TIC4001 - Project notes.

Brainstorm idea

1. Video downloader via cli. (is it allowed to download video?)
    a. User can download video if they have the link to view the video. (youtube)
        i. Internet access + storage space
    b. Search the video that the user downloaded.
    c. Rename the video file. (organisation related function)
    d. Delete the videos that you do not want anymore

    Remarks:

2. Money expenses tracker
    a. Able to create daily expenses tracker
        i. Add / remove / edit
            - Per transaction
            - Recurring setup
        ii. Different Account category
            - Saving / Credit Card expenses / Cash
    b. Search for expenses with the keyword
        i. Example category, description
        ii. Show the list of expenses for Today.
            - Can it be configurable? (we can consider)
    c. daily/weekly/monthly report (pdf
        i. Pdf generator or Excel format
        ii. Custom tips saving (based on the spending/saving)

    Remarks:

3. Booking/reservation hotel
    a. Able to search/book an hotel via the cli.
    b. Users will be able to book/cancel/update their own bookings.
        i. Optional: Add on room services, add more days, check in and check out.
    c. History of your bookings
        i. Your search history too.

Remark: require hotel's API or some api to use. (not recommended)

Question 1: Product name
NUSExpensesHelper.

Question 2: Target user profile
NUS TIC4001 Classmate (Students/Lecturer)

Question 3: Value proposition

1. No internet access.
2. Secured tracker cause only on your PC.
3. Generating Reports (Open in Excel).
4. One glance view of weekly/monthly spending.

User stories in trello.
https://trello.com/b/2290RLsD/nusexpenseshelper

AB3 SAMPLE:

NUS Expenses Helper/Tracker User Guide

Introduction

Nus Expenses Tracker is an app for managing expenses, optimized for use via a Command Line Interface (CLI) while still having the benefits of a Graphical User Interface (GUI).

-- Let me think of a cooler intro -- will update

Features for v1.0

- Add Expenses (add) (Zi Wen)
- View Expenses (view) (Lai Ping)
- Delete Expenses (delete <IDX>) (Li Yi)
- Search with Keyword to find the expenses record. (SEARCH <keyword>) (Adi)
- Show Total Expenses Incurred - (show) (Olivier)

Add a expense entry: add

Add an expense entry with description, amount and optionally specify the date

Format: add <KEYWORD> $<AMOUNT> <DATE>[optional]

- Add the record with the <KEYWORD> and <AMOUNT>, these two fields are compulsory
- The third input <DATE> will be optional, if the user enters the date, this expense will be recorded as the date the user entered, otherwise it will be default today's date.

Examples:
Add Lunch $4 returns
Record (1. Lunch $4.00) has been successfully added.

Search for a expense entry: search

Search for any expense entries containing the given keyword.

Format: search <KEYWORD>

- The search is case-insentive. E.g. wonton will match Wonton
- The order of the keywords does not matter. E.g. Wanton will match Fried Wonton
- Only the description is searched.
- Only full words will be matched. E.g Wonton will not match Wontons
- Expenses matching at least one keyword will be returned.(i.e OR search). E.g. Wonton will return Fried Wonton $2.50, Wonton Soup $5.50

Examples:
Search pau returns
1. 2020-09-16 char siew pau $2.50
2. 2020-09-17 chicken pau $400.00
3. 2020-09-18 veggie pau $800.00

Show total expenses: show

Calculate the total amount spent from all the expenses that user entered.

Format: show

Examples:
show returns The amount you spent till today is $42000

Delete expenses record: delete id

Delete the expense record by unique ID number.

Format: delete 1

- Delete the record with the specific ID entered
- The ID will be generated according the the index position
- The ID must be a positive integer 1,2,3,...

Examples:
delete 1 returns
Record (1.Lunch $4.00) has been successfully deleted.

When user want to delete the report

View expenses record : display

To display all data user entered when the user key display
Format : display

- Display according to sequence user input. The first entered will display first.
- Go through ArrayList and print out

Examples :
<sn> <Date> <Description> <Amount>
1. 2020-09-16 Dinner $5.00
2. 2020-09-17 Breakfast $800.00

*Sn is an auto increment.

