



V0.1 Report

cs2103jan13-t13-4j

Iteration v0.1 Interim Report

LIU BOHUA
LIN ZHIYUAN
PANG KANG WEI, JOSHUA
WONG JING PING
HAO WEI

A0091879J
A0091859M
A0087809M
A0086581W
U080159R

Table of Contents

1. Project Scope	3
1.1 Baseline Features	3
1.2 Good-to-Have features.....	4
2. User Guide	5
2.1 Overview of Main Panel.....	5
2.2 Add Transaction Panel	6
2.3 Edit Transaction Panel	7
2.4 Delete Transaction Panel.....	7
2.5 Intra-asset / Intra-liability Transfer Panel	8
2.6 Rename Category Panel	8
2.7 Things to take note	9
3. Developer's Guide	10
3.1 Software Architecture.....	10
3.2 Components.....	11
3.3 Main Algorithms	14
3.4 Project Process and Administration	17
3.5 Development Infrastructure	17
3.6 Glossary	18



1. Project Scope

Our software will be targeted at working adults, with 1 user in mind. Given the busy schedules that working adults have and the multiple types of complex transactions that they make daily, we have decided to make a robust personal accountant that is not only informative, but also intuitive and easy to use. As such, we distinguish this piece of software from other personal accounting software in following aspects:

- Able to handle every possible of transaction that a working adult would require
- An intuitive graphical interface that would allow the user to get a bird's eye view of his/her state of finances at a glance at our program

As such, we have come up with a list of features to be implemented, categorizing them according to their priority, with the highest priority features in the Baseline Features, and the additional features under Good-to-have Features.

1.1 Baseline Features

- CRUD operations 

As a working adult, the user handles transactions that are more complex than just receiving incomes and making expenses. Sometimes, he takes loan from bank, applies for credit cards, and pays bill through credit cards and so on. Moreover these transactions may not involve cash directly, but involve complex accounts like credit card and loan balances. Hence, in order to keep track of all the transactions correctly in a more systematic manner, we decided to adopt a double-entry accounting framework. As such, we would need to monitor four main sets of data, which are the user's asset, liability, income and expense. The table below shows how this is to be done by using "Add Transaction" function as an example.

Transactions	Asset	Liability	Income	Expense
Income Received	Increase		Increase	
Expense by Asset	Decrease			Increase
Expense By Credit		Increase		Increase
Repay Loan		Decrease		Increase
Take Loan	Increase	Increase		

As you can see from this example, every time when a transaction is made, 2 sets of data will be affected. Same principles are applied to other CRUD operations as well.

- Display of asset, liability, income and expense subtotals by categories

Four panels representing four different sets of data will be displayed on the main panel. To distinguish between the 4 main types, we have identified them with different colours. Within each panel, bar charts are used to show values of different categories.

- Renaming of categories
Users are able to rename categories if necessary.
- Simple search function
This function allows the user to find a particular transaction history based on its date and transaction type.
- Undo function
When a user does a wrong / unwanted operation, our system allows the user to undo and go back to the last step.
- Alert
When the balance is negative, or when the user enters invalid information in the field, there will be an alert to warn the user.

