

WIX1002:Fundamentals of Programming

Assignment 1

Personal Finance System

TAN JO YI (WEM150029)

TAN AI LOON (WEM150028)

HIBA MOHAMMAD NOOR (WEM150702)

SEVELEN JAIMIN (WED150021)

LECTURER : DR. YAMANI IDNA BIN IDRIS

# Features of the system

How much time do you spend managing your finances? Just the day-to-day stuff – paying bills, transferring money from one account to another, and at cetera. If it’s more than an hour, a week or even a month, one will face difficulty in managing their own financial activities. Hence, we as a software programmer have created a software to ease the user to handle their own money. In addition, the system also focus on giving you specific money management tips such as advice on handling irregular income or expense. Here are the features of the system :-

1. Provide your log-in details and it fetches the transactions from your accounts. The Accounts screen shows your accounts and balances. You can see recent transactions for the last week, 30 or 90 days. Many transactions are automatically categorised, such as cash withdrawals, credit card payments, and utilities like gas and electric.
2. Analyses and categorises transactions and then lets you view each category. You can view both incoming and outgoing money. It does not predict future balances based on income and spending and you cannot set budgets for categories to help you to take control of future spending. It is still a reasonably good service though and its simplicity is its strongest feature.

**Login module**

This is a module used to determine whether the username and password entered by user matches with the ones which resides in the NameList.txt . Only the corresponding usernames with its respective password can gain access to further data linked with that account.

**Expected outcome**

**-Login status**

Among the methods that exist in this class are:

**LoginModule()**

* Empty Constructor

**LoginModule(String username , String password)**

* Not used in this case

**LogInMenu()**  (Return type: void)

* Method used to display the front page once the user uses this application.

**userLogIn()**  (Return type: void)

* Method used to accept the name and password from the user

**getUsername()**  (Return type: String)

* Method used to return the username accept in <method>userLogIn</method>

**databaseCheck()**   (Return type: void)

* Method used to check whether the userdatabase file exists . If it does not exist, a new userdatabase file will be created.

**databaseCreate()**  (Return type: void)

* Method used to create the database , for user with no prior records

**checkDatabaseLines()**  (Return type: int)

* Method used to check the number of lines in the NameList.txt.

**checkUserName()** (Return type: void)

* This method will check whether username exist. String split() method is used to split and get existing username from each line. The existing username will store into array (usernamecheck) to compare with the username input by user. A counter (passwordLocation) is used to check the username lies in which line so that only that line is checked when matching up the password input by the account owner, further description can be found below..

**checkPassword()** (Return type: void)

* Method used to check whether the password matches the username. It will only perform password checking when reading the line where the authenticated username lies. This ensures the account is not logged in in cases when there are different user accounts with the same password.

**checkLogInStatus()** (Return type: void)

* Method used to display the user’s log in status .

**LoginModule()** (Return type: boolean)

* This method combines all the methods arranged in the desired order. It returns whether the log in is authenticated so that other processes can proceed in the main method.

**logOut()** (Return type: boolean)

* This method erases the username and password used to sign into the account. It will go back to the main screen for new user to log in or sign up or to close the program ( For safety purposes ).

**Sign up module**

This is a module that handles the signing up of new accounts where each username that exist is unique and there is not repetition to ensure no data mishandling will occur as the username is most important key used to bind all the user data. It will prompt user to enter their desired username where a check will be performed upon username entry to make sure that username is not used in prior account sign ups. Password confirmation will be performed too to ensure the new user is entering it correctly. When both the conditions are true, the account will be created.

* It will prompt user to enter their desired username where it will be checked to make sure username is not used in prior account sign ups.
* Password confirmation will be performed to ensure new users are entering it correctly. When both conditions are true , account will be created .

**Expected outcome**

**-Create new Account**

Among the methods that exist in this class are:

**databaseCheck()**   (Return type: void)

* Method used to check whether the database file exists . If it does not exist, a database file will be created.

**checkDatabaseLines()** (Return type: int)

* Method used to check and return the number of lines are there in the NameList.txt

**getUserName()**  (Return type: void)

* This method used to accept the username form the new user.

**checkExistingUserName()**  (Return type: void)

* Method used to check the existing username in the database to ensure there is no overlapping name else prompt the user to enter a new name.

**checkPasswordInput()**  (Return type: void)

* Method used to accept the password form user and a reconfirmation of password needed to ensure the user type the right password . If both password and reconfirmation password are matched . User will be lead to log in class .

**checkSignUpStatus()** (Return type: boolean)

* Method used to return the new username and password if the account has been created successfully .

**addUserAccount()** (Return type: void)

* **Method** used to add the new username and password into NameList.txt.

**signup()** (Return type: void)

* This method combines all the methods in sign up class arranged in the desired order.

**checkSignUpStatus**() (return type: boolean)

* This method will return whether the new username and password are created   
   successfully.

