Duke$$$

An NUS Software Development Project

By: AY1920S1-CS2113T-F09-1

Since: August 2019

Contents

1. Introduction

DUKE$$$ is designed for exchange students who prefer to use a desktop app for managing their finances during their period of exchange.

DUKE$$$ extends from getting users' current financial status to calculating and checking their day-to-day expenditure. On top of that, it offers the user a wide range of functionalities to manage their expenses.

DUKE$$$ is optimized for those who prefer to work with a Command-line Interface(CLI) while still having the benefits of a Graphical User Interface.

Jump to section 2, "Quick Start" to get started. Enjoy!

2. Quick Start

1. Ensure you have Java `11` or above installed in your Computer.
2. Download the latest `Duke$$$.jar` here.
3. Copy the file to the folder you want to use as the home folder for your finance tracking.
4. Double-click the file to start the app. The CLI should appear in a few seconds.
5. Type the command in the command box and press `Enter` to execute it.
   e.g. typing `help` and pressing `Enter` will open the help window.
6. Some example commands you can try:
   - `[status]`: Shows the current balance in the account
   - `[out] $5.00 /date 2019-10-31 /tags food` adds an entry of `food` of value `5` to the memory
   - `[delete] 3`: deletes the 3rd entry on the current day's list
   - `[bye]`: exits the app
7. Refer to Section 4, "Commands" for details of each command.

3. Features

This section displays the features that you can expect from **DUKE$$$**

Expenses System

- Adds records of your expenses and income.
- Manage these records by editing or deleting them.
- View your expenses categorically.
- Include recurring expenses

{In Progress….}

4. Commands

This section shows the various commands that the user can use to access the different components of the application and features of **DUKE$$$**

Command Format

- Words in [...] are the commands used by the user to invoke the action. For example :

  `expendedmonth`

- Words in `<...>` are the parameters to be supplied by the user. For example :

  `expendmonth <month> /year <year>`

4.1. Viewing help: `help`

Lists out all the commands available for the user to interact with the application

Format: **`[help]`**

4.2. Financial Status: `status` `(TBC)`

Gives the current account balance and the total expenditure thus far

Format: **`[status]`**

Examples: `[status]`

- `balance 2000`
- `expenditure 500`

4.3. Currency Converter: `convert`

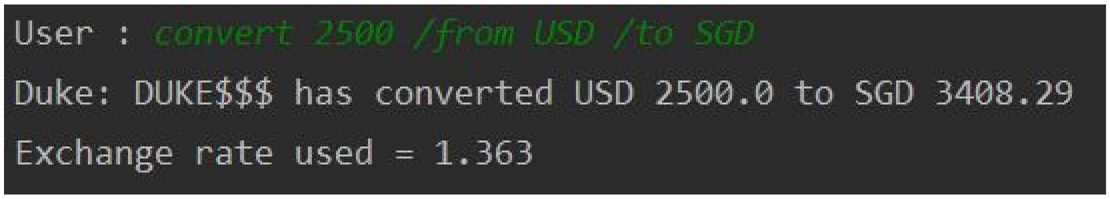
Converts user-specified amount from base currency to required currency. The base currency and required currency are given in ISO standard unique 3 character country code.

Base currency country code : BCC

Required currency country code : RCC

Format: **`convert <amount> from/<BCC > to/<RCC>`**

Example: **`convert 2500 /from USD /to SGD`**



```
User : convert 2500 /from USD /to SGD
Duke: DUKE$$$ has converted USD 2500.0 to SGD 3408.29
Exchange rate used = 1.363
```

Figure 4.3.1. Converting Currency between two countries

4.4. Calculator: `cal` (TBC)

Performs basic arithmetic operations on user inputs. add → Addition, sub → Subtraction, mul → Multiplication, div → Division.

