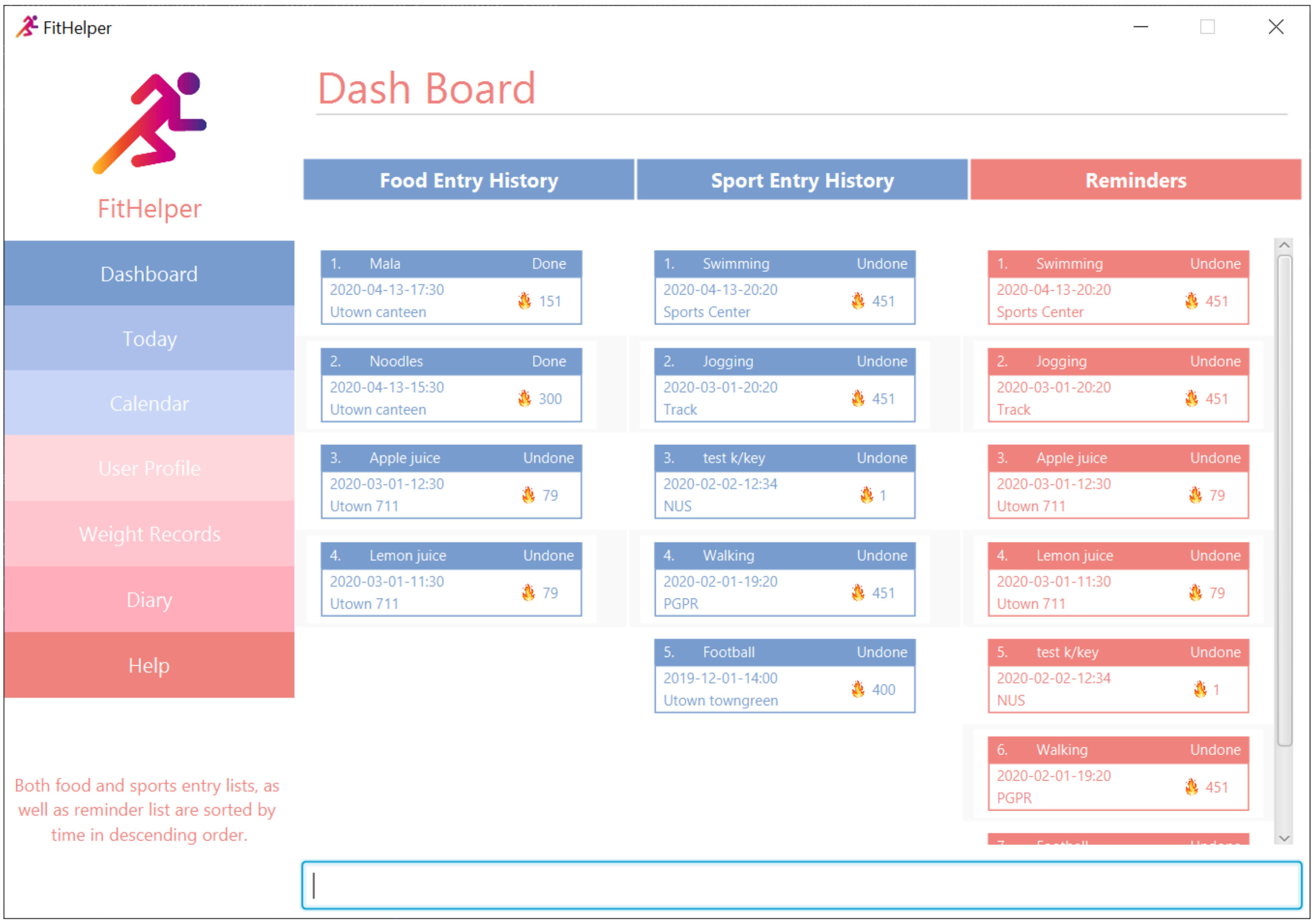
Sort food/sports entry list: `sort`

Sorts food or sports entry list, or both of them, based on starting time or calorie value or name of the entry (case insensitive), in either ascending or descending order. In addition, the reminders entry list will also be sorted in the same manner specified by the user. The user must specify the sorting criterion `cal/c OR time/t OR name/n`. If the user does not specify the type of entry list, both lists will be sorted. And if the user does not specify the sorting order `a OR d`, the default order is descending (i.e. entry that starts later or entry with higher calorie value or entry whose name starts with later alphabet comes first).

