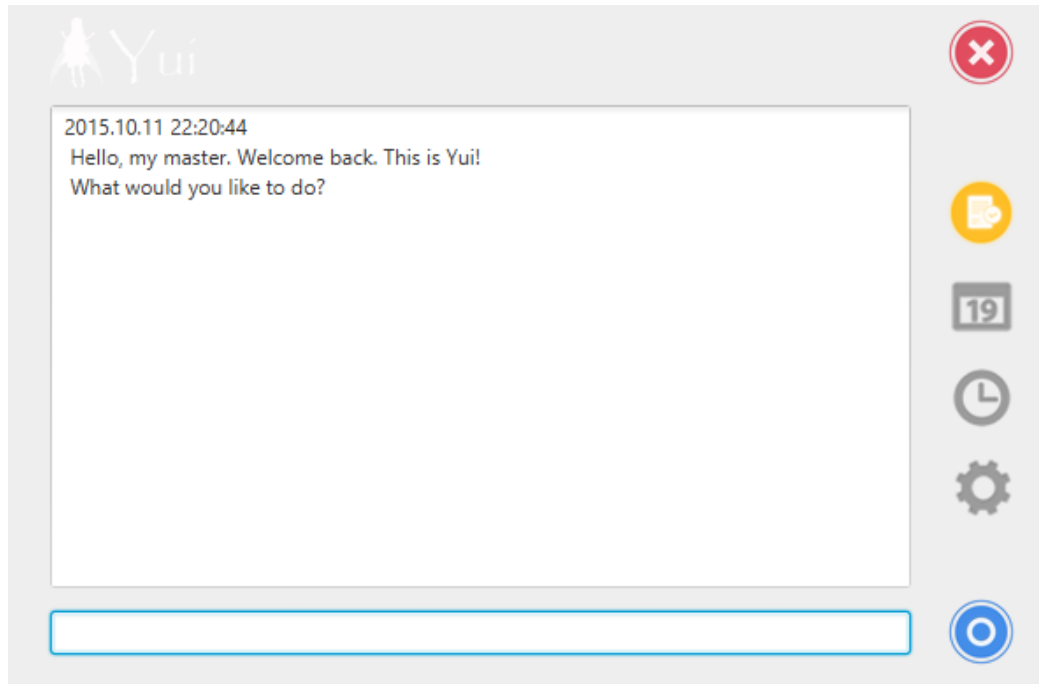






Yui Project Menu



Supervisor: Karan Kamath Extra feature: Recurring Task & FlexiCommands

 <p>Liu Hanyang</p> <p>Team lead</p> <p>Git expert and Eclipse expert</p> <p>UI Designer</p>	 <p>Justin Erh</p> <p>Documentation</p> <p>Code Quality</p>	 <p>Mao Chenyang</p> <p>Testing</p> <p>Integration</p>	 <p>Le Nguyen</p> <p>Scheduling and tracking</p> <p>Deliverable and deadline</p>
---	--	--	---

Credit: We used JavaFx API for our UI design. More information can be found in the UI component detail on page 21.



User Guide

Liu Hanyang (A0133992X)

Mao Chenyang (A0127142R)

Le Nguyen (A0127821J)

Justin Erh (A0125622N)

Contents

1	Introduction	5
2	Quick Start	5
3	Basic Functions.....	5
3.1	Add events.....	6
3.2	Show events.....	6
3.3	Update events.....	7
3.4	Delete events	7
3.5	Search events.....	7
3.6	Undo and Redo.....	8
3.7	Help.....	10
3.8	Exit	10
4	More possibilities.....	10
4.1	Add events with deadlines.....	10
4.2	Add events with start and end times	10
4.3	Outline.....	11
4.4	Mark as 'complete'.....	11
4.5	Comments.....	11
5	Extra feature	11
5.1	Import and Export data	11
5.2	Recurring task.....	12
6	Cheat Sheet.....	12
7	Contact Us.....	13
8	Appendix A: User stories. As a user, I	13
8.1	[Likely]	13
8.2	[Unlikely].....	14
9	Appendix B: Non Functional Requirements.....	15
10	Appendix C: Product survey	15

1 Introduction

Yui is a Personal Events Management Assistant software. You can manage your upcoming tasks and events using this software and it will give you reminders when the day of a particular event is approaching. You can use Yui solely on your keyboard. Now, you can refer to section 2 for a quick start in using Yui.

2 Quick Start

Thank you for choosing Yui! To start, double click on the Yui.jar file. This will start our app. If you cannot start the app, please proceed to download JDK 8u60 or later from the Oracle official website: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

Once you open the app, a window similar to Figure 2.1 will appear and you can start to manage your own event list. To learn more, please read section 2 on more commands and functions. You can also type in "help" to see the list of commands available.

