

TIC2002

# INTRODUCTION TO SOFTWARE ENGINEERING

Semester 1 - AY19/20

## Duke Project Report

November 18, 2019

---

|                 |                    |
|-----------------|--------------------|
| NAME            | Li Shihao          |
| MATRIX NO.      | A0165362E          |
| GITHUB USERNAME | asmaww             |
| EMAIL           | e0166067@u.nus.edu |

## User Stories

1. Duke task checklist supports multiple users to use the application;
2. As users who prefer faster entries, it would be a great fit for this kind of audience to be able to quickly note down and manipulate tasks using various commands, which are triggered by a few key strokes. It's faster than clicking buttons by mouse in GUI;
3. When the next time users start the program, Duke should still remember the tasks that users had left from last time;
4. There are different types of task that fit in different use cases;
5. There are commands that mark task status such as completion and provide views of list of tasks;
6. System should be fast, reliable and provide message to guide correct user behaviour.

## Non-functional requirements

1. Message from the system should be fun and intimate;
2. The system should be smart to guide the users with meaningful alert of what to do rather than just exit;
3. Minimum visual elements/clusters which create distraction for users want to be fast. The aesthetic aims for simplicity and tidiness.

## Level-1 The output Duke shows when launching the program

|   |  |
|---|--|
| <pre>Knock knock Σ √( . ∪ . ? )    ____    _ \ _ _     ____                / / _ \                &lt; __/   ____/ \__/_    \ \____            \      /     Hey! \      / here, living in a ...     pod... 🍷     Who is there summoning me?     li shihao     Welcome back, li shihao</pre> | <pre>Knock knock Σ √( . ∪ . ? )    ____    _ \ _ _     ____                / / _ \                &lt; __/   ____/ \__/_    \ \____            \      /     Hey! \      / here, living in a ...     pod... 🍷     Who is there summoning me?     Linus T     New master registered ^o^/</pre> |
|---|--|

Figure 1: Screenshot - Start Greeting Page Existing/New User

## Level-4 Describe the commands for adding different types of tasks

```

todo do laundry
Got it. I've added this task:
[T][x] do laundry
Now you have 11 tasks in the list.
deadline finish milk /by 2019-11-29
Got it. I've added this task:
[D][x] finish milk (by: Nov 29)
Now you have 12 tasks in the list.
event sleep after exam /at 2019-12-04 3:30-7
Got it. I've added this task:
[E][x] sleep after exam (at: Dec 04 03:30-07:00)
Now you have 13 tasks in the list.

```

Figure 2: Screenshot - todo, deadline, event

## Level-2 Describe the commands for listing tasks

```

Knock knock Σ √( . ∪ . ? )

----
| _ \ _ _ | | -----
| | | | | | | / / _ \
| | | | | | | < _ /
| ____ / \ _ _ | | \ \ ____ |
      \      /
Hey! \      / here, living in a ...
pod... 🏠
Who is there summoning me?
li shihao
Welcome back, li shihao
list
Here are the tasks in your list:
1.[T][x] read book Algorithms
2.[D][✓] return book (by: Feb 28)
3.[E][✓] project meeting (at: Oct 11 02:00-04:00)
4.[E][✓] project management (at: Dec 25 02:00-13:15)
5.[T][x] join sports club
6.[T][x] meeting at 5F when free
7.[D][✓] duke (by: Nov 17)
8.[D][x] exam (by: Dec 04)
9.[E][x] conf.call with customer (at: Nov 18 10:30-12:00)
10.[D][x] security report (by: Nov 20)

```

Figure 3: Screenshot - list

## Level-3 Describe the commands for marking/unmarking tasks as done.

```

11.[T][✓] do laundry
12.[D][×] finish milk (by: Nov 29)
13.[E][×] sleep after exam (at: Dec 04 03:30-07:00)
done 12
Nice! I've marked this task as done:
[D][✓] finish milk (by: Nov 29)
do 3
Noted! I've marked this task as incompletd:
[E][×] project meeting (at: Oct 11 02:00-04:00)

```

Figure 4: Screenshot - done, do + task no.

