Lim Ting Wei - Project Portfolio

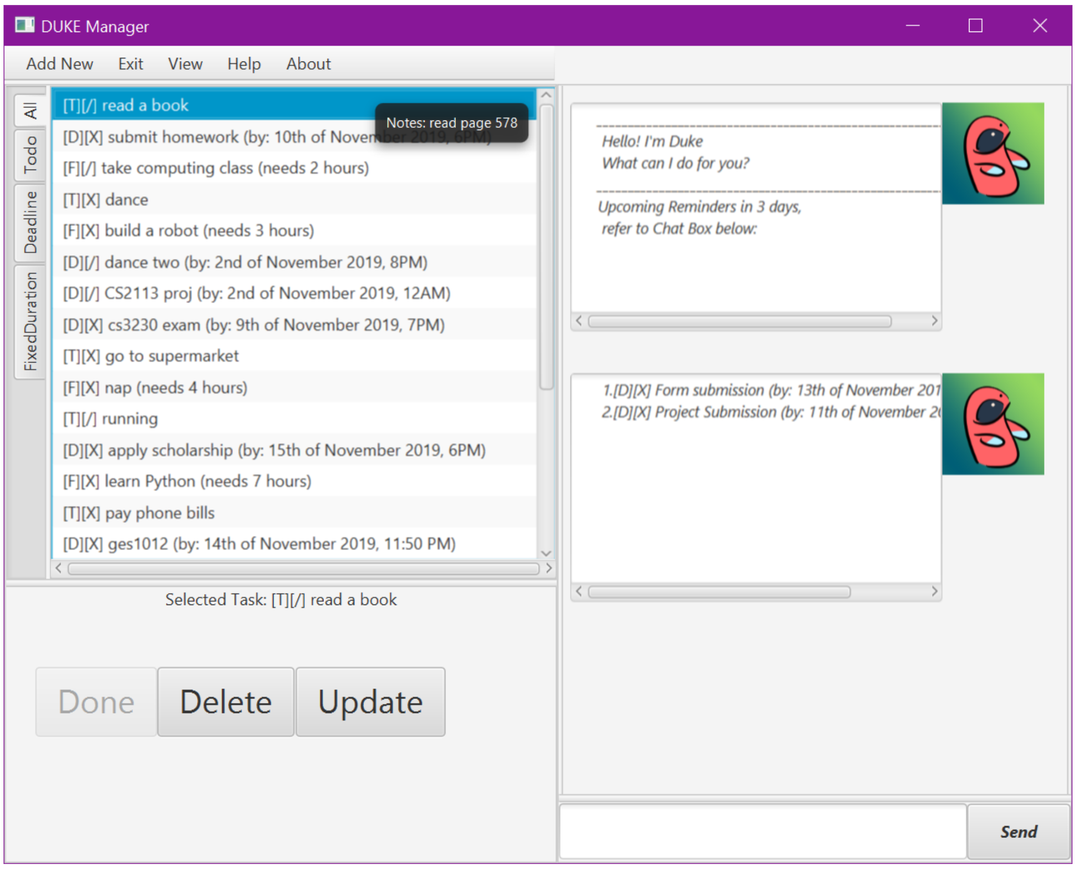
PROJECT: Duke Manager

Overview

DUKE Manager is an electronic handbook (desktop application) which is resulted from extending the project from DUKE in CS2113, using Java and Object-Oriented Programming. Our team of 4 students and I decided to create this desktop application to allow undergraduate engineering students to manage school-related tasks, store contact information of their peers and professors, and keep track of their own budget.

DUKE Manager has a user-friendly Graphical User Interface (GUI) which helps users to quickly familiarize and be comfortable for daily use. Also, it has an option for users who prefer Command Line Interface style by using the text box.

This is what our project looks like:



[T][/] read a book
Notes: read page 578
[D][X] submit homework (by: 10th of November 2019, 6PM)
[F][/] take computing class (needs 2 hours)
[T][X] dance
[F][X] build a robot (needs 3 hours)
[D][/] dance two (by: 2nd of November 2019, 8PM)
[D][/] CS2113 proj (by: 2nd of November 2019, 12AM)
[D][X] cs3230 exam (by: 9th of November 2019, 7PM)
[T][X] go to supermarket
[F][X] nap (needs 4 hours)
[T][/] running
[D][X] apply scholarship (by: 15th of November 2019, 6PM)
[F][X] learn Python (needs 7 hours)
[T][X] pay phone bills
[D][X] ges1012 (by: 14th of November 2019, 11:50 PM)

Selected Task: [T][/] read a book

Hello! I'm Duke
What can I do for you?

