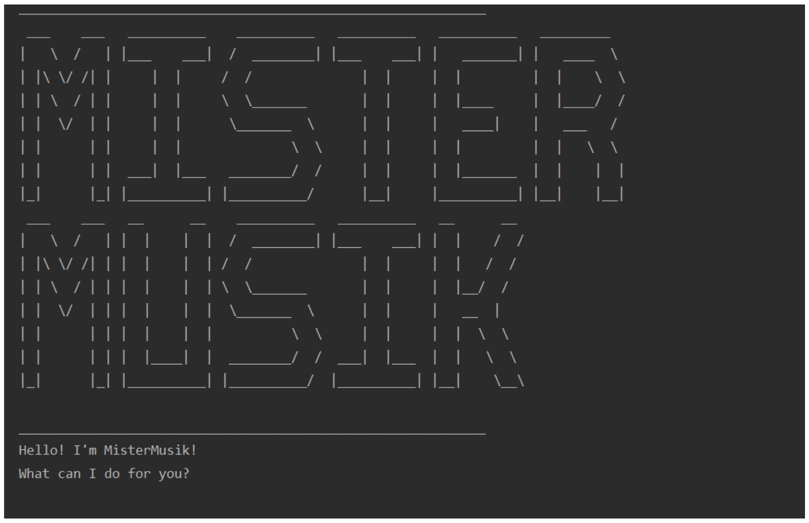
Zhang Yihan – Project Portfolio for MisterMusik

About the Project

My team of 5 Computer Engineering students were required to develop and enhance a command line interface desktop scheduler software for Software Engineering and OOP module project. We decided to further develop it into a powerful scheduling system for Music students of National University of Singapore (NUS) named **MisterMusik**. This enhanced scheduler software aims to automate and streamline the scheduling and organizing of various events, such as exams, lessons, practice sessions, etc., and help NUS music students to pursue a musical career.

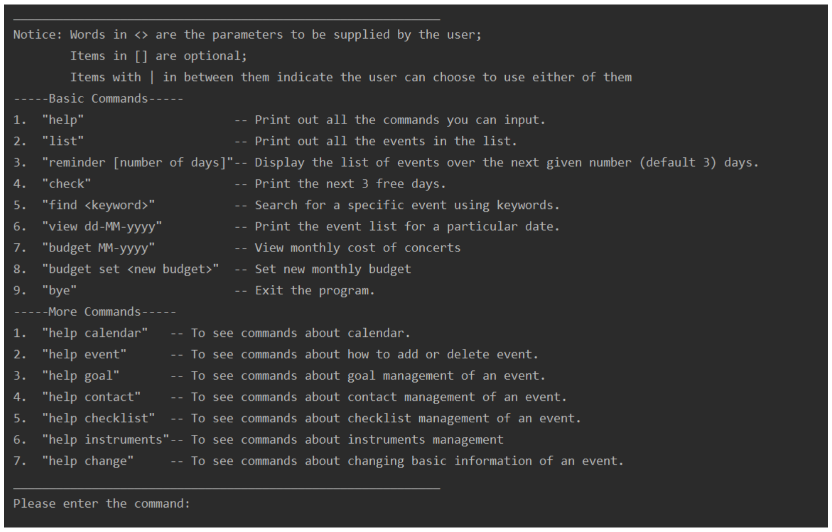
This is what our project looks like:



```
____________________________________________________________
 ___  ___  _________  ________  _________  ________  ________
|   \/   ||___   ___|/  ______||___   ___||   _____||   ___  \
| |\  /| |    | |    / /           | |    | |       | |    \ \
| | \/ | |    | |    \ \_______    | |    | |___    | |___/  /
| |    | |    | |     \_______ \   | |    |  ___|   |  ___  /
| |    | |    | |             \ \  | |    | |       | |   \ \
| |    | | ___| |___  ________/ /  | |    | |______ | |    | |
|_|    |_||_________||_________/   |_|    |________||_|    |_|
 ___  ___  __    __  _________  ________  __    __
|   \/   || |   | ||  _______||___   ___|| |   / /
| |\  /| || |   | || |            | |    | |  / /
| | \/ | || |   | || |_______     | |    | |_/ /
| |    | || |   | ||_______  \    | |    |  _  |
| |    | || |   | |         \ \   | |    | | \ \
| |    | || |___| | ________/ / __| |__  | |  \ \
|_|    |_||_______||_________/ |_______| |_|   \_\


____________________________________________________________
Hello! I'm MisterMusik!
What can I do for you?
```