Format: **`[cal add] <x y>`**

`[cal sub] <x y>`

`[cal mul] <x y>`

`[cal div] <x y>`

Examples:

- `[cal div] 15 5`
  15 is divided by 5 to return 3
- `[cal add] 5 10 10`
  Adds 5 10 and 10 to return 25

4.5. Total Expenditure

Provides the expenditure according to categories: `Day, Week, Month, Year`

4.5.1 Expenditure for the: `day` (TBC)

Shows the total expenditure for the day.

Format: **`[expendedday]`**

Example: `expendedday`

Outputs the total amount of money spent on that day. I.e. 12.50

4.5.2 Expenditure for the: `week` (TBC)

Shows the total expenditure for the day.

Format: **`[expendedweek]`**

Example: `expendedweek`

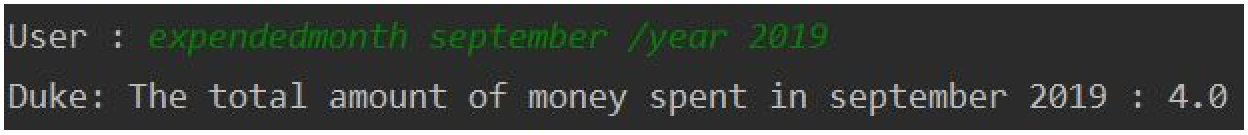
Outputs the total amount of money spent in that week. I.e. 50.50

4.5.3 Expenditure for the: `month`

Shows the total expenditure for the day.

Format: **`[expendedmonth] <month> [/year] <year>`**

Example: `expendedmonth september /year 2019`



```
User : expendedmonth september /year 2019
Duke: The total amount of money spent in september 2019 : 4.0
```

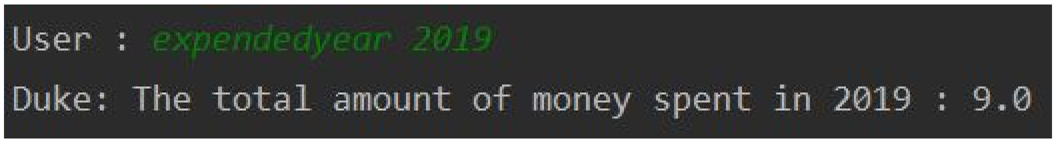
Figure 4.5.3.1. Getting the expenditure of a month

4.5.4 Expenditure for the: `year`

Shows the total expenditure for the day.

Format: **`[expendedyear] <year>`**

Example: `expendedyear 2019`



```
User : expendedyear 2019
Duke: The total amount of money spent in 2019 : 9.0
```

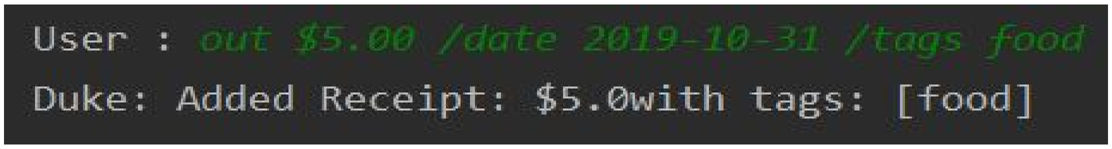
Figure 4.5.4.1. Getting the expenditure of a year

4.6. Add an expenditure entry: `out`

Adds an entry to the list as an expenditure.

Format: **`[out] <amount> [/date] <yyyy-mm-dd> [/tags] <tag>`**

Example: `out $5.00 /date 2019-10-31 /tags food`



```
User : out $5.00 /date 2019-10-31 /tags food
Duke: Added Receipt: $5.0with tags: [food]
```

Figure 4.6.1 Adding an expenditure entry

4.7. Search for an entry: `find` (TBC)

Finds entry descriptions that contain any of the keywords and returns the date and the amount of money spent. It is NOT case sensitive.

Format: **`[find] <keyword>`**

Examples:

- `find transport`
  Returns `transport 3.50 18/9/2019`
- `find movie`
  Returns all the entries that contain the keyword movie and the money spent and the date as well.