## Level-5 Describe what kind of errors Duke can handle

```

Command make coffee
😞 OOPS!!! I'm sorry, but I don't know what that means :-(
done finish milk
😞 OOPS!!! Please input a Task Number instead ~
done 0
😞 OOPS!!! Please input a valid Task No. ~
done 99
😞 OOPS!!! Please input a valid Task No. ~
done 12
🎉 Hooray! This task has already been marked done ~
do 1
😞 Yes you should! This task was not completed in the first place =_|||
deadline watch Joker
😞 OOPS!!! Separate content and date with " /by "
event see fashion week
😞 OOPS!!! Separate content and time block with " /at "
deadline watch movie /by 2019-13-40
😞 OOPS!!! Please input a date in format as " yyyy-mm-dd "
event watch movie /at 12-12 0-2
😞 OOPS!!! Please input a time in format as " yyyy-mm-dd time-time (24h)"
find eat banana
No matching task, dear ~

```

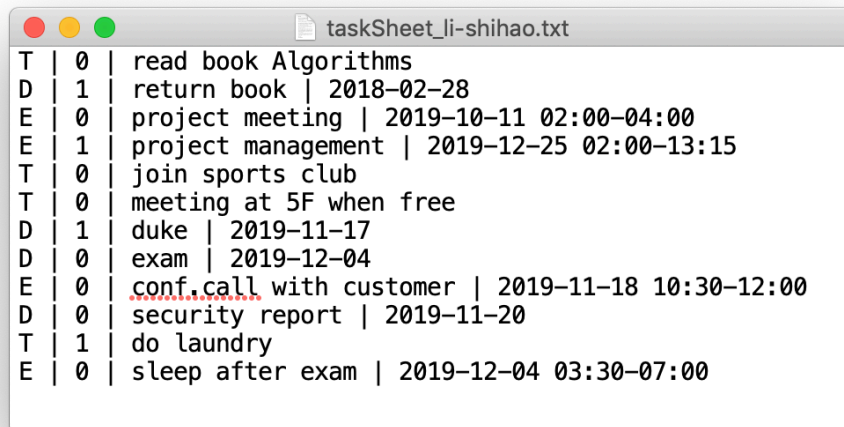
Figure 5: Screenshot - There are also error messages handling file access error

## Level-6 Describe the commands for deleting tasks

```
delete 0
  😞 OOPS!!! Please input a valid Task No. ~
delete 12
  Noted. I've removed this task:
  [D][✓] finish milk (by: Nov 29)
  Now you have 12 tasks in the list.
```

Figure 6: Screenshot - delete + task no.

## Level-7 Give a sample of the tasks as they are stored in the hard disk



| Type | ID | Description             | Date/Time              |
|------|----|-------------------------|------------------------|
| T    | 0  | read book Algorithms    |                        |
| D    | 1  | return book             | 2018-02-28             |
| E    | 0  | project meeting         | 2019-10-11 02:00-04:00 |
| E    | 1  | project management      | 2019-12-25 02:00-13:15 |
| T    | 0  | join sports club        |                        |
| T    | 0  | meeting at 5F when free |                        |
| D    | 1  | duke                    | 2019-11-17             |
| D    | 0  | exam                    | 2019-12-04             |
| E    | 0  | conf.call with customer | 2019-11-18 10:30-12:00 |
| D    | 0  | security report         | 2019-11-20             |
| T    | 1  | do laundry              |                        |
| E    | 0  | sleep after exam        | 2019-12-04 03:30-07:00 |

Figure 7: Screenshot - File taskSheet\_user-name.txt

## Level-8 Explain how Duke uses dates/times

Date is a property for Deadline type of task, time is property for Event type of task. Date can be use to sort tasks of Deadline type in chronological order.

