# Lim Kang Yee – Project Portfolio for treasurerPro (tP)

## 1. Introduction

This project portfolio briefly introduces the project, treasurerPro and outlines my contributions to the project and showcases the key features I implemented.

### 1.1. About the team

My team of 5 members consists of 4 year 2 Computer Science Undergraduate students, including me, and another year 4 Computer Engineering Undergraduate student.

### 1.2. About the project

This project is part of the module 'Software Engineering Project CS2103T' where we were tasked to develop a basic command line interface desktop application by morphing or enhancing an existing AddressBook desktop application. Our team decided to incorporate and morph the AddressBook application as part of our all-in-one application which enables treasurers or members of Co-Curricular Activities (CCA) Clubs and Societies to manage their club finances, reimbursements, inventory and member's contact details easily. The duration of our project was 13 weeks.

### 1.3. Key to the icons and formatting used in the document

This symbol indicates extra information or definition.

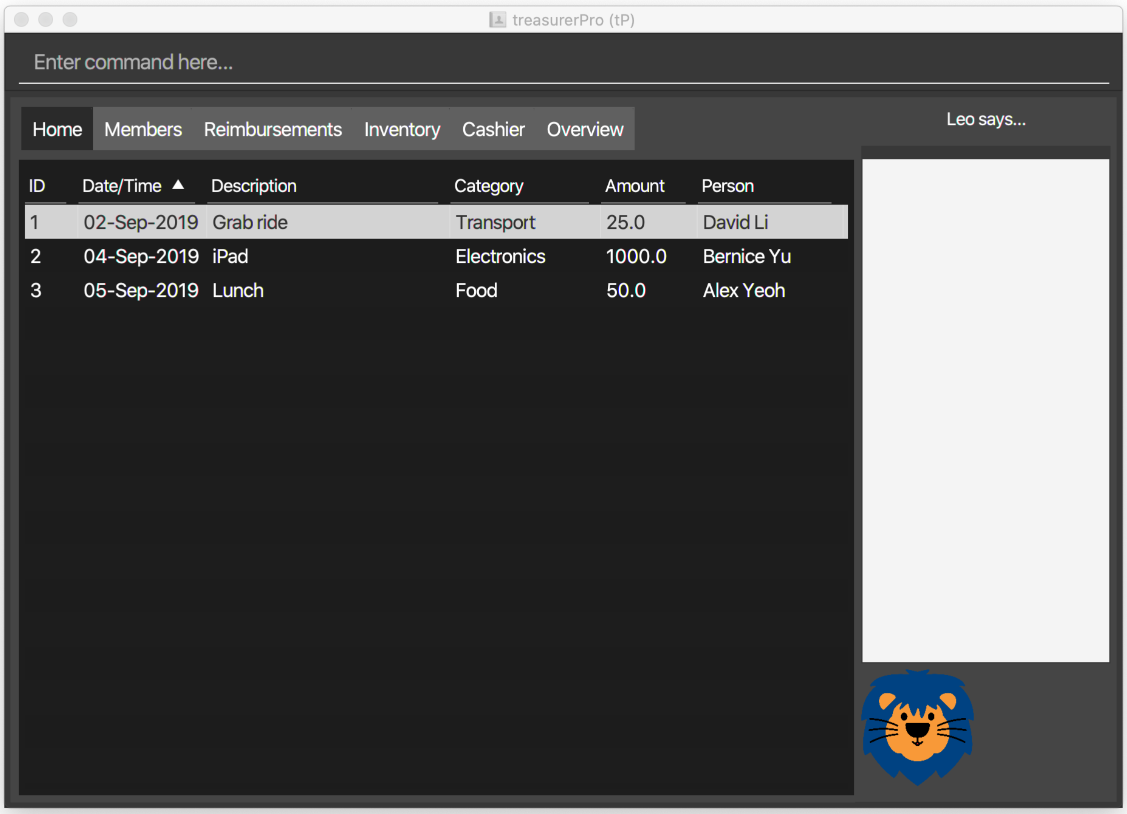
`Model` : Text with grey highlight indicates a component, class or object in the architecture of the application.

`command` : Text with blue font and grey highlight indicates a command that can be inputted by the user.

### 1.4. Introduction of treasurerPro

This desktop application consists of 6 tabs, a command box for users to input their commands and a response box for Leo, our lion mascot. Each tab serves a different purpose that helps treasurers and members better manage their club or Society's finances.