4.8. Deleting an entry: `delete` (TBC)

Deletes the specified entry based on the `index`, the number shown in the displayed entry list. The index must be a positive integer.

Format: **`[delete] <index>`**

Examples:

- `delete 2`
  Deletes the 2nd entry in the list.

- `find transport`
  `delete 1`
  Deletes the 1st entry in the results of the `find` command.

4.9. Clearing all entries: `clear`

Clears all entries from the expenditure list.

Format: **`[clear]`**

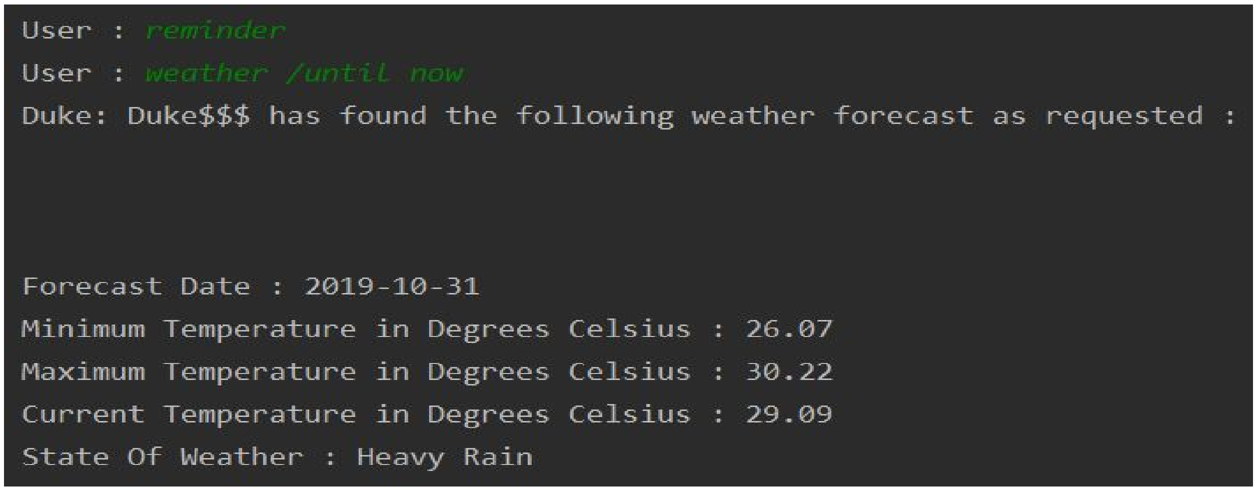
4.10. Display Weather status: `weather`

User is able to request for three different weather status based on the period entered.

- Current instant weather status only by entering : now
- Current instant weather status and the weather forecast for the next day by entering : tomorrow
- Current instant weather status and the weather forecast for the next four days by entering : later