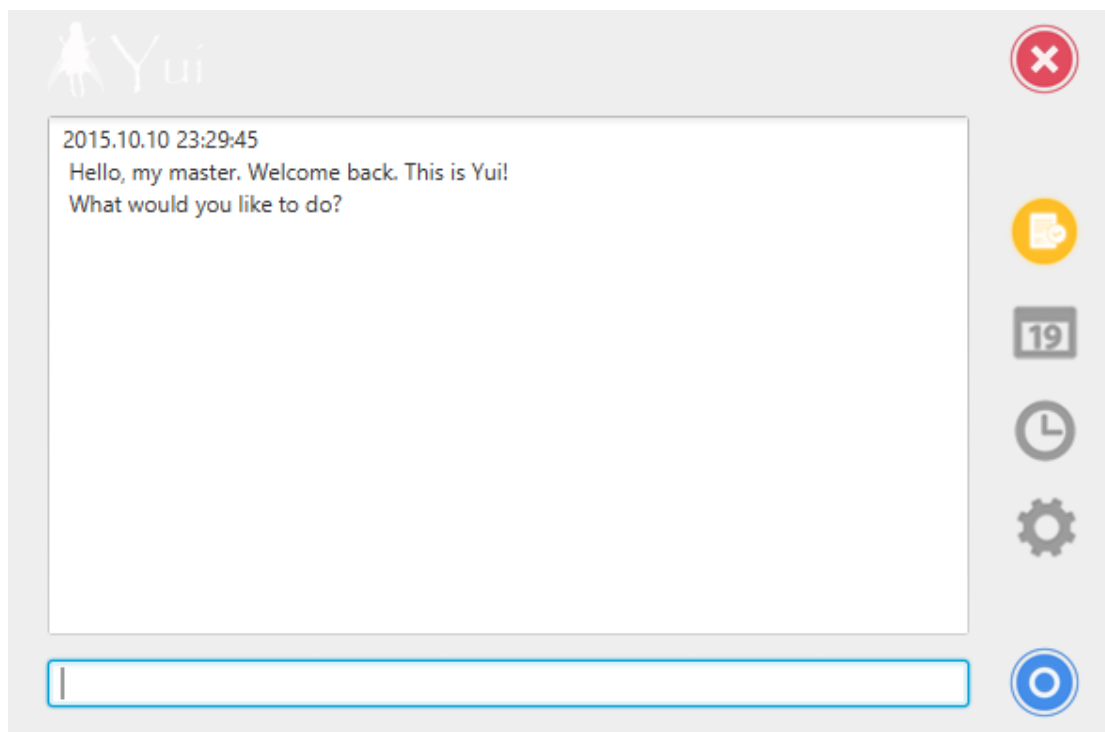


Figure 2.1. Demo of the interface of Yui

3 Basic Functions

This section will give you an idea about what you can do with Yui and guide you to start using it. You will find this section most useful if you are a light user. If you want to explore more on this app, you should proceed to read section 3 after finishing this section.

3.1 Add events

To add simple (floating) events to Yui, type “add <event>” into the software then hit ‘Enter’. This command is best if the event you wish to add does not have any further details associated with it.

Examples:

- add have a CS2103T class
- add date with girlfriend
- add have lunch with tutor

3.2 Show events

Type “show” to display all the events that have been added. You can refer to Figure 3.1 and 3.2 for the possible outcomes. (Figures below only capture the useful information for a clear demonstration of the function)

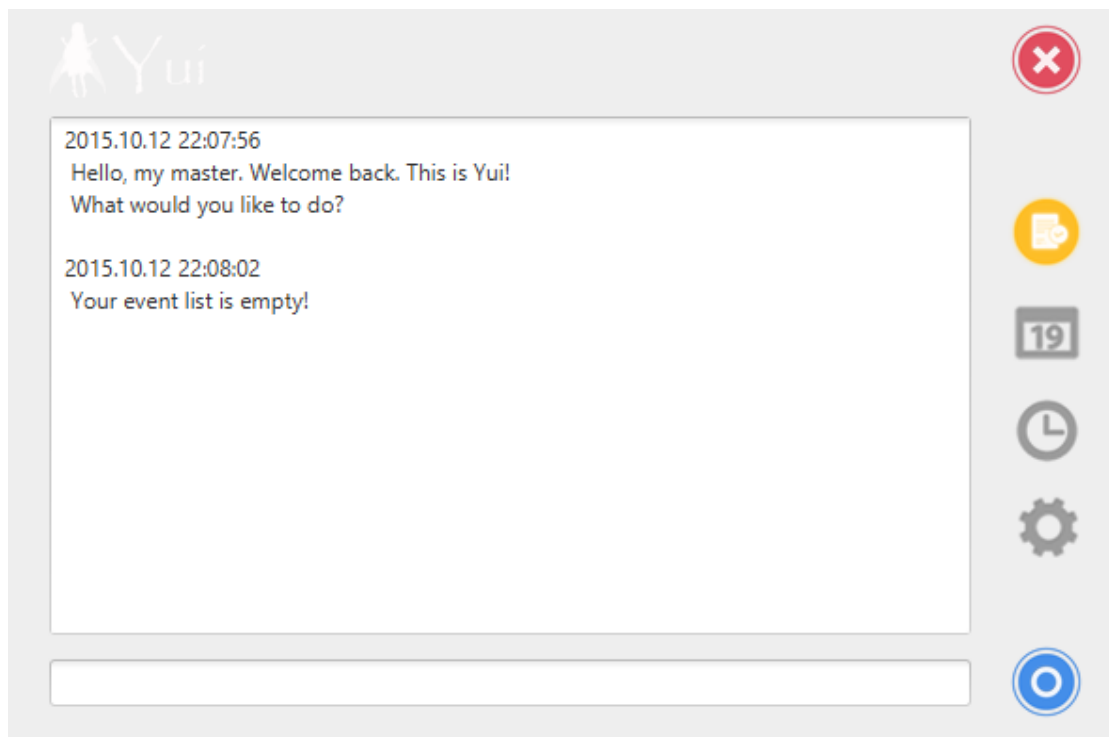


Figure 3.1. Response from Yui if you have not entered any event

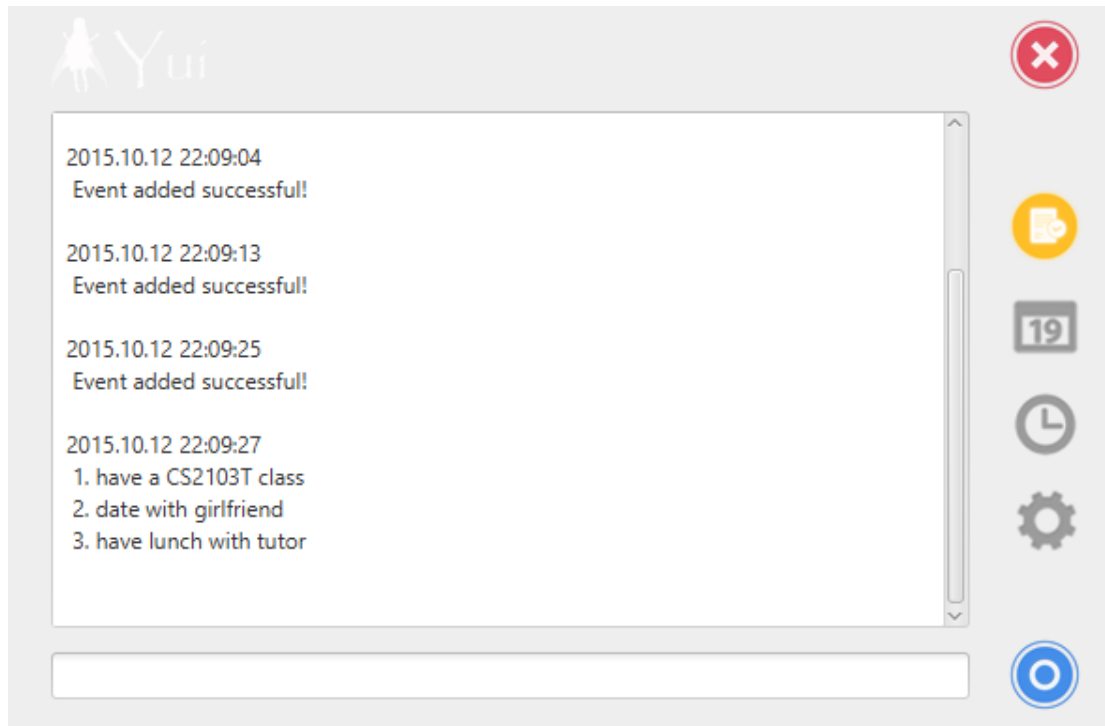


Figure 3.2 Response after adding 3 events and type “show”

3.3 Update events

To update events with new details, type “update<event number><event>”.

