Duke$$$

An NUS Software Development Project

By: AY1920S1-CS2113T-F09-1

Since: August 2019

Contents

1. Introduction

Welcome to Duke$$$!

For new users, you may choose to jump to Section 2, “Quick Start”, to quickly get started. Enjoy!

1.1. About Duke$$$

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.

1.2. About this User Guide

This user guide aims to guide new users to quickly and easily get started with Duke$$$ while offering veteran users with a comprehensive and detailed documentation of all the various features that Duke$$$ has to offer.

For ease in navigation, you may use the table of contents above to quickly transport you to the page containing the information that you seek.

For ease in communication, this document will use the following markups:

- `command`
  A grey highlight with black text indicates that this is a command that may be run on Duke$$$ command line

2. Quick Start

We have outlined a few necessary steps to ensure that your first experience with Duke$$$ will be a smooth one. These steps are:

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 Graphical User Interface should appear in a few seconds.

Congratulations, you have successfully set up Duke$$$ on your local device and are ready to begin responsibly tracking your finances.

To get started, type in a command into the command line and press `Enter` to execute it.

Below are some example commands you can try:

- **`help`**: Shows the current balance in the account
- **`out $5.00 /date 2019-10-31 /tags food`**: Adds a receipt with a value of 5, registered on the date 31/10/2019 and tagged with the tag “food”
- **`delete 3`**: deletes the 3rd entry on the current day’s list
- **`bye`**: exits the app

Refer to Section 4, “Commands” for the full details of each command.

3. Features

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

Task System

- Add Events, Deadlines or ToDos to your Task Tracker

Expenses System

- Add records of your expenses and income
- Manage these records by editing or deleting them
- View your expenses categorically
- Convert money from one currency to another

Export System

- Easily export your data into a CSV file

Bonus

- Check the Weather Forecast for today

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>`

Notes

- Duke$$$ has **intentionally** been optimized to interpret user input to the best of its ability. This means that so long as the Command is properly spelled and any required inputs come after it, Duke$$$ will still be able to interpret and execute the desired command.

  This was done to remove the frustration of typing in a long command only to have it fail because of a missing whitespace. For more on the design considerations and the implementation, please refer to our Developer Guide where we discuss in-depth on why such a decision was made. Nonetheless, it is recommended to follow the format stated for the best performance.

  Example:

  Consider the input, `out $5.00 /date 2019-10-31 /tags food`, which adds a receipt with a value of $5, registered on the date 31/10/2019 and tagged with the tag “food”.

  Typing the following will also produce the same result:

  no whitespace
  `out$5.00/date2019-10-31/tagsfood`

  rearranging flags
  `out $5.00 /tags food /date 2019-10-31`

  random capitalization of command
  `OuT $5.00 /date 2019-10-31 /tags food`

random preceding input
`xyzabout $5.00 /date 2019-10-31 /tags food`

random flags
`xyzabout $5.00 /date 2019-10-31 /random ??? /tags food`

However, typing in the following commands will not yield the same result:

random capitalization of flags
`out $5.00 /dATe 2019-10-31 /tAGS food`

Misspelling the command
`ouut $5.00 /date 2019-10-31 /tags food`

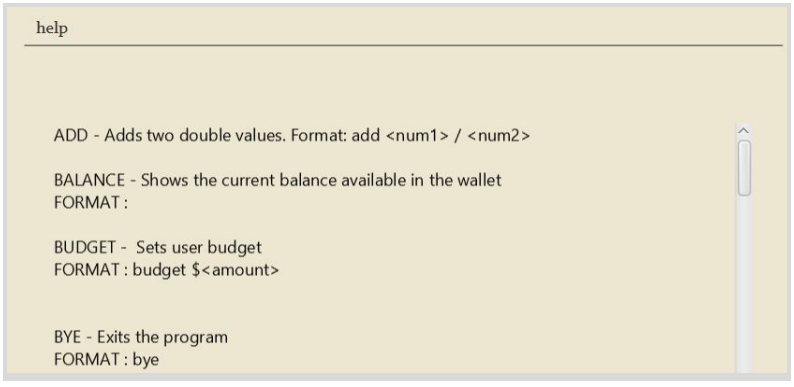
random trailing output
`outxyzab $5.00 /date 2019-10-31 /tags food`