This is what our application looks like when it is first opened. (graphical user interface for treasurerPro):



| ID | Date/Time ▲ | Description | Category | Amount | Person |
|---|---|---|---|---|---|
| 1 | 02-Sep-2019 | Grab ride | Transport | 25.0 | David Li |
| 2 | 04-Sep-2019 | iPad | Electronics | 1000.0 | Bernice Yu |
| 3 | 05-Sep-2019 | Lunch | Food | 50.0 | Alex Yeoh |

*Figure 1. Graphical User Interface of treasurerPro*

## 2. Summary of contributions

My role was to design and write the code for the features of the Home Tab. The following sections shows a summary of these features, as well as the relevant documentation I have added to the user and developer guides in relation to these features.

### Enhancements

- Deleting Transactions of a Specific Member
  - What it does: The Delete Command allows all transactions linked to the specified member in the command to be deleted at once.
  - Justification: This feature allows the user to clear all transaction records linked to a specific member if the member has left the CCA since a member's details can only be deleted in the Members Tab if the member does not have existing transaction records.
  - Highlights: This feature is a faster way to delete transactions of a specific member, especially if the member has many transaction records. This works well with the deletion of member feature in Members Tab.
- Sorting Transactions in a Specific Order
  - What it does: Each transaction records consists of a date, description, category, amount of money and person accountable for the transaction. Thus, the Sort Command helps to sort the transactions by the alphabetical order of the person's name, by the date (from oldest to most recent) or amount (from smallest to largest).

    - Justification: It is useful for users to keep track of transactions and view the transaction records according to different priority.
    - Highlights: This command can be extended easily to allow for sorting of transactions in the reverse order or with other different orders.

**Code contributed**

Please click these links to view the code I have contributed for Home Tab: Functional Code , Test Code

**Other contributions**

- Community
    - Reviewed Pull Requests (with non-trivial review comments): (PR #98)
    - Added Detailed Explanation of Added Code in Pull Requests to Help Understanding By Other Members: (PR #42), (PR #201)
    - Helped to Debug Code in Other Member’s package: (PR #126)
- Integration
    - Integrated `Transaction Tab` with `Reimbursement Tab`: (PR #49)
    - Integrated the Original `AddressBook` into the `Members Tab` in Graphical User Interface: (PR #42)
    - Integrated the Edit and Delete Command of `AddressBook` with the Logic of `Transaction Tab`: (PR #49), (PR #85)
- Documentation
    - Added implementation details for Home Tab, Model Component and Storage Component to the Developer Guide: (PR #154), (PR #199)
    - Added to User Stories and Guide to Use the Home Tab to the User Guide: (PR#22), (PR #197)

# 3. Contributions to the User Guide

The following section shows my additions to the treasurerPro User Guide for the `Home Tab` features.

## 3.1. Current enhancement

{Start of First Extract from User Guide}

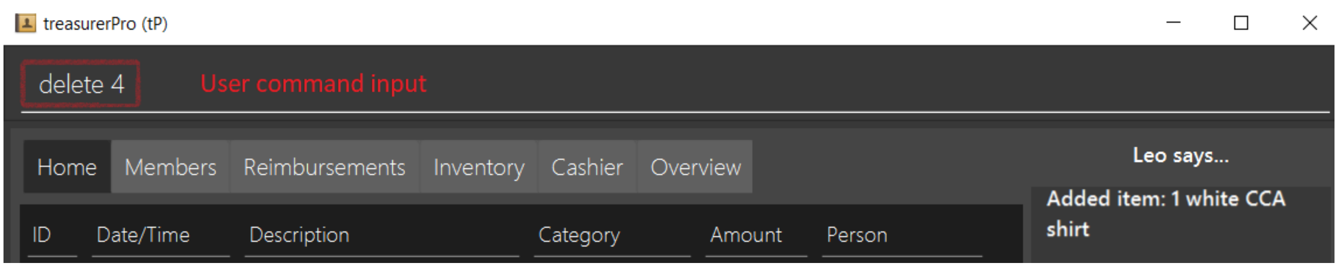
### 5.1.2 Delete Transaction(s)

This command allows you to delete either all transactions of a person or a single transaction of a specific ID from the table.

