

Laurent Huang (laurenthg)- Project Portfolio

Project: Le Duc

Overview

Leduc is a task management application geared to help high school students manage their academic and extracurricular workloads. The user enters commands via the command line and is able to perform a vast array of operations.

Summary of contributions

Features implemented:

Major enhancement: modify a task

¥ Several commands are implemented: `reschedule`, `postpone`, `snooze`, `edit` and `prioritize`.

! snooze and postpone a homework task:

- " This command is important as the deadline of a homework could have to change.
- " Verification of that the new date is after the old one is made.

! reschedule a event task:

- " This command is important as a event could be moved to another date, could be extended
- " Verification of that the new event period have not conflict with other tasks is made.

! Edit a task:

- " This command allows the user to edit the description or date/period of a task.
- " This is a primary command as a task description or date are likely changed if the user initially created a incorrect task.
- " This is managed either in a interactive way which the multi-steps command or in one shot command.
- " Verification of conflict dates for event task is made when the user wants to change the period of an event task.
- " With edit command, user is allowed to give a new date to a homework task which is earlier than the old one.

! Prioritize a task:

- " A priority field is added to each task. So, the user can now create a task with a priority of

his choice (from 1 to 9) or the priority 5 by default.

" This command allows to change the priority of a task and this is very important as the priority of a task generally increase with the time.

Minor enhancement: Sort the task list

¥ Implemented the second feature "sort the task list":

! This feature is very useful for the user as he could now see the task list either in the description, date, priority, type of task or either the task is done or not order.

Minor enhancement: Set the initial task priority when adding a task

¥ When adding a todo, homework or event task, the priority is set by default to 5.

¥ This feature allows the user to set the priority of the task when adding the task. This is very useful for the user as it allows to directly set the priority when adding the task instead of modify the default priority after have added the task.

Code contribution:

¥ [Dashboard: <https://nuscs2113-ay1920s1.github.io/dashboard/#=undefined&search=laurenthg>]

¥ [[PostponeCommand](#)][[SnoozeCommand](#)][[RescheduleCommand](#)][[EditCommand](#)][[PrioritizeCommand](#)][[SortCommand](#)][[TodoCommand](#) priority feature][[HomeworkCommand](#) priority feature][[EventCommand](#) priority feature]

Other contributions:

¥ Gradle

! Set up and managed gradle : [build.gradle](#)

" Gradle allows to manage the build for running, automate the Junit test and so improve the team productivity.

" It allows also to generated the jar file

¥ Manage releases

! Managed releases with setting for milestones in Github, creation of jar files for the different milestones thanks to gradle: [leduc-v1.1.jar](#), [leduc-v1.2.jar](#), [leduc-v1.2.1.jar](#) and [leduc-v1.3.jar](#).

! Make jar file work without initial data file : [Generate data file when executing jar file](#)

¥ Manage Github

! Set up the labels and milestone.

! Set up the issues corresponding to different user stories.

! Set up the pull request for requiring reviews É

¥ Write tests

! JUnit : [[PostponeCommandTest](#)][[SnoozeCommandTest](#)][[RescheduleCommandTest](#)][[EditCommandTest](#)][[PrioritizeCommandTest](#)][[SortCommandTest](#)][[TodoCommandTest](#)][[HomeworkCommandTest](#)][[EventCommandTest](#)]

[eworkCommandTest priority part](#)]

! Text-ui-test : [\[Edit text-ui-test for multi-steps part\]](#)

¥ Debugging after the PE

! Pull requests: [#140](#), [#141](#), [#142](#), [#143](#), [#157](#) and [#162](#)

¥ Community

! Reviews important pull request: [#21](#), [#75](#) É

¥ Documentation contribution

! Write the Readme file [README.adoc](#)

! Contribute to write the User Guide file [\[AY1920S1-CS2113-T16-1\]-\[Le Duc\]-UG.adoc](#)

! Contribute to write the Developer Guide [\[AY1920S1-CS2113-T16-1\]-\[Le Duc\]-DG.adoc](#)

Contributions to the User Guide

Introduction

For each command description bellow, examples are provided in the user guide. Please refer to the user guide to see the examples.

Snooze a homework task : [snooze](#)

To snooze a homework task: [snooze](#) [INDEX](#) The snooze time is fixed at 30 minutes.

Postpone a homework task : [postpone](#)

To postpone a homework task: [postpone](#) [INDEX](#) /by [DATE](#).

[DATE](#) is the new date of the homework task. The new date should be after the old one.

Reschedule an event task : [reschedule](#)

To reschedule an event task: [reschedule](#) [INDEX](#) /at [DATE](#) - [DATE](#).