Examples:

- update 1 read lecture notes before CS2103T lecture
- update 2 buy flowers before dating with girlfriend
- update 3 ask about project when having lunch with tutor

3.4 Delete events

In the event that you wish to delete events that are no longer applicable or events that were erroneously keyed in, type “delete <event number>”.

3.5 Search events

To look for events with specific details, type “search <keyword>” into the program. Please refer to examples below and Figure 3.3.

Examples:

- search tutorial
- search date
- search lunch

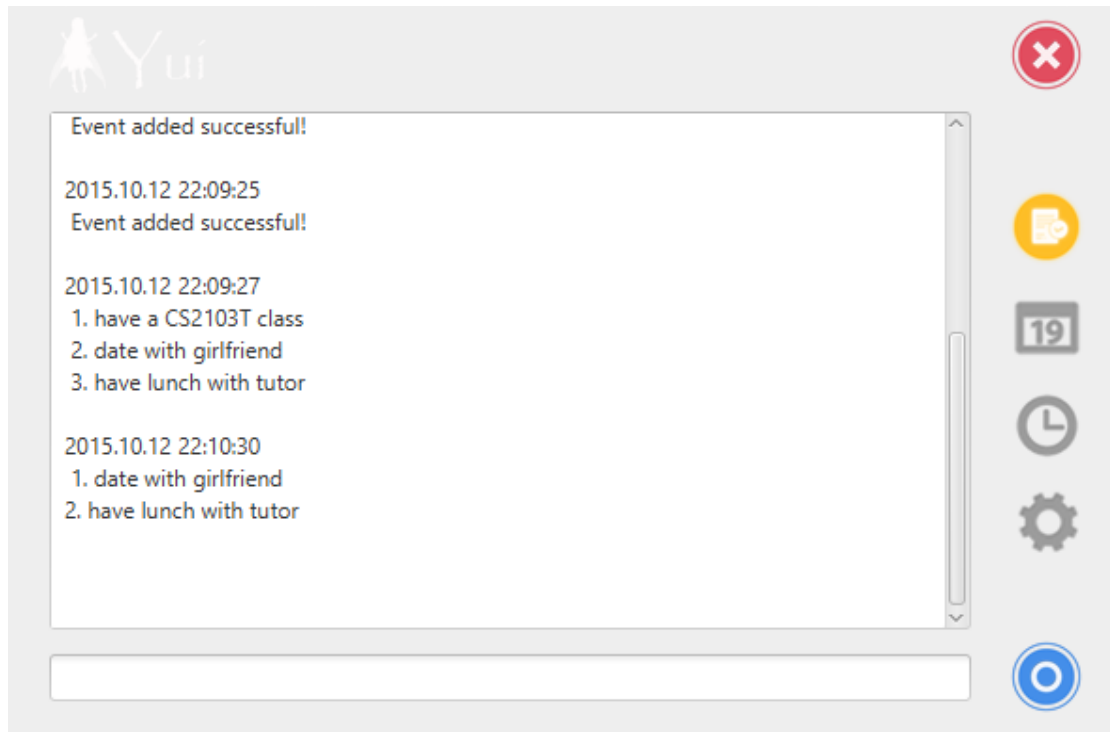


Figure 3.3. Search result of “with”

3.6 Undo and Redo

If the user wants to undo your last change in the program, type “undo” into the program. Commands such as “show” or “search” will be ignored by “undo” because you did not make any changes to the list. In addition, you cannot “undo” twice in a row! You can refer to Figure 3.4. To reverse your undo command, type “redo” into the program. Figure 3.5 is the “redo” after Figure 3.4.

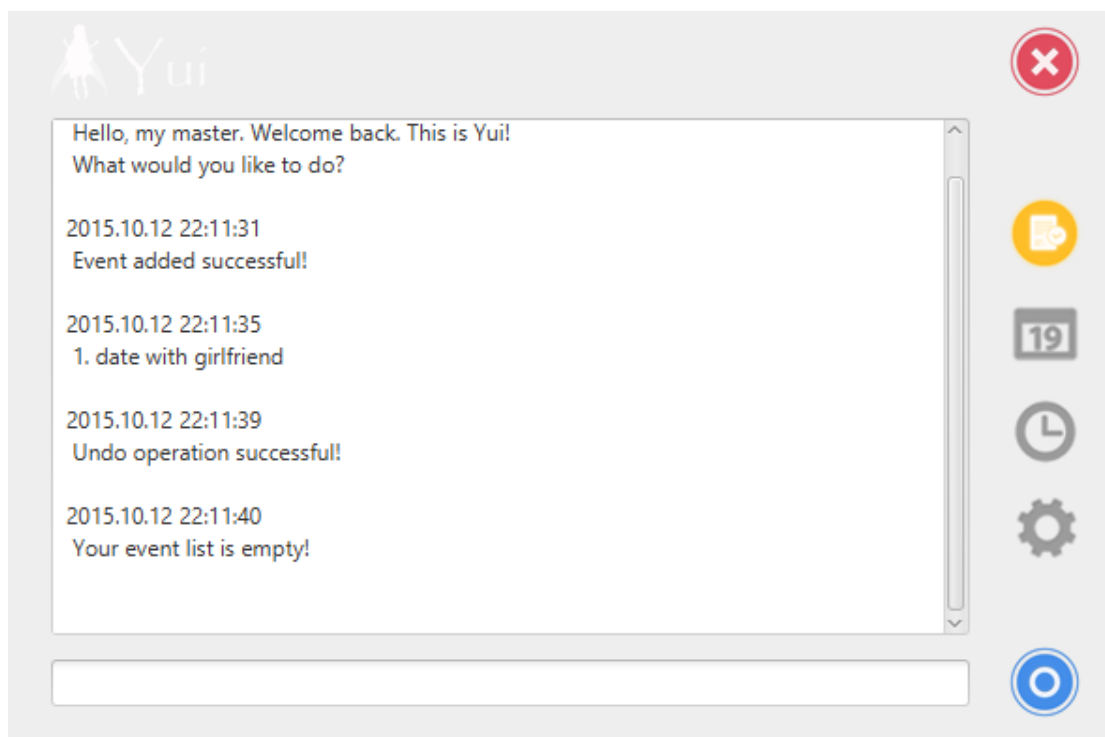


Figure 3.4. “Undo” after adding one event and “show”