*Figure 1. The welcome message from **MisterMusik**.*



```
____________________________________________________________
Notice: Words in <> are the parameters to be supplied by the user;
        Items in [] are optional;
        Items with | in between them indicate the user can choose to use either of them
-----Basic Commands-----
1.  "help"                      -- Print out all the commands you can input.
2.  "list"                      -- Print out all the events in the list.
3.  "reminder [number of days]"-- Display the list of events over the next given number (default 3) days.
4.  "check"                     -- Print the next 3 free days.
5.  "find <keyword>"            -- Search for a specific event using keywords.
6.  "view dd-MM-yyyy"           -- Print the event list for a particular date.
7.  "budget MM-yyyy"            -- View monthly cost of concerts
8.  "budget set <new budget>"   -- Set new monthly budget
9.  "bye"                       -- Exit the program.
-----More Commands-----
1.  "help calendar"    -- To see commands about calendar.
2.  "help event"       -- To see commands about how to add or delete event.
3.  "help goal"        -- To see commands about goal management of an event.
4.  "help contact"     -- To see commands about contact management of an event.
5.  "help checklist"   -- To see commands about checklist management of an event.
6.  "help instruments"-- To see commands about instruments management
7.  "help change"      -- To see commands about changing basic information of an event.
____________________________________________________________
Please enter the command:
```

*Figure 2. The command line user interface for **MisterMusik**.*

My Roles in the project

- Created calendar table for a great improvement in visualizing schedule.
- Created checklists for events, so that students can easily manage a checklist for various situations.
- Created check free day functionality to easily check the coming 3 days that are completely free.

Note the following formatting used in this file:

Calendar	Text highlighted in grey indicates this is a command that can be inputted into the command line for the execution of certain features.
calendarView	Text in blue and highlighted in grey indicates this is a class, object or component in the architecture of the application.

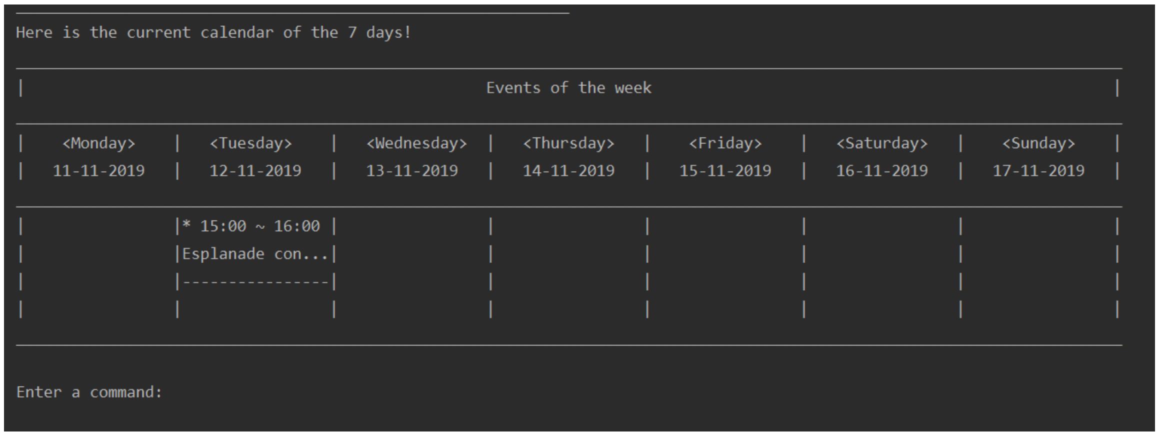
Summary of Contributions

1. Overall code contributions

Overall code contributions can be viewed here.

Or: https://nuscs2113-ay1920s1.github.io/dashboard/#search=ZhangYihan

2. Major enhancement - Calendar table interface



```
____________________________________________________________
Here is the current calendar of the 7 days!

________________________________________________________________________________________________________________________
|                                                 Events of the week                                                   |
________________________________________________________________________________________________________________________
|    <Monday>    |   <Tuesday>    |  <Wednesday>   |   <Thursday>   |    <Friday>    |   <Saturday>   |    <Sunday>    |
|   11-11-2019   |   12-11-2019   |   13-11-2019   |   14-11-2019   |   15-11-2019   |   16-11-2019   |   17-11-2019   |
________________________________________________________________________________________________________________________
|                |* 15:00 ~ 16:00 |                |                |                |                |                |
|                |Esplanade con...|                |                |                |                |                |
|                |----------------|                |                |                |                |                |
|                |                |                |                |                |                |                |
________________________________________________________________________________________________________________________

Enter a command:
```

Figure 3. The calendar table interface.