## Level-9 Describe the commands for searching for tasks.

```
find re
```

```
Here are the matching tasks in your list:
```

- 1.[T][×] read book Algorithms
- 2.[D][✓] return book (by: Feb 28)
- 6.[T][×] meeting at 5F when free
- 10.[D][×] security report (by: Nov 20)

```
find project
```

```
Here are the matching tasks in your list:
```

- 3.[E][×] project meeting (at: Oct 11 02:00-04:00)
- 4.[E][✓] project management (at: Dec 25 02:00-13:15)

Figure 8: Screenshot - Search tasks (notice that the task no. is correct)

## Level-10 Individual feature: If you implemented an individual feature, describe that feature

Sort Deadline tasks in order that oldest date on the top:

```
sort
```

- [D][✓] return book (by: Feb 28)
- [D][✓] duke (by: Nov 17)
- [D][×] security report (by: Nov 20)
- [D][×] exam (by: Dec 04)

Figure 9: Screenshot - Deadlines sort

## Other features Describe other features you implement

Multiple users can use this program, as long as they remember their usernames, they could get back their tasks. After starting the program, first thing is to enter your username. If you are an existing user, duke will greet you with your username and you can list out all the tasks you had created since last time; If you are a new user, type your desired username, and next time use it to log in to the system.

## A-MoreOOP Give a class diagram to match your code

Refer to Appendix *Figure13* ↔ *click*

## A-MoreOOP Give at least one object diagram illustrating the state of your program at some point

Refer to Appendix *Figure14* ↔

## A-MoreOOP Give at least one sequence diagram illustrating an object interaction in your product

Refer to Appendix *Figure15* ↔

## A-JavaDoc: Give at least 2 javadoc comments from you code

```
/**
 * HashMap Stores all the keywords and their respective function
 * Upon matching Key with correct format (Exception handles the rest, calling Message ui)
 * Value is Lambda expression method will run automatically
 * Values are all <Command()> Interface with run() method |
 * Updating the TempTaskList first if necessary and then write to the text file storage
 *
 * @param list
 * @param file
 */
public CommandList(TempTaskList list, Storage file) {
    keywords = new HashMap<String, Command>();
    ui = new Message();

    keywords.put("bye", new Command() {
        public void run(String content) {
            // Do absolutely nothing
        }
    });
}

/**
 * This method will be called when keyword "done" is matched
 * It sets the status of a given task (by task no.) to completed (true)
 * It also catches many Exception to show error msg
 * such as the String followed bt "done":
 * is not an valid task number
 * is not number at all
 * the task is already completed
 * Lastly it call function to print the message which task is done
 * @param content
 * @param list
 * @throws Exception
 */
private void cmdMarkDone(String content, TempTaskList list) throws Exception {
    try {
        int listIndex = Integer.parseInt(content) - 1;
        if (listIndex < 0 || listIndex > list.size()) {
            throw new IndexOutOfBoundsException();
        }
        if (list.get(listIndex).getCompleted()) {
            ui.doneAlreadyMessage();
        } else {
            list.get(listIndex).setCompleted();
            printMarkDone(list, listIndex);
        }
    } catch (NumberFormatException e) {
        ui.doneTaskNoMessage();
    } catch (IndexOutOfBoundsException e) {
        ui.doneValidTaskNoMessage();
    }
}
```

Figure 10: Screenshot - JavaDoc snippets

## A-JUnit: Give 2-3 JUnit test methods from your code

```
package duke.parse;

import duke.task.Task;
import duke.task.TODO;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class ParserTest {
    Task test = new TODO( content: "to do a new one");

    @Test
    void taskToText() {
        assertEquals( expected: "T | 0 | to do a new one", new Parser().taskToText(test));
    }
}

ParserTest

✓ Tests passed: 1 of 1 test – 40 ms

/Library/Java/JavaVirtualMachines/jdk-11.0.4.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

```
package duke.command;

import ...

class CommandListTest {

    String test = "todo This is a sentence";
    String[] expect = {"todo", "This is a sentence"};

    @Test
    void splitKeywordTest() throws IOException, DukeException {
        assertEquals(expect, new CommandList(new TempTaskList(), new Storage()).splitKeyword(test));
    }
}