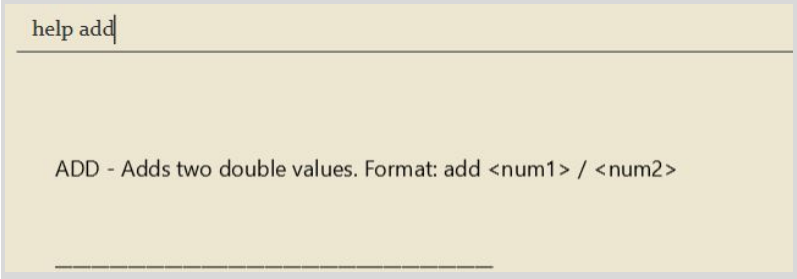
4.1. Viewing help: `help`

This lists out all the commands available for the user to interact with the application.

- You can get the description for all the commands: ***`help`***
- You can get the description for a specific command: ***`help add`***

Format: **`[help]`** **`or`** **`[help] <command_name>`**





4.2. Currency Converter: `convert`

This command 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. The list of countries (and their respective ISO codes) which allow for this currency conversion is available in this link.

Base currency country code : **`BCC`**

Required currency country code : **`RCC`**

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

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

Note:

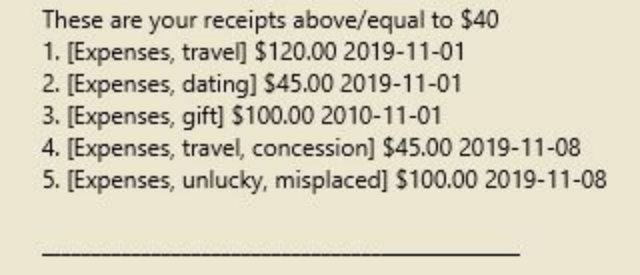
- This command makes use of an API call and hence it is advisable that user has an active internet access

4.3. Major expenses

This gives the list of receipts that have cash spent property higher than or equal to the user's integer input.

Format: `[majorexpense] <integer>`

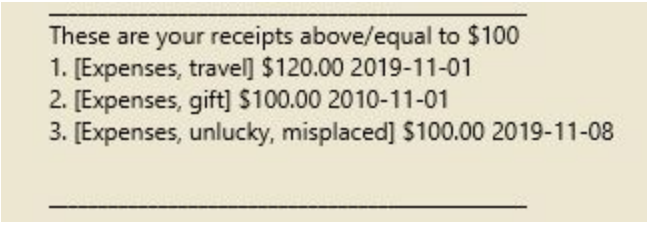
Example: `majorexpense 40`



When command is typed without any primary input, a list of receipts with cash spent property above or equal to $100 is printed.

Format: `[majorexpense]`

Example: `majorexpense`



4.4. Calculator

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

4.4.1 Addition: `add`

Adds two numbers

Format: `[add] <num1>/<num2>`

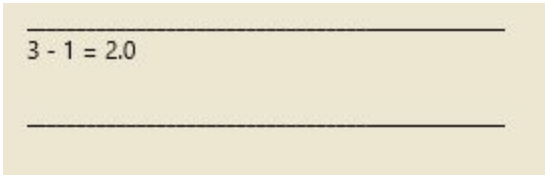
Example: `add 1/2`

4.4.2 Subtraction: `sub`

Subtracts two numbers

Format: `[sub] <num1>/<num2>`

Example: sub 3/1

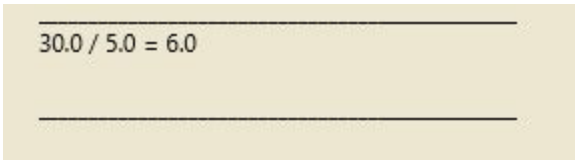


4.4.3 DIvision: `div`

Divides two numbers.

