Rahul Shah (rshah918) - Project Portfolio

Product Name: Le Duc

Overview

Le Duc is a schedule management software aiming to assist high school students manage their academic and extracurricular tasks, with the goal of increasing their time management/productivity. User-Software interaction occurs on the command line, from which they have access to a variety of commands/operations.

Contributions Summary

- **Major Enhancement**: **Added the ability to generate a reminder for upcoming tasks**
  - What it does: Analyzes the user's tasklist, and will generate a reminder that highlights the top 3 upcoming tasks that are not yet complete.
  - Justification: A user with a busy schedule could easily forget about a task that he/she must do. This feature will scan for upcoming tasks and remind the user of them, thus preventing the user from falling behind.
- **Major Enhancement**: **Created an Advanced Search Feature**
  - What it does: Uses an advanced character matching algorithm to search for tasks by descriptions, despite partial queries or query typos. The feature also includes many other commands that allow the user to search for tasks by other metrics, such as completion, a date window, etc.
  - Justification : The user may accidentally make a typo in their query, or make broad queries (search for tasks with only partial words). An advanced search feature allows for relevant results to be displayed despite imperfect queries.
  - Contributions: Search by day/week/month was implemented by Jérôme Huang
- **Major Enhancement**: **Generate Statistics from the Tasklist**
  - What it does: Will analyze the tasklist and display interesting metrics to the user, such as % of tasks completed, the count of each type of task, etc. User can view up to 23 detailed metrics, grouped by flags. There is one group of general statistics, one group of priority statistics, and one group of completion statistics.
  - Justification: This feature will help the user to make future decisions. For example, in a case where a large percentage of tasks are not complete, he/she might refrain from taking on an additional workload. If the user has a large number of high priority tasks, the user could make the choice to finish those first, instead of completing tasks in chronological order. This is a rather unintuitive scheduling optimization that can be made if the user was presented

with these priority statistics.

    - Highlights: This enhancement will allow for interesting insights to be generated from the users schedule, most of which are not necessarily intuitive to see without aggregating all historical tasks over time.

- **Major Enhancement**: **Set a Customized Welcome Message for Le Duc**
    - What it does: User can customize the message that gets displayed upon Le Duc startup. An internal refactoring of the UI class now allows for certain UI elements to be saved in a data file instead of being hard-coded, to allow for increased flexibility.
    - Justification: The user may want to customize his/her software to distinguish it from that of other Le Duc users.
- **Major Enhancement**: **Major Refactoring for New Task Types**
    - What it does: Modified most of the software to allow compatibility with the Homework task type. Please note (for RepoSense purposes) that this required line by line changes in almost every file that cannot be tagged in blocks. All former mentions of the Duke "Deadline" task was converted for the new "Homework" task type.
    - Justification: Le Duc is targeting High School students, thus a Homework task type is necessary for our users.
- **Minor enhancement**: Code refactoring: offloaded helper methods to higher layers to increase cohesion between the Command and Ui classes.
- **Code contributed**: [Project Code Dashboard][Remind Command] [Find Command] [UnfinishedCommand][Set Welcome Message] [Generate Statistics] [Create Homework Object][Filter and Sort Tasklist][Ui-English Refactoring][Ui-French Refactoring]
- **Other contributions**:
    - Project management:
        - Assisted with releases `v1.1` `v1.2` `v1.3` `v1.4`
        - Cohesion Improvement Refactoring (Portion of Pull Request #145)
        - Project wide Code Refactoring (Pull Request #96)
        - PR reviews: (Pull requests #24, #25, #76)
    - Documentation:
        - Contributed to the User Guide
        - Contributed to the Developer Guide

Contributions to the User Guide

Overview: I added user descriptions for my implemented features, and did general formatting/grammar adjustments document wide. Our User Guide is here

Search for a task by relevancy : `find`

To find a task by character relevancy : `find QUERY`

The find command allows the user to search for tasks via character matching (NOT keyword matching). The command will return the top 5 tasks ordered by ascending relevancy.

Typos in the user query will not affect performance.