Format: **`weather /until <period>`**

Example 1: **`weather /until now`**



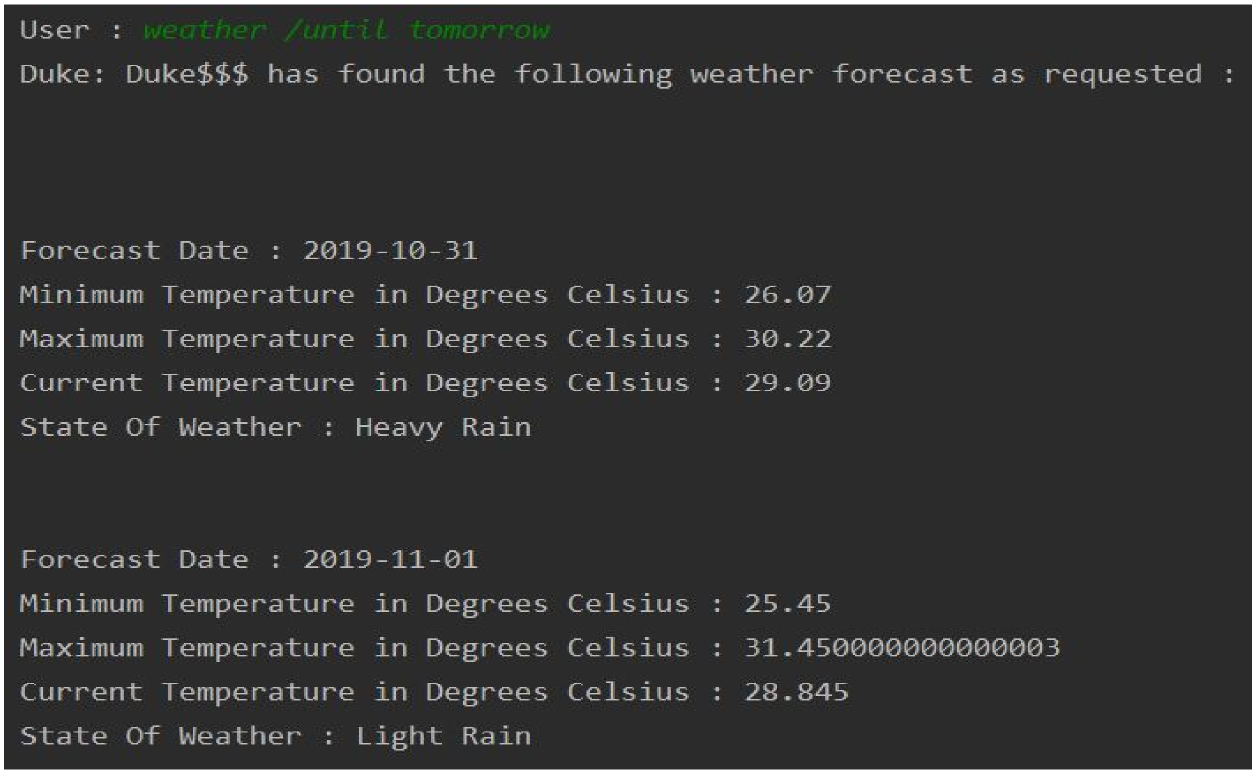
```
User : reminder
User : weather /until now
Duke: Duke$$$ has found the following weather forecast as requested :



Forecast Date : 2019-10-31
Minimum Temperature in Degrees Celsius : 26.07
Maximum Temperature in Degrees Celsius : 30.22
Current Temperature in Degrees Celsius : 29.09
State Of Weather : Heavy Rain
```

Figure 4.10.1 Weather Status Display of current instant

Example 2: ***weather /until tomorrow***



```
User : weather /until tomorrow
Duke: Duke$$$ has found the following weather forecast as requested :



Forecast Date : 2019-10-31
Minimum Temperature in Degrees Celsius : 26.07
Maximum Temperature in Degrees Celsius : 30.22
Current Temperature in Degrees Celsius : 29.09
State Of Weather : Heavy Rain


Forecast Date : 2019-11-01
Minimum Temperature in Degrees Celsius : 25.45
Maximum Temperature in Degrees Celsius : 31.450000000000003
Current Temperature in Degrees Celsius : 28.845
State Of Weather : Light Rain
```

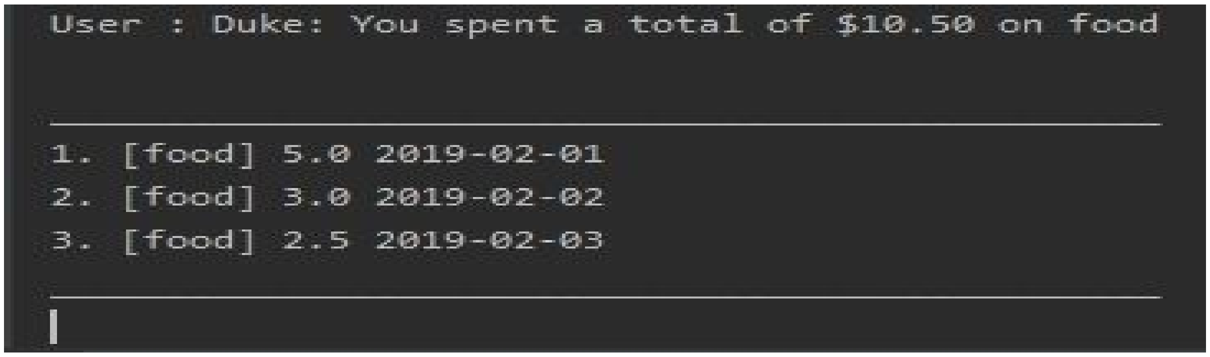
Figure 4.10.2 Weather Status Display until tomorrow