format: `sort [x/TYPE] by/SORT_BY [o/ORDER]`

Examples:

- `sort x/f by/time o/a` (sort food entry list and reminders list in ascending order of recording time, i.e. the older entry comes first)
- `sort by/c` (sort both food and sports entry list, as well as reminders list, in descending order of calorie value, i.e. entry with higher calorie intake/consumption comes first)



Food Entry History

1. Mala Done
2020-04-13-17:30
Utown canteen
151

2. Noodles Done
2020-04-13-15:30
Utown canteen
300

3. Apple juice Undone
2020-03-01-12:30
Utown 711
79

4. Lemon juice Undone
2020-03-01-11:30
Utown 711
79

Sport Entry History

1. Swimming Undone
2020-04-13-20:20
Sports Center
451

2. Jogging Undone
2020-03-01-20:20
Track
451

3. test k/key Undone
2020-02-02-12:34
NUS
1

4. Walking Undone
2020-02-01-19:20
PGPR
451

5. Football Undone
2019-12-01-14:00
Utown towngreen
400

Reminders

1. Swimming Undone
2020-04-13-20:20
Sports Center
451

2. Jogging Undone
2020-03-01-20:20
Track
451

3. Apple juice Undone
2020-03-01-12:30
Utown 711
79

4. Lemon juice Undone
2020-03-01-11:30
Utown 711
79

5. test k/key Undone
2020-02-02-12:34
NUS
1

6. Walking Undone
2020-02-01-19:20
PGPR
451

Both food and sports entry lists, as well as reminder list are sorted by time in descending order.

Figure 1. Dashboard page after all 3 lists are sorted by time in descending order (later entry comes at top)

Check calorie intake/consumption of common food/sports: `check`

Searches through the pre-set database for calorie intake/consumption information about common food/sports (i.e. a datum) whose name matches/contains keywords specified by the user. Note that due to space constraint, at most 3 matching data will be shown at the left-bottom corner of the application. Each datum shows the name of the food/sports and its calorie intake/consumption information: for food, it will be in cal per serving (weight per serving in gram will also be shown); for sports, it will be in cal per hour for a 70kg person. Please note after the command is executed, the application will automatically go to Dashboard page **without resetting the display status**.