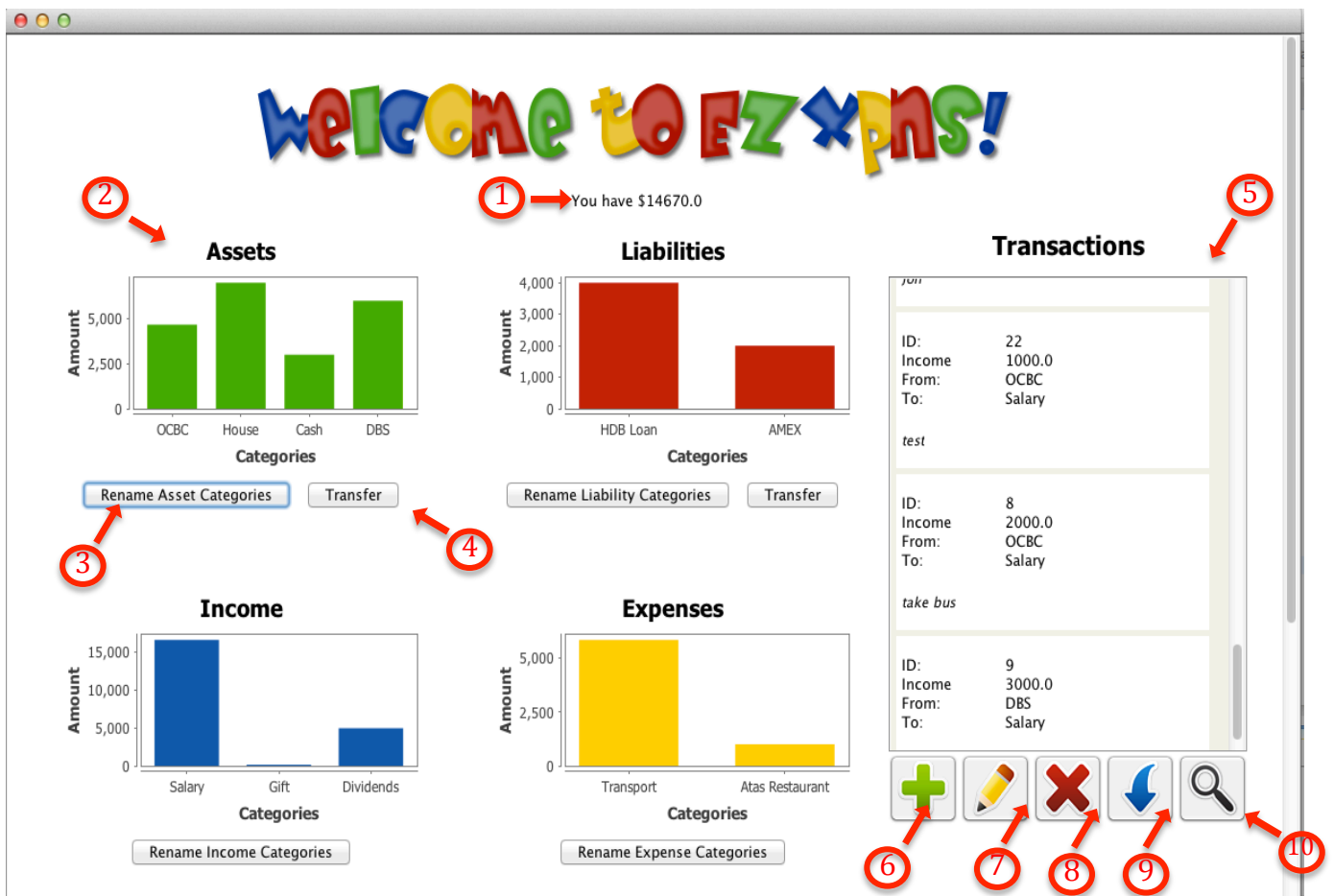
1.2 Good-to-Have features

- Well-designed graphical user interface
We have decided to make a sleek graphical user interface that uses charts to show the user the state of his finances, as well as a news-feed style scrolling pane on the right that displays his history of transactions.
- Export to Excel. In the next iteration, we will be implementing an excel export function to export data and a simple report to an excel sheet.

2. User Guide


2.1 Overview of Main Panel

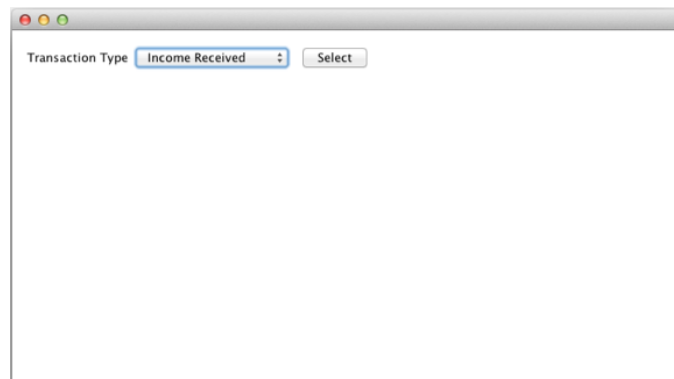
EzXpns has a very interactive and easily understood Graphic User Interface (GUI). Users can get all the information at a glance from the main user panel. Various transaction operations are to be done in a separate pop-up window. Pressing the respective button can easily activate these pop-up windows. The diagram below is a screenshot of the main user panel and its important information can be found in the table after the diagram.



Number	Name	Description
1	Balance Field	To show the balance according to the formula: $\text{Balance} = \text{Asset} - \text{Liability}$
2	Asset/ Liability/ Income/ Expense Panel	Use bar charts to display the values of different categories within each panel
3	Rename Asset/ Liability/ Income/ Expense categories	Press the button to change the name of a category for asset/liability/income/expense
4	Intra-transfer asset/ Liability	Press the button to transfer money from one asset/ liability category to another
5	Scrollable Transaction List	To display various transactions made by the user
6	Add Transaction	Press the button to add new transaction
7	Edit Transaction	Press the button to edit an existing transaction
8	Delete Transaction	Press the button to delete an existing Transaction
9	Undo Transaction	Press the button to undo the last transaction
10	Search	Press the button to search a particular transaction based on its date category