**Asset module**

This is a module used to record all the input Asset and output Asset of the user. It determines whether the username matches with the respective file name. Only the corresponding file that matches with the user's name will be opened. The text file contains the item name of the user's assets and its value. This is a module used by users to add, delete or edit their assets. All their assets are saved in own file named after their usernames (username+"\_asset.txt").

**Expected outcome**

**- Data of the user asset’s records**

Among the methods that exist in this class are:

**AssetModule(String username)**

* constructor which initialize the value of username taken from LogInModule

**assetFileCreate()**  (Return type: void)

* This method used to create username\_asset.txt.

**assetFileCheck()**  (Return type: void)

* Method used to check whether the username\_asset.txt exists . If it does not exist, A new file based on the username will be created.

**checkAssetDatabaseLines()**  (Return type: int)

* Method used to return the total number of lines that exists in username\_asset.txt.

**assetAccessRecord()** (Return type: void)

* Method used to access to the user's records in username\_aset.txt , and the records will be displayed on the screen for the user.

**deleteAssetRecord()** (Return type: void)

* Method used to delete the informations that already saved in the file .

**assetCalculate()** (Return type: void)

* Method used to calculate the total value of each asset that is already saved in the file username\_asset.txt and display the total value on the screen right after the user access to this Asset Module .

**assetAdd()**  (Return type: double)

* This is a method used to accept the asset entered by the user in the sequence of item followed by its’ value . All the informations are stored in the array named <item> and <value> . All the records will be displayed on the screen to ensure the informations are correctly entered.

**assetTempCalculate()**  (Return type: void)

* Method used to calculate the total value of the user's assets by adding all the values of asset that stored is array <value>. The total value will be displayed each time the user entered a new asset.

**assetDelete()**  (Return type: void)

* This method used to accept the line that need to be deleted . When the user chooses the record to delete, the line will be removed from the array by rewriting the specified line[n] of record with “null” for String array or “0” for double array . Besides , shift the array after [n] to [n-1].

**editAsset()**  (Return Type: void)

* This method used to edit a specified line with new information . This method can activities can only be done after saving the record . First , transfer all the informations in data into array<name> , <value> <Date> . Next , delete the specific line that has been chosen by the user (choose the number that has been displayed on the screen) . Followed by getting the new information entered by the user an replaced the deleted line with the new information . Lastly , transferring all the information from array into file username\_asset.txt.

**assetMenu()**  (Return type: void)

* Method used to ease the user to choose the numberhe wants . User will choose the number of activities that he would like to continue and the user will be led to the specific method . [1] to add [2] to edit[Save records before editing] [3] to delete [0] to save records

**userFileTransfer()** (Return type: void)

* Method used to transfer all the information that stored in Item array , value array and the date accessed from date() into the text file named username\_asset.txt.

**Asset()**  (Return type: void)

* This method combines all the asset methods arranged in the desired order. This method will only go through once , that is after the access Asset method . After that , assettMenu<method> will take over the step.

**sortAsset()** (Return type: Void)

* This method will divide the aset input by user into different fields of arrays. Each record is divided into different fields but are assigned with the same index, e.g.: N-th record has index of [N-1] and to access all its different fields, just use assName[N-1], assValue[N-1], assDate[N-1],

**getTotalAsset()** (Return type: Void)

* This method will total up all the value in assValue array and return total asset.

**getAssetNameList()** (Return type: Void)

* This method will return ass(array).

**getAssetValueList(**) (Return type: Void)

* This method will return assValue (array).

**getAssetDateList()** (Return type: Void)

* This method will return assDate (array).

**Income module**

This is a module used to record all the income the user has. It enables the user to add, delete and edit his or her income.All their expenses are saved in a file named after their username which is username\_income.txt.

**Expected outcome**

**-Data of the user income’s records**

Among the methods that exist in this class are:

**IncomeModule(String username)**

* Method used to accept the condition from the user whether to add, edit or delete the income. It initializes the username taken from LoginModule.

**IncomeFileCreate()**  (Return type: void)

* Method used to check whether the database of the user exists else a text file named after the username will be created. Username\_income.txt

**IncomeFileCheck()** (Return type: void)

* Method used to check whether the file of the username\_income.txt exists . else a file will be created in <method>IncomeFileCreate</method>

**CheckIncomeDatabaseLines** (Return type: Int)

* Method used to return the number of lines that exists in username\_income.txt.

**accessIncomeRecords()** (Return type: void)

* Method used to access to the user's records in username\_income.txt , and the records will be displayed on the screen for the user.

**IncomeModule()** (Return type: void)

* Method used to ease the user to choose the number he wants . User will choose the number of activities that he would like to continue and the user will be led to the specific method . [1] to add [2] to edit [3] to delete [0] to exit

**IncomeCalculate()** (Return type: void)

* Method used to calculate the total value of income in the username\_income.txt. First , transferring all the data from textfile into array <item1> , <value1> and <Date> . Add the total value of array <value1> and display it to the user each times the user make some changes to the records .

**getDate()** (Return type: String)

* This method will set the date of the input. The format of date is day/month/year.