Examples:

Partial word query

```
find scie

____________________________________________________________________________
         Here are the most relevant tasks in your list:
         7. [H][V] science by: 05/05/2005 05:05 [Priority: 2]


____________________________________________________________________________
```

Typo in Query

```
find homewqeuktest

____________________________________________________________________________
         Here are the most relevant tasks in your list:
         6. [H][V] homeworktest by: 04/04/2004 04:04 [Priority: 5]


____________________________________________________________________________
```

If all tasks have <50% matching characters with the query

```


____________________________________________________________________________
         There is no matching tasks in your list

____________________________________________________________________________
```

Customize the welcome message : `setwelcome`

To customize the welcome message: `setwelcome WELCOME`

Example:

- Original welcome message:

```
____        _
|  _ \ _   _| | _____
| | | | | | | |/ / _ \
| |_| | |_| |   <  __/
|____/ \__,_|_|\_\___|


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

--------------------------------------------------------------------------------
```

- setwelcome `hello World`

```
setwelcome Hello World

--------------------------------------------------------------------------------
         The welcome message is edited: Hello World

--------------------------------------------------------------------------------
```

- New welcome message:

```
 ____        _
|  _ \ _   _| | _____
| | | | | | | |/ / _ \
| |_| | |_| |   <  __/
|____/ \__,_|_|\_\___|


--------------------------------------------------------------------------------
        Hello World

--------------------------------------------------------------------------------
```

Be careful:

*Reverting to the previous welcome message is not possible once a new message is set. *Ensure the folder "data" is in the correct location. The welcome message is stored in this folder.

Display Statistics: `stats`

Display statistics : `stats`

Display useful statistics about your tasklist.

Enter command `stats` to view general statistics, `stats -p` to view detailed priority statistics, or `stats -c` to view detailed completion statistics.

Example:

- `stats`

General Statistics Example:

```
stats

--------------------------------------------------------------------------------
Here are some general statistics about your task list:
Number of tasks: 8.0
Number of Todo's : 3
Number of Events: 1
Number of Homeworks: 4
Number of Uncompleted Tasks: 5
Number of Completed Tasks: 3
Percent Complete: 37.5%

--------------------------------------------------------------------------------
```

- `stats -c`

Completion Statistics Example

```
stats -c

--------------------------------------------------------------------------------
Here are some completion statistics about your task list:
----COMPLETION COUNTS----
Number of incomplete Homeworks remaining: 2
Number of incomplete Todos remaining: 2
Number of incomplete Events  remaining: 1
----COMPLETION PERCENTAGES----
Percent of incomplete Homework: 50.0%
Percent of incomplete Todo: 66.66666666666666%
Percent of incomplete Events: 100.0%

--------------------------------------------------------------------------------
```

- `stats -p`

Priority Statistics Example:

```
stats -p

--------------------------------------------------------------------------------
Here are some priority statistics about your task list:
----PRIORITY COUNTS----
Number of tasks with priority 9: 0
Number of tasks with priority 8: 0
Number of tasks with priority 7: 0
Number of tasks with priority 6: 0
Number of tasks with priority 5: 12
Number of tasks with priority 4: 0
Number of tasks with priority 3: 0
Number of tasks with priority 2: 1
Number of tasks with priority 1: 0
----PRIORITY PERCENTAGES----
Percent of tasks with priority 9: 0.0%
Percent of tasks with priority 8: 0.0%
Percent of tasks with priority 7: 0.0%
Percent of tasks with priority 6: 0.0%
Percent of tasks with priority 5: 92.3076923076923%
Percent of tasks with priority 4: 0.0%
Percent of tasks with priority 3: 0.0%
Percent of tasks with priority 2: 7.6923076923076925%
Percent of tasks with priority 1: 0.0%

--------------------------------------------------------------------------------
```

View Unfinished tasks : `unfinished`

Find and display all unfinished tasks : `unfinished`

Example:

- `unfinished`
- Output:

```
 unfinished

--------------------------------------------------------------------------------
         Here are the unfinished tasks in your list:
         1. [T][X] td1 [Priority: 5]
         2. [E][X] e at: 21/09/2019 00:00 - 28/10/2019 22:22 [Priority: 5]
         3. [H][X] math by: 11/11/2011 01:01 [Priority: 5]

--------------------------------------------------------------------------------
```