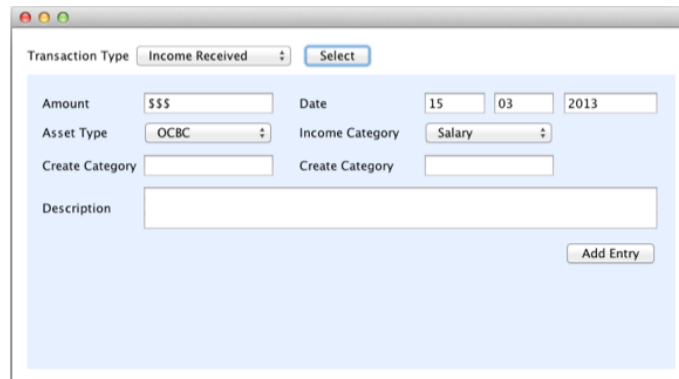
2.2 Add Transaction Panel

When Add Transaction button  is pressed, a pop-up window that looks like the diagram below will come out. The user is required to select a transaction from the drop-down list.



Upon selecting a transaction, a new panel will pop out in the current window. Over here, the user is to fill in the transaction information before pressing the Add Entry button to confirm. If the user has a new Asset Type or Income Category that is not in the drop-down list, he/she is able to add new type/category as long as it does not


have the same name as the existing ones. A transaction ID is given to every successful transaction.

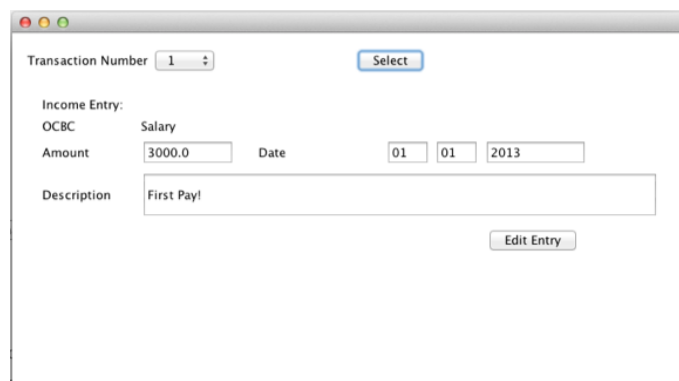


A screenshot of a web application window titled 'Add Entry'. The window has a light blue background. At the top, there is a 'Transaction Type' dropdown menu set to 'Income Received' and a 'Select' button. Below this, there are several input fields: 'Amount' with a '\$\$\$' prefix, 'Date' with fields for '15', '03', and '2013', 'Asset Type' with a dropdown set to 'OCBC', 'Income Category' with a dropdown set to 'Salary', and two empty 'Create Category' fields. A large text area for 'Description' is at the bottom. An 'Add Entry' button is located at the bottom right.

2.3 Edit Transaction Panel

Edit transaction function is for users to change the transaction information of a particular transaction based on its ID. The ID of the transaction can be easily found

from the Scrollable Transaction List. When Edit Transaction Button  is pressed, Edit Transaction Panel pops out.




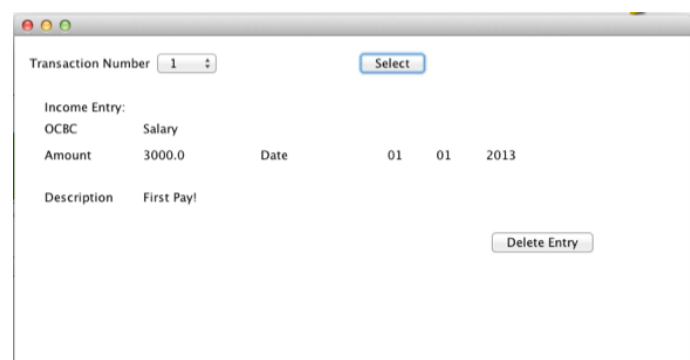
A screenshot of a web application window titled 'Edit Entry'. The window has a light blue background. At the top, there is a 'Transaction Number' dropdown menu set to '1' and a 'Select' button. Below this, there is a section for 'Income Entry' with a dropdown set to 'OCBC' and a label 'Salary'. There are input fields for 'Amount' (3000.0), 'Date' (01/01/2013), and a text area for 'Description' (First Pay!). An 'Edit Entry' button is located at the bottom right.

User can just edit the information that he / she wants to change before pressing the Edit Entry button to confirm.