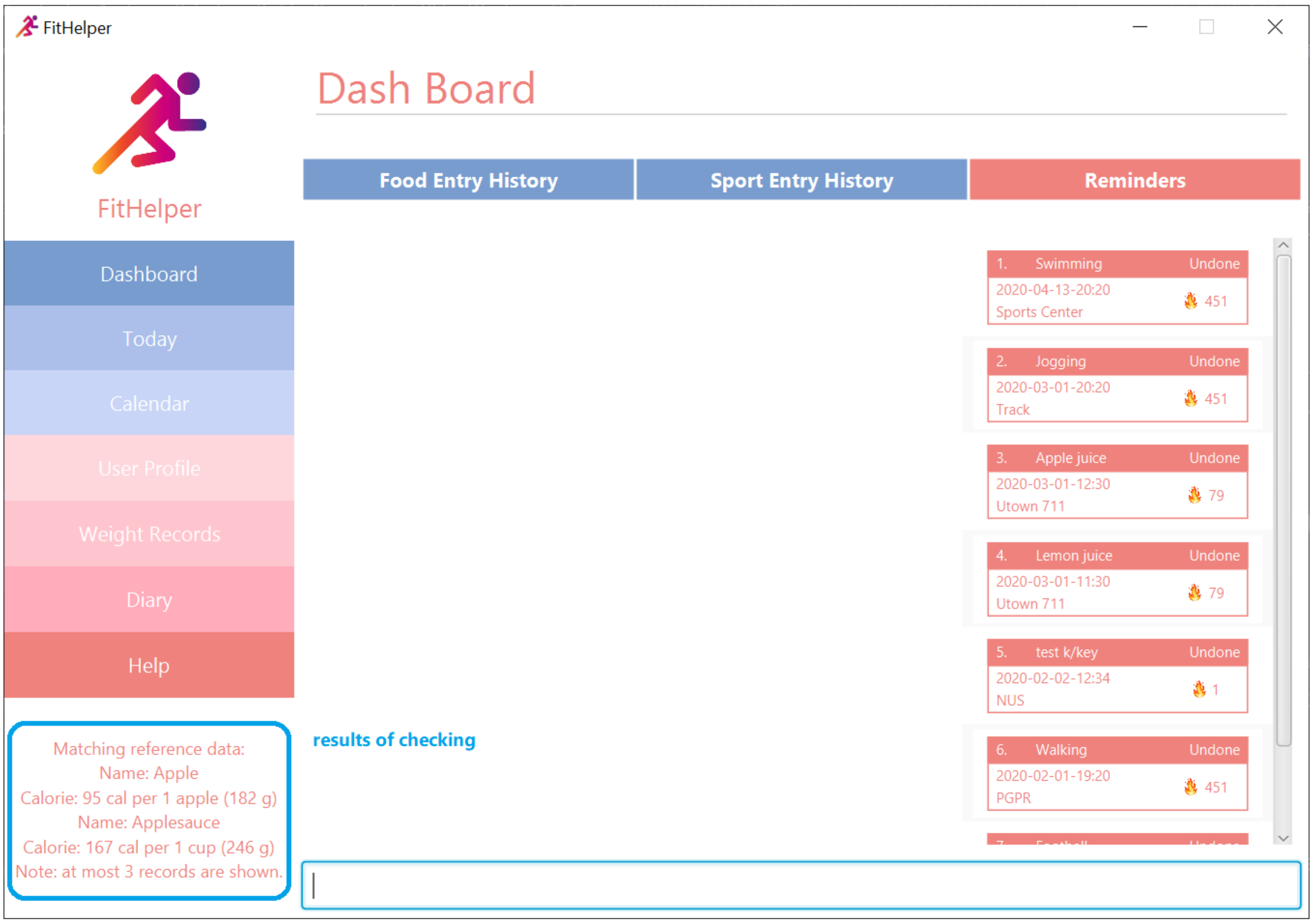
The searching works as follows:

- Retrieves food/sports data based on check type specified by the user.
- Checks and adds food/sports datum whose name **equals one of the keywords**. If fewer than 3 food/sports data are added so far, checks and adds food/sports datum whose name **contains the keywords as a whole**. If still fewer than 3 food/sports data are added so far, checks and adds food/sports datum whose name **contains one of the keywords**.
- If no datum is added after all the steps, replies user that search failed. Otherwise, shows user the data.

format: `check x/TYPE k/ONE_OR_MORE_KEYWORDS`

Examples:

- `check x/food k/apple` (searches for calorie intake of food whose name match/contain keyword "apple")



Matching reference data:
Name: Apple
Calorie: 95 cal per 1 apple (182 g)
Name: Applesauce
Calorie: 167 cal per 1 cup (246 g)
Note: at most 3 records are shown.

results of checking

1. Swimming Undone
2020-04-13-20:20 451
Sports Center

2. Jogging Undone
2020-03-01-20:20 451
Track

3. Apple juice Undone
2020-03-01-12:30 79
Utown 711

4. Lemon juice Undone
2020-03-01-11:30 79
Utown 711

5. test k/key Undone
2020-02-02-12:34 1
NUS

6. Walking Undone
2020-02-01-19:20 451
PGPR

Figure 2. Check command result (matching data found)

Given below are sections I reviewed and edited in the User Guide.