Format: `[div] <num1>/<num2>`

Example: div 30.0/5.0

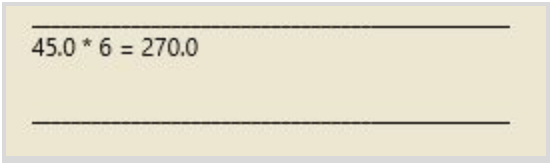


4.4.4 Multiplication: `mul`

Multiplies two numbers

Format: `[mul] <num1>/<num2>`

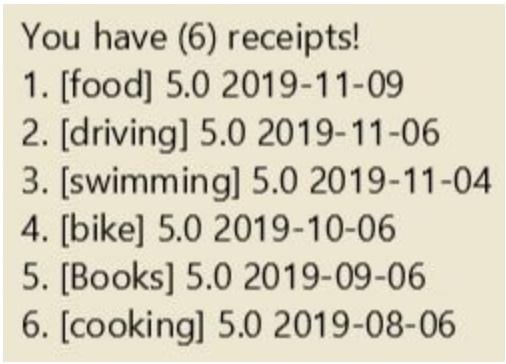
Example: mul 45.0/6



4.5. Total **Expenditure**

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

Assume the following is the list of receipts:



4.5.1 Expenditure for the: `day`

This shows the total expenditure for the day.

User can get different expenditures based on

- Expenditure for today: ***`today`***
- Expenditure for yesterday: ***`yesterday`***
- Expenditure for date: ***`YYYY-MM-DD`***

Format: **`[expendedday today]`**

`[expendedday yesterday]`

`[expendedday <YYYY-MM-DD>]`

Example: **`expendedday today`**

This is the data from the current example list.

This is the output that the user will see.

```
The total amount of money spent today(2019-11-09) is $5.0
```

4.5.2 Expenditure for the: `week`

This shows the total expenditure for the day and the number of days left to the end of the week. Monday is taken as the starting day of the week.

Format: **`[expendedweek]`**

Example: **`expendedweek`**

This is the data from the current example list.

```
1. [food] 5.0 2019-11-09
2. [driving] 5.0 2019-11-06
3. [swimming] 5.0 2019-11-04
```

This is the output the user will see.

```
The total amount spent this week is $15.0 and there is/are 1day(s) to end of week
```

4.5.3 Expenditure for the: `month`

This shows the total expenditure for the month given in input.

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

Example: **`expendedmonth september /year 2019`**

This is the data from the current example list.

```
1. [food] 5.0 2019-11-09
2. [driving] 5.0 2019-11-06
3. [swimming] 5.0 2019-11-04
```

This is the output that the user will see.

```
The total amount of money spent in november 2019 : $15.0
```

4.5.4 Expenditure for the: `year`

This shows the total expenditure for the given year.

Format: `[expendedyear] <year>`

Example: `expendedyear 2019`

This is the data from the current sample list.

```
1. [food] 5.0 2019-11-09
2. [driving] 5.0 2019-11-06
3. [swimming] 5.0 2019-11-04
4. [bike] 5.0 2019-10-06
5. [Books] 5.0 2019-09-06
6. [cooking] 5.0 2019-08-06
```

This is the output that the user will see.

```
The total amount of money spent in 2019 : $30.0
```

4.6. Add an income entry: `in`

This adds an entry to the list as an income.

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

Example: **`in $10.00 /date 2019-10-31 /tags bank`**

4.7. Add an expenditure entry: `out`

This 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`**

4.8. Search for an entry: `find`

This 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.

