

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.

```
C:\Users\andre\IdeaProjects\main\build\libs>java -jar Planner-v1.4.0.jar

Welcome to ModPlanner, your one stop solution to module planning!
Begin typing to get started!

show module
All modules in the list!
1. [not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade: | 09:00 - 12:00 on MONDAY
2. [not taken] CS1010 | ModuleCode:CS1010, MC:4.0, SU: can S/U, grade:
3. [not taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:
4. [not taken] MA1511 | ModuleCode:MA1511, MC:2.0, SU: can S/U, grade:
5. [not taken] MA1512 | ModuleCode:MA1512, MC:2.0, SU: can S/U, grade:
6. [taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:S
7. [not taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade: | 14:00 - 17:00 on TUESDAY
8. [not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
9. [taken] CS2040C | ModuleCode:CS2040C, MC:4.0, SU: cannot S/U, grade:A-
10. [taken] EE2026 | ModuleCode:EE2026, MC:4.0, SU: cannot S/U, grade:B+
11. [not taken] GEQ1000 | ModuleCode:GEQ1000, MC:4.0, SU: cannot S/U, grade:
12. [taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B
13. [taken] CS2113T | ModuleCode:CS2113T, MC:4.0, SU: cannot S/U, grade:B+
14. [taken] CG2027 | ModuleCode:CG2027, MC:2.0, SU: cannot S/U, grade:B+
15. [not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on WEDNESDAY
16. [not taken] UTC2701 | ModuleCode:UTC2701, MC:4.0, SU: can S/U, grade:
17. [not taken] UTS2707 | ModuleCode:UTS2707, MC:4.0, SU: can S/U, grade:
18. [taken] CG2023 | ModuleCode:CG2023, MC:4.0, SU: cannot S/U, grade:A
19. [taken] ST2334 | ModuleCode:ST2334, MC:4.0, SU: cannot S/U, grade:A-
20. [not taken] CG2271 | ModuleCode:CG2271, MC:4.0, SU: cannot S/U, grade: | 10:00 - 12:00 on THURSDAY

show cca
All ccas in the list!
1. [C] NUSSU | 18:00 - 21:00 on MONDAY
2. [C] Volleyball | 21:00 - 23:59 on MONDAY, 21:00 - 23:59 on WEDNESDAY
3. [C] Tchoukball | 22:00 - 00:00 on THURSDAY
4. [C] Dodgeball | 22:00 - 00:00 on TUESDAY
5. [C] Basketball | 18:00 - 20:00 on FRIDAY
6. [C] NUS Investment Club | 18:00 - 19:30 on WEDNESDAY
```

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.

3.7. 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
```

3.7.1. Sorting and printing the CCAs : `sort cca`

Sorts the cca list according to alphabetical order and prints the cca list.

Format: `sort cca`

```
sort cca
Here are your sorted cca:

[C] BASKETBALL | 14:00 - 20:00 on FRIDAY
[C] NUSSU | 10:00 - 12:00 on TUESDAY
[C] SOCCER | 16:00 - 18:00 on MONDAY
[C] Ultimate Frisbee | 22:00 - 00:00 on THURSDAY
```

3.7.2. 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`.

- Note that if two ccas have the same name, only the first one will be considered for sorting.

Example:

```
sort time Tuesday
```

```
sort time tuesday
Here are your sorted time:

NUSSU[10:00 - 12:00 on TUESDAY]
CG2028[14:00 - 17:00 on TUESDAY]
```

3.7.3. 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`

```
sort module code
Here are your sorted module:

[taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:A-
[not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:B+ | 09:00 - 12:00 on MONDAY
[taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:A-
[not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
[taken] CS1010 | ModuleCode:CS1010, MC:4.0, SU: can S/U, grade:A-
[taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:S
[taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
[taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:B
[taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:B+
[not taken] EE2026 | ModuleCode:EE2026, MC:4.0, SU: cannot S/U, grade: | 14:00 - 17:00 on WEDNESDAY
[not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
[not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:
[taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:CS
[not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
[not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
```

3.7.4. Sorting and printing the modules : sort module grade

Sorts the module by the grade entered and prints the module list.

Format: sort module grade

```
sort module grade
Here are your sorted module:

[taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:A-
[taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:A-
[taken] CS1010 | ModuleCode:CS1010, MC:4.0, SU: can S/U, grade:A-
[not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:B+ | 09:00 - 12:00 on MONDAY
[taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
[taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:B+
[taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:B
[taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:S
[taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:S
[not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
[not taken] EE2026 | ModuleCode:EE2026, MC:4.0, SU: cannot S/U, grade: | 14:00 - 17:00 on WEDNESDAY
[not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
[not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:
[not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
[not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
```

3.7.5. 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

```
sort module level
Here are your sorted module:

[taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:S
[taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:A-
[taken] CS1010 | ModuleCode:CS1010, MC:4.0, SU: can S/U, grade:A-
[not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
[not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:B+ | 09:00 - 12:00 on MONDAY
[taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:A-
[taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:S
[taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:B+
[not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
[not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
[not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:
[not taken] EE2026 | ModuleCode:EE2026, MC:4.0, SU: cannot S/U, grade: | 14:00 - 17:00 on WEDNESDAY
[not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
[taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
[taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:B
```

3.7.6. 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`

```
sort module mc
Here are your sorted module:

[not taken] MA1513 | ModuleCode:MA1513, MC:2.0, SU: can S/U, grade:
[not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
[taken] GER1000 | ModuleCode:GER1000, MC:4.0, SU: can S/U, grade:S
[taken] ACC1002 | ModuleCode:ACC1002, MC:4.0, SU: can S/U, grade:A-
[taken] CS1010 | ModuleCode:CS1010, MC:4.0, SU: can S/U, grade:A-
[not taken] GEH1056 | ModuleCode:GEH1056, MC:4.0, SU: can S/U, grade:
[taken] CS1231 | ModuleCode:CS1231, MC:4.0, SU: can S/U, grade:S
[taken] EC1301 | ModuleCode:EC1301, MC:4.0, SU: can S/U, grade:B+
[not taken] MA1508E | ModuleCode:MA1508E, MC:4.0, SU: can S/U, grade:
[not taken] GEK2020 | ModuleCode:GEK2020, MC:4.0, SU: can S/U, grade:
[not taken] EE2026 | ModuleCode:EE2026, MC:4.0, SU: cannot S/U, grade: | 14:00 - 17:00 on WEDNESDAY
[taken] CS2101 | ModuleCode:CS2101, MC:4.0, SU: can S/U, grade:B+
[taken] CS3240 | ModuleCode:CS3240, MC:4.0, SU: cannot S/U, grade:B
[not taken] CG1111 | ModuleCode:CG1111, MC:6.0, SU: can S/U, grade:B+ | 09:00 - 12:00 on MONDAY
[taken] CG1112 | ModuleCode:CG1112, MC:6.0, SU: can S/U, grade:A-
```

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`

```
add module CG2028 --begin 14:00 --end 17:00 --dayOfWeek Tuesday
Got it, added the follow module!
[not taken] CG2028 | ModuleCode:CG2028, MC:2.0, SU: cannot S/U, grade: | 14:00 - 17:00 on TUESDAY
```

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

4.4. Sorting

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.