Be careful : when rescheduling an event, two dates can't clash

Edit a task : [edit](#)

¥ Multi-steps command: to edit a task, follow these instructions:

1. [edit](#)
2. All of the tasks will be displayed, you have to choose a task [INDEX](#)
3. Depending on the type of task:
 - " If it is a todo task, you have to enter the new [DESCRIPTION](#)

" If it is not a todo task, you have to choose 1) if you want to edit the description or 2) if you want to edit the date

" Then, enter the new DESCRIPTION or the new DATE of the task

¥ For one shot command:

! edit the description: `edit INDEX description DESCRIPTION`

! edit the date of an homework task: `edit INDEX /by DATE`

! edit the period of an event task: `edit INDEX /at DATE - DATE`

Sort by: `sort`

Sort all task by date, description, priority, type of task or either it is done or not: `sort SORTTYPE`

SORTTYPE is either date, description, priority, type or done

Be careful:

¥ Sorting by date will sort tasks in chronological order

¥ Sorting by description will sort the descriptions in alphabetical order

¥ Sorting by priority will sort tasks in ascending urgency

¥ Sorting by type will sort tasks depending on its task type (event, homework, todo)

¥ Sorting by done will sort tasks depending on it the task is done or not

Prioritize: `prioritize`

Giving priority to task: `prioritize INDEX prio INDEX`

The first INDEX is the task index

The second INDEX is the priority (goes from 1 to 9)

Be careful:

¥ The second INDEX can't be less than 1 nor greater than 9.

¥ 1 is the less urgent, 9 is the most urgent

¥ When creating a task, specifying the priority is optional. When the priority is not specified, the task will automatically have a priority of 5.

Contributions to the Developer Guide

Contribution to 2.1. Class Diagram

The following class diagram represents in details the abstract class `Command` with all its inherited concrete class.

Modify a Task

Several commands allow the user to modify a task: `reschedule`, `postpone`, `snooze`, `edit` and `prioritize`. As every other command, these commands extend `Command`. As these commands relate to the modification of tasks, each command need to write into the data file after its execution.

Reschedule an event task

When rescheduling an event, two dates can not clash. This verification is done with the `verifyConflictDate` method which is in the `TaskList` class. Indeed, all task dates are needed to verify if there is a conflict. So, this allows to improve the cohesion.

Please refer to the Developer Guide to find the Sequence Diagram for the `reschedule` command.

Snooze an homework task

Snooze is applicable to a homework task. The snooze time is fixed at 30 minutes(it could be easily

changed in the `snoozeLocalDateTime()` method of `Date`.

Postpone an homework task

Postpone is also only applicable to a homework task. The new date should be after the old one. This is verified inside the execution of the `postponeCommand`.

Edit a task

¥ Multi-steps command: to edit a task, the user has to follow these instructions:

1. `edit`
2. All of the tasks will be displayed, you have to choose a task INDEX
3. Depending on the type of task:
 - " If it is a todo task, you have to enter the new DESCRIPTION
 - " If it is not a todo task, you have to choose 1) if you want to edit the description or 2) if you want to edit the date
 - " Then, enter the new DESCRIPTION or the new DATE of the task

The sequence diagram shows the interactions between different classes when the user wants to edit the description of an homework or event task with a multi-steps edit command.

Please refer to the Developer Guide to find the Sequence Diagram for the edit command in multi-steps.

¥ For one shot command:

- ! edit the description: `edit INDEX description DESCRIPTION`
- ! edit the date of an homework task: `edit INDEX /by DATE`
- ! edit the period of an event task: `edit INDEX /at DATE - DATE`

The sequence diagram shows the interactions between different classes when the user input `edit 2 description DESCRIPTION`.

Please refer to the Developer Guide to find the Sequence Diagram for the one shot edit command.

Prioritize a task

A task has initially a priority of 5. The priority of a task goes from 1 to 9. This command allows the user to change the priority of a task.

The sequence diagram show the interactions between different classes when the user wants to change to priority of the first task to 2.

Consideration

There are two different commands for modifying the priority (`prioritize`) and the description/date (`edit`) of a task. Indeed, the edit command is considered to be used when a user have initially created a incorrect task, whereas the prioritize command is supposed to be used regularly as the priority of a task generally increase with the time. However, these two commands are obviously easy to combine into one command.

Sort the task list

Sort all task by date/description/priority/type of task/ done or not: `sort SORTTYPE` SORTTYPE is either date, description, priority, type, done

- ¥ Sorting by date will sort tasks in chronological order
- ¥ Sorting by description will sort the descriptions in alphabetical order
- ¥ Sorting by priority will sort tasks in ascending urgency
- ¥ Sorting by type will sort tasks depending on its task type (event, homework, todo)
- ¥ Sorting by done will sort tasks depending on it the task is done or not.

To implement the sort command, the comparing static method of Comparator interface introduced in Java 8 is used. So, here the sort key are the description or the priority of the task.