Upcoming Reminders in 3 days,
refer to Chat Box below:

1.[D][X] Form submission (by: 13th of November 201
2.[D][X] Project Submission (by: 11th of November 2

My role is to create a GUI to the DUKE Manager, as well as implementing some enhancements like Update, Filter, Detect Anomalies, and Notes to improve the efficiency for the user. The following sections will explain these enhancements in detail, as well as the documentation that I have contributed to the user and developer guides.

Note the following symbols and formatting used in this document:

ℹ	This symbol indicates important information.
update	A grey highlight (called a mark-up) indicates that this is a command that can be inputted into the command line and executed by the
UpdateCommand	Blue text indicates a component, class or object in the architecture of the application.

Summary of contributions

Enhancement added: I added the ability to update existing tasks.

- What it does: This enhancement allows Duke Manager to be able to update either task description, date and time, or task's type to existing tasks.
- Justification: This feature improves the efficiency of DUKE manager as the user may want to update an existing task instead of deleting and then adding a new task. This saves time for the user, so that the user can manage the list of tasks quickly and efficiently.
- Highlights: The update command can be used for both existing and new types of task. However, the update command is only implemented for majority of the tasks. Some parts of a certain type of task are too narrow for them to be included in update command. For example, the user will not be able to update the duration and unit of FixedDuration. To compensate that, it is set to 1 hour for the duration and unit by default.

Other Enhancements:

- **Notes**: added notes so that users will be able to add or update, delete, and show notes of a task in DUKE Manager.
- **Detect Anomalies**: added code to detect tasks that clash with the same date and time when adding a new task or updating an existing task in DUKE Manager.
- **Filter**: added filter so that the users can filter out tasks in the task list based on the type of task.

Code contributed: [RepoSense]

- **Update Related Code**: [Update Command]
- **Notes Related Code**: [Add Notes Command][Delete Notes Command][Show Notes Command]
- **Filter Related Code**: [Filter Command]

Other contributions:

- Project management:
  - I have managed 7 releases from mid-v1.1 - v1.4 on GitHub.
- Enhancements to existing features:
  - Designed and updated GUI from its tutorial JavaFx code (Pull requests #99, #120, #129)
  - Improve DetectDuplicate's code to work with every type of task (Pull requests #172)
  - Created an algorithm to enable jar to read and write text files, but was scrapped in later versions (Pull requests #124, #150, #211)
- Documentation:
  - Helped populating the user guide during its initial stage. (Pull requests #48, #85)
- Tools:
  - Integrated shadowJar and checkstyle into the project. (Pull requests #1, #36)
  - Integrated JUnit into the project. (#1)
  - Integrated Gradle and JavaFx into the project. (Pull requests #1, #1)

Contributions to the User Guide

The following is an excerpt from our DUKE Manager User Guide, showing additions that I have made for `update` and `notes`.

This section also contains an excerpt for history feature that I have planned for next version (v2.0) of DUKE Manager.

Updates an existing task in task list: `update`

Updates the task, either task description, date and time, or type of task in Duke Manager.

Example:
Let's say there is an update about a certain task, and you would like to change what was stored in the task without deleting and adding again.

To update:
1. Type `update <task number> <type of update> <description to be updated>` into the command box, and press Enter to execute it.
2. The chat bot will display the message "Nice! I've updated this task ^^" with the task that you have updated.
3. And you can see the updated task in the list.

- `update 1 /desc homework`
  Updates the 1st task description in the task list.

- `update 5 /date 17/09/2019 1222`
  Updates date and time of the 5th task in the task list.

- `update 2 /type deadline`
  Updates type of the 2nd task to Deadline in the task list.

Format: `update <task number> <type of update> <description to be updated>`
•Updates the task at the specified `<task number>`.
•The task number refers to the index number shown in the displayed task list.
•The index **must be in between 1 and the size of task list**. e.g. For a task list that contains 4 tasks, only numbers 1 to 4 are allowed.
•The type of update refers to either `/desc`, `/date`, or `/type`.
•`/desc` represents updating the task description.
•`/date` represents updating the date and time of the task.
•`/type` represents updating the type of task.
•The description to be updated refers to either description of task, date and time, or type of task depending on `<type of update>`.
•Date and time must be inputted in this format `<dd/MM/yyyy HHmm>`.
•Returns an error if a task does not contain date and time when the user tries to update date and time of the task.
•Returns an error if the task type is invalid.
•This `update` command does not work on notes, there is a separate command catered to notes only. However, notes will be lost when there is a change of type of task.