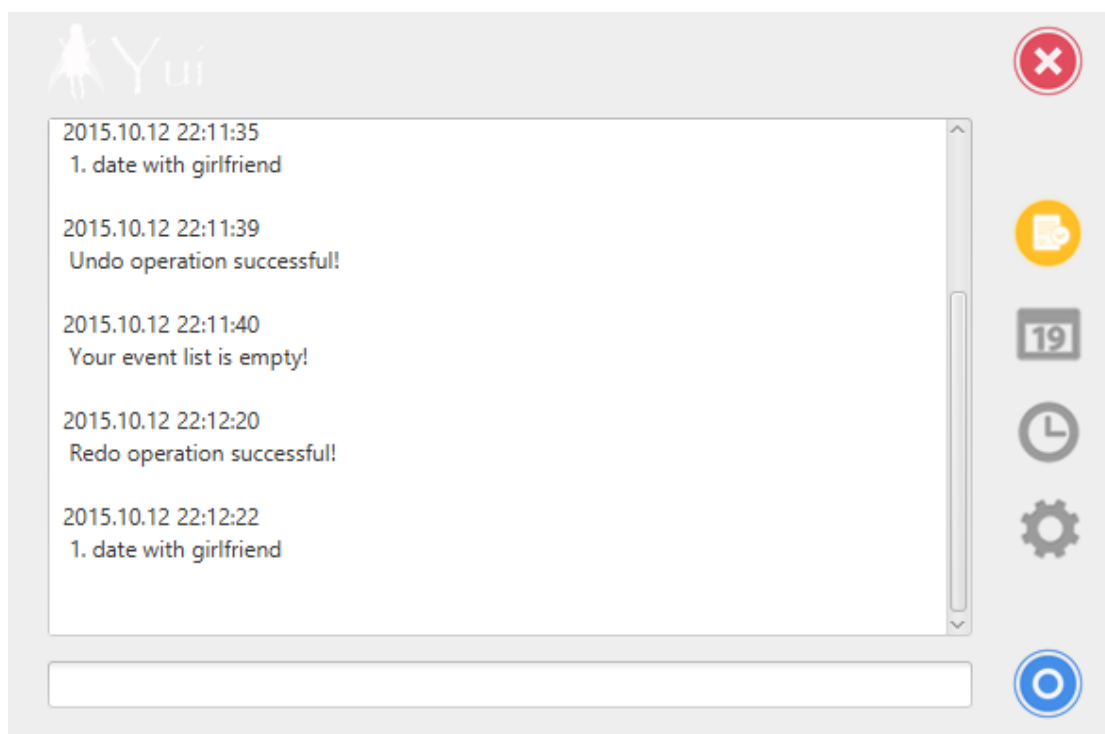


Figure 3.5. “Redo” the last operation and “show”

3.7 Help

When you feel lost and do not know what to do, type “help” to call out the cheat sheet with all the commands and their functions.

3.8 Exit

When you want to exit Yui, simply type “exit”.

4 More possibilities

This section gives information on more commands. With special permutations of commands, you can do more amazing stuff. After reading this section, you should be able to use most of the functions we currently provide in Yui. However, feel free to read on section 4 for the extra features we provide for heavy users!

4.1 Add events with deadlines

If the event that you wish to add must be carried out by a specific date and time, type “add <event> by <HH:mm> <DD/MM/YY>”, where <HH:mm> represents the time and <DD/MM/YY> represents the date. You may also substitute the date with either “today” or “tomorrow”.

Examples:

- add finish CS2103T tutorial by 09:00 tomorrow
- add buy flowers by 20:00 05/10/2015
- add cook lunch by 12:00 today

4.2 Add events with start and end times

Type “add <event> from <HH:mm> to <HH:mm> <DD/MM/YY>” to add a task that starts and ends at a specific time. Like events with deadlines, you may substitute the date with “today” or “tomorrow”.

Examples:

- add CS2103T lecture from 09:00 to 11:00 tomorrow
- add date with girlfriend from 20:00 to 24:00 05/10/2015
- add have lunch with father from 12:00 to 13:00 today

4.3 Outline

To display the events that need to be completed today or tomorrow, type “outline” (or “show today”, “show tomorrow”). These events will be shown in a table.

4.4 Mark as ‘complete’

When you have finished an event, type “mark <event number> complete” to set it as completed. The event will no longer appear in your reminder list.

4.5 Comments

To add comments to events, type “comment <event number> <comment>”. This is useful if the event has some additional details that you are otherwise unable to put into the event title.

Examples:

- comment 1 E3-06-09
- comment 2 budget is only \$5
- comment 3 save the eggs for Sunday

5 Extra feature

This section talks about the add-ons and extensions to Yui we have already implemented. You have to learn to use the features but they will definitely help to increase your productivity if you are a heavy user!

5.1 Import and Export data

To change where the data is located, type locate <folder directory> into the program

Example:

- locate D:\cs2103T\calendarFolder

5.2 Recurring task

If the user want to set up a task to do every Friday or maybe at the 1st day of the month, we also provide the needed tool for user to do so by using the command `recur <event number> <day of the week/ day of the month>`

Examples:

- `recur 3 Friday`
- `recur 2 12`

6 Cheat Sheet

Command	Description
<code>add <event> <XX:XX> <DD/MM/YY></code>	Add a new event with deadline to users' to-do list
<code>add <event></code>	Add a new event without details
<code>add <event> <XX:XX> <XX:XX> <DD/MM/YY></code>	Add a new event with specific do-time
<code>mark</code>	Mark event as done
<code>comment <Your Comment></code>	Comment on users' current event
<code>show</code>	Show all events
<code>locate</code>	Show the file in the file explorer to allow users to import/export or backup
<code>update <index> <new content></code>	Update the details of users' events
<code>undo</code>	Undo the previous operation (This is also able to do by Ctrl + Z)
<code>redo</code>	Redo the undo operation
<code>delete <index></code>	Delete the event
<code>-help</code>	Call the help menu to view all the commands available

outline	Show today and tomorrow outlines to users
recur <days>	Create recurring tasks
search <Key Word>	Search for the event in your list
exit	Exit Yui

7 Contact Us

Whenever you encounter problems when using Yui, feel free to contact our developing team through email. The followings are the email addresses:

Liu Hanyang (A0133992@u.nus.edu)

Mao Chenyang (A0127142@u.nus.edu)

Le Nguyen (A0127821@u.nus.edu)

Justin Erh (A0125622@u.nus.edu)

8 Appendix A: User stories. As a user, I ...

8.1 [Likely]

(For some functions in the user stories, we have not figured out a smart implementation. To avoid ambiguity, we did not put them in the user guide. However, we will update them in future versions.)