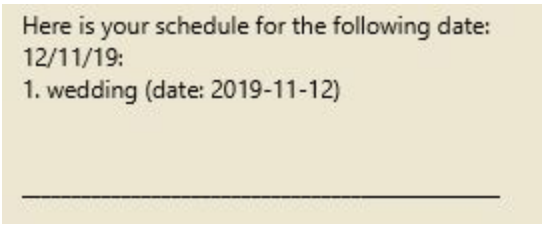
4.9. View Schedule for a day: `viewschedule`

This lists out all the tasks corresponding to the user input date. It is NOT case sensitive.
Example:

Format: **`[viewschedule] <DD/MM/YY>`**

Example: `viewschedule 12/11/19`

Returns



4.10. Deleting an entry: `delete`

This deletes the specified entry based on the `index`, the number shown in the displayed entry tasklist. 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.

4.11. Deleting a receipt: `deletereceipt`

This deletes the specified receipt based on the `index`, the number shown in the displayed receipt list. The index must be an integer.

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

Examples:

- `deletereceipt 2`
  Deletes the 2nd entry in the receiptlist.

4.12. Clearing all entries: `clear`

This clears all entries from the GUI display screen.

Format: **`[clear]`**

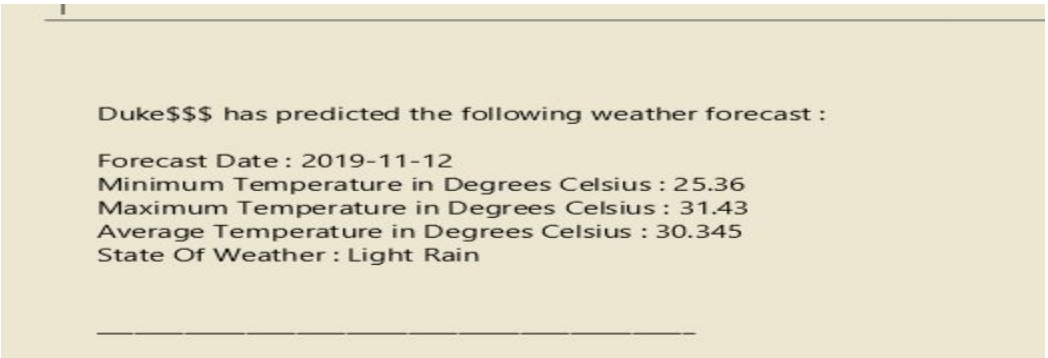
4.13. Display Weather status: `weather`

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

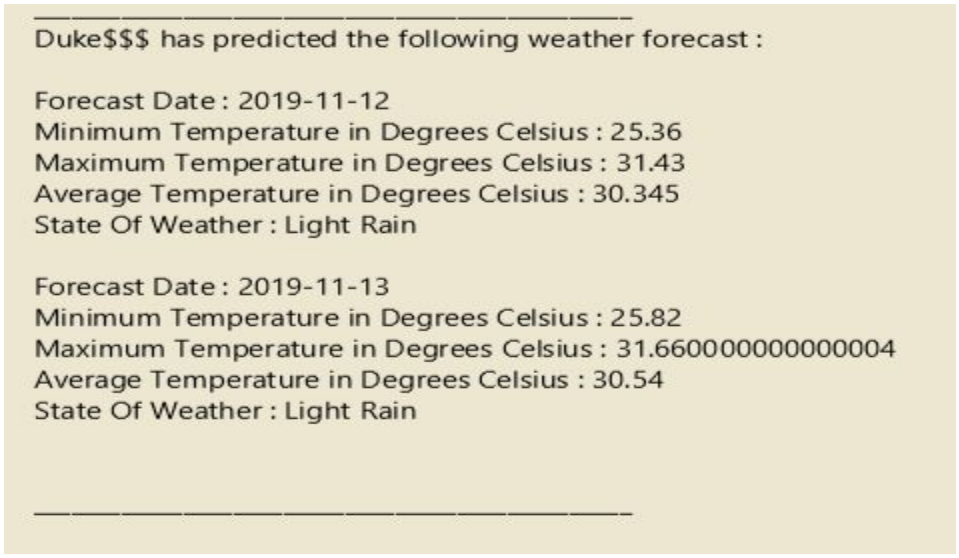
- Current instant weather status only by specifying: ***`now`***
- Current instant weather status and the weather forecast for the next day by specifying : ***`tomorrow`***
- Current instant weather status and the weather forecast for the next 5 days by specifying : ***`later`***

