EVELYN CHEN – PROJECT PORTFOLIO FOR MODPLAN

ABOUT THE PROJECT

My team of 4 computer engineering students and I were tasked with enhancing a basic command line interface desktop personal assistant application for our Software Engineering project. We chose to morph it into a module planning system - ModPlan. This enhanced application expands on NUSMODS to help NUS CEG students to plan their modules for all four years of their bachelor's degree. Other functionalities include calculating cap, generating module reports, sorting modules and ccas.

This is what our project looks like.

Figure 1. The command-line interface of ModPlan

My role was to design and write the codes for the sort feature. The following sections illustrate these enhancements in more detail, as well as the relevant documentation I have added to the user and developer guides in relation to these enhancements.

SUMMARY OF CONTRIBUTIONS

This section shows a summary of my coding and documentation.

Enhancement added:

I added the ability to sort the modules and/or ccas in a way which the users desires

- What it does: The sort command can separately sort the modules and/or ccas in ascending or descending order.
- Justification: In the event that users want to see all their added modules/ccas in a sorted order, whether it's according to the module's code, grade, level or mcs, the sort command gives them this option.
- Highlights: In order for the users to sort together the modules and ccas for a timetable, I've allowed the users to add modules with optional weekly time periods.

Code contributed:

Please click on these links to see a sample of my code: [Functional code] [Test Code]

CONTRIBUTIONS TO THE USER GUIDE

We had to update the original duke User Guide with instructions for the enhancements that we had added. The following is an excerpt from our ***ModPlan User Guide***, showing additions that I have made for the sort feature.

3.6. Sorting and printing the modules and/or ccas : sort

Sorts out modules and/or ccas accordingly. For all the sorting methods listed below, enter the optional flag --r to sort in the reverse order.

Example:
sort cca --r
sort module code --r

3.6.1. Sorting and printing the CCAs : sort cca

Sorts the cca list according to alphabetical order and prints the cca list.
Format: sort cca

3.6.2. Sorting and printing the CCAs and modules of a certain day of the week : sort time

Sorts the cca and modules together list according to alphabetical order and prints the cca list.
Format: sort time DAY_OF_WEEK , replace DAY_OF_WEEK by any
of monday tuesday wednesday thursday friday saturday and sunday

Example:
sort time monday

3.6.3. Sorting and printing the modules : sort module code

Sorts the module list according to alphabetical order and prints the module list.
Format: sort module code

3.6.4. Sorting and printing the modules : sort module grade

Sorts the module by the grade entered and prints the module list.
Format: sort module grade

3.6.5. Sorting and printing the modules : sort module level

Sorts the module list by the numerical order and prints the module list.
Format: sort module level

3.6.6. Sorting and printing the modules : sort module mc

Sorts the module list according to the number of mcs and prints the module list.
Format: sort module mc

3.1.2. Searching and adding a module with time period of the week: add module

Searches for a module and adds it to the module list, if the module is present in the list of NUS modules.
Format: `add module MOD_CODE --begin BEGIN_TIME --end END_TIME --dayOfWeek DAY_OF_WEEK`

- The `MOD_CODE` must be in the format of the NUS module code name.
- The `BEGIN_TIME` and `END_TIME` must be in the format of HHmm.
- The `DAY_OF_WEEK` must be in the format of the day names such as, "Monday", "Tuesday", etc.
- Example: `add module CG2028 --begin 14:00 --end 17:00 --dayOfWeek Tuesday`

The following section shows my additions to our *ModPlanner User Guide* for the sort feature.

4.4. SortCommand

4.4.1. Current implementation

The sort feature is operated by the SortCommand class, which is called by the Parser class. Upon user input of sort TOSORT TYPE, the Parser will return a new SortCommand.

Since SortCommand inherits the ModuleCommand class, it must override the execute method to specially execute the SortCommand. From the Parser, SortCommand also receives two additional variable inputs from the user:

- The object to sort, e.g. module, cca or time.
- The type of sorting the user specifies. e.g. for modules they can sort by code, grade, level and mcs; and for time they must specify a day of the week.

The parameter TYPE can take three main forms according to the user input:

- sort cca Where the user receives a sorted list of their ccas in alphabetical order.
- sort time DAY_OF_WEEK Where the user receives a list of ccas and/or modules in a timely order for the day of week they choose.
- sort module BY Where the user receives a sorted list of their modules in the order they specify.

These TYPE parameters will be parsed by the Parser class and pass the corresponding argument of toSort into the SortCommand class. A switch case statement will handle the toSort argument, and choose to execute the corresponding sorting.