General rules about command format

- A prefix followed by a word in `UPPER_CASE` form a parameter to be supplied by the user.

Example: in `add n/NAME`, `n/NAME` is a parameter which can be `n/running`.

- Parameters in square brackets are optional e.g `n/NAME [r/REMARK]` means both `n/swimming r/fun` and `n/swimming` are both valid user inputs.
- Parameters not in square brackets are compulsory e.g. `x/TYPE k/KEYWORDS` means you must input both `TYPE` and `KEYWORDS` parameters.
- Items with `…` after them can be used multiple times including zero times e.g. `[r/remark]…` can be used as (i.e. 0 times), `r/really fun`, `r/really fun r/helps me lose weight` etc.
- Parameters can be in any order e.g. if a command format is `add n/NAME t/TIME`, then `add n/Bob t/2020-04-10-08:00` and `add t/2020-04-10-18:00 n/Bob` are both acceptable.
- The field `TYPE` in this document refers to type of entries. The only valid values are food OR sports, or in acronym f OR s.
- The field `TIME` needs to be in the format of yyyyy-mm-dd-hh:MM, e.g. `2020-04-10-03:00`
- The field `DATE` should be entered in the format of `yyyy-mm-dd`, e.g. `2020-02-02`.
- Our time parser will auto-correct some invalid date/time entered. More specifically, as long as the month value is valid and the day value is between 1 and 31, the parser will auto-correct an invalid date to the last valid date of the year-month specified by the user.

Example: `2020-02-31-12:34` will be auto-corrected as `2020-02-29-12:34` since `2020-02-29` is the last valid date of Feb 2020. `2020-22-31` and `2020-02-40` will not be auto-corrected since the month value is invalid or the day value is beyond 31.

- The field `INDEX` should be a positive integer representing a one-based index of an entry in an list.
- The field `DURATION` should be a positive double number, representing duration in hours. e.g. `dr/1.5` for 1.5 hours.
- You may input multiple parameters of a same prefix, but only the last input value will be read.

Example: `add n/Alice n/Bob n/Charlie` is equivalent to `add c/Charlie`.

- Please do not input parameters not required by a particular command.

Example: the format of sort command is `sort [x/TYPE] by/SORT_BY [o/ORDER]`, so you should not input other parameters like `n/NAME`, `k/KEYWORDS` or `v/VALUE`.

- The flag `-f` should be added **just after the command word**, e.g. `update -f`.

Contributions to the Developer Guide

Given below are sections I wrote in the Developer Guide.

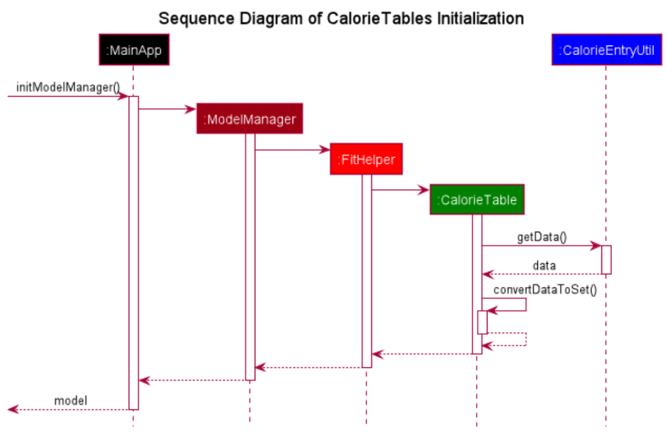
Check calorie intake/consumption of some common food/sports

Implementation

The check function is achieved by calling the FitHelper inside the ModelManager to search through either FoodCalorieTable or SportsCalorieTable for CalorieDatum that contain the keywords specified by the user.