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

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



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



Note:

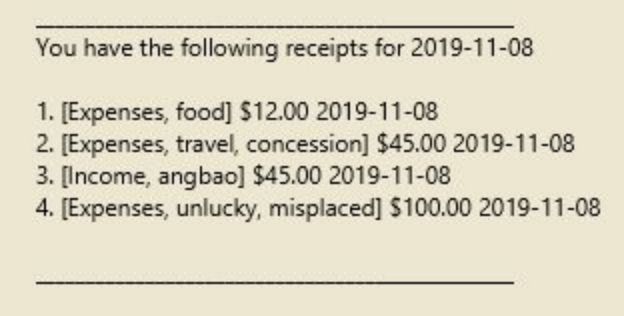
- If user enters just weather, Duke$$$ understands that the user might be in a rush and hence the current instant weather forecast is given
- This command makes use of an API call and hence it is advisable that user has an active internet access

4.14. Listing entries based on date: `datelist`

User is able to get the list of all receipts for a particular day. Date entry must be in the correct format (yyyy-mm-dd)

Format : `[datelist] <yyyy-mm-dd>`

Example: `datelist 2019-02-01`



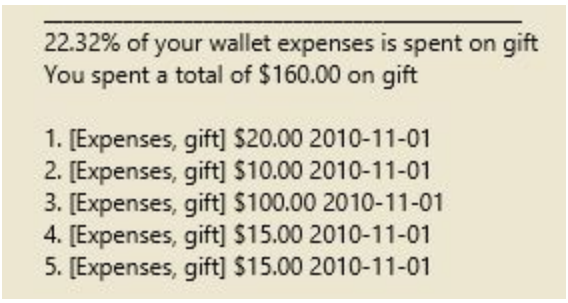
4.15. View statistics for a tag: `stats`

Statistics command allows users to get an overview of the statistics for a particular tag in the receipt list. This list of statistics include :

1. The percentage of all wallet expenses that went to that tag
2. The total expenditure for the tag
3. List of receipts containing the tag

Format: `[stats] <tag>`

Example: **`stats gift`**



Note:

- This feature only applies to tags that already pre-exist. Command will not go through if the tag input does not exist in the receipt list.
- This feature does not take income receipts into account for the first two attributes i.e Percentage of wallet expenses on tag and total expenditure

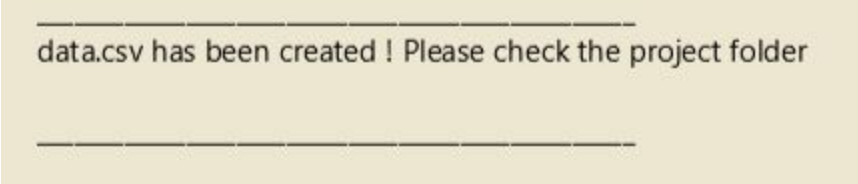
- The command lists both income and expense receipts

4.16. Exporting wallet into csv : `export`

Export command allows users to convert wallet data into a csv file with meaningful headers for tracking. The created file will be saved in the project folder containing the jar file of Duke$$$.

Format : **`[export]`**

Example: **`export`**

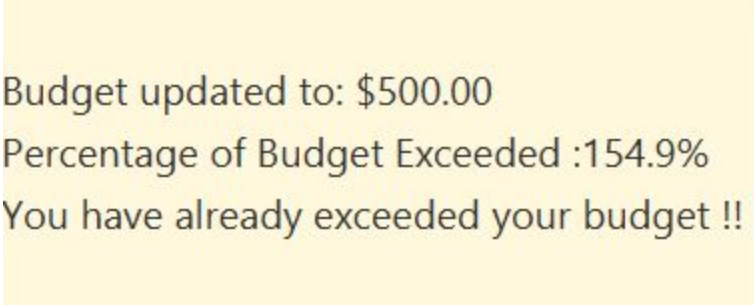


4.17. Budget Setting : `budget`

This command aims to allow the user to be able to set a budget. By setting a budget, the user will also be prompted with percentage statistics on how much has he or she used up or exceeded the budget with their expenditure (expenses receipts).

