MooMooMoney - User Guide

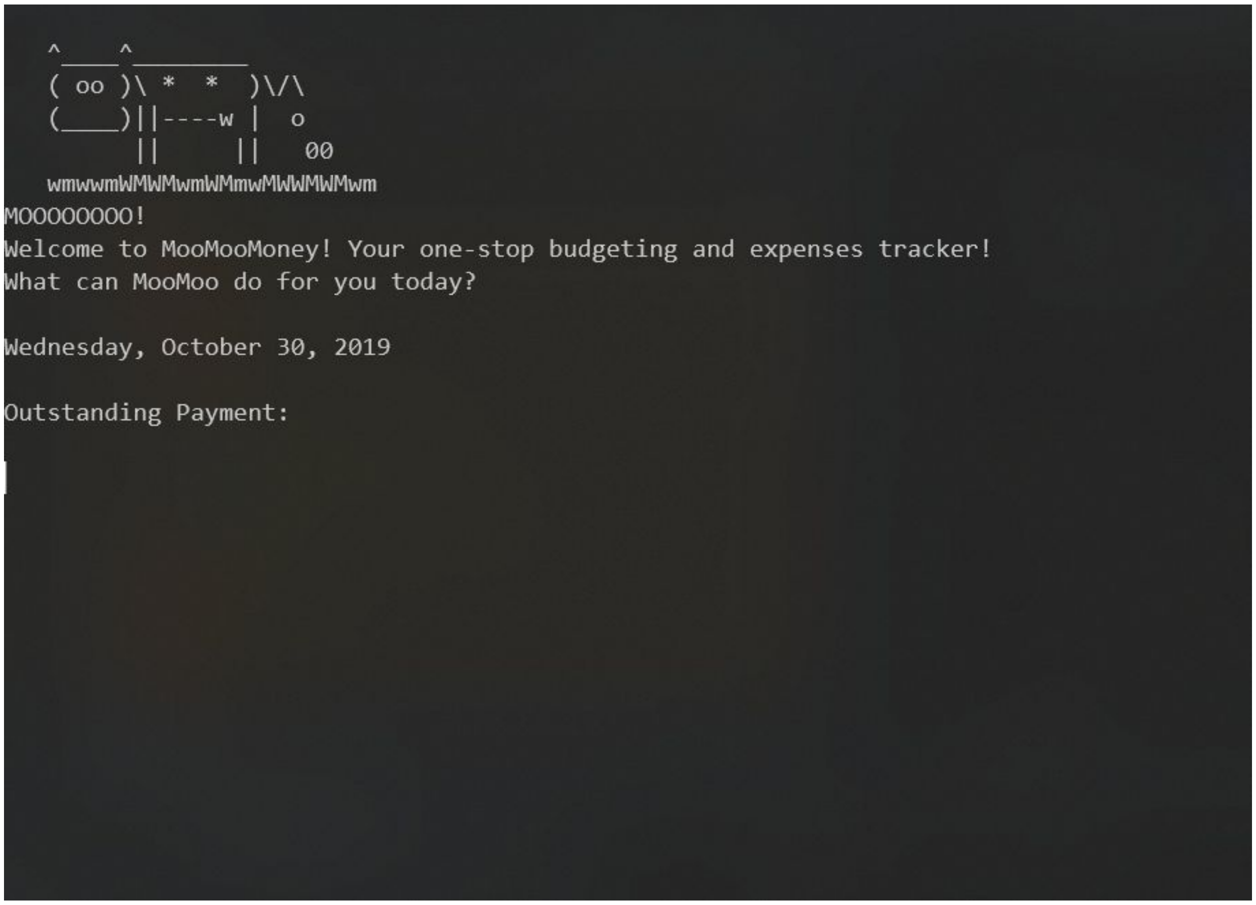
By: `AY1920S1-CS2113T-F14-1`   Since: `Oct 2019`

1. Introduction

MooMooMoney (MMM) is for those who prefer to use a desktop app for managing monetary expenditure. More importantly, MMM is optimized for those who prefer to work with a Command Line Interface (CLI). If you can type fast, MMM can help you with your money management faster than traditional GUI apps. Interested? Jump to the 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 `MooMooMoney.jar` here. (Link will be updated)
3. Copy the file to the folder you want to use as the home folder for MooMooMoney.
4. To run the file, the application should be run using the provided terminal (Linux/Mac) or Command Prompt/Powershell (Windows) using “java -jar FILENAME.jar”



```
   ^____^________
   ( oo )\ *  *  )\/\
   (____)||----w |  o
         ||     ||   00
   wmwwmWMWMwmWMmwMWWMWMwm
MOOOOOOOO!
Welcome to MooMooMoney! Your one-stop budgeting and expenses tracker!
What can MooMoo do for you today?

Wednesday, October 30, 2019

Outstanding Payment:
```

5. Some example commands you can try:
    - `moo` : makes the cow go “Mooooo”.
    - `category c/Food n/Drinks:` Change the category “`Food`” to “`Drinks`”
    - `add c/Shopping d/Nike Shoes a/50:` Add the entry `“Nike Shoes”` of the price `“$50”` to the `“Shopping”` category.
    - `bye` : exits the app
6. Refer to Section 3, “Features” for details of each command.