2.4 Delete Transaction Panel


If a transaction is not long valid, a user can choose to delete it by pressing the Delete

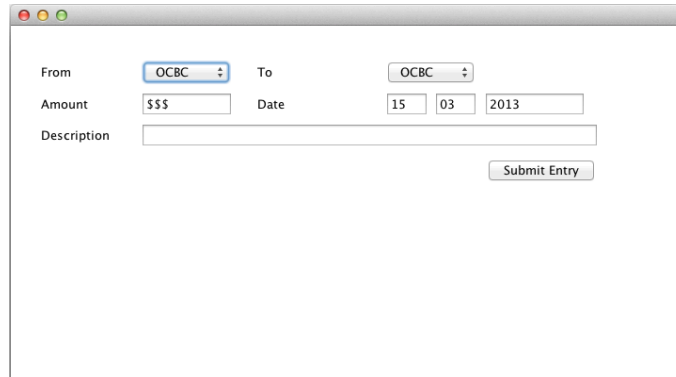
Transaction button . The Delete Transaction panel looks something like this. Press Delete Entry button to confirm.



A screenshot of a web application window titled 'Delete Entry'. The window has a light blue background. At the top, there is a 'Transaction Number' dropdown menu set to '1' and a 'Select' button. Below this, there is a section for 'Income Entry' with a dropdown set to 'OCBC' and a label 'Salary'. There are input fields for 'Amount' (3000.0), 'Date' (01/01/2013), and a text area for 'Description' (First Pay!). A 'Delete Entry' button is located at the bottom right.

2.5 Intra-asset / Intra-liability Transfer Panel

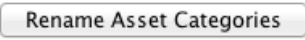
Sometimes, a user wants to transfer money from one category to another category within asset or liability. Examples for such scenarios can be a transfer of money from one bank account to another bank account (intra-asset transfer) or use a credit card to pay loan (intra-liability transfer). Press the Transfer button  located below Asset/Liability panel to perform this operation.

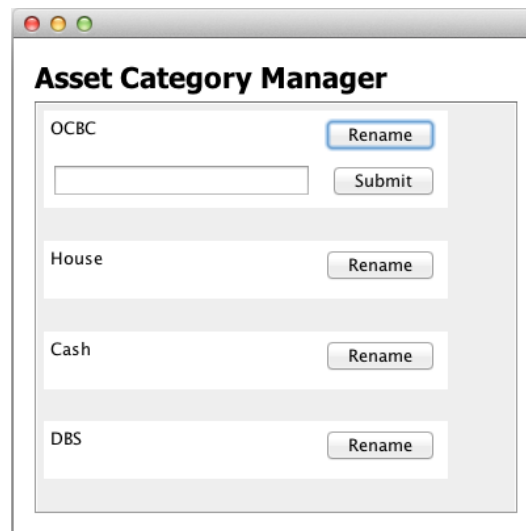


The screenshot shows a window titled "Intra-asset / Intra-liability Transfer Panel". It contains the following fields and buttons:

- From:** A dropdown menu with "OCBC" selected.
- To:** A dropdown menu with "OCBC" selected.
- Amount:** A text input field containing "\$\$\$".
- Date:** Three text input fields for day, month, and year, containing "15", "03", and "2013" respectively.
- Description:** A large text input field.
- Submit Entry:** A button located at the bottom right of the form.

2.6 Rename Category Panel

A user is also able to change a category name by pressing the Rename Categories Button  located below Asset / Liability / Income / Expense Panel.



The screenshot shows a window titled "Asset Category Manager". It contains a list of categories with a "Rename" button next to each:

- OCBC:** A text input field with "OCBC" and a "Rename" button.
- House:** A text input field with "House" and a "Rename" button.
- Cash:** A text input field with "Cash" and a "Rename" button.
- DBS:** A text input field with "DBS" and a "Rename" button.

2.7 Things to take note

There are a few things to take note when a user keys in information during transaction operations. They are:

Type	Things to Take Note
Date Format	dd/mm/yyyy <ul style="list-style-type: none">• Date value range is $0 < dd \leq 31$• Month value range is $0 < mm \leq 12$• Year value range is $1900 \leq yyyy \leq$ current year
Amount Value	<ul style="list-style-type: none">• All amount value filled in cannot be negative• The amount value after any transaction operation cannot be negative e.g. cash is \$1000, a transfer of \$1500 from cash to OCBC bank account is an invalid transaction• Balance can be negative. A alert sign will be shown if it is so
Category Name, Description	No Pipe sign “ ” is allowed