CommandListTest

✓ Tests passed: 1 of 1 test – 43 ms

/Library/Java/JavaVirtualMachines/jdk-11.0.4.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

Figure 11: Screenshot - JUnit



## A-Assertions: Give at least 2 code segments that contain assertions you added to your code

```

public void read(TempTaskList to) throws FileNotFoundException {
    Scanner s = new Scanner(file);
    while (s.hasNext()) {
        String[] linewords = Parser.fileLineBreak(s.nextLine());
        assert linewords.length < 2;
        String taskType = linewords[0].trim();
        String isCompleted = linewords[1].trim();
        String content = linewords[2].trim();

        if (taskType.equals("T")) {
            Todo task = new Todo(content);
            if (isCompleted.equals("1"))
                task.setCompleted();
            to.add(task);
        }

        String username = in.nextLine();
        String filename = Parser.convertFileName(username);
        assert !filename.equals("") : "empty filename no good";

        try {
            file = new Storage(filename);
            if (file.get().createNewFile()) {
                ui.newUser();
            } else {
                ui.existingUser(username);
            }
        }
    }
}

```

Figure 12: Screenshot - Assertion

## Suggested test commands Give a list of commands a tester can execute in sequence to examine your product. Cover all features in a reasonable order:

|   |            |
|---|------------|
| li shihao                                 | do 1       |
| list                                      | done 1     |
| hello world                               | do 1       |
| todo read reports                         | sort       |
| deadline mark paper /by 2019-12-24        | find money |
| event marathon /at 2019-12-25 10:15-15:30 | find exam  |
| event firework party /at 2020-01-01 0-4   | delete 12  |
| delete 99                                 | list       |
| done 0                                    |            |
| done 2                                    | bye        |

