Winter 2023

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Issue** | **Description** | **Author** |
| February 13, 2023 | Assignment 2 | Adding Glossary for assignment1 | Zaki Hasan Ali |
| February 13, 2023 | Assignment 2 | Adding system actors | Zaki Hasan Ali |
| February 15, 2023 | Assignment 2 | Adding system use cases | Zaki Hasan Ali |
| February 15, 2023 | Assignment 2 | Adding use case diagram | Zaki Hasan Ali |
| March 4, 2023 | Assignment 3 | Produce a list of candidate classes | Zaki Hasan Ali |
| March 4, 2023 | Assignment 3 | Filtering the list of candidate classes | Zaki Hasan Ali |
| March 4, 2023 | Assignment 3 | Produce a potential class diagram | Zaki Hasan Ali |
| March 4, 2023 | Assignment 3 | Identified potential class attributes | Zaki Hasan Ali |
| March 4, 2023 | Assignment 3 | Revised the diagrams | Zaki Hasan Ali |
| March 5, 2023 | Assignment 3 | Update the glossary | Zaki Hasan Ali |
| March 5, 2023 | Assignment 3 | Create the database schema | Zaki Hasan Ali |
| April 5, 2023 | Assignment 4 | Create Sequence Diagram for Manager Groups | Zaki Hasan Ali |
| April 5, 2023 | Assignment 4 | Detailing the Operations of Manager Groups | Zaki Hasan Ali |

Table of Contents

[iFINANCE System Glossary 4](#_Toc131803932)

[1.1 Introduction 4](#_Toc131803933)

[1.2 Glossary 4](#_Toc131803934)

[iFINANCE System Actors 5](#_Toc131803935)

[iFINANCE System Use cases 6](#_Toc131803936)

[iFINANCE System Use Case Diagram 7](#_Toc131803937)

[List of Candidate Classes 8](#_Toc131803938)

[Potential iFINANCE Classes 9](#_Toc131803939)

[Potential iFINANCE class diagrams (entity classes only) 12](#_Toc131803940)

[Revised iFINANCE UML class diagram 13](#_Toc131803941)

[iFINANCE Sequence Diagram Manager Groups 16](#_Toc131803942)

[Detailing the Operations of Manager Groups 18](#_Toc131803943)

# iFINANCE System Glossary

## Introduction

This document is used to define terminology specific to the problem domain, explaining terms, which may be unfamiliar to the reader of the use-case descriptions or other project documents. Often, this document can be used as an informal *data dictionary*, capturing data definitions so that use-case descriptions and other project documents can focus on what the system must do with the information.

## Glossary

The glossary contains the working definitions for the key concepts in the iFINANCE System.

|  |  |
| --- | --- |
| Term | definition |
| Authentication | Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. |
| User account | A user is a person who uses iFINANCE system. Each user should have an account in order to be identified by iFINANCE. To login to an account, a user is typically required to authenticate himself/herself with a password or other credentials for the purposes of accounting, security, logging, and resource management. |
| Double-entry bookkeeping | It is a system of accounting in which every transaction has a corresponding positive and negative entry (debits and credits). |
| Assets | Tangible and intangible items that the company owns that have value (e.g. cash, computer systems, patents). |
| Liabilities | The money that the company owes to others (e.g. mortgages, vehicle loans). |
| Income | The money the company earns from its sales of products or services, and interest and dividends earned from marketable securities. |
| Expenses | The money the company spends to produce the goods or services that it sells (e.g. office supplies, utilities, advertising). |
| Chart of Accounts | A chart of accounts is a listing of the names of the accounts that a company has identified and made available for recording transactions in its general ledger. A company has the flexibility to tailor its chart of accounts to best suit its needs, including adding accounts as needed. |
| Master Accounts | The Master Accounts are a super-set of all allowable accounts in the chart of accounts. Transactions are never posted to the Master Accounts. |
| UI component | UI stands for User Interface. It is a junction between a user and a computer program. An interface is a set of commands or menus through which a user communicates with a program. |
| PDF Document | Portable Document Format (PDF) is a file format used to present and exchange documents reliably, independent of software, hardware, or operating system. |
| Windows-based computers | A personal computer powered by Microsoft Windows operating system in a form intended for regular use at a single location desk/table due to its size and power requirements. |
| Database Schema | The term schema refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases). |