What it does: Calendar table is a table printed that displays all the events information of 7 days in the format of an actual calendar. With the dates, day of week, event descriptions and event times, the calendar table provides users a much clearer way of visualizing the schedule of a week. When `calendar` entered, a table of the week will be printed, containing information of all the events of this week. `calendar next`/`calendar last` can be used to show calendar of the next/last week of the current displaying week.

Justification: Since MisterMusik is a command line based application, normally events would be presented to users in lists. However, we are more used to looking at calendars in the format of tables, which is why presenting events to users in calendar tables could be much better.

Highlights: This enhancement was challenging because in the terminal, application only prints line by line. However, all the info of an event cannot be fully displayed within one line. Thus, it is not possible to print everything right after receiving the relevant data. I managed to overcome this by temporarily storing the data of events in a week, with a customized formatting followed before printing to screen.

3. Major enhancement - Checklist management

What it does: Checklist management allows users to create a checklist for every event. They can put for example "items to bring" into the checklist. The items in checklists can be viewed, added, removed or edited easily.

Justification: It is very normal to want to have a checklist for different events such as lessons, exams or concerts. However, placing the items inside description makes it very hard to manage. Thus, having a checklist for every event would be convenient and useful.

Highlights: This enhancement was challenging because we have to not only navigate to specific events, but also specific items in the checklist if we were to operate them. However, once implemented, it is helpful to the software and it works well along with all other enhancements.

4. Minor enhancement – Check free days

What it does: Checking free days is a functionality that provides to the user the next 3 days that are completely free, which contains no events.

Justification: Checking free days is a very useful enhancement as students often want to find a day to arrange or attend certain activities. This made the process so easy that users only can get the free days in less than 5 seconds.

Highlights: This enhancement was overall less challenging comparing to the major enhancements. The only part that was challenging was to iterate through the whole list while increasing java `Date` at the same time.

5. Other contributions

- Project management

    - Set goals and objectives for the project team to smoothly work together.
    - Refactored code architecture for more logical structure
    - Fixed code styles for better readability and fewer mistakes.
    - Wrote various test cases for better coverage of testing.
    - Revised pull requests of teammates to avoid unnecessary mistakes and errors.
- Documentation
    - Added missing javadocs for better readability and clearer structure
    - Wrote corresponding user guide and developer guide.

Contributions to User Guide

We had to update the original user guide with instructions of our new enhancements. The followings are excerpts of MisterMusik user guide, which shows the additions I made to the file according to my enhancements. It also includes features that are coming in version v2.0.

3.7. Calendar Table: `calendar`

The calendar table is generated from the EventList. It prints on the screen a table of calendar of 7 days starting from a specified day, including the events within this time period.

3.7.1. Commands for CalendarView

- *`calendar` This prints the calendar table of this 7 days.*
- *`calendar next` This prints the calendar table of the next 7 days.*
- *`calendar last` This prints the calendar table of the last 7 days.*
- *`calendar on` Allow the calendar to be printed after every command execution.*
- *`calendar off` Not allowing the calendar to be printed after every command execution.*

3.9. Checking for free days: `check`

The user can check for the nearest days that are free. This will list the next 3 free days on the users' schedule. A day is considered free if there are no events scheduled. ToDos are not counted as events.

Future enhancements of this function will be to check free hours.

3.16. Checklists: `checklist`

Checklist of each event can be used to remind users of certain items (e.g. bring glasses to concert). This is implemented by storing an array list of strings in `Event` objects. Storing checklist data to files will be available in v2.0.

Checklist implementation contains 4 operations:

3.16.1. add checklist item

`checklist add <event index>/<checklist item>` This adds an item to a specific event’s checklist.

3.16.2. view checklist

`checklist view <event index>` This displays the checklist of a specified event.

3.16.3. edit checklist item

`checklist edit <event index> <item index>/<new item>` This edits a specific item in the checklist of an event.

3.16.4. delete checklist item

checklist delete <event index> <item index> This deletes an item from the checklist of an event.

Contributions to Developer Guide

The follow section is my additions to MisterMusik developer guide.

3.1. Calendar Table

The calendar table is generated from the EventList. It prints on the screen a table of calendar of 7 days starting from a specified day, including the events within this time period.

3.1.1. Sequence diagram

Below is the sequence diagram of calendarView.