4.11. Listing entries based on tag: `taglist`

Lists all the receipts corresponding to the keyword/tag that user entered and shows the total expenditure for those receipts as well

Format : **`[taglist] <tag>`**

Example: taglist food



```
User : Duke: You spent a total of $10.50 on food

____________________________________________________________
1. [food] 5.0 2019-02-01
2. [food] 3.0 2019-02-02
3. [food] 2.5 2019-02-03
____________________________________________________________
```

Figure 4.11.1 List entries based on tag

4.11. Exiting the program: `bye`

Exits the program.

Format: `[bye]`

4.12. Saving the data

Expenditure data entries are saved in the hard disk automatically after any command that changes the data

5. Future Enhancements [coming in v2.0]

Our current implementations have many usabilities, however, it does have many more features that could be added to the application to increase the user experience. The possible enhancements include :

1. Allowing the user to have a chatting option with others using the application.
2. Encrypting data files to increase the security of personal data.
3. Allowing the user to directly update the application if they do online shopping or use PayLah kind of applications for payments.
4. Syncing with the user’s bank account to provide more monitoring.

6. FAQ

Q: How do I transfer my data to another Computer?

A: Install the app in the other computer and overwrite the empty data file it creates with the file that contains the data of your previous DUKE$$$ folder.

Q: Where is my data stored?

A: It is stored in the /main directory.

7. Command Summary

Identifier	Description	Format	Example
`BALANCE`			
`BLANK`			
`BYE`	Exits the program	`[bye]`	`bye`
`CONVERT`	Converts user-specified amount from base currency to required currency. The base currency and required currency are given in ISO standard unique 3 character country code. Base currency country code : BCC Required currency country code : RCC	`convert <amount> from/<BCC> to/<RCC>`	`convert 2500 /from USD /to SGD`
`DELETE`	Deletes the chosen entry	`[delete] <index>`	`delete 3`
`EXPENDEDMONTH`	Gives the total amount of money expended in the month	`[expendedmonth] <month> [/year] <year>`	`expendedmonth march /year 2019`
`EXPENDEDYEAR`	Gives the total amount of money expended in the year	`[expendedyear] <year>`	`expendedyear 2019`
`EXPENSES`			
`FIND`			
`HELP`	Gives a description of all the commands available for the user to	`[help]`	`help`

	use.		
IN	Takes in an entry that indicates the income.	`[in] <amount> [/date] <yyyy-mm-dd> [/tags] <tag>`	`in $5.00 /date 2019-10-31 /tags food`
LIST	Lists out all the entries in the receipttracker.	`[list]`	`list`
TAGLIST	Lists out all the entries corresponding to that specific tag that user inputs	`[taglist]`	`taglist transport`
OUT	Takes in an entry which indicates the expenditure.	`[out] <amount> [/date] <yyyy-mm-dd> [/tags] <tag>`	`out $5.00 /date 2019-10-31 /tags food`
QUEUE			
REMINDER	Displays the tasks which are due on the current date.	`reminder`	`reminder`
SETBALANCE			
TAGLIST			
VIEWSCHEDULE			
WEATHER	Displays weather forecast based on the period requested by user.	`weather /until <period>`	`weather /until later`