# iFINANCE System Actors

The first useful step to analyze the system functionality is to look in the problem statement at the things that interact with the system. In UML use case analysis, these external things are called **actors**. Actors are identified based on the following:

* Actors are always external to the system – they are therefore outside our control.
* Actors interact directly with the system.
* Actors represent roles that people and things play in relation to the system, not specific people or specific things.
* Each actor has a unique name and description.

|  |  |
| --- | --- |
| Actor | Description |
| iFINANCE User | A general user interacting with iFINANCE system. This general user can be the day-to-day regular user (we called non-admin user) or the system administrator. In order to this general user to use iFINANCE, he/she needs to successfully login to the system. |
| Administrator | A special type of iFINANCE User who has the privileges to add user accounts into the system, edit and delete their profiles. The administrator user account will be shipped with the system. |
| Non admin user | A day-to-day iFINANCE user who wish to use iFINANCE system to control the his/her personal finances, keeping track of bank accounts, cash, credit cards, and investment accounts. |

# iFINANCE System Use cases

|  |  |
| --- | --- |
| Use Case | Description |
| Authenticate User | iFINANCE shall provide a functionality to authenticate its user by using a username and encrypted password. Once the user has logged in, the set of controls dynamically changes to match the permissions of this account. Note that, each user should be successfully authenticated before using any of iFINANCE functionalities. |
| Manage User Account | iFINANCE shall provide the functionality for the system administrator to create user accounts for the regular iFINANCE users according to a set of access controls predefined for each user type. The administrator account itself is shipped with the system. |
| Change Password | iFINANCE shall provide the functionality for its users to change their secret passwords. |
| Manage Account Group | iFINANCE shall provide the ability for its non-admin users to add, update, or delete a custom sub-categories (called Groups). This includes groupID, groupName, groupElement, i.e., Assets, Liabilities, Income, and Expenses, and groupParent. Each group can be decomposed to further level of sub-groups (the groupParent field is responsible to maintain the parent-children link). |
| Manage Chart of Accounts | Manage Chart of Accounts UC shall provide a graphical user interface GUI to help non-admin users to add, update, or delete Master Account (e.g., cash, bank, and credit card). Each Master Account in the Chart of Accounts record will include accountID, accountName, openingAmount, closingAmount, and accountGroup (to specify a group/sub-group to which the account belongs). |
| Manage Double-Entry  Transactions | iFINANCE shall provide a tool for its non-admin users to manage accounts’ transactions (called double entry bookkeeping or double entry accounting). Each transaction will always have two effects: a debit entry and a credit entry.  This tool will facilitate the non-admin user to add, update, or delete a transaction's entry in a Master-Detail style form. The Master part of the form includes the transaction number, date, the total amount of debit, the total amount of credit in addition to a comments/notes field. The Detail part of this form is a grid that has in each line the account number and name, the debit amount, the credit amount and a field of comments. |
| Generate Reports | iFINANCE shall allow the non-admin users to generate valuable financial reports including the Trial Balance, Balance Sheet, a Profit and Loss statement, or a cash flow statement from the user financial data.  This UC includes two UCs Show Results on Screen and Print out results in PDF files. |
| Show Results on Screen | This is a supplier UC for the Generate Reports UC that shall allow the non-admin users to see the generated reports in an easy to see in the computer screen. |
| Print out results in PDF files | This is a supplier UC for the Generate Reports UC that shall allow the non-admin users to export the generated reports in a formatted style to PDF file type. These PDF files can be printed out later on. |