# Appendices

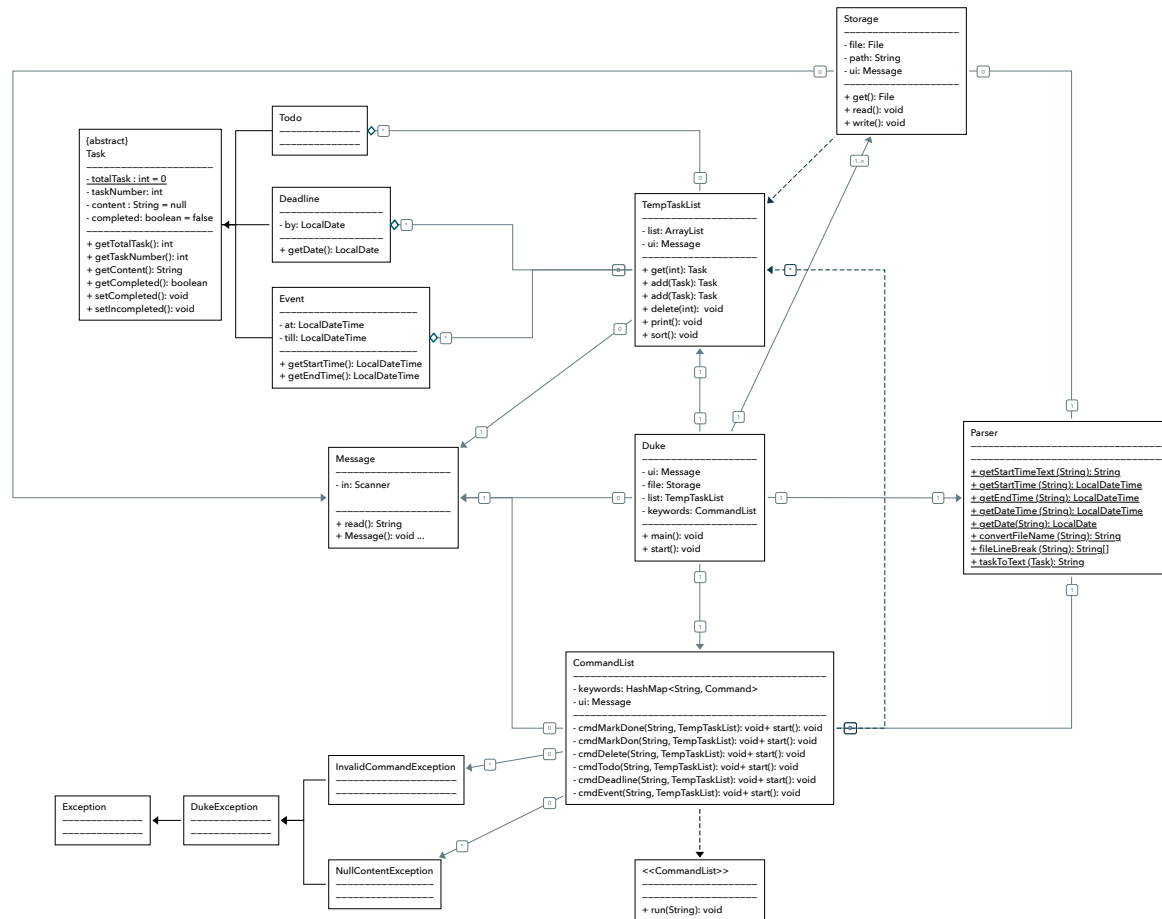


Figure 13: Screenshot - Class Diagram

When just started the program, a new Duke object will be initiated to call the start() method. Duke initiates Message ui, Storage file, TempTaskList list, and pass file and list as parameters to initiate CommandList keywords.