3. Features

Command Format

- Words in `UPPER_CASE` are the parameters to be supplied by the user e.g. in `c/CATEGORY`, `CATEGORY` is a parameter which can be used,e.g `category c/CATEGORY`
- These words are preceded by an `identifier/`, which allows the program to understand the user's input, e.g in `c/CATEGORY`, `c/` is the identifier which lets the program know that the `UPPER_CASE` parameter that follows is a category type. Other examples include:

  - `d/` (date)
  - `n/` (description)
  - `b/` (budget)
  - `a/` (amount)

- The order of the commands does not matter, e.g if the command specifies `n/DESCRIPTION a/AMOUNT`, `a/AMOUNT n/DESCRIPTION` will also give the same result
- Words in `[]` are optional fields that users can choose to fill

3.1. Viewing help : `help`

List all valid commands.

Format: `help`

- Shows a list of all commands a user can enter and gives a brief description of what each command does, as well as an example using it.

3.2. Edit categories: `category`

Edit Categories - Change category name.

Format: `category c/CATEGORY n/NEW_NAME`

- Replace the current `CATEGORY` name with `NEW_NAME`

Examples:

- `category c/Food n/Drinks`
- `category c/Shop n/Shopping`

3.3. Remove categories : `remove`

Delete a category.

Format: `remove c/CATEGORY`

- All expenditure under deleted `CATEGORY` will be deleted as well
- If deleted `CATEGORY` does not exist, the user will be prompted with a list of categories available to delete

Examples:

- `remove c/Food`
- `remove c/Shopping`

3.4. Manage expenditure : `add`

Add an entry for expenditure - what the user has spent money on.

Format: `add c/CATEGORY n/DESCRIPTION a/AMOUNT`

- If `CATEGORY` does not exist, the system automatically adds the category
- `CATEGORY` is case insensitive

Examples:

- `add c/Food n/Laksa a/5.50`
- `add c/Shopping n/Nike Shoes a/50`

3.5. Manage budget: `budget`

Set budget amount for category.

Format: `budget set c/CATEGORY b/BUDGET`

- `CATEGORY` must be an existing category.
- The budget must be a numerical number with no other symbols and at most 2 decimal places.
- Budget should be placed after a category.
- Multiple categories can be added by specifying more pairs of c/ and b/.
- *CATEGORY* is case insensitive.
- *CATEGORY* should not have a budget set.

Examples:

- `budget set c/Food b/1000 c/Shoes b/500`
- Sets budget amount to be `1000` for the `food` category and budget amount to be `500` for the `shoes` category. Budget amount and category will be displayed.

- `budget set c/Food b/800`
- If budget has already been set, an error will be displayed.

Edit budget amount for a category.

Format: `budget edit c/CATEGORY b/BUDGET`

- `CATEGORY` must be an existing category.
- The budget must be a numerical number with no other symbols.
- `BUDGET` should be placed after a `CATEGORY`.
- Multiple categories can be added by specifying more pairs of c/ and b/.

Examples:

- `budget edit c/Food b/750 c/Shoes b/250`
- Change budget amount from previous amount to `750` for the `food` category and budget amount from previous amount to `250` for the `shoes` category. Budget amount and category changed will be displayed.

- `budget edit c/laptop b/100`
- If budget has not been set, an error will be displayed.

List currently set budget.

Format: `budget list [c/CATEGORY]`

- `CATEGORY` must be an existing category.
- If `CATEGORY` is not specified, budget for all categories will be listed.

Examples:

- `budget list`
- Lists the `budget` for all categories set with a `budget`.

- `budget list c/Food c/Shoes`
- Lists the `budget` for the `food` and `shoes` category.

View the savings for each category

Format: `budget savings [c/CATEGORY] s/STARTMONTHYEAR [e/ENDMONTHYEAR]`

- `CATEGORY` must be an existing category.
- `STARTMONTHYEAR` and `ENDMONTHYEAR` should be a month and year value in this format: `02/2019` (February 2019). `ENDMONTHYEAR` should be a month and year after `STARTMONTHYEAR`.
- If there are no expenditures in the category, savings will equal budget.