- **Remind section omitted. Please see the user guide**

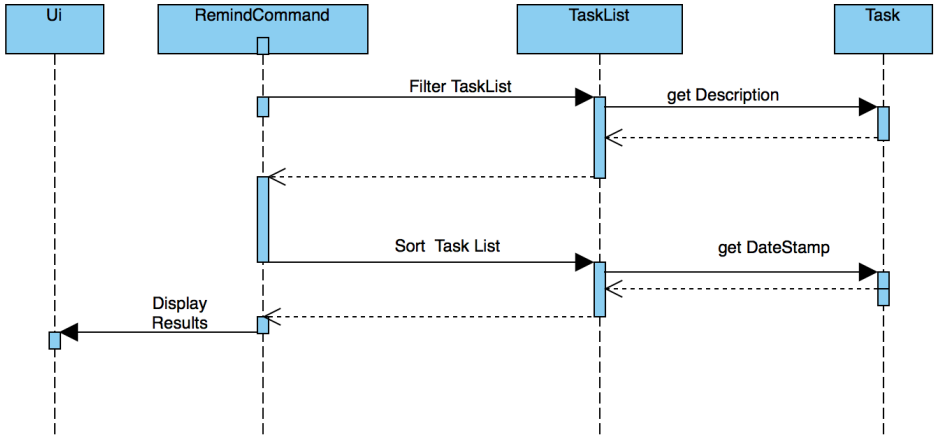
Contributions to the Developer Guide

Overview: I have added the implementations of my features to the Developer Guide : remind, statistics, and find. I also created Appendix Sections A-F and G1-G11. Our Developer Guide is here.

Remind

The Remind feature is done by the RemindCommand. Along with all of the other implemented commands, it extends Command. The feature will process each tasks date/timestamps to order them, and then remind the user of the top 3 upcoming tasks. The following methods were implemented in this feature:

- `filterTasks` - Extracts the Homework and Event tasks into a seperate ArrayList
- `sort` - Orders the filtered TaskList in chronological order.
- **Sequence Diagram of the Remind Feature:**



There are 4 cases:

- TaskList contains only Homework and Event objects
- TaskList contains only Todo objects
- TaskList contains a mix of all objects
- TaskList contains no objects

The task of sorting the tasklist in chronological order becomes challenging as not all tasks have associated timestamps. This problem is divded into 4 subproblems, each of which are handled seperately.

TaskList contains only Homework/Event tasks

- The original TaskList is passed through a filter to isolate the Homework and event tasks.
- The TaskList.extractTodo() method will attempt to isolate all todo tasks. It will return an empty array in this case
- After filtering by TaskList.filterTasks(), the filtered TaskList will be equal to the original

TaskList, as there are no Todo objects to filter out in this particular case.

- The filtered TaskList will then be sorted by TaskList.sort().
- The sort() method will call each tasks task.getDate() method in preparation for sorting. For "Event" tasks, the first timestamp will be used for the purposes of sorting.
- After performing insertion sort, the array of Todo's will be appended to the end of the sorted list. In this case, there are no "todo" tasks, so nothing will be appended.
- The first 3 most upcoming tasks will be displayed to the user.
- **Output:**

```
remind
1. [H][X] d1 by: 14/09/2019 22:33 [Priority: 5]
2. [E][X] e1 at: 21/09/2019 00:00 - 28/10/2019 22:22 [Priority: 5]
3. [H][X] d2 by: 22/09/2019 22:33 [Priority: 5]
```

TaskList only contains Todo tasks

- The original TaskList is passed through a filter to isolate the Homework and Event tasks. Because there are only Todo tasks, it will return an empty array.
- The TaskList.extractTodo() method will attempt to isolate all todo tasks. It will return the original tasklist in this case, becase every task is a "todo"
- The filtered TaskList will then be sorted by TaskList.sort(). The method will return an empty array, because the input array containing Homework and Event tasks is empty.
- The sort method will return an empty array, as the input array of Homework/Event tasks is empty in this case.
- The array of Todo's is appended to the empty, returned array from TaskList.sort(), resulting in the original tasklist of only Todo tasks.
- **The original tasklist is considered to be the sorted tasklist. The order in which the Todo's were created are treated as an "implicit order".**
- The first 3 most upcoming tasks will be displayed to the user.
- **Output:**

```
remind
1. [T][X] todo1 [Priority: 5]
2. [T][X] todo2 [Priority: 5]
3. [T][X] todo3 [Priority: 5]
```

Last 3 sections are omitted. Please see Developer Guide