Given below are example usage scenario:

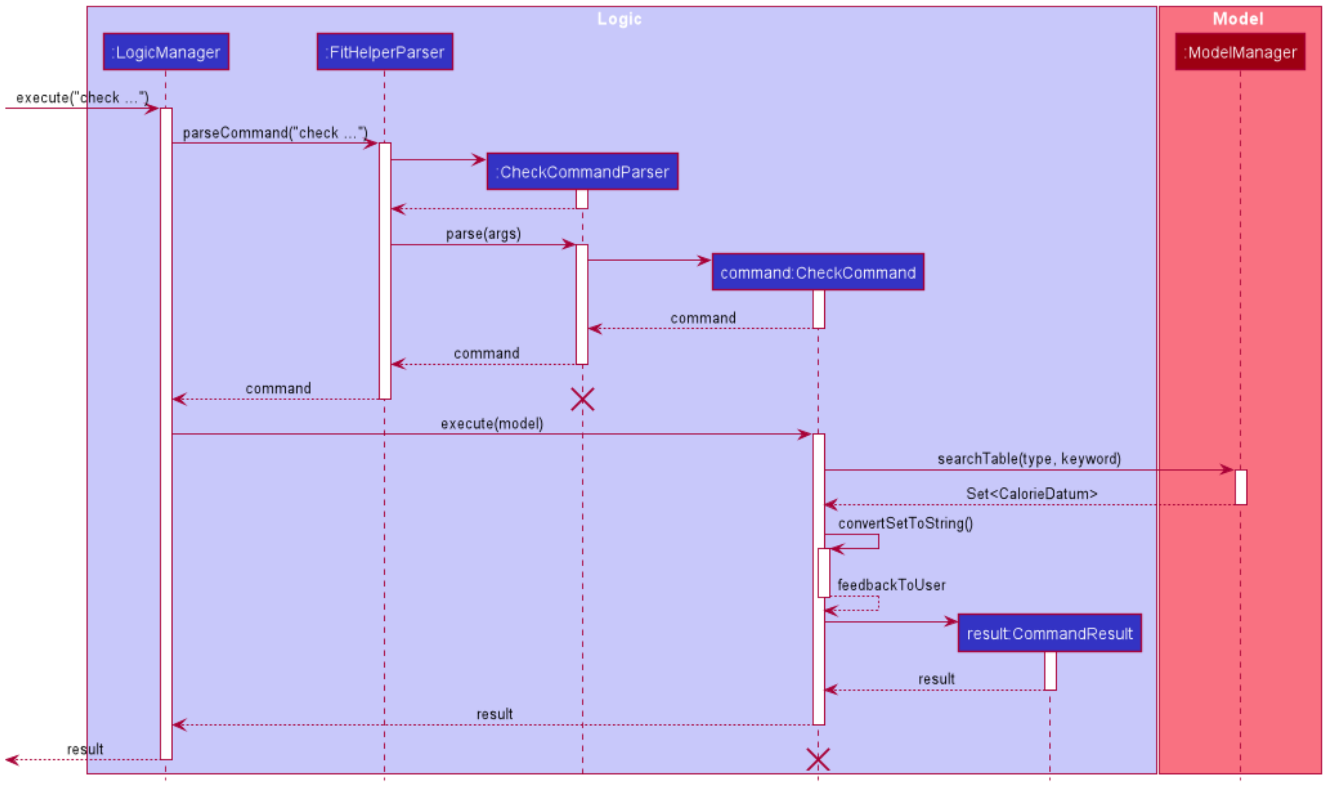
Initialization: when the application is launched, ModelManager will initialize a FitHelper, which will in turn initialize both FoodCalorieTable and SportsCalorieTable to contain pre-set data which is a LinkedHashSet of one type of CalorieDatum (either FoodCalorieDatum or SportsCalorieDatum).