Examples:

- `budget savings c/Food s/02/2019 e/10/2019`
- Views the total `savings` (budget set for that category - total expenditure of that month) for the `food` category from `February 2019` to `October 2019`.

- `budget savings s/01/2019`
- Views the total savings (budget set – total expenditure) for all categories for January 2019.

3.6. View Expenditure : `view`

View the list of transactions based on category or all transactions.

Format: `view [CATEGORY]`

- View transactions within the `CATEGORY` specified
- View all transactions if `CATEGORY` not specified
- Search case for `CATEGORY` insensitive. E.g `food` will display `Food` expenditures

Examples:

- `view FOOD`
  Displays list of expenditure spent on `FOOD`
- `view`
  Displays list of all expenditures

3.7. Delete an entry : `delete`

Delete an entry in a category.

Format: `delete c/Category INDEX`

- User selects the entry to be deleted by indicating the entry number in `INDEX`
- A prompt (y/n) will appear to confirm the execution of the command

Examples:

- `delete c/Food 2`
  Delete the second transaction in the food category

3.8. Display data as a graph: `graph`

Display data in a visual graph format.

Format: `graph c/CATEGORY d/MONTH`

- User selects the category and month to be displayed as a bar graph.
- The parameter `MONTH` is to be entered as a number from 1 - 12, corresponding to each of the 12 months of the year (i.e `1` = January, `2` = February, etc)

- Entering `graph total` instead displays a bar graph for all categories instead of expenditure of a particular category.

Examples:

- `graph c/Food d/1`
  Displays a bar graph of all the expenditure in the category `Food` for the month of January

3.9. Setting a scheduled payment : `schedule`

Set a payment or bill in advance.

Format: `schedule d/DATE a/AMOUNT n/DESCRIPTION`

- User sets the payment in advance by providing a DATE, AMOUNT and DESCRIPTION input.
- The DATE parameter should be entered in the format dd/mm/yy. (i.e 31st October 2019 = 31/10/19)
- The AMOUNT parameter must be entered as a number (i.e 4, 3.30, 1050, etc)
- The DESCRIPTION parameter allows spaces (i.e pay bills, bills)
- Once a scheduled payment has been set, the user will be prompted when he runs the app on that day automatically. The program should display user's due payment in the following format -
  "Outstanding Payments :
  Pay bills"

Examples:

- `schedule d/01/11/19 a/50 n/pay school fees`
  Set a payment of $50 to pay school fees on 1st November 2019
- `schedule d/23/01/20 a/102.50 n/electricity bill`
  Set a payment of $102.50 to pay electricity bill on 23rd January 2020

3.10. Exiting the program : `exit`

Exits the program.

Format: `exit`

3.11. Saving the data

All data are saved automatically to a file on the hard disk after any command that changes data is ran. As such, there is no need to do any manual saving.

3.12. Calculator `[coming in v2.0]`

A calculator will be available in the application to allow users to calculate their expenditure or savings without requiring an external program.

Format: `calculator`

3.13. Additional Feature : `moo`

Program will return a response “moo” back to user.

Format: `moo`

4. FAQ

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

A: A folder called **data** will be created in the same location as the jar file, transfer the folder to another computer in the same location as the jar file.

Q: Is it possible to edit the data that is saved?

A: The budget is saved to **data/budget.txt** in the format of CATEGORY | BUDGET. The values can be edited to your liking so long as it fits the format.

Q: Is an internet connection required?

A: No, all files are stored locally on the hard disk and as such, it is important that you backup the files as necessary.

5. Command Summary

- Help : `help`
- Edit categories : `category`
  e.g. `category c/Food n/Drinks`
- Remove categories : `remove`
  e.g. `remove c/CATEGORY`
- Manage Expenditure : `add c/CATEGORY n/DESCRIPTION a/AMOUNT`
  e.g. `add c/Food n/Nasi Lemak a/3`
- Manage Budget : `budget a/AMOUNT d/DEADLINE`
  e.g. `budget a/500 d/31/10/2019`
- View Expenditure : `view c/CATEGORY`
  e.g. `view c/FOOD`
- Delete Expenditure : `delete c/CATEGORY INDEX`
  e.g `delete c/Food 2`
- Displaying Graph: `graph c/CATEGORY d/MONTH`
- Schedule Payments : `schedule d/DATE a/AMOUNT n/DESCRIPTION`
- Exiting the program : `exit`
- Additional Feature : `moo`