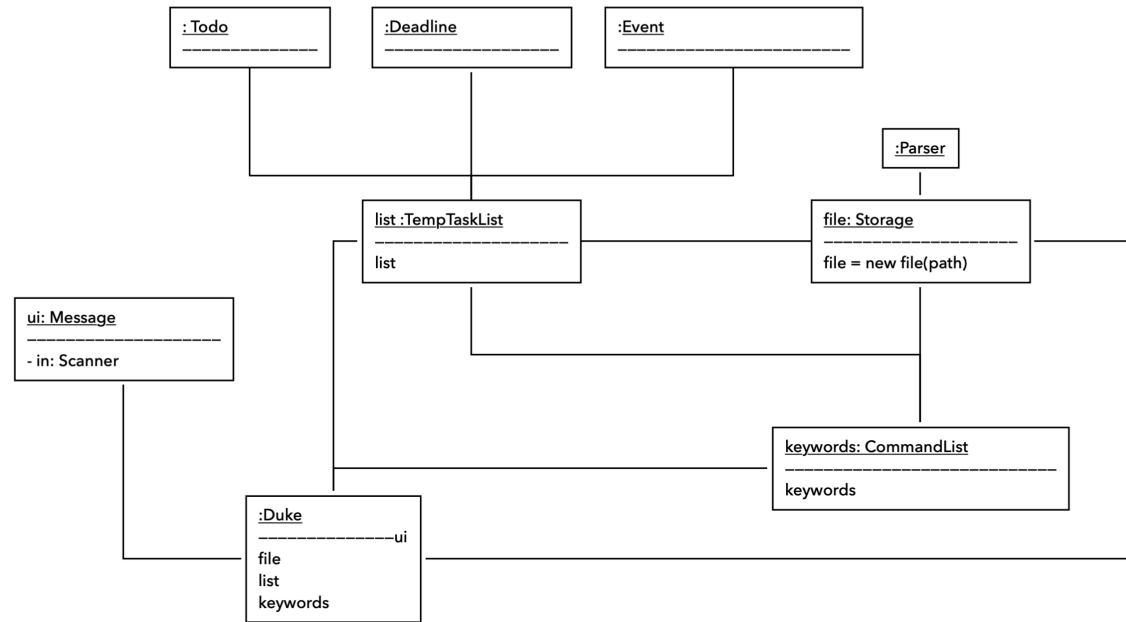


Figure 14: Screenshot - Object Diagram

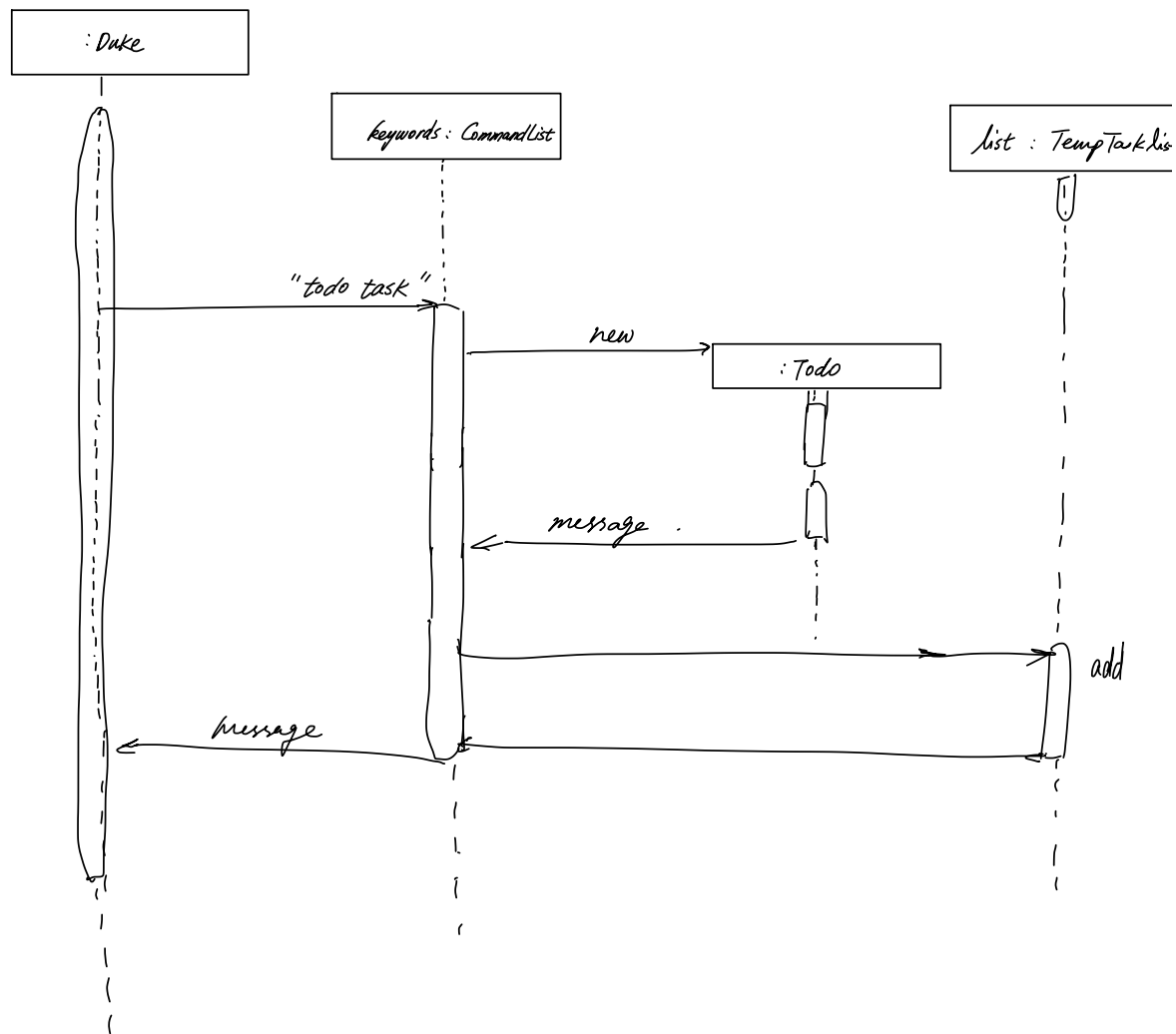


Figure 15: Screenshot - Sequence Diagram