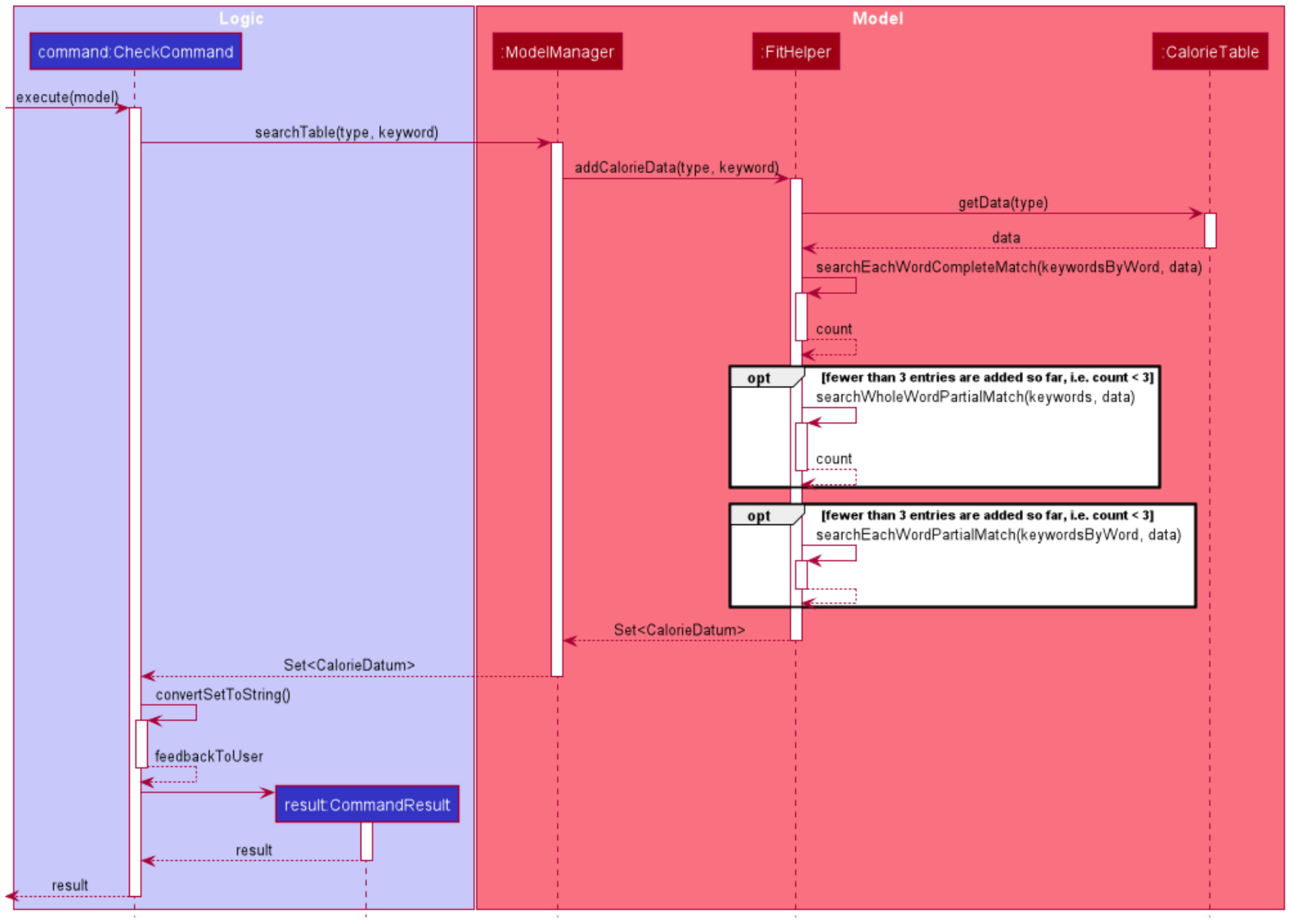
Case 1: when the user enter the command `check x/sports k/swim`, the LogicManager will create a CheckCommand, which asks ModelManager to let FitHelper to search through SportsCalorieTable to add first 3 CalorieDatum whose name matches the keyword swim into a set, and return the set to CheckCommand. Since the set contains at least one CalorieDatum (meaning there is some matching data), the CheckCommand returns a CommandResult whose feedbackToUser contains a success message followed by the string representation of each matching datum.