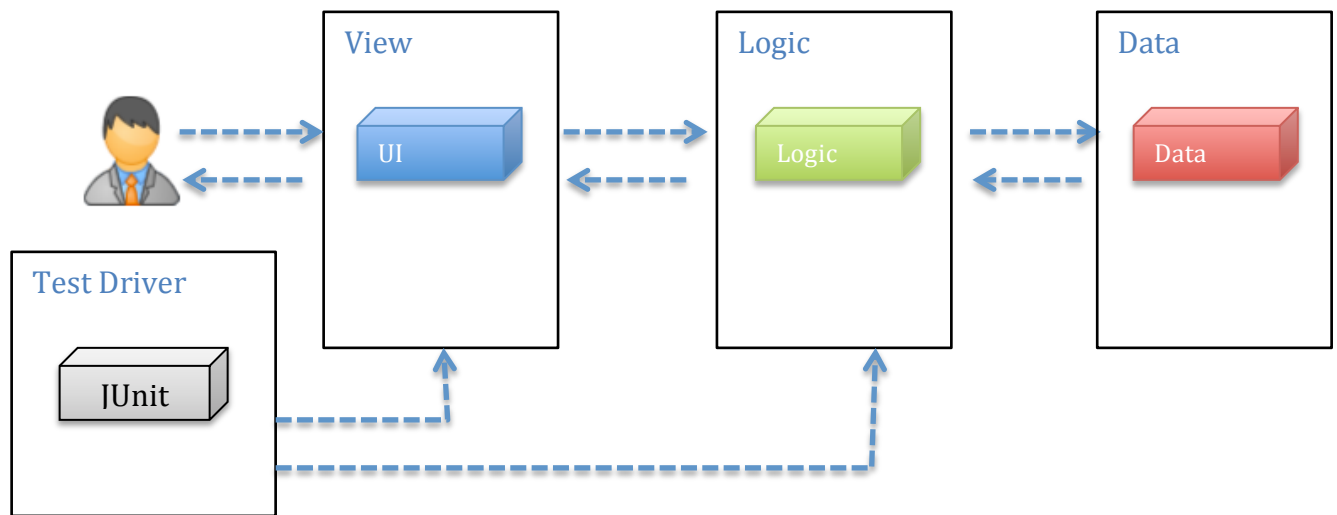
3. Developer's Guide

3.1 Software Architecture

EzXpns adopts the n-tier architectural style and it consists of 4 main components at different levels. They are View, Logic, Data and Testing.

View and Logic components have their own managers that store all the methods required for EzXpns systems. View managers are used to communicate between frontend display and backend logic, deciding which button activates which method.

Logic managers contain methods that are essential to perform different operations and it is also in charge of writing data into database, in this case .txt files and extracting relevant information that is to be displayed on frontend display.