# iFINANCE System Use Case Diagram

Diagram

Description automatically generated

# List of Candidate Classes

bank accounts

cash

credit cards

investment accounts

users

deposits

expenses

reports

financial health

double-entry

people

accounting knowledge

financial accounting categories

assets

liabilities

income

expenses

debits type

credits type

financial transactions

user-defined sub-categories

balance sheet

profit and Loss statement

cash flow statement

financial data

master accounts

money

desktop computers

tablet PCs

clubs

self-employed

small businesses

home

user name

encrypted password

access controls

permissions

system administrator

privileges

user accounts

profiles

administrator account

custom group

main form

groupID

groupName

groupElement

groupParent

level of sub-groups

parent-children link

Chart of Accounts

UI component

accountID

accountName

openingAmount

closingAmount

accountGroup

report

master-detail style form

grid

line

account number

account name

the debit amount

the credit amount

field of comments

trial balance

cash flow statement

tabular layout

# Potential iFINANCE Classes

|  |  |  |
| --- | --- | --- |
| Class name | Type | Brief Description |
| UserLoginForm | Boundary | The UserLoginForm will be displayed by the iFINANCE system asking the user to enter its username and password to be authenticated. |
| iFINANCE User | Entity | The common information of the iFINANCE non admin users and the iFINANCE admin such as User ID and User’s name will be maintained by this entity supper class. |
| Non admin user | Entity | Non admin user is a specification class inherits the iFINANCE User class and add specific information about the user (user profile) such as name, address and email. |
| Administrator | Entity | iFINANCEAdmin is a specification class inherits the iFINANCE User class and add specific information about the system administrator such as the creation date of the account. |
| UserAccount | Entity | The UserAccount class store the iFINANCE user account information that include a password related information like an encrypted password of a user, the day the password expires, whether or not the password has to be changed from time to time, the minimum and maximum time between password changes, etc. |
| UserLoginController | Control | A control class to accept and validate the iFINANCE username and its corresponding password. It takes the decision whether the iFINANCE user is allowed to access the iFINANCE services or not. |
| ManageUserAccountsForm | Boundary | iFINANCE displays the ManageUserAccountsForm when the system admin initiates the process for adding/modifying iFINANCE user account. This form will help the admin to enter the user general information like name and addresses and the secret information like password and the associated access roles. |
| ManageUserAccountsController | Control | A control class to manage (create, modify, delete) iFINANCE users’ information. It also determines whether the entered username has been used by other user or not, it encrypts the password before storing, and finally assign a specific system access role to the user. |
| ChangePasswordForm | Boundary | The ChangePasswordForm will be displayed by the iFINANCE system when the non admin user ask to change his/her password. This form will ask the user to enter the old password and the new password two times for confirmation. |
| ChangePasswordController | Control | A control class to accept and validate the non admin user old and new password. The old password needs to be related to this user and the two new passwords should match each other. This control will update the UserAccount entity class. |
| CustomGroupForm | Boundary | CustomGroupForm is a boundary class that show all exiting financial group names in a Tree View style control, allowing the user to edit, add, or delete a groups and subgroups. |
| Group | Entity | Group class defines the basic information of the system financial group names and its parent-child relationship. It also determines the type of the group whether it is an Assets, Liabilities, Income, or Expenses. |
| AccountCategory | Entity | An entity class that defines the account types such as Income, Expense, Asset. |
| CustomGroupController | Control | This class defines the attributes and the methods needed to retrieve the information of all account groups and subgroups names from the Group objects and display it in a TreeView control in CustomGroupForm organized by the main account type categories. It also allows for edit, add, or delete functionalities using the same form. |
| ChartOfAccountForm | Boundary | A boundary class to define a graphical user interface GUI element that display a list of all Master accounts at iFINANCE information such as name, group, type, closing and opening amount. The form allows the user to add new master account, edit or delete exiting ones. |
| MasterAccount | Entity | This entity class defines the basic information of the system financial Masters Accounts such as name, group, type, closing and opening amount. |
| ChartOfAccountController | Control | This class defines the attributes and the methods needed to retrieve the existing master accounts data and display it in a grid control in ChartOfAccountForm. It allows the user to add new master account, edit or delete exiting ones using the same form. |
| iFINANCETransactionForm | Boundary | iFINANCE provides a that facilitate the user to add, update, or delete a transaction's entry in a Master-Detail style form. The iFINANCETransactionForm consists of a master part that includes the transaction number, date, the total amount of debit, the total amount of credit in addition to a comments/notes field. The Detail part of this form is a grid that has in each line the account number and name, the debit amount, the credit amount and a field of comments. |
| Transaction | Entity | This entity class defines the data involved in master part of the iFINANCETransactionForm. This data includes: the transaction number, date, the total amount of debit, the total amount of credit in addition to a comments/notes field. |
| TransactionLine | Entity | This entity class defines the data involved in detail part of the iFINANCETransactionForm. This data includes: the account number and name, the debit amount, the credit amount and a field of comments in a grid format. |
| iFINANCETransactionController | Control | A control class to define the application logic that add, edit, or delete accounts’ transactions (called double entry bookkeeping or double entry accounting). In each transaction it will recognize one of two processes: a debit entry and a credit entry, such that for every Debit entry, there will always be an equal Credit entry. In the Debit entries we do one of the following: increase assets account, increase expense account, decrease liability account, or decrease income account. However, In the Credit entries we do one of the following: decrease assets account, decrease expense account, increase liability account, or increase income account. iFINANCE deals with two basic transactions—deposits and withdrawals. |
| GenerateReportForm | Boundary | iFINANCE provides their non admin users with a form represented by the GenerateReportForm boundary class by which the user can enter specific criteria and option to generate reports such as Trial Balance, Balance Sheet, a Profit and Loss statement, or a cash flow statement. |
| TrialBalanceReport | Boundary | TrialBalanceReport class presents the closing balance of all master accounts in a tabular layout with the debit accounts on one side and the credit accounts on the other. The sum of all credit balances should always match the sum of all debit balances. The Trial Balance is the basis of preparing the Profit and Loss account and the Balance Sheet. |
| BalanceSheetReport | Boundary | BalanceSheetReport class shows Assets plus Inventory on one side and Liabilities + Profit or Loss (as derived above) on the other. These two sides should be equal. |
| ProfitLossStatement | Boundary | ProfitLossStatement class is all about subtracting all Expenses from Income to derive a Profit or Loss figure. Thus: Profit or (Loss) = Income – Expenses. |
| CashFlowStatement | Boundary | CashFlowStatement is a boundary class that shows how changes in balance sheet accounts and income affect cash and cash equivalents, and breaks the analysis down to operating, investing and financing activities. |
| GenerateReportController | Control | This control class defines the methods that read the user options in which report they would like to generate and to read the specific criteria that is related to the chosen report, such as date range, account types, or account categories. Then calculate and generate the desire report. It also determines whether the output will be displayed on the screen or saved as PDF. |