**addIncome()** (Return type: Void)

This method will allow user to add new income. Prompt user to name the income and set income amount. setDate() method will get the date of input made. All the information will be combine into a line in the format: incomeName , incomeValue,date to append into user’s file (username + “income.txt”).

**editIncome()** (Return type: Void)

* This enables the user to edit the name, type and the amount of the income that has previously entered by replacing the old information with the new information entered by the user

**deleteIncome()** (Return type: Void)

* This enables the user to delete the income that has been entered if the user entered his income wrongly.

**Income()** (Return type: Void)

* This method combines all the Income methods arranged in the desired order . After that , IncomeMenu<method> will take over the step.

**sortIncome()** (Return type: Void)

* This method will divide income input by user into different fields of arrays. Each record is divided into different fields but are assigned with the same index, e.g.: N-th record has index of [N-1] and to access all its different fields, just use incName[N-1], incDate[N-1], incValue[N-1].

**getTotalIncome()** (Return type: Void)

* This method will total up all the value in incValue array and return total asset.

**getIncomeNameList()** (Return type: Void)

* This method will return inc(array).

**getIncomeValueList(**) (Return type: Void)

* This method will return incValue (array).

**getIncomeDateList()** (Return type: Void)

* This method will return incDate (array).

**Expense Module**

This is a module used by user to manipulate their expenses. Users can choose to add, delete or edit their expense. All their expenses will be saved in a file called username\_expense.txt.

**Expected outcome**

**-Data of the user expense’s records**

**ExpenseModule(String username)**

* constructor which initialize the value of username taken from LogInModule

**expenseFileCreate()**  (Return type: void)

* This method used to create username\_expense.txt.

**expenseFileCheck()**  (Return type: void)

* Method used to check whether the username\_espense.txt exists . If it does not exist, A new file based on the username will be created.

**checkExpenseDatabaseLines(**)  (Return type: int)

* Method used to return the total number of lines that exists in username\_expense.txt.

**expenseAccessRecord()** (Return type: void)

* Method used to access to the user's records in username\_expense.txt , and the records will be displayed on the screen for the user.

**expenseCalculate()** (Return type: void)

* Method used to save the history expense value into a array and then calculate the sum of the value and display it.

**tempExpenseMenu()** (Return type: void)

* Method used to get the number from the user to proceed with the activities that the user wants . Press[1]to delete, Press[2] to back to Expense menu . Next the Expense Menu will be overtaken by the <method>ExpenseMenu</method>

**tempDeleteExpenseRecord()** (Return type: void)

* This method is to delete the specific line of information chosen by the user . First , transfer all the data from username\_expense.txt into array . Display the information in array with the digit at the left [n+1] , it will ease the user to choose which line fo information to be delete . After deleting the information , save the new data into the text file named after the user .

**setExpenseType()** (Return type: void)

* enables the user to choose the type of expenses. It is more systematic when it is displayed out.

**expenseCalculate()** (Return type: void)

**expenseAdd()** (Return type: void)

* enables the user to enter the name of Item and its value followed by storing the information into array Item and value.
* **expenseTempCalculate()** (Return type: void) is used to calculate the temporary total value of the user's expense and display it. The history of assets is not included.
* **expenseEdit()** (Return type: void)
* **expenseDelete()** (Return type: void) enables the user to delete the asset if the user entered wrongly or there is something need to be edited.
* **expenseMenu()** (Return type: void) enables the user to choose whether to continue with adding or editing the expense to be stored into the array Item and value.
* **userFileTransfer()** (Return type: void) is used to transfer the Item array and value array date into the username\_expense.txt which is from array to file.

**expense()** (Return type: void)

* is the collection of all the methods to manipulate the expense of the user.
* **Date()** (Return type: String) is used to set the date of the input.

**sortIncome()** (Return type: Void)

* This method will divide income input by user into different fields of arrays. Each record is divided into different fields but are assigned with the same index, e.g.: N-th record has index of [N-1] and to access all its different fields, just use incName[N-1], incDate[N-1], incValue[N-1].

**getTotalIncome()** (Return type: Void)

* This method will total up all the value in incValue array and return total asset.

**getIncomeNameList()** (Return type: Void)

* This method will return inc(array).

**getIncomeValueList(**) (Return type: Void)

* This method will return incValue (array).

**getIncomeDateList()** (Return type: Void)

* This method will return incDate (array).

**Reporting**

ReportMethod enables the user to divide his income to different expense area and calculate the balance that can be used. For example, 15% on medical, 30% on loans, 30% on family, 15% on traveling and 10% on saving.

**Expected outcome**

**-Records of the information based on daily , weekly or monthly**

Log in

Main menu

System terminate

False

No

Close system

Sign up

No

True

Yes

Enter username and password

Enter username and password

No

3.Expense

2.Income

1.Asset

Choose aspect

5.Log out

4.Financial Report

Yes

No

No

No

Yes

Yes

Yes

Yes

Yes

Aspect management