ID	I can ... (i.e. Functionality)	so that I can ... (i.e. Value)
create	add new event	record tasks that I want to do some day
read	display all events that I have created	read all the events I have created
undo	undo the last command	delete my last wrong input easily
redo	revert the undo command	redo my last command that I accidentally undo
delete	delete an event	delete events that I no longer want to keep track of

edit	edit information from existing event	modify or add more information to an event
search	search and display specific event	search of specific event easily
extraCommand	have various command words to execute the same command	input command in a shorter form or have an easier to remember command
readToday	display only events from that day	read only the events I have to do by today
comment	add extra comment for an event	add more extra information on an event
reminderType	have at least 3 set of basic reminder type for low, medium, high priority	assign various events with a more favorable reminder settings easily
customReminder	reminder type can be customized	further customize the reminder setting if I need to
import/export	store and transfer event data between 2 devices	transfer my event list between my laptop and desktop
mark	mark an event as done	turn off reminder for that event
help	display available commands	find the command input I need if I forget/ do not know
caseInsensitive	input command, search event without the need of correct case	not worry about case sensitivity for long commands and event names
recurTask	input recurring events	do not have to type in the same recurring event multiple times
clearUI	have a clear UI	see my events more clearly

8.2 [Unlikely]

ID	I can ... (i.e. Functionality)	so that I can ... (i.e. Value)
multilingual	input command and event name in Chinese/ different languages	have support for user who are not familiar with English

emoji	input emoji	express more feelings and expression toward certain events
specialSyntax	allow command input from special syntax	fasten the process of inputting command

9 Appendix B: Non Functional Requirements

1. The software should run on any device with Java Environment.
2. The software can respond to user input faster than a blink of the eye.
3. The software can be launched quickly (less than a second).
4. The software works perfectly even without Internet connection

10 Appendix C: Product survey

<p>Product: Google Calendar Documented by: Le Nguyen</p> <p>Strengths:</p> <p>Mouse centric, 'drag and drop' design</p> <p>Offer easy color code for different events</p> <p>Friendly GUI</p> <p>Weaknesses:</p> <p>Only provide "1 hour before" reminder, user needs to set up reminder setting if they want to have a more specific reminder</p> <p>Not keyboard centric, lack good input command from keyboard</p> <p>Lack support for multi-day event (event that last between 10pm-2am next day)</p> <p>Limited offline functionality</p>
<p>Product: Google Keep Documented by: Mao Chenyang</p> <p>Strengths:</p>

Works both online and offline, fast response

Captures events with different needs well

Can be used without Internet

Can mark something as done and archive it

Weaknesses:

Cannot add quickly (does not support quick add)

Short cut is not easy to set (no build-in setting of short cut)

Product: Google Calendar **Documented by:** Liu Hanyang

Strengths:

Support different languages

Have a clear UI, which is convenient for users to use.

With an account, users can check their calendars in any devices linked to the Internet.

It is easy to share your calendar to friends

It would send emails to users to remind what is going to happen

It is easy to search the events in the calendar.

Weaknesses:

Must be used online, which people without Internet cannot use.

It is linked with an account. If users lose their account, it cannot be used any more.

If users only the website version, they may miss some reminder.

Information on the Internet is not so safe.

Product: Fantastical 2 **Documented by:** Justin Erh

Strengths:

Natural language to enter events and reminders

Supports time zones

Available also for mobile devices – calendars can sync using a cloud service

GUI looks neat and clean

Reminder list (separate from events)

Localized in various (though not all) languages

Weaknesses:

Mac only - not for Windows

Not free and expensive

Mobile version is separate, requires extra payment

No Web version, just offline (with cloud sync) application



Developer Guide

Liu Hanyang (A0133992X)

Mao Chenyang (A0127142R)

Le Nguyen (A0127821J)

Justin Erh (A0125622N)

Contents

1	Introduction of Yui.....	20
2	Architecture Briefing.....	20
3	Components.....	21
3.1	Commands Flow.....	21
3.2	UI.....	21
3.3	Logic.....	23
3.4	Parser.....	24
3.5	Storage.....	25
4	How to use APIs.....	25
5	Start your codes	27
5.1	UI.....	28
5.2	Logic and Parser	28
6	Testing.....	29
7	Appendix - API Library.....	30
7.1	ToDoList API.....	30
7.2	Action API.....	30
7.3	Event API.....	31
7.4	Parser API	31
7.5	Storage API	31

1 Introduction of Yui

Yui is an advanced task management software. It can be used to record your upcoming tasks and give you reminders when necessary. Users are able to control and use all the functionalities in this software such as adding tasks, searching tasks and setting reminders by just keyboard commands.

This developer guide serves to help new incoming developers who would like to understand the internal design of Yui and hope to extend the program with more advanced functions which they deem useful.

We will start with the overall architecture of Yui ([Section 2](#)) and proceed to functions of the major components of Yui ([Section 3](#)). After having a brief idea about Yui, you should refer to respective sub-sections in existing APIs ([Section 4](#)), coding demonstration ([Section 5](#)) and testing framework ([Section 6](#)) to make your code fit into Yui.

Now, let's get started!

2 Architecture Briefing

Here is a general view of the design of Yui. There are four major components in Yui, including UI, Logic, Parser and Storage. Each component has its own interface. The main functions of each component is shown as follow:

- UI Component: Control the appearance of Yui
- Logic Component: Handle user commands entered and modify the event list
- Parser Component: Parse commands received and return information to Logic
- Storage Component: Save and load data files

You can refer to Figure 1 below for a better understanding.

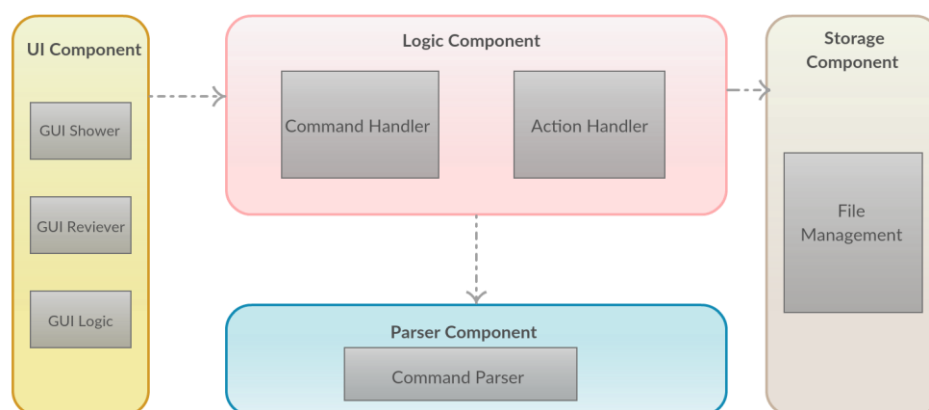


Figure 1: Software architecture of Yui

3 Components

3.1 Commands Flow

In Yui, all the operations are initiated by users. When users type in a new command in the user interface, it would be sent to the Logic components. The Logic component will then send the command to Parser component first and get back essential information. After that, it would send the feedback message to the UI and save the event data into Storage component at the same time. The Storage component would save the data received into local file, and it would also read data from the local file and pass it to the Logic component if the Logic component request a “show”. This process is reflected in the following diagram:

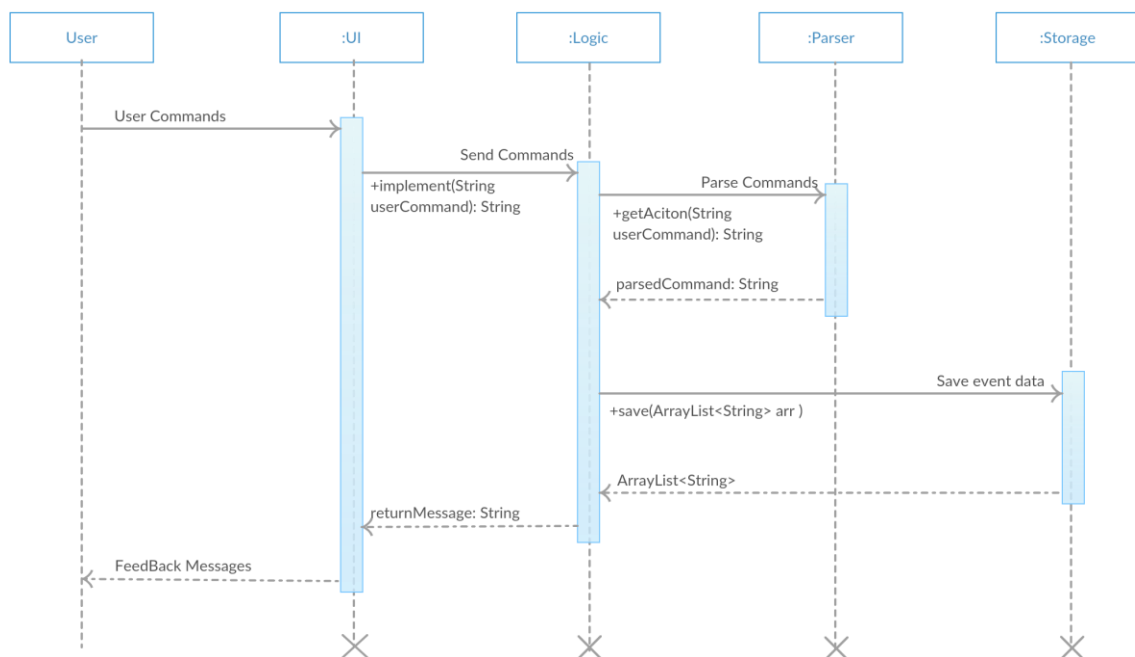


Figure 2: Sequence diagram for Yui to deal with new commands

3.2 UI

Here you can check how the UI component works. If you want to modify the interface or methods, you can go through this part.

GUI component is mainly written with the JavaFx API. Here we assume you have been familiar with JavaFx. If not, you can refer to <http://docs.oracle.com/javase/8/javase-clienttechnologies.htm> to learn about JavaFx.

3.2.1 Classes Structure

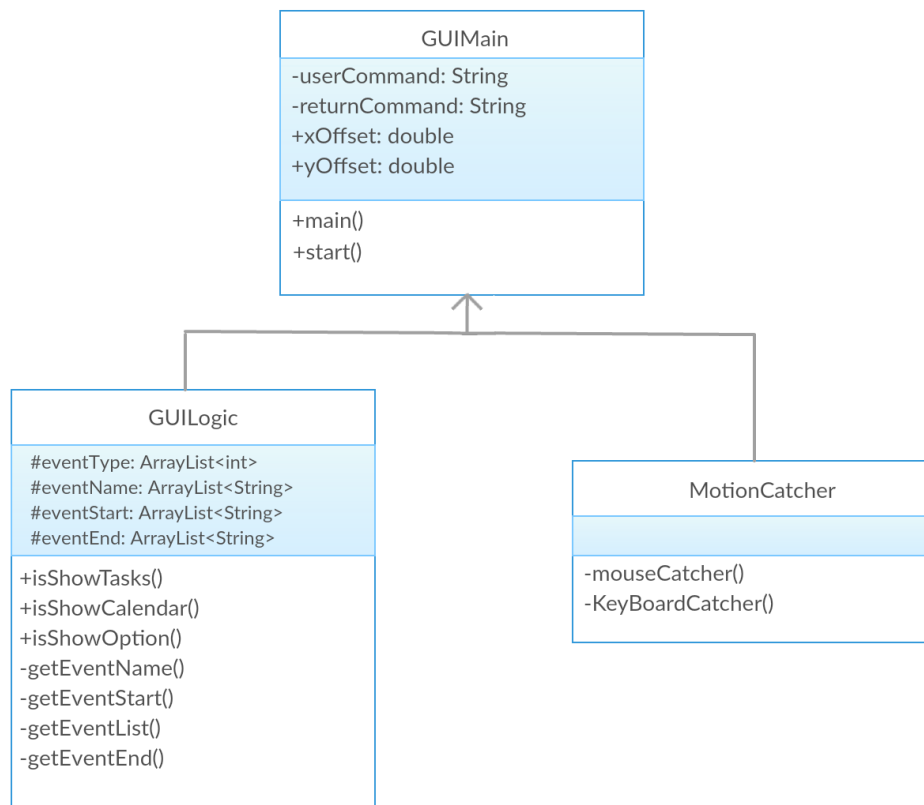


Figure 3: Class diagram for UI component

3.2.2 GUIMain

This main class mainly describes how to draw the UI with codes. If you want to change the appearance of GUI, you just need to modify this class first.

The methods, `main()` and `start()` are just for launching the graphical interface.

3.2.3 GUILogic

This class is used to change the appearance of UI with commands. It usually receives the handled results from logic and uses them to alternate the appearance of UI.

It controls the appearance of task lists, command boxes and the calendar diagram. Also, it can handle the different kinds of events and show them separately.

3.2.4 MotionCatcher

This class is used for catching the actions of users. It would catch the keyboard actions to control the showing of task lists, the undo or redo operation and exiting operations.