# Potential iFINANCE class diagrams (entity classes only)

**Diagram

Description automatically generated**

# Revised iFINANCE UML class diagram

1. **iFINANCEUser Class**

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Brief Description |
| ID | int | This is the primary key of the iFINANCE user info class. Every registered iFINANCE user is assigned automatically a sequential unique user ID. |
| name | String | The name attribute stores the full name of the iFINANCE user. |

1. **NonAdminUser Class**

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Brief Description |
| address | String | The address attribute stores the address of the iFINANCE non admin user. |
| email | String | The email attribute stores the email address of birth of the iFINANCE non admin user. |

1. **Administrator Class**

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Brief Description |
| dateCreated | String | The date when the administrator account is created in iFINANCE’s database. |

1. **UserAccount Class**

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Brief Description |
| userName | String | The name attribute is the primary key to store the use account name of the iFINANCE user. |
| encryptedPassword | String | The userEncryptedPassword attribute stores the encrypted version of the non admin user password. A salted hash will be used in order to encrypt the password (optional). |
| passwordExpiryTime | Int | From time to time the system requires the user to change the password. The passwordExpiryTime attribute stores this period of time. |
| userAccountExpiryDate | String | The userAccountExpiryDate attribute stores the expiry date of the user account if any. |
| accountType | String | To determine whether the account is for admin or non-admin user |

1. **AccountCategory Class**

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Brief Description |
| name | String | The name attribute is the primary key of the AccountCategory class and stores the iFINANCE pre-defined account category name. Example, Assets, Liabilities, Income, and Expenses. |
| type | String | The type attribute stores the type of account category, this could be Debit or Credit. |

1. **Group Class**

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Brief Description |
| ID | int | This is the primary key of the Group class. The value of this attribute is a sequential unique number generated automatically to distinguish one account group from the other. |
| name | String | The name attribute stores the name of the group or its subgroups. |

1. **MasterAccount Class**

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Brief Description |
| name | String | The name attribute is the primary key of the master account class, stores the full name of the master account. |
| openingAmount | double | The openingAmount attribute stores the balance brought forward at the beginning of an accounting period. |
| closingAmount | double | The openingAmount attribute stores the amount remaining in an account within the chart of accounts, positive or negative, at the end of an accounting period or year end. |