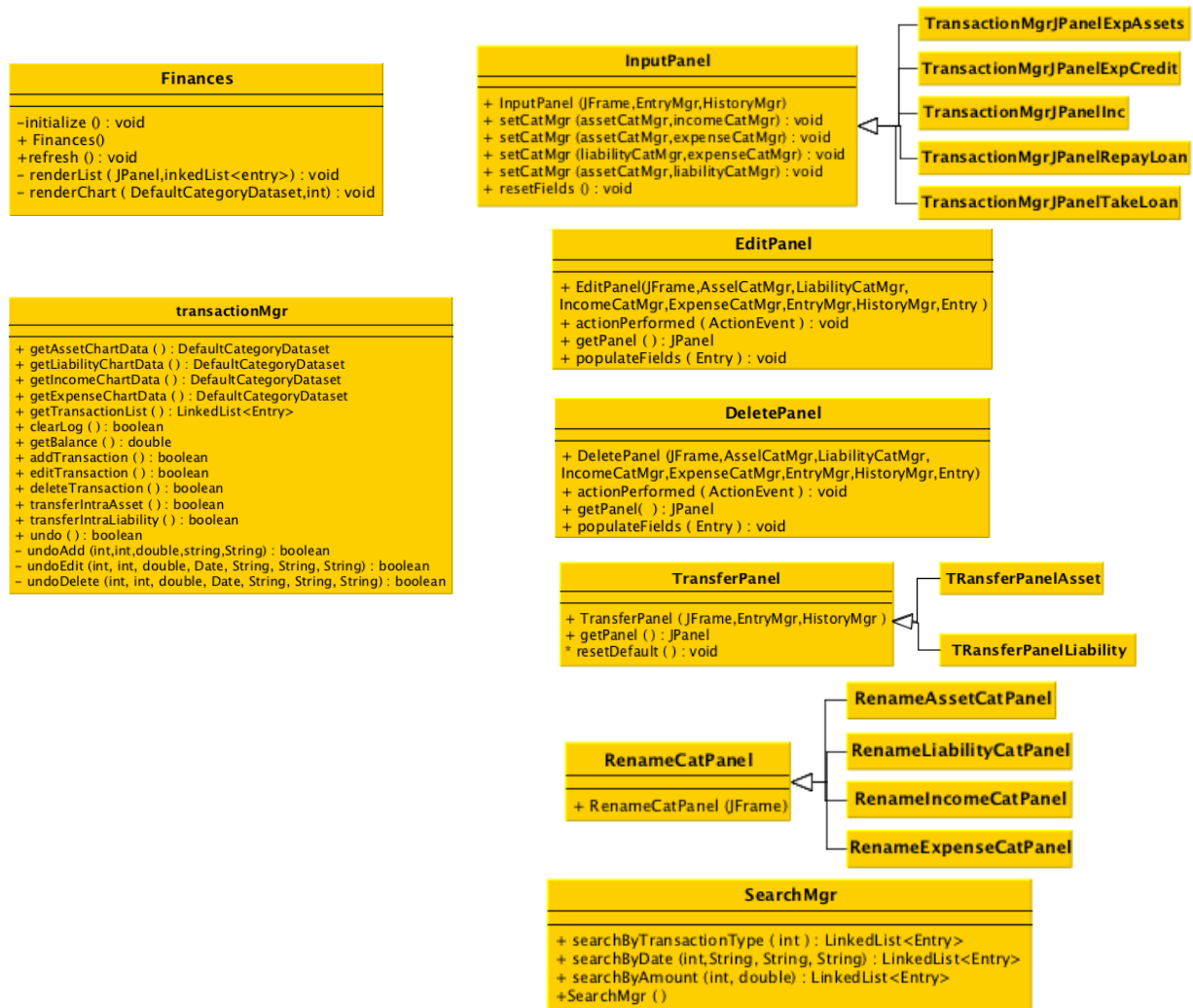


3.2 Components

- View

The user interface is designed to interact with users, as well as to display information that is relevant to users' needs.

User Interface



Finances class is the main GUI that displays the main window. TransactionMgr class handles all the requests from the user. There are 6 classes that serve to open individual operation panels, namely InputPanel, Edit Panel, DeletePanel, TransferPanel, RenameCatPanel and SearchPanel. For InputPanel, TransferPanel and RenameCatPanel classes, they have their own inheritance classes. These inheritance classes are used to create panels for more specific operations.

In GUI classes, we have standardized a naming convention. All GUI elements will be named in this way: [classname][what we want to call it]_[shortform of the type]. For example, Asset Panel will be named financeAsset_PNL.

The shortforms will be as such:

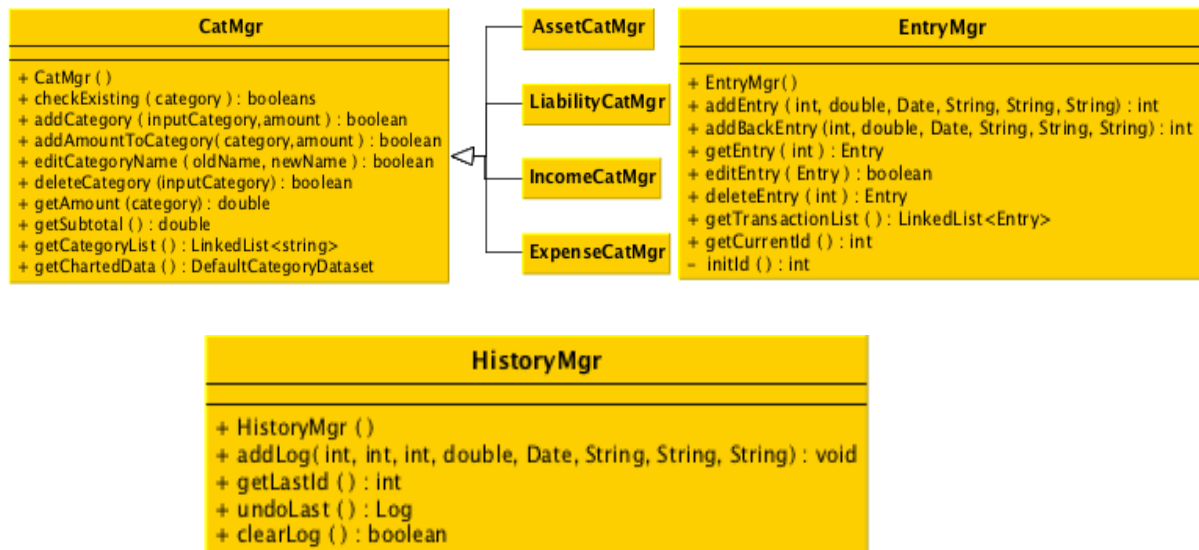
Shortform	Description
PNL	Panel
LBL	Label
SCP	Scrollable Panel
FRM	Frame
CB	Combobox
BTN	Button
TF	Text Field

Currently we are using JAVA original date API, but it's use is not optimal due to deprecated methods present. We are considering refactoring the coding by implementing Joda Time API in the next iteration.

Our GUI uses actionlistener and windowlistener to activate logic/data processing.