Notes in a task: `notes`

Notes can be either added or updated to an existing task in the task list. They can also be removed from an existing task. Notes of an existing task can be shown in the task list.

Example:

Let’s say you would like to add notes to the first task in the list to have a better idea on what to do and view it later. After finishing the task, the notes can be either be removed or updated.

To use notes:
1. Type `notes <task number> <type of notes> <(Optional) notes description>` into the command box, and press Enter to execute it.
2. The chat bot will display the message with the notes that you have queried.

- `notes 1 /add read page 578`
  Adds/Updates notes of the first task in the task list.

- `notes 2 /show`
  Shows notes of the second task to the user in the task list.

- `notes 3 /delete`
  Deletes notes of the third task in the task list.

Format: `notes <task number> <type of notes> <(Optional) notes description>`
- Adds/Updates/Deletes/Shows notes of the task at the specified <task number>.
- The task number refers to the index number shown in the displayed task list.
- The index **must be in between 1 and the size of task list**. e.g. For a task list that contains 4 tasks, only numbers 1 to 4 are allowed.
- The type of notes refers to either `/add`, `/delete`, or `/show`.
- `/add` represents adding/updating notes of the task.
- `/delete` represents removing existing notes from the task.
- `/show` represents showing the notes of the task.
- The notes description is **only required when adding or updating notes**, not when deleting or showing notes.

Storing a history of previous commands: `history` [coming in v2.0]

The user can retrieve a list of commands that were entered previously. These previous commands can also be reused again by pressing up or down arrow keys.

Example:

Let’s say you would like to re-enter a command you have executed moments ago but you have forgotten after executing many commands in a single session. You would be able to retrieve the command by either listing the whole history via executing history command or pressing up key for a few times to traverse through the previous commands until you have found the correct command.

Contributions to the Developer Guide

The following is an excerpt from our DUKE Manager Developer Guide, showing additions for update.

Update function

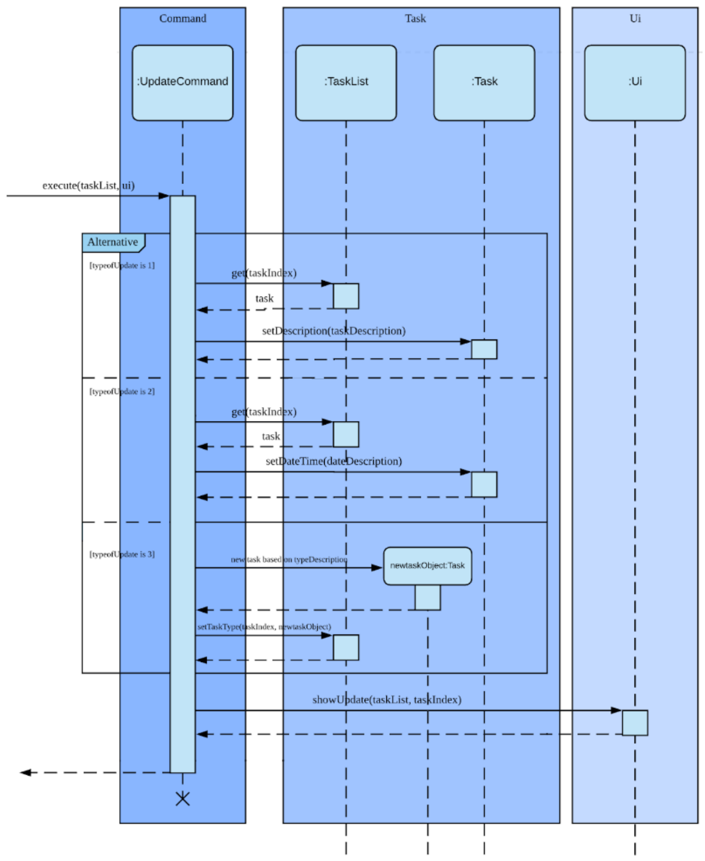
The update function will allow the user to update the existing tasks in the task list of Duke Manager. This will allow the user to update either task description, date and time of the task, or type of task.