1. **Transaction Class**

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Brief Description |
| ID | int | This is the primary key of the transaction header class. The value of this attribute is a sequential unique number generated automatically and links the transaction line class objects. |
| date | String | The date attribute stores the date and time of the transaction. |
| description | String | The description attribute stores the detail comments about the transaction. |

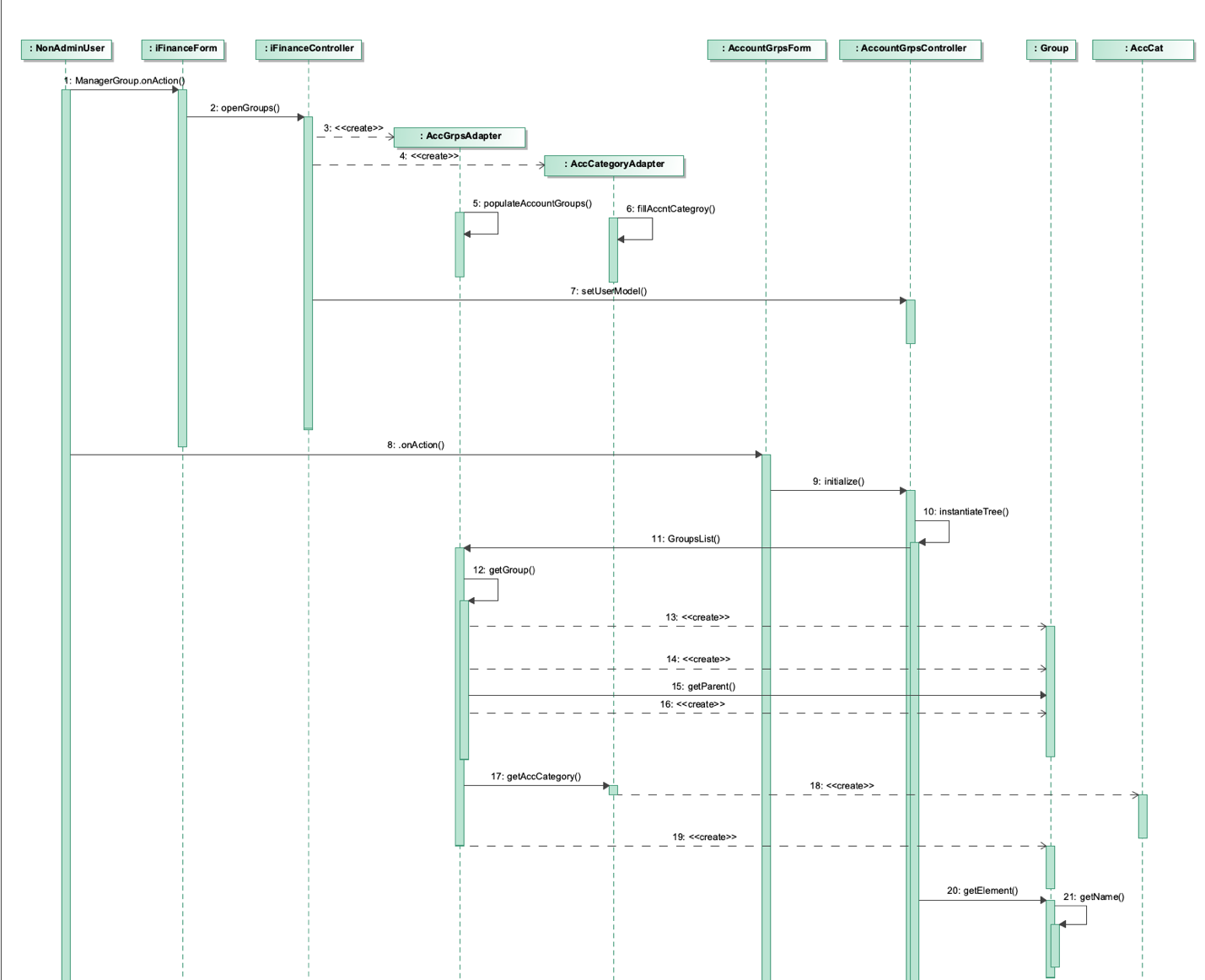
1. **TransactionLine Class**

|  |  |  |
| --- | --- | --- |
| Attribute name | Type | Brief Description |
| ID | int | This is the primary key of the TransactionLine class. The value of this attribute is a sequential unique number generated automatically. |
| creditedAmount | double | The creditedAmount attribute stores the credited amount of the transaction. |
| debitedAmount | double | The debitedAmount attribute stores the debited amount of the transaction. |
| comments | String | The comments attribute stores the detail comments about the transaction. |