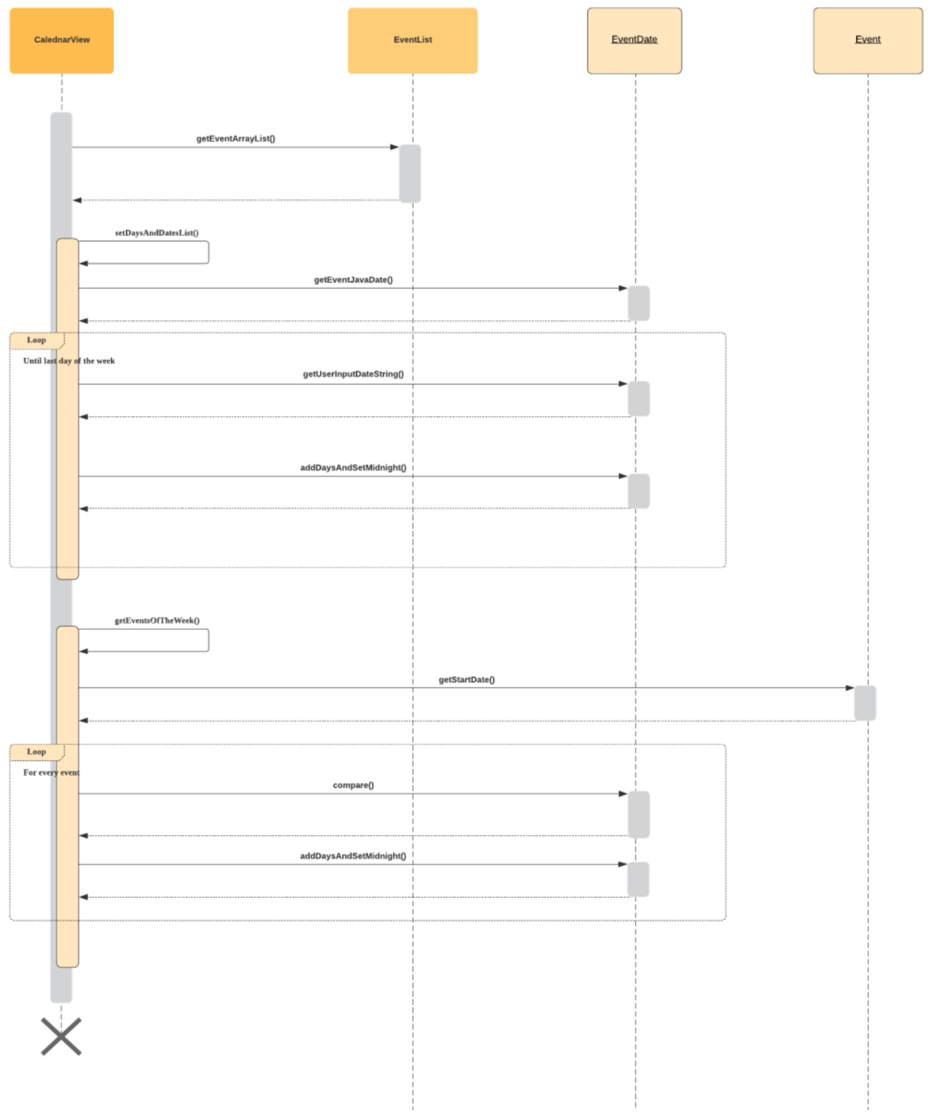


Figure 4. Sequence diagram of calendar command.

3.1.2. How it is implemented

Given below is the sequential steps of how a calendar table is generated and printed.

Step 1. User enters calendar to start the initialization process of a calendar table with today as the starting day.

Step 2. The program checks the date of the given start day to generate a list of 7 days, starting from given day. It also gets the day of the 7 days (e.g. Monday, Tuesday, etc). This sets the dates info of the table.

Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday

26/10/2019	27/10/2019	28/10/2019	29/10/2019	29/10/2019	30/10/2019	31/10/2019

Figure 5. Example in calendar command.

Step 3. The program find all events in the EventList that is within the 7 days, and store them correspondingly into 7 queues, representing the 7 days. This is for further printing.

Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday
26/10/2019	27/10/2019	28/10/2019	29/10/2019	29/10/2019	30/10/2019	31/10/2019
<ToDo>	<Exam>		<Recital>		<Practice>	<Lesson>
	<Concert>				<ToDo>	
					<Exam>	

Figure 6. Example in calendar command.

Step 4. The program now have all the information of these 7 days and is then able to print the calendar table.