- Command: `delete ID` or `delete p/PERSON`
- Examples:
    - `delete 1`

    - `delete p/Alex Yeoh`

- Steps for Deleting by ID:
    1. Type the command with the ID of the transaction to be deleted as shown in the screenshot below:



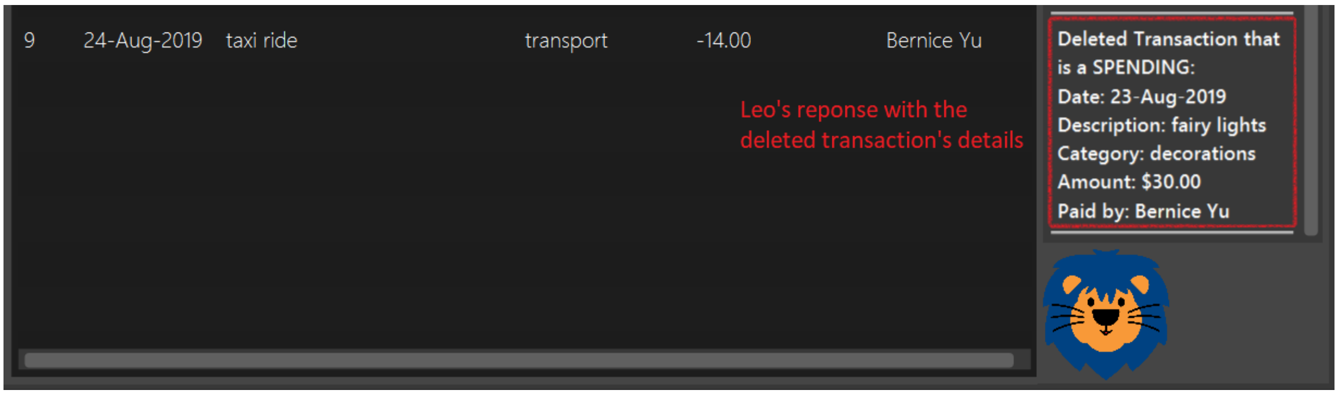
treasurerPro (tP)

delete 4

Added item: 1 white CCA shirt

2. Hit `Enter`.

Leo will respond with a success message and the transaction will be removed from the table as shown below:

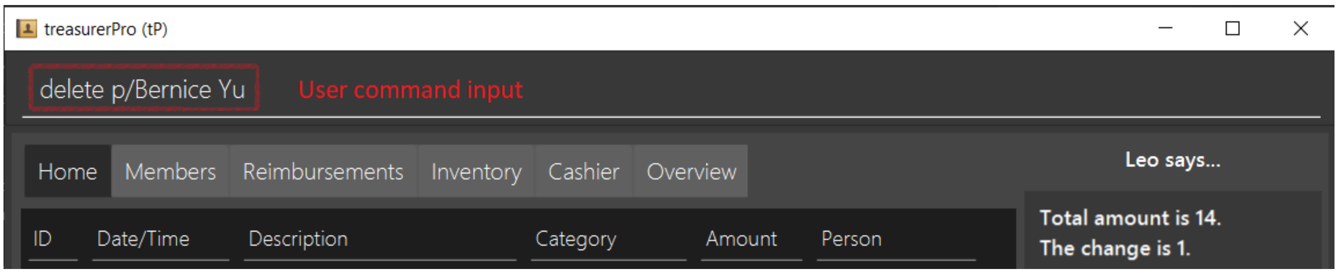


| | | | | | |
|---|---|---|---|---|---|
| 9 | 24-Aug-2019 | taxi ride | transport | -14.00 | Bernice Yu |

Deleted Transaction that is a SPENDING:
Date: 23-Aug-2019
Description: fairy lights
Category: decorations
Amount: $30.00
Paid by: Bernice Yu

*Figure 2. Screenshot of a successful user input for Delete by ID Command in Home Tab*

- Steps for Deleting by Person:
    1. Type the command with the person's name to delete all transactions related to that person, as shown in the screenshot below:



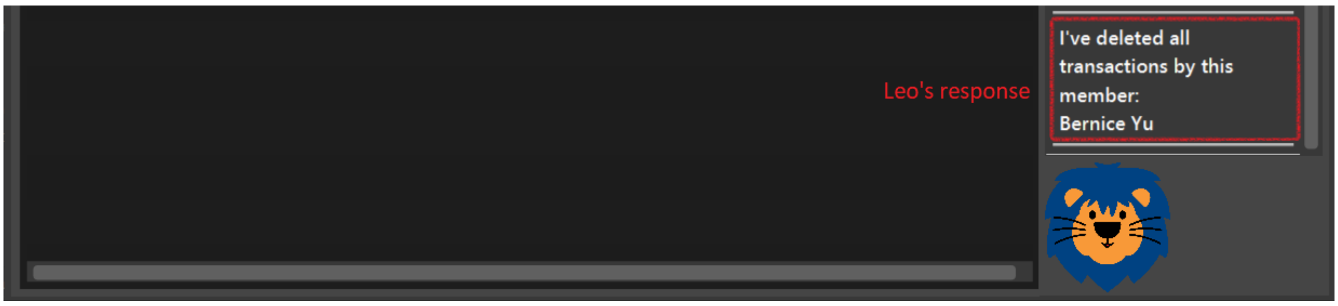
treasurerPro (tP)

delete p/Bernice Yu

Total amount is 14.
The change is 1.