**Diagram

Description automatically generated**

# iFINANCE Sequence Diagram Manager Groups



Chart

Description automatically generated with low confidence

Graphical user interface

Description automatically generated with low confidence



# Detailing the Operations of Manager Groups

|  |  |  |
| --- | --- | --- |
| **Class Name** | **Receiving Message (operation)** | **Brief Description** |
| iFinance-view | onAction() | When the user clicks on the iFinance-view button, this triggers and opens the iFinance-view fxml. |
| iFinanaceController | openGroups() | This method opens the ManagerGroup form, where users can manage groups by adding, modifying, or deleting them |
| AccGrpsAdapter | updateGroup | This void method updates the name of a group in the SQL table and takes in two string variables: the old group name and the new group name. |
| AccGrpsAdapter | createGroup | This void method adds groups to the SQL table and takes in a string name, int parent, and string element as parameters. |
| AccGrpsAdapter | getID | This method returns the ID of a group based on its name, taking in a string name as a parameter. |
| AccGrpsAdapter | GroupsList | This method returns an ArrayList of group objects used to convert initial groups into group objects and put them in an array list. It takes in no parameters. |
| AccGrpsAdapter | delGroup | This void method deletes a group from the SQL table based on its ID and takes in an int ID as a parameter. |
| AccGrpsAdapter | getGroup | Given an int ID, this method returns a group object. It takes in the ID as a parameter. |
| AccGrpsAdapter | fillAccGrps | This void method populates the AccountGroups table with the initial groups and takes in no parameters. |
| AccGrpsAdapter | getElement | Given a string name, this method returns the element of a group. It takes in the name as a parameter and returns a string. |
| AccountCategoryAdapter | fillAccCategory() | This void method populates the AccountCategory table with the four different types of account categories and takes in no parameters. |
| AccountCategoryAdapter | getAccCategory() | Given a string name, this method returns an AccountCategory object and takes in the name as a parameter. |
| AccountGroups-view | onAction() | This method is triggered when the user clicks on the AccountGroups-view button, which opens the AccountGroups-view form. |
| AccountGroupsController | setUserModel() | This void method sets the AccountGroupsAdapter and AccountCategoryAdapter for the controller to access the SQL tables and takes in the two adapters as parameters. |
| AccountGroupsController | loacteTreeItem() | Given a TreeItem and a string name as parameters, this method recursively searches the tree for the given item and returns it when found. It returns a TreeItem. |
| AccountGroupsController | remAndHideTreeItem() | This void method removes a tree item and stops showing it in the treeview. It takes in a TreeItem and TreeView. |
| AccountGroupsController | Save() | This is a parameterless void method that gets triggered when the user clicks on the save button. Its functionality involves creating a new group or modifying an existing group's name based on the option selected in the selectItem() method. |
| AccountGroupsController | selecItem() | This method is triggered when a user clicks on an item in the tree view. It is a void method that doesn't take any parameters. Depending on the clicked item, a context menu is displayed, offering options to add a new group, modify the group name, or delete the group. The method then redirects the cursor's focus to a titled pane, depending on the selected menu item. |
| AccountGroupsController | fillTree() | This void method adds the initial groups to the treeView and takes in no parameters. |
| AccountGroupsController | Initialize() | This void method initializes the form, disables some FXML elements, and calls the populateTree method when the form is opened. It takes in URL and ResourceBundle. |
| AccountGroupsController | Exit() | This method will close the form for the AccountGroups-view. |
| Group | Group() | To create a new group object, this constructor method requires an integer, a string, a Group, and an AccountCategory as parameters. |
| Group | getElement() | With no arguments provided, this method returns an AccountCategory object. |
| Group | getParent() | With no arguments provided, this method returns a Group object. |
| Group | getName() | By invoking this method with no arguments, it returns the string name associated with the group. |
| AccountCategory | getName() | To retrieve the name of the AccountCategory, use this method. It returns the name as a string and does not require any parameters. |