3.3 Logic

This component is used to process the commands passed from UI.

3.3.1 Classes Structure

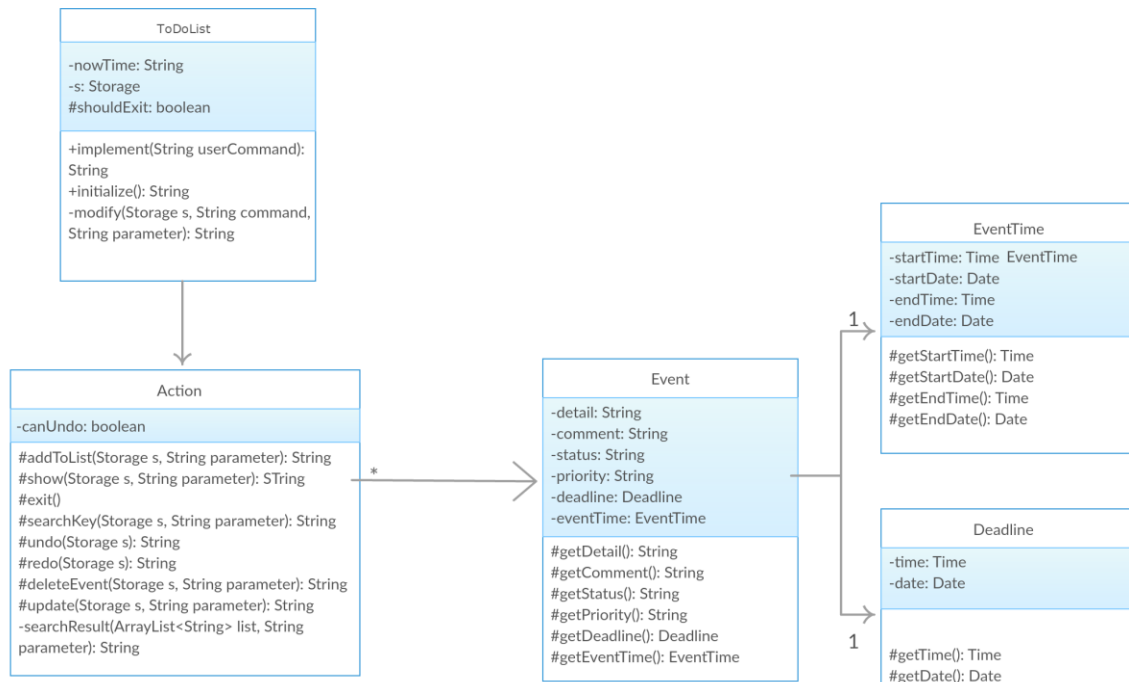


Figure 4: Class diagram for Logic component

3.3.2 ToDoList

This is the main class of Logic component. It is used to initialize the storage or read existing file and implement the user commands.

This class would call **Parser** first when receiving a new command, and get the parsed command to modify.

The method `modify` will decide on which operation to call. It has a switch to choose `add`, `show`, `delete`, `search`, `update`, `undo`, `redo`, `exit`, etc.

3.3.3 Action

This class is used to realize various functions. When parsed commands are passed to action, it would deal with the data saved. It would be called by `modify` and return the messages to show operations are successful or errors are detected.

3.3.4 Event

It is an object to record the event details. It covers comment, status, priority, deadline and eventTime. Also, it has methods to show these properties.

3.3.5 EventTime

It is an object to record the start time and end time for the event. The start and end time are both divided into two parts, time and date. It also has methods to return these properties.

3.3.6 Deadline

It is an object to record the deadline of the event. It only covers the date and time of the deadline.

3.4 Parser

3.4.1 Class Structure

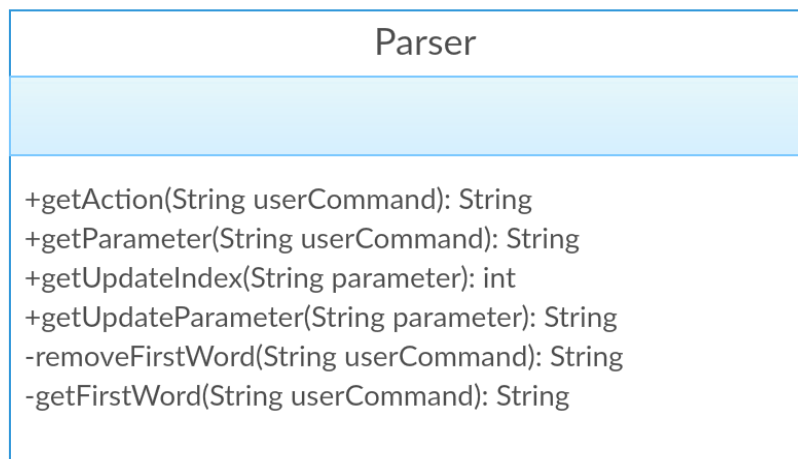


Figure 5: Class diagram for Parser component

3.4.2 Parser

This class is used to deal with the commands passed from the Logic component. It would cut the first word of the command and return what the operations back to the logic.

For example, for command “add CS2101 tomorrow”, the parser would parse it and return “CS2101 tomorrow” to the add method.

3.5 Storage

3.5.1 Class Structure

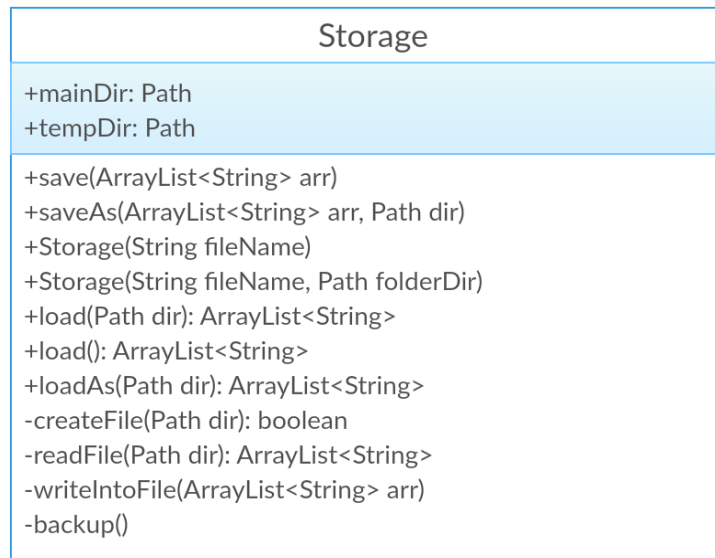


Figure 6: Class diagram for Storage component

3.5.2 Storage

This class is used to save and load data file. It would save the commands or events passed from the Logic component and read the local data file to pass events to Logic component.