Case 2: when the user enter the command `check x/f k/swim`, the LogicManager will create a CheckCommand, which asks ModelManager to let ` FitHelper` to search through FoodCalorieTable to add first 3 CalorieDatum`s whose name contains the keyword `swim into a set, and return the set to CheckCommand. Since the set contains no CalorieDatum (meaning there is no matching data), the CheckCommand returns a CommandResult whose feedbackToUser contains a failure message followed by the string representation of the keyword.

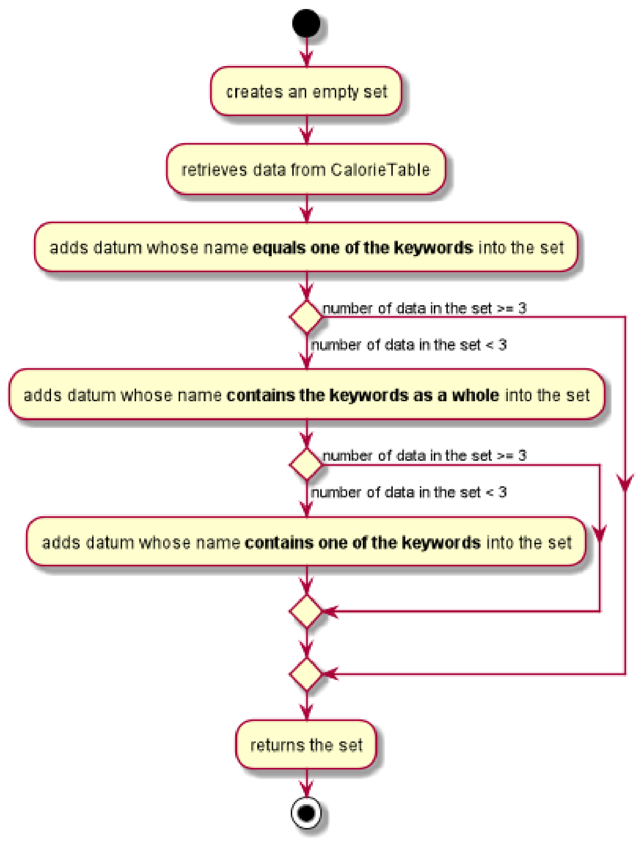
Sequence Diagram of Check Command (Logic part)



Sequence Diagram of Check Command (Model part)


Logic
Model
command:CheckCommand
:ModelManager
:FitHelper
:CalorieTable
execute(model)
searchTable(type, keyword)
addCalorieData(type, keyword)
getData(type)
data
searchEachWordCompleteMatch(keywordsByWord, data)
count
opt
[fewer than 3 entries are added so far, i.e. count < 3]
searchWholeWordPartialMatch(keywords, data)
count
opt
[fewer than 3 entries are added so far, i.e. count < 3]
searchEachWordPartialMatch(keywordsByWord, data)
Set<CalorieDatum>
Set<CalorieDatum>
convertSetToString()
feedbackToUser
result:CommandResult
result
result

The detailed searching mechanism is illustrated in the following activity diagram:


creates an empty set
retrieves data from CalorieTable
adds datum whose name equals one of the keywords into the set
number of data in the set >= 3
number of data in the set < 3
adds datum whose name contains the keywords as a whole into the set
number of data in the set >= 3
number of data in the set < 3
adds datum whose name contains one of the keywords into the set
returns the set