1. It initiates an empty string to store all info of the calendar and for later printing.
2. It puts the header of the table into the string.
3. It puts the days of week and dates info into the string.
*4. To add in events, each event takes 3 rows (time info, description, and ashes) to print. For each 3 rows, there can be at most 7 events. The events are added per 3 rows. For each 3 rows, the program creates an array of 3 * 7 to store the details. Whenever there exists an event at the position of a day, details of the event will be added to the corresponding 3 rows (1 column) of the array. The array is then added by rows into the string.*

	* 18:00 – 20:00				* ToDo	
	Concert at …				Return book	
	--------------------				--------------------	

Figure 7. Example in calendar command.

3.1.3. Commands for CalendarView

- `calendar` *This prints the calendar table of this 7 days.*
- `calendar next` *This prints the calendar table of the next 7 days.*
- `calendar last` *This prints the calendar table of the last 7 days.*
- `calendar on` *Allow the calendar to be printed after every command execution.*
- `calendar off` *Not allowing the calendar to be printed after every command execution.*

3.3. Checklists

Checklist of each event can be used to remind users of certain items (e.g. bring glasses to concert). This is implemented by storing an array list of strings in `Event` *objects. Checklist implementation contains 4 operations:*

3.3.1. add checklist item

`checklist add <event index>/<checklist item>` *This adds an item into a specific event's checklist.*

3.3.2. view checklist

`checklist view <event index>` *This prints on the screen the checklist of an event.*

3.3.3. edit checklist item

`checklist edit <event index> <item index>/<new item>` *This edits a specific item in the checklist of an event.*

3.3.4. delete checklist item

`checklist delete <event index> <item index>` *This deletes an item from the checklist of an event.*

3.2. Check Free Days

CheckFreeDays is a command that allows the program to search for the next 3 days *without any events (except ToDos).*

1. When the user enters check, starting from the current day, the program checks all the events whether any is in this day.

2. If not, this day will be added into a list.

3. Above process will continue until the list has 3 days, which will then be printed.

The following activity diagram shows how check free days is implemented.

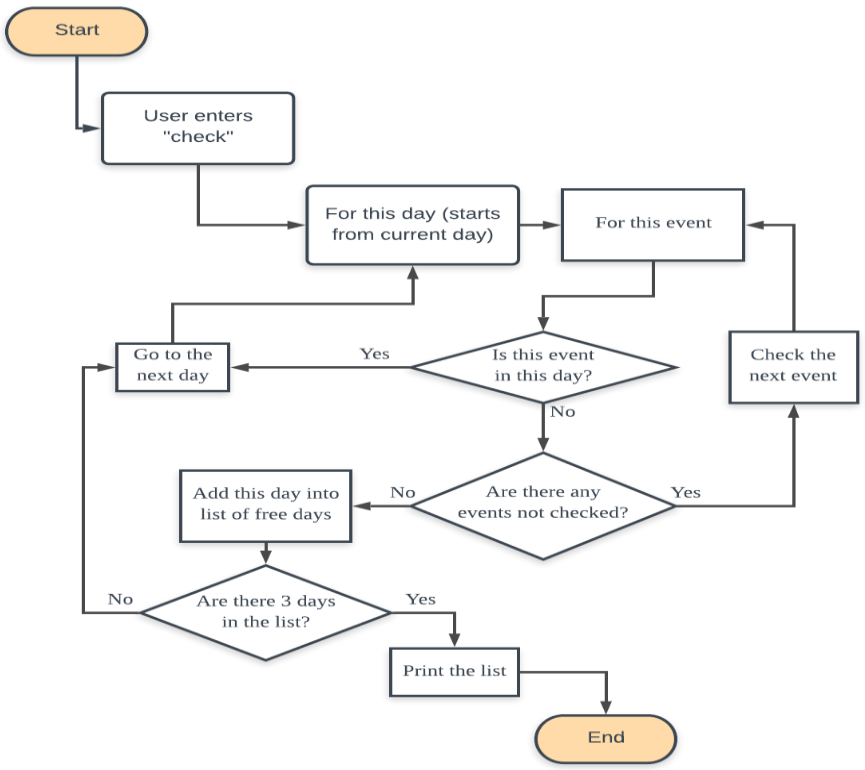

Start
User enters "check"
For this day (starts from current day)
For this event
Is this event in this day?
Yes
Go to the next day
No
Check the next event
Are there any events not checked?
No
Yes
Add this day into list of free days
Are there 3 days in the list?
No
Yes
Print the list
End

Figure 8. Activity diagram of checkFreeDays command.