- Logic

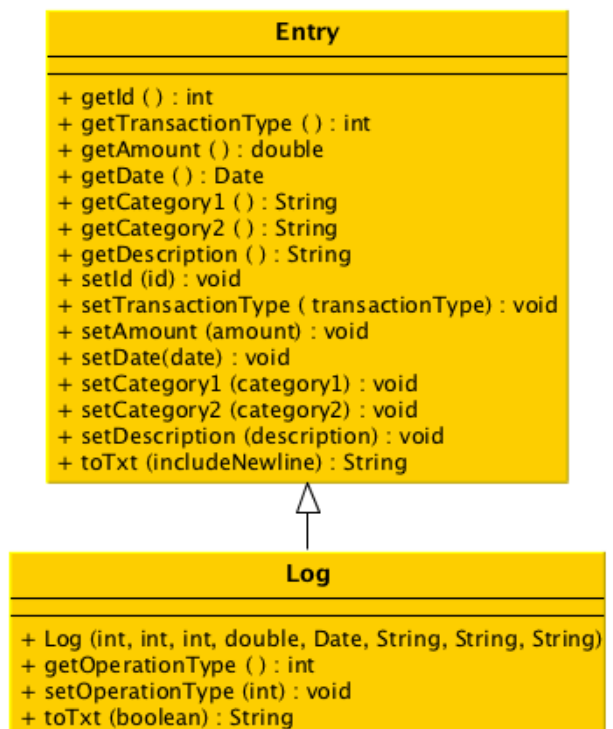
Logic



There are 3 Logic classes in EzXpns. They are CatMgr, EntryMgr and HistoryMgr. CatMgr extends to 4 sub-classes that are used to handle the 4 main types (Assets, Liabilities, Income, and Expenses).

- Data

Data



- Testing

We will be using JUnit for the testing procedures of View and Logic. Currently we are just using a blackbox approach to test for bugs in our program. During the next iteration, we will use write JUnit test-cases to do our testing.

3.3 Main Algorithms

- Create transaction algorithms

The standard parameters are int transactionType, double amount, Date date, String category1, String category2, String description.

Depending on the transaction type of the Entry, category 1 and category 2 changes accordingly as seen below.

Selection Index:


- 0 – Income Received (Addition to Assets, Addition to Income)
- 1 – Expenses by Assets (Subtraction from Assets, Addition to Expenses)
- 2 – Expenses by Credit (Addition to Liabilities, Addition to Expenses)
- 3 – Repaying Loan (Subtraction from Liabilities, Subtraction from Assets)
- 4 – Take Loan (Addition to Assets, Addition to Liabilities)
- 5 – Intra Asset Transfer (Subtraction from Assets, Addition to Assets)
- 6 – Intra Liability Transfer (Subtraction from Liabilities, Addition to Liabilities)

Thus after keying in the relevant information into the system, the relevant panel in the transactionMgr invokes the Asset/Liability/Expense/Income Category manager accordingly to update the latest category information. These category information are also displayed on the charts in the Finances Window.

No matter what is the transaction type, these Entries are to be recorded. Entries are created using EntryMgr with parameters passed down from the TransactionMgr.

The entry manager uses the function: addEntry(int transactionType, double amount, Date date, String category1, String category2, String description) to add an Entry into EntryList.txt. Algorithms for this will be: Gets the latest used ID of the last entry and adds +1 for the next Entry. A new Entry is invoked with parameters passed from transactionMgr. A scanner initialised to read Entries from the txt file. A buffered writer is initialised to write the Entry into the txt file. If scanner reads no Entries, writer writes Entry from start, else writer appends Entry. Closes scanner and writer, Updates current Entry ID

- **Read Algorithms**

When another Manager requires a particular Entry information, it calls on the EntryMgr's function: getEntry (int id 

Algorithm:

A buffered reading is initialised and used to read the lines in EntryList.txt file in the designated directory. An Entry variable to read the entry, line variable to read the Strings in the txt file, and a boolean variable to test whether the particular ID has been found.

```
while( there is a next line to read, read in new line){  
    Using a String tokenizer, look for delimiters of the special character, |, to  
    break the string into the format of an Entry, int transactionType,  
    double amount, Date date, String category1, String category2, String  
    description.  
    If the ID of the entry matches the searched ID, it returns the Entry  
}  
Closes buffered reader
```

- **Update Algorithm**

When another Manager requires to edit a particular Entry information, it calls on the EntryMgr's function: editEntry (Entry entry)

Algorithm:

An int variable is used to take note of the particular Entry ID what is to be updated.
A boolean variable, edited, to take note whether the entry has been edited already.
A buffered reader is used to read the information from the EntryList.txt.
A buffered writer initialised to write into a temporary txt file, tempFile.txt (this temporary txt file will have 100% of the information of the EntryList.txt, inclusive of the updated Entry)

```
While read next line is not null, {  
    Use string tokenizer to break the string into format of an Entry  
    If read Entry ID is not is not particular entry's ID,  
        append a direct copy of the Entry into tempFile.txt  
    Else  
        append the latest entry into tempFile.txt  
}  
Boolean variable, edited, is then set to true. Close buffered file reader and  
writer.  
Delete the original EntryList.txt, Rename tempFile.txt to EntryList.txt
```

- **Delete Algorithm**

When another Manager requires to delete a particular Entry information, it calls on the EntryMgr's function: deleteEntry (int id)

Algorithm:

A boolean variable, idFound, whether the entry ID is found
A buffered reader is used to read the information from the EntryList.txt
A buffered writer initialised to write into a temporary txt file, tempFile.txt
While read next line is not null, {

Use string tokenizer to break the string into format of an Entry

If read Entry ID is not is not particular entry's ID,
append a direct copy of the Entry into tempFile.txt

Else
set idFound to true

}

If idFound is false,

print error message and return null

Close buffered file reader and writer

Delete the original EntryList.txt,

Rename tempFile.txt to EntryList.txt

- **Search Algorithm**

SearchMgr (Search Manager) requires to search Entry information on certain parameters, it calls on the EntryMgr's function: **getTransactionList()**, to return all the Entries, and in which the search manager filters out the sorted Entries.

Search by TransactionType Algorithm:

Calls EntryMgr.getTransactionList() and stores to a LinkedList<Entry>
allEntries

Initialise a LinkedList<Entry> resultEntries

For all Entries {

If the transaction type of Entry matches the given transaction type,
add the entry to resultEntries}

Set the output label on the GUI how many entries has been found

Return resultEntries

Search by Date Algorithm:

Calls EntryMgr.getTransactionList() and stores to a LinkedList<Entry>
allEntries

Initialise a LinkedList<Entry> resultEntries

Format the day, month and year search parameters into date format

For all Entries {

If the date of Entry matches the input date,
add the entry to resultEntries

}

Set the output label on the GUI how many entries has been found

Return resultEntries

Search by Amount Algorithm:

Calls EntryMgr.getTransactionList() and stores to a LinkedList<Entry>
allEntries

Initialise a LinkedList<Entry> resultEntries



```

For all Entries {
    If the amount of Entry matches the input amount,
        add the entry to resultEntries
}
Set the output label on the GUI how many entries has been found
Return resultEntries

```

3.4 Project Process and Administration

Roles and responsibilities

Team Member	Role
Liu Bohua	Timekeeper (ensure we meet our deadlines) 
Lin Zhiyuan	Programme tester
Pang Kang Wei, Joshua	Programme code
Wong Jing Ping	GUI designer
Hao Wei	Documentations

Project Timeline

Week	Tasks
9	Implement good-to-have features. Do testing. Prepare for live demo
10	Continue to do whatever that is needed to be done
11	Prepare V0.2
12	Prepare V0.2

3.5 Development Infrastructure

These are tools that we are using in our development process:

Development Tool	Eclipse Juno & Windows Builder
Third-party Library	jFreeChart, charts4j
Versioning Tool	TortoiseHG & Google repository
Team communication & documentation tool	Google groups & Google docs

3.6 Glossary

The following table contains a list of terms and our specific interpretation in the context of our software

Term	Meaning
Asset	An asset has cash value, and can be converted into cash if necessary. Examples include a bank account, cash deposits, stock holdings, and even physical possessions of significant value such as property, land, or vehicles.
Liability	A liability is a form of debt, which has to be paid off. Examples include credit card debt, cash loans, and house loans.
Income	An income is a source that increases one asset. Examples include salary, dividends, gifts, rebates and handouts.
Expense	An expense is an accrual of expenditure. Here we use the GAAP (general accepted accounting practices) standard, which recognizes expenses as they are accrued, even before they are paid in cash. This allows for the accounting of debt, for example when the user spends beyond his or her present ability to pay. Examples include phone bills, money spent on meals, transport, rent, utilities and taxes. In more complicated scenarios, intangible losses such as depreciation of property or vehicles, or loss through stock trading can be represented as an expense.
Transaction	An exchange between the user and a third party involving some form of value. Examples include receiving a salary, spending cash on meals, paying bills with a credit card, or signing up for a loan. In more general cases (also for our case), the exchange can be within the user's own assets or liabilities. Examples include topping up one's ez-link card with cash, drawing cash from a bank's ATM, or paying a credit card bill with another credit card.
GUI	Graphical User Interface. It is what the user sees when he/she uses the program, and it is the means by which he/she gives interacts with the program.