Implementation

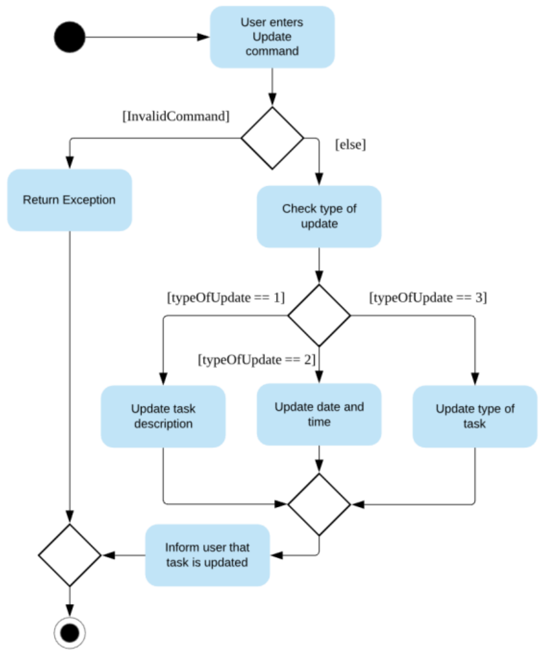
Upon entering the class UpdateCommand, the following steps will be done:

1. If the typeofUpdate equals to 1, this means that it is updating the task description. It proceeds to call get(taskIndex), where taskIndex refers to index of task, to retrieve the task to be updated. Next, it calls setDescription(taskDescription), where taskDescription refers to the new task description, to overwrite the description of the retrieved task.
2. If the typeofUpdate equals to 2, this means that it is updating the date and time of the task. It proceeds to call get(taskIndex), where taskIndex refers to index of task, to retrieve the task to be updated. Next, it calls setDateTime(dateDescription), where dateDescription refers to the new date and time in string format, to overwrite the date and time of the retrieved task.
3. If the typeofUpdate equals to 3, this means that it is updating the type of task. It proceeds to create a new task object based on the typeDescription, where typeDescription refers to the new type. Next, it calls setTaskType(taskIndex, newtaskObject), where taskIndex refers to index of task and newtaskObject refers to the task object that was created moments ago, to overwrite the type of the task.
4. After either Step 1, 2, or 3 is completed, it then calls showUpdate(taskList, taskIndex), where taskList refers to the task list and taskIndex refers to index of task, to tell the user that the update has been completed.

Below is the sequence diagram of updating task.



The following diagram shows the activity diagram of the process of calling the update command in DUKE manager.



Design Considerations

Aspect: Flexibility of update

- Alternative 1 (current choice): Updates either task description, date and time, or task type, one at a time.

    - Pros: Easy to implement.
    - Cons: Can be very inefficient if the user wants to execute two or more types of update of an existing task.

I decided to proceed with this option because I want the command to be as straightforward as possible. The user should not have to spend more time trying to figure out how it works. Furthermore, updating task one at a time reduces the confusion for the user, making it user-friendly.

- Alternative 2: Updates up to all types of update in one command.
    - Pros: Very efficient for the user when updating many types of update of an existing task.
    - Cons: Difficult to implement, requires complex algorithm.

History function [coming in v2.0]

The history function will allow the user to retrieve a list of commands that were entered previously. By pressing up or down arrow keys, the user can traverse through the previous commands. When history command is entered, it passes a string that contains "history" to HistoryCommand. When up or down arrow key is pressed instead, it passes a string that contains "up" or "down" to HistoryCommand.

Implementation

Upon entering the class HistoryCommand, the following steps will be done:

1. If the string contains "history", this means that it is calling the list of previous commands. It proceeds to call showHistoryList(), to display the list of previous commands to the user.
2. If the string contains "up", this means that it is traversing backwards, where the first command is the oldest, and the last command is the latest. It proceeds to deduct previousCommandIndex by one, where previousCommandIndex refers to the index of the list of previous commands.
3. If the string contains "down", this means that it is traversing forward, where the first command is the oldest, and the last command is the latest. It proceeds to add previousCommandIndex by one, where previousCommandIndex refers to the index of the list of previous commands.
4. After either Step 2 or 3 is executed, it proceeds to call showpreviousCommand(previousCommandIndex), to display the command to the user.

If the previousCommandIndex has reached the index of either oldest or latest command, it would not deduct to a negative number nor add to a number that exceeds the size of the history list.