Introduction

Nus Expenses Tracker is an app for managing expenses, optimized for use via a Command Line Interface (CLI) while still having the benefits of a Graphical User Interface (GUI).

Quick Start

1. Ensure you have Java 11 or above installed in your Computer.
2. Download the latest addressbook.jar from here.

3. Copy the file to the folder you want to use as the *home folder* for your ExpensesTrackerApp.

4. Double-click the file to start the app. The GUI like the below should appear in a few seconds. Note how the app contains some sample data.

<Image>

5. Type the command in the command box and press Enter to execute it.

Some example commands you can try:

- **add**: Add an expense entry with description, amount and optionally specify the date
- **search**: Search for any expense entries containing the given keyword.
- **show**: Calculate the total amount spent from all the expenses that user entered.
- **delete1**: Delete the expense record by unique ID number.
- **display**: To display all data user entered when the user key display
- **exit**: Exits the app.

6. Refer to the Features for v1.0 below for details of each command.

Features for v1.0

Add an expense entry: add

Format:

add <KEYWORD> $<AMOUNT><DATE> [optional]

- Add the record with the <KEYWORD> and <AMOUNT>, these two fields are **compulsory**
- The third input <DATE> will be optional, if the user enters the date, this expense will be recorded as the date the user entered, otherwise it will be default today's date

Sample input/output:

Add Lunch $4

Record (1. Lunch $4.00) has been successfully added.

Search for an expense entry: search

Format:

search <KEYWORD>

- The search is **not case-sensitive**. E.g. wonton will match Wonton
- The order of the keywords does not matter. E.g. Wanton will match Fried Wonton
- Only the description is searched
- Only full words will be matched. E.g. Wonton will not match Wontons
- Expenses matching at least one keyword will be returned. (i.e. OR search). E.g. Wonton will return Fried Wonton $2.50, Wonton Soup $5.50

Sample input/output:

Search pau

1. 2020-09-16 char siew pau $2.50

2. 2020-09-17 chicken pau $400.00

3. 2020-09-18 veggie pau $800.00

Show total expenses: show

Format:

show

Sample input/output:

show

The amount you spent till today is $42000

Delete expenses record: delete

Format:

delete <id>

- Delete the record with the specific ID entered
- The ID will be generated according the index position
- The ID must be a positive integer 1,2,3, (…)

Sample input/output:

delete 1

Record (1. Lunch $4.00) has been successfully deleted.

View expenses record: display

Format:

display

- Display according to sequence user input.
- The first entered will display first.

Sample input/output:

display

1. 2020-09-16 Dinner $5.00

2. 2020-09-17 Breakfast $800.00

FAQ

Command Summary

```
public static final Pattern ADD_COMMAND_FORMAT
        =
Pattern.compile("(?<description>[^$]*)(?<amount>\\${1}\\d+\\.\\d{2})(?<date>
.*)",Pattern.CASE_INSENSITIVE);
```

```
package seedu.duke.commands;

import org.junit.jupiter.api.Test;
import seedu.duke.data.ReadOnlyTransaction;
import seedu.duke.data.TransactionList;
import seedu.duke.utilities.SetupTransactionData;

import java.util.Collections;
import java.util.List;

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

public class SearchCommandTest {
    private SetupTransactionData setupData = new SetupTransactionData();
    private TransactionList transactionList = setupData.loadTransactionData();
    //Expenses list shown to the user recently.
    private List<? extends ReadOnlyTransaction> lastShownList = Collections.emptyList();

    @Test
    public void searchCommand_Test(){
        SearchCommand command = new SearchCommand("chicken");
        command.setData(transactionList, lastShownList);

        assertEquals("1 transactions listed!", command.execute().feedbackToUser);

        SearchCommand command2 = new SearchCommand("rice");
        command2.setData(transactionList, lastShownList);

        assertEquals("3 transactions listed!", command2.execute().feedbackToUser);

    }
}
```

```
Pattern.compile("(?<description>[^$]*)(?<amount>\\${1}\\d+\\.?\\d{0,2})(?<date>.*)",Pattern.CASE_INSENSITIVE);
```