2. Hit `Enter`.

Leo will respond with the success message and the transaction(s) will be removed from the table as shown below:



I've deleted all transactions by this member: Bernice Yu

For both delete commands, if the transaction(s) deleted was part of a pending reimbursement record, it will also be removed from that reimbursement record. On the other hand, if the person entered is not part of our database as shown in the Members Tab, Leo will respond with a message to inform you. If the person does not have any transactions, Leo will also respond with a message to inform you.

{End of First Extract}

{Start of Second Extract from User Guide}

### 5.1.4 Sort Transactions in the Table

This command sorts the table of transactions into a specified order for viewing and carrying out of subsequent commands.

- To sort:
  - By date (from oldest to most recent): `sort date`
  - By name (from alphabetical order of name): `sort name`
  - By amount (from smallest to largest in amount): `sort amount`
  - Undo sort: `sort reset`

The undo sort command allows you to view the table of transactions in the order originally shown when the application was initially opened.

{End of Second Extract}

# 4. Contributions to the Developer Guide

The following section shows my additions to the treasurerPro Developer Guide for the `Home Tab` features.

## 4.1. Current enhancement

{Start of Extract from Developer Guide}

### 3.1.2 Delete Feature

This section explains the implementation of the Delete Command in Home Tab. This feature allows for 2 types of deletion, by the index shown in the table or by the person’s name. Inputting the person’s name will cause all transactions linked to that person to be deleted.

The following sequence diagram shows how the delete by name command works which is referenced in 2.3. Logic component: Figure 5:

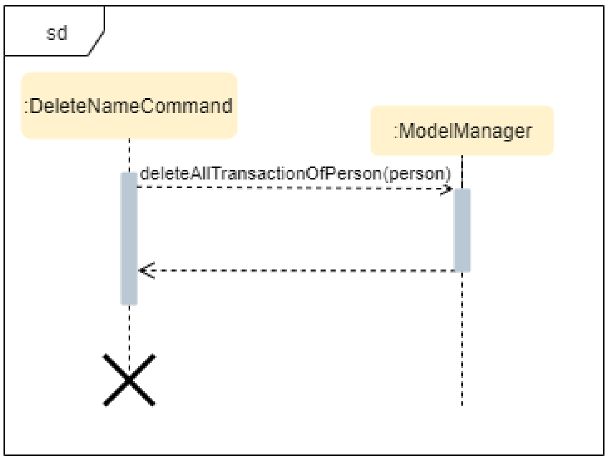


*Figure 16. Sequence Diagram of Delete Command in Home Tab (transaction package)*

In addition, the `ModelManager#resetPredicate()` method is not called in the `DeleteNameCommand`. Thus, the UI table will continue to show the filtered transaction list. If the prior input is a Find Command and the list at the start of the activity diagram shows a filtered list by the Find Command’s keywords, it will continue to show the filtered list at the end of the command. To view the full transaction list, the user would be required to input the Back Command where `BackCommand` calls `resetPredicate()`. The sequence diagram for the `BackCommand` is shown in the following section 3.1.3 BackCommand

After this, the list of transactions and reimbursement tab is updated as shown in 3.1.1. Add Command feature: Figure 13 and 3.1.1. Add Command feature: Figure 14 respectively. The delete by index implementation would be similar but does not require interaction with the `Model` from the `AddressBook` in the person package. The following activity diagram shows the steps needed to delete a new transaction:

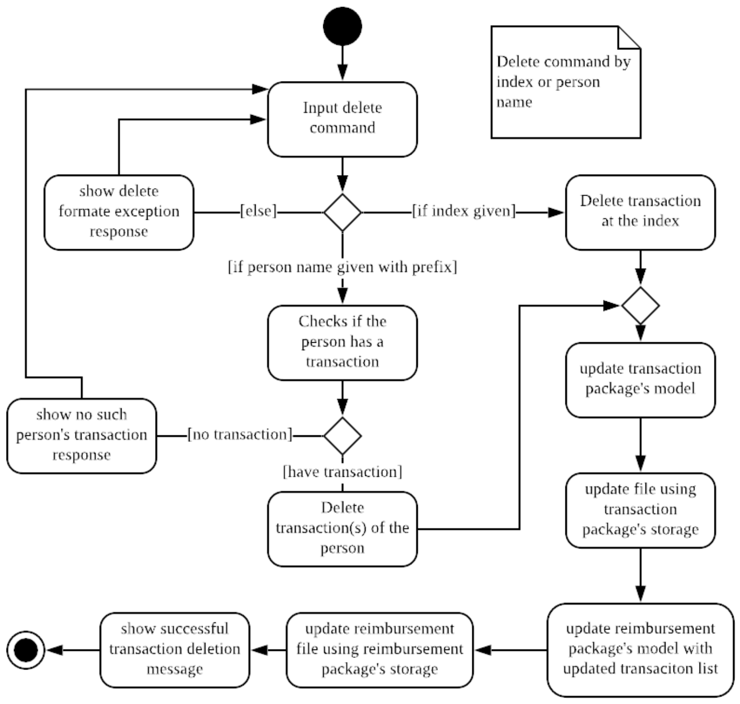

Delete command by index or person name
Input delete command
show delete formate exception response
[else]
[if index given]
Delete transaction at the index
[if person name given with prefix]
Checks if the person has a transaction
update transaction package's model
show no such person's transaction response
[no transaction]
[have transaction]
Delete transaction(s) of the person
update file using transaction package's storage
show successful transaction deletion message
update reimbursement file using reimbursement package's storage
update reimbursement package's model with updated transaciton list

*Figure 17. Activity Diagram of Delete Command in Home Tab (transaction package)*

The above activity diagram assumes the index to be within the bounds of the table but if it is not, a response will be shown about the incorrect input. Also, as shown above, responses will be shown to indicate if an input is incorrect or when a successful deletion is done.

### 3.1.3 Back Command Feature

This section explains the implementation of the Back Command feature in Home Tab. The `BackCommand` is not initialised by a specific command parser as shown in as shown in 2.3. Logic component: Figure 5 but initialised by the `TransactionTabParser` instead. The following detailed sequence diagram shows how the back command works:

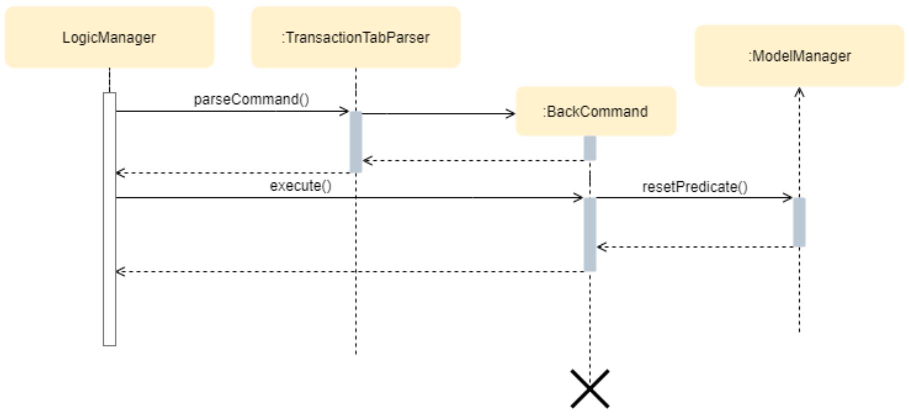


*Figure 18. Sequence Diagram of Back Command in Home Tab (transaction package)*

### 3.1.4 Sort Command Feature

This section explains the implementation of the Sort Command feature in Home Tab. The `SortCommand` allows for 3 types of sort, by name in alphabetical order, by amount (from least to most) and by date (from oldest to most recent).

The following sequence diagram shows how the sort command works:

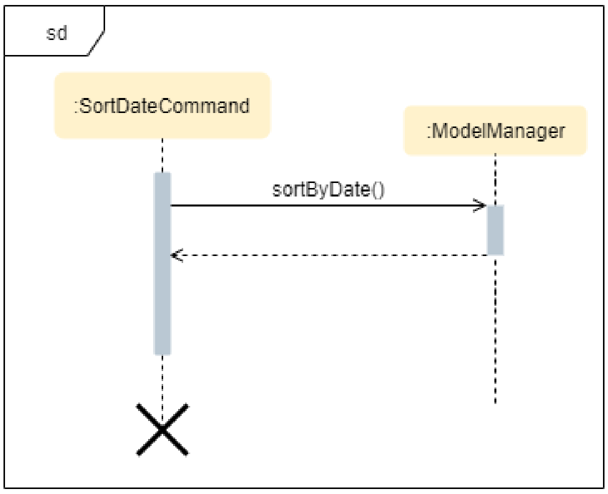


*Figure 19. Sequence Diagram of Sort Command in Home Tab (transaction package)*

When a user inputs the sort command, it is only checked that it is one of the 3 types or it will show a response about the incorrect user input. When it is successfully sorted, there will also be a response message shown.