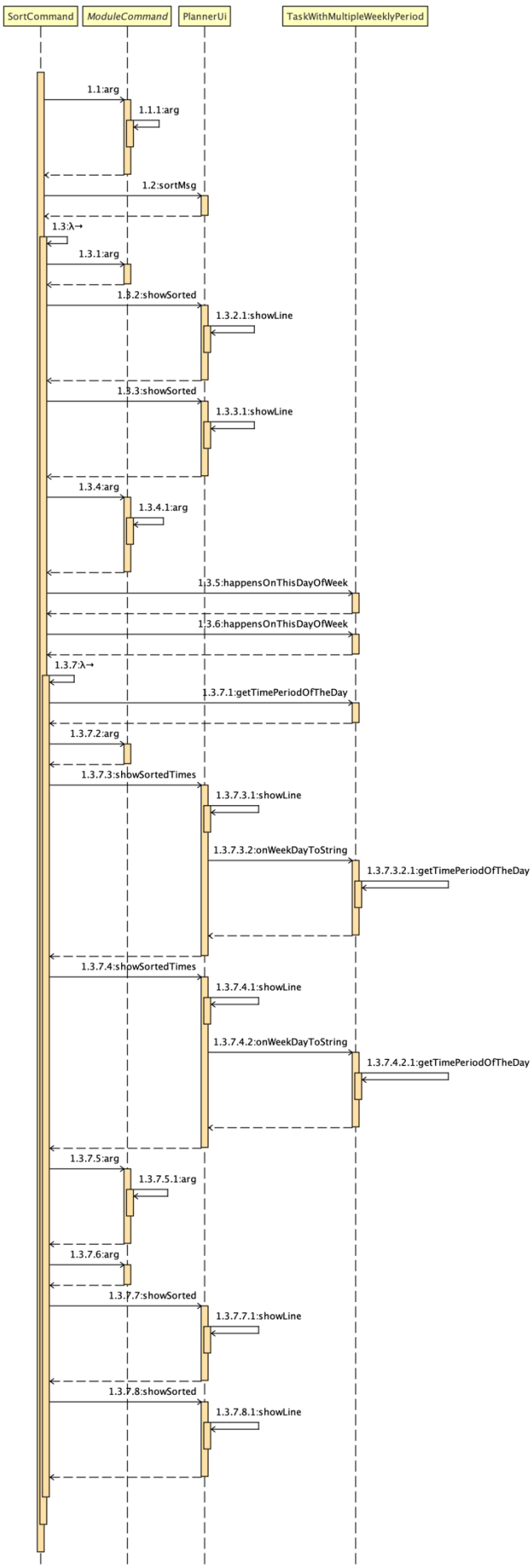
As stated above, there are three forms that can be executed depending upon the TYPE the user inputs.

- Case 1: sort cca
  If the argument read for toSort is “cca”, the cca list will be sorted in alphabetical order, and printed to the users.
- Case 2: sort time DAY_OF_WEEK
  If the argument read for toSort is “time”, a new list of TaskWithMultipleWeeklyPeriod will be initialized, and all the ccas and modules that happens on the specified DAY_OF_WEEK will be added to the new list. This list is then sorted in ascending order of the starting time of the ccas/modules, and this list is printed to the users.
    - Note it will only take into account modules that was initially given a time period of the week attached to its details.
- Case 3: sort module BY
  If the argument read for toSort is “module”, another switch case statement will handle the BY argument, and choose to execute the corresponding sorting of the modules. The options for BY are code, grade, level and mc.

For all the above forms, the users may choose to enter an optional flag of `--r` to have all their modules/ccas sorted in the reverse order. This is done by the parser setting the argument as true (default is set as false). Then if the argument is true, the previously sorted list will be reversed then printed to the users.

Below is a Sequence Diagram showing how SortCommand works. SortCommand inherits the attributes and methods from the ModuleCommand, which is shown by the 1.1 arg arrow connecting SortCommand and ModuleCommand. The ModuleCommand calls itself as it uses its own attributes method as shown by 1.1.1 args. The SortCommand also calls certain methods of PlannerUi as shown by the arrows from 1.2 and 1.3 args. This is because the SortCommand uses the methods in PlannerUi such as the sortMsg, showSorted and showSortedTimes.

As both the modules and ccas inherit from the class TaskWithMultipleWeeklyPeriod, when the users indicate they want to sort time DAY_OF_WEEK, the SortCommand calls the happensOnThisDayOfWeek method to check whether the module or cca in the list happens on the DAY_OF_WEEK specified by the user, then adds to a temporary list. The method getTimePeriodOfTheDay from TaskWithMultipleWeeklyPeriod is used to sort the modules and ccas in a timely order.


SortCommand
ModuleCommand
PlannerUi
TaskWithMultipleWeeklyPeriod
1.1:arg
1.1.1:arg
1.2:sortMsg
1.3:λ→
1.3.1:arg
1.3.2:showSorted
1.3.2.1:showLine
1.3.3:showSorted
1.3.3.1:showLine
1.3.4:arg
1.3.4.1:arg
1.3.5:happensOnThisDayOfWeek
1.3.6:happensOnThisDayOfWeek
1.3.7:λ→
1.3.7.1:getTimePeriodOfTheDay
1.3.7.2:arg
1.3.7.3:showSortedTimes
1.3.7.3.1:showLine
1.3.7.3.2:onWeekDayToString
1.3.7.3.2.1:getTimePeriodOfTheDay
1.3.7.4:showSortedTimes
1.3.7.4.1:showLine
1.3.7.4.2:onWeekDayToString
1.3.7.4.2.1:getTimePeriodOfTheDay
1.3.7.5:arg
1.3.7.5.1:arg
1.3.7.6:arg
1.3.7.7:showSorted
1.3.7.7.1:showLine
1.3.7.8:showSorted
1.3.7.8.1:showLine