Design Considerations

Aspect: Data structure to store entries

- **Alternative 1 (current choice):** Use an `LinkedHashSet` as a field in CalorieTable to store the entries.
  - Pros: Easy to implement partial-key search (compare the keyword with the name of each entry in the set). Ensure that the database contains no duplicate data since a Set does allow duplicate elements.
  - Cons: O(n) complexity for finding matching entries, where n is the number of entries in the set.
- **Alternative 2:** Use a `HashMap` as a field in CalorieTable to store the entries. The key is the name of the entry and the value is the entry.
  - Pros: (theoretically) O(1) time complexity for finding an entry given a complete keyword, regardless of how many entries are in the HashMap.
  - Cons: hard to implement partial-key search (i.e. the keyword is only part of the name of the entry).

Given below are sections I reviewed and edited in the Developer Guide.

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	record my basic information such as name and gender	have a more complete profile
* * *	user who is concerned about body shape	record and update my current height and weight	have a clear view of my current body condition
* * *	user who wants to lose weight	set my target weight	have a clear target to work towards
* * *	user who wants to set diet plans	add a food entry	can plan my diet
* * *	user who wants to control calorie intake	view the calorie in each food entry	can keep track of my calorie intake
* * *	user who wants to set sports plans	add a sport entry	can plan for my sport exercises
* * *	user who wants to increase calorie consumption	view the calorie consumption for each sport entry	can keep track of my calorie consumption
* * *	user who wants to adjust my diet/sports plans	edit a food/sports entry	can have my plans and records updated
* * *	user who wants to remove my diet/sports plans	delete a food/sports entry	
* * *	user who wants to search for an entry	search by keywords in the entry name	can find related entries without having to scan through all the entries
* * *	user who wants see today’s plans	switch to Today Page and view the daily food/sports plans	can have a general idea of the daily diet/sports arrangements
* * *	user who needs some suggestions for my daily plan	switch to Today Page and view FitHelper feedback	I know whether my daily food/sports plan is suitable
* * *	user who wants to know my daily performance	switch to Today Page and view my performance report	I know my food calorie intake distribution and my task completion
* * *	user who types wrongly sometimes	undo my previous command	I do not need to delete explicitly using a long command
* * *	user who types wrongly sometimes	redo my previous undo command	I can re-executed a undone command
* * *	user who wants to keep a diary	add a diary log for a specific day	note down my schedules, feelings, goals and so on as a self-encouragement

Priority	As a ...	I want to ...	So that I can...
* * *	user who wants to append more content to a previous diary	append new content to existing diaries	enrich my previous diaries' content
* * *	user who wants to replace the content of a previous diary with new content	edit existing diaries	modify the content to an updated version
* * *	user who wants to remove some diary logs	delete existing diaries	keep abandon some diary logs that I do not want to keep
* * *	user who wants to clear my diary	clear all existing diaries	I can re-start my diary from a white paper
* *	user who wants keep fit	acknowledge my weight change trend according to time	keep track of my weight change easily
* *	user who wants to lose weight	compare between my current weight and target weight	know the gap clearly
* *	user	update my basic information such as address and name if necessary	have an updated profile at any time
* *	user	view pending tasks and status of daily calories goals in a calendar	have cleaner display of data
* *	user who wants to have a clean user interface	clear entries regularly	do not need to see irrelevant information
* *	user	leave the application when I need	It does not occupy additional space in my computer
* *	user	list all entries by certain criteria	I can filter the tasks by what I am looking for
* *	user	get reminders for tasks not done	I can focused on these tasks and complete them
* *	user who do not know very well about dieting and exercising	check calorie intake/consumption of common food and sports	I can input calorie intake/consumption without having to search about these information online.
* *	first-time user	view help page	I can know the functions of the application quickly