4 How to use APIs

As you reach here, you have a basic understanding of the architecture and components of Yui. Now you can start to work on code level and advanced feature. Each component has had a set of public APIs (Application Programming Interfaces), which could be used in other components' classes.

For Logic developer,

you can use the APIs of Logic components through its main class, `ToDoList.java`.

The following are the APIs of `ToDoList`.

return type	description
static String	initialize() initialize the storage file and return welcome message

static String	implement() handle the commands got from UI and return the corresponding messages
static ArrayList<String>	getFloatingEvents() get the list of floating events
static ArrayList<String>	getDeadlineEvents() get the list of events with deadlines
static ArrayList<String>	getDetailedEvents() get the list of events with start time and end time

Note

To use the logic, you must call `initialize()` in GUI components before any other operations that handle commands.

For Parser developer,

you need to extract the required details from the commands that have been input by the user. The following is the public APIs of the `Parser` class.

return type	description
static String	<code>getAction(String userCommand)</code> parse the command and return what the action is
static String	<code>getParameter(String userCommand)</code> parse the command and get the parameter of the pending event

For Storage developer,

you need to manage saving and loading of both the main file as well as the temporary file (backup of the file before last editing to allow undo of last operation). You will also store the directory required for all the actions inside. The following is the public APIs of the `Storage` class.

Constructor
<code>Storage(String filename)</code> Initialize the file and storage class, generate a blank file with default directory (not recommended)
<code>Storage(String filename, Path dire)</code> Initialize storage class, generate a blank file with specified directory (recommended)

return type	description
void	save(ArrayList<String> arr) save the arrayList into the main directory
void	saveAs(ArrayList<String> arr, Path dir) save the arrayList into the main directory also change main directory to the new Path
ArrayList<String>	load() read file from the main directory
ArrayList<String>	load(Path dir) read file from the specified directory
ArrayList<String>	loadAs(Path dir) read file from the specified directory also change main directory to the new Path
Path	mainDir() return the directory for the main file
Path	tempDir() return the directory for the temporary file

5 Start Coding

Now you have known the overall design of Yui. You should have understood the APIs and algorithms used in Yui. Therefore, you can start to extend your own codes! For different components, you can refer to different sections. You can read [Section 5.1](#) for UI writing, [5.2](#) for Logic writing.

5.1 UI

Your main work is to pass the user commands to the Logic component and show the meaningful feedback on the screen. The following are the method which are used to catch commands and return feedbacks:

```
userCommandBox.setOnKeyPressed(new EventHandler<KeyEvent>() {
    @Override
    public void handle(KeyEvent event) {
        if(event.getCode().equals(KeyCode.ENTER)) {
            userCommand = userCommandBox.getText();
            //link with logic
            System.out.print(userCommand);
            userCommandBox.clear();
            //link with logic
            if(!userCommand.equals("")) {
                try {
                    returnCommand = ToDoList.implement(userCommand);
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                showBox.appendText(returnCommand + "\n" + "\n");
            }
        }
    }
});
```

The userCommand is used to get the commands typed into the text box, and the returnCommand is used to get the feedback messages from the Logic component.

5.2 Logic

Logic are used to process the commands passed from the UI component. Your work is to add more actions in the Logic component and add some parsing methods in the Parser component to support the new functions. Here are some important codes which you can refer to and make improvements on:

Codes for judging operations:

```
private static String modify(Storage s, String command, String parameter) throws
IOException {
    switch (command) {
        case "add": {
            return Action.addToList(s, parameter);
        }
        .....
    }
}
```

Some codes of it have been emitted, but when you want to add some new actions, just add new cases in the switch. Then, you can add new actions in Action.java and Parser.java.

6 Testing

Now we have not had various test methods, such as JUnit Test (White Box). We can only test the software directly in the System. Here is how you can test with typing in the text box.

Test function	Command in text box	Input Data	Expected Result
Add event	add CS2101 tomorrow	String: "add CS2101 tomorrow"	Return "Event added successfully" and add an event in the data file
Show event	show	String: show	1. CS2101 tomorrow
Delete event	delete 1	String: delete	Return "Delete successfully" and delete the event in the data file
Undo operation	undo	String: undo	Return "Undo operation successfully" and undo the last operation
Search event	search CS2101	String: search CS2101	1. CS2101 tomorrow
Update event	update 1 updated CS2101 10.12.2015	String: update 1 updated CS2101 10.12.2015	Return "Event updated successfully" and change the event in the data file as "CS2101 10.12.2015"
Redo operation	redo	String: redo	Return "Redo operation successfully" and redo the last undo operation
Exit the software	exit	String: exit	Exit the software

7 Appendix - API Library

7.1 ToDoList API

return type	description
static String	initialize() initialize the storage file and return welcome message
static String	implement() handle the commands got from UI and return the corresponding messages
static ArrayList<String>	getFloatingEvents() get the list of floating events
static ArrayList<String>	getDeadlineEvents() get the list of events with deadlines
static ArrayList<String>	getDetailedEvents() get the list of events with start time and end time

7.2 Action API

return type	description
static String	addToList(Storage s, String parameter) add the parameter into the storage
static String	show(Storage s, String parameter) get the event list from the storage and pass it to logic
static void	exit() exit the software
static String	searchKey(Storage s, String parameter) get the result of searching for certain parameter
static String	update(Storage s, String parameter) update a certain event and save it into the data file
static String	deleteEvent(Storage s, String parameter) delete a certain event and delete it from the date file

static String	undo(Storage s) undo the last operation
static String	redo(Storage s) redo the last undo operation

7.3 Event API

return type	description
static String	getDetail() get the details of the event
static String	getComment() get the comment of the event
static String	getStatus() get the status description of the events
static String	getPriority() get the priority of the event
Deadline	getDeadline() get the deadline the event
EventTime	getEventTime() get the start and end time of the event

7.4 Parser API

return type	description
static String	getAction(String userCommand) parse the command and return what the action is
static String	getParameter(String userCommand) parse the command and get the parameter of the pending event

7.5 Storage API

return type	description
void	save(ArrayList<String> arr) save the arrayList into the main directory

void	saveAs(ArrayList<String> arr, Path dir) save the arrayList into the main directory also change main directory to the new Path
ArrayList<String>	load() read file from the main directory
ArrayList<String>	load(Path dir) read file from the specified directory
ArrayList<String>	loadAs(Path dir) read file from the specified directory also change main directory to the new Path
Path	mainDir() return the directory for the main file
Path	tempDir() return the directory for the temporary file