Format : **`[budget] $<amount>`**

Example: **`budget $500`**



Note :

- The example above compares the budget against preloaded data.

4.18. Edit

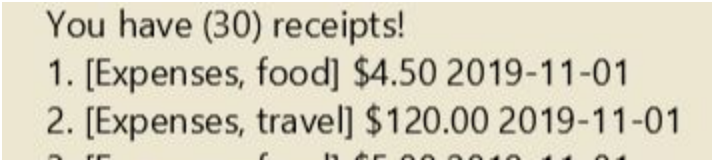
This allows the user to edit parts of the already existing receipts.

- Has 3 different flags that could be used to edit the tag,date, cash value.

Format: `[edit] <index> [/tag] <new_tag>`
`[edit] <index> [/tag] <new_value>`
`[edit] <index> [/tag] <new_date>`

Example: **`edit 1 /tag swimming course`**

This is the current list before command:



This is the list after command(Notice receipt 1)

4.19. Expenses Tag Tracker: `track/untrack`

This command allows you to track and display a barchart that allows you to compare how much you spend on a particular category of items as compared to others.

Format: `[track] <tag>`

Example: **`track food`**

Say you have the following list of receipts and you wanted to get a sense of how much more you spend on food than you do on your lover.

```
You have (5) receipts!
1. [Income, angbao] $1000.00 2019-11-12
2. [Expenses, food] $10.00 2019-11-12
3. [Expenses, food] $10.00 2019-11-12
4. [Expenses, food] $10.00 2019-11-12
5. [Expenses, lover] $10.00 2019-11-12
```

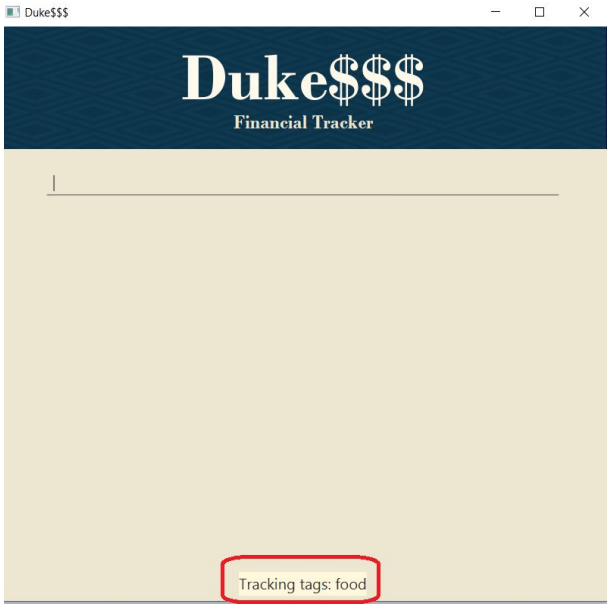
Having the raw numbers are great but sometimes, what you really need is a visualization of all this data.

Instead of going to excel and manually creating a barchart to really appreciate the difference, you can easily take advantage of Duke$$$'s in-built command: `track`.

Here's what you need to do.

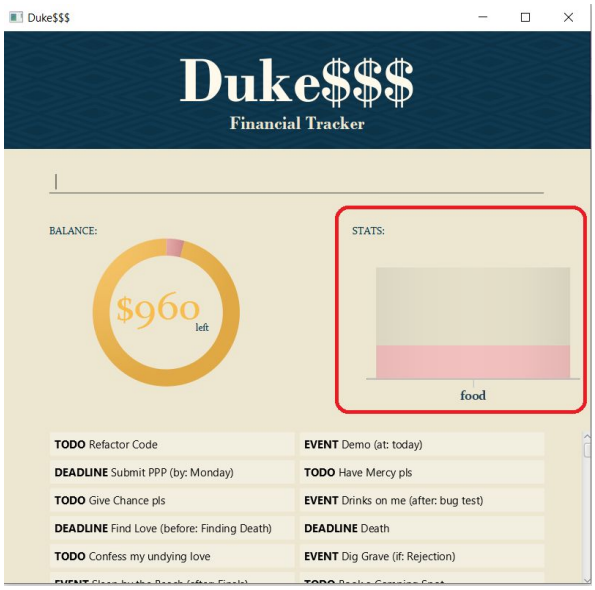
Step 1: Type `track` and the tag you want track.