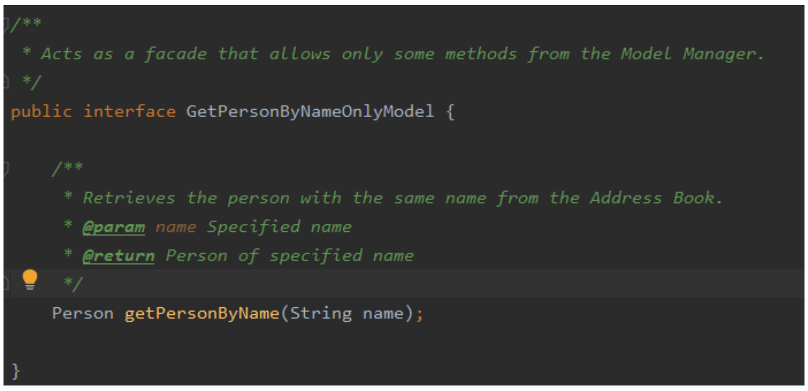
Similar to the Delete Command, the `resetPredicate()` method in `ModelManager` is not called.

### 3.1.5 Overall Design Considerations

This section's table explains the design considerations for some implementations in the Home Tab.

| **Alternative 1** | **Alternative 2** | **Conclusion and Explanation** |
|---|---|---|
| `ModelManager` contains 2 variables that point to a `TransactionList` object in original order and a `TransactionList` object for viewing that can be sorted such that when `sort reset` is called, the shown `TransactionList` can be set to be equals to the original one. | `ModelManager` contains only the shown `TransactionList` that can be sorted and reads from the data file to get the `TransactionList` object in original order when `sort reset` is called. | Alternative 1 was implemented. Alternative 1 allowed exporting of the data file in the desired order anytime while treasurerPro was running while alternative 2 meant that the data file would be updated only when treasurerPro is exited. The implementation is shown in Figure 21 |
| The Members Tab's `Model` interface is passed as parameters into Transaction Tab's `Logic` to give `Logic` access to all public methods of `ModelManager`. | A new interface is made to allow the only used method of Members Tab's `ModelManager` to be accessed in Transaction Tab's `Logic`. | Alternative 2 was implemented. The new interface acts as a facade for `ModelManager` which prevent unwanted modifications to `AddressBook`. The interface implemented is `GetPersonByNameOnlyModel` as shown in Figure 20 |

| Alternative 1 | Alternative 2 | Conclusion and Explanation |
|---|---|---|
| An `ArrayList` is used to store `Transaction` objects in `TransactionList`. | A `LinkedList` is used to store `Transaction` objects in `TransactionList`. | Alternative 1 was implemented. An `ArrayList` has better performance for the set and get methods than a `LinkedList` which would beuse frequently in `ModelManager`. |

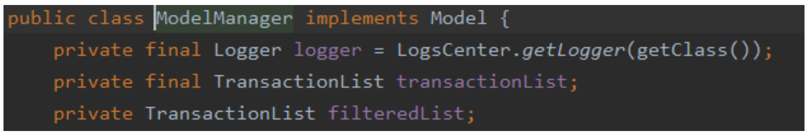


```java
/**
 * Acts as a facade that allows only some methods from the Model Manager.
 */
public interface GetPersonByNameOnlyModel {

    /**
     * Retrieves the person with the same name from the Address Book.
     * @param name Specified name
     * @return Person of specified name
     */
    Person getPersonByName(String name);

}
```

*Figure 20. Code Snippet of `GetPersonByNameOnly` facade class for `ModelManager` from Members Tab*



```java
public class ModelManager implements Model {
    private final Logger logger = LogsCenter.getLogger(getClass());
    private final TransactionList transactionList;
    private TransactionList filteredList;
```

*Figure 21. Code Snippet of `ModelManager` class with 2 `TransactionList` objects*

{End of Extract}