Step 2: Press `Enter`. You should see the following pop-up message that indicates that Duke$$$ is now tracking that tag of yours.

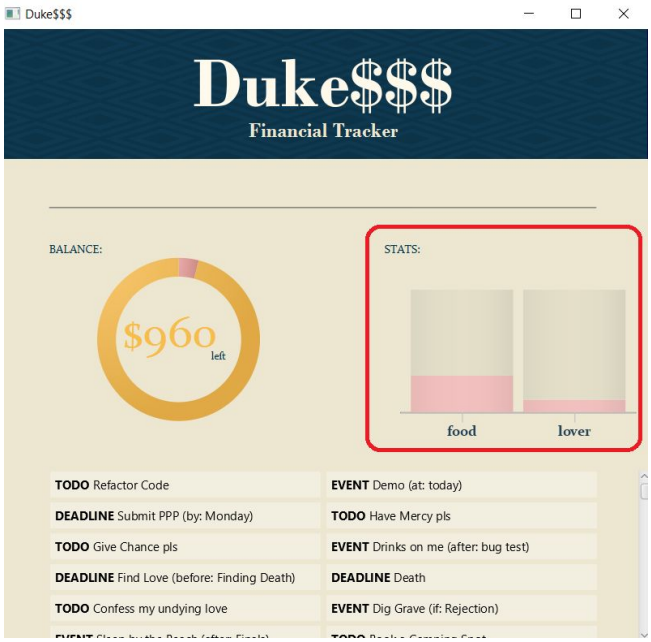


Step 3: Return to the home display by typing `home` and pressing `Enter`.

You should see the following:



Step 4: Repeat steps 1 to 3 for the tag “lover”. The resulting graph should look something like this.



Congratulations, you have now created a barchart that will continuously update as you add more and more receipts with the tags “food” or “lover”. This way, you’ll always be sure that you’re spending more on the things that matter.

Note:

- Each time you track a new tag, do refresh the home display by typing the command `home`. The current Duke$$$ does not fully support live updating of data as this will come in **Version 2.0.**
- To untrack a tag, simply follow the steps above, replacing `Track` with `Untrack`.

4.20. Reminder for today: `reminder`

This command lists out the reminders for today. However, if the date entered by the user is not interpreted properly, the program assigns the deadline of a task to be of the current day. Hence the list of reminders could be huge. Thus if user enters the tasks as per required user input the correct tasks will be displayed as a reminder.

Format: `[reminder]`

Example: **`reminder`**

```
Here are the tasks for today
1. test
2.
3.
4. remove
5. Refactor Code
```

4.21. Creating a new Task: `event/deadline/todo`

This command allows you to create a new Task for Duke$$$ to track.

Format: **`[TaskType] <nameOfTask>`**

4.22. Exiting the program: `bye`

This command allows user to exit the program.

Format: **`[bye]`**

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. Implement cross-platform capability such that users can access their data on any desktop environment.
3. Encrypting data files to increase the security of personal data.
4. Allowing the user to directly update the application if they do online shopping or use transaction related applications such as PayLah or Grabpay for payments.
5. Syncing with the user's bank account to provide more monitoring.
6. Integrate cloud computing to store the data.
7. Have information regarding Singapore tourism such that exchange students can explore Singapore more easily

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.

Q: Where do I find the exported .csv file?

A: It is located in the /main directory.

7. Command Summary

Identifier	Description	Format	Example
ADD	Adds two numbers	`[add] <num1>[/] <num2>`	`add 1/2`
BALANCE	Shows the current balance in the User's wallet	`[balance]`	`balance`

BUDGET	Sets the budget for the user	`[budget] <amount>`	`budget $50.00`
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`
DATELIST	Lists all receipts for a particular date.	`[datelist] <yyyy-mm-dd>`	`datelist 2019-02-01`
DEADLINE	Adds a task with deadline	`[deadline] <taskdescription> /by dd/mm/yy HHmm`	`Deadline project /by 06/11/19 2359`
DELETE	Deletes the chosen task	`[delete] <index>`	`delete 3`
DELETERECEIPT	Deletes the chosen receipt	`[deletereceipt] <index>`	`deletereceipt 3`
DIV	Divides two numbers	`[div] <num1>[/]<num2>`	`div 10/4`
DONE	Marks task as done	`[done] <index>`	`Done 3`
EVENT	Adds an event task	`[event] <taskdescription>`	`Event party`

EXPENDEDDAY	Gives the total amount of money expended in a day	`[expendedday] <yyyy-dd-mm>`	`expendedday 2019-08-30`
EXPENDEDMONTH	Gives the total amount of money expended in the month	`[expendedmonth] <month> [/year] <year>`	`expendedmonth march /year 2019`
EXPENDEDWEEK	Gives the total amount of money expended in the current week	`[expendedweek]`	`expendedweek`
EXPENDEDYEAR	Gives the total amount of money expended in the year	`[expendedyear] <year>`	`expendedyear 2019`
EXPENSES	Shows the total amount of money spent	`[expenses]`	`expenses`
EXPORT	Export the wallet data as csv	`[export]`	`export`
FDURATION	Adds a task with a fixed duration	`[fduration] <taskdescription> need <duration>`	`Fduration assignment need 5 hours`
FIND	Checks if input exists in the list of tasks	`[find]`	`find food`
HELP	Gives a description of all the commands available for the user to use.	`[help]`	`help`
HOME	Displays the home page	`[home]`	`home`
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`
MAJOREXPENSE	Gives the user a list of	`[majorexpense]`	`[majorexpense]`

	receipts that have cash spent property above or equal to user input. Also gives list of receipts above $100 when the command is entered without any primary input	`<integer cash input>`
		
		`Or`
		
		`[majorexpense]`
			`50`
			
			`Or`
			
			`[majorexpense]`

MUL	Multiplies two numbers	`[mul] <num1>[/]<num2>`	`mul 45/56`
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`
PERCENT	Gives the percentage of wallet expenses on a particular tag	`[percent] tag`	`Percent transport`
QUEUE	Queues the user inputted task in the list.	`[queue] <taskindex> /task <taskdescription>`	`Queue 3 /task prepare dinner`
RECUR	Adds a recurring task to the list.	`[recur] <taskdescription> /for <duration>`	`recur quiz submission /for weekly`
STATS	Gives the statistics for expenditure of a particular tag	`[stats] <tag>`	`stats transport`
SUB	Subtracts two numbers	`[sub] <num1>[/]<num2>`	`sub 3/1`
TAGLIST	Lists out all the entries corresponding to that specific tag that user	`[taglist]`	`taglist transport`

	inputs.		
TRACK	Tracks expenditure based on tag.	`[track] <tag>`	`track transport`
UNTRACK	Untrack a particular tag.	`[untrack] <tag>`	`untrack food`
VIEWSCHEDULE	Allows user to view their task schedule for a particular date.	`[viewschedule] <yyyy-mm-dd>`	`viewschedule 2019-02-01`
WEATHER	Displays weather forecast based on the period requested by user.	`[weather] [/until] <period>`	`weather /until later`
EDIT	Edits parts of the already existing receipts requested by user.	`[edit] <index> [/<flag>] <new_value>`	`edit 1 /tag food`
REMINDER	Gets the reminder for today	`reminder`	`reminder`