Expense Management System

Samuel J. Rengifo A00404150

<u>Analysis</u>

Description: The program will work as a savings system which will allow the **user** to keep his accounts up to date, the user will be able to register an expense, query the total expenses for a specific user and query the average spending of all users.

Input:

1. **String** username
   -The user will input a string which will work to compare to the stored values on the array and in that order display if the username typed matches with a recorded value.

2. **String[]** usernameArray
   -The user will type a string which will be stored in a determined position of the array.

3. **double[]** expenseArray
   -The user will type a double value which will be stored in a determined position of the array.

Output:

1. **double** totalExpensesUniqueUser
   -This variable will work to calculate the expenses of a user and then, finally show to the user the total expenses of a specific user.

2. **String** username
   -As output this variable is part of the answer when selecting option (2), it will indicate the user that matches with the compared records.

3. **double** averageAllUserExpenses
   -This variable will work to calculate the average of all user expenses, first, it will sum all the expense values and then it will divide it by the amount of value expenses higher than 0, to finally show a value at the end of the execution.

Example:

1.
The user is asked to select an option from the main menu [1, 2 ,3 ,4]
  - -*The user chooses the option* ***1***
The program asks for an username
  - -*The user says* ***josee4***
The program asks for the expense
  - -*The user says* ***100000***

2.
The user is asked to select an option from the main menu [1, 2 ,3 ,4]
  - -*The user chooses the option* ***2***
The program asks for an username
  - -*The user says* ***josee4***
The program says to the user that ***josee4*** expenses are: $*100000*

<u>Contract methods</u>

Method 1: Welcome | METHOD int

Description: This method will welcome the user and it will ask him for a menu selection.
@return **menuSelectOption** This option will work as a param to the switch to evaluate.

Method 2: Register-An-Expense | METHOD int
Description: This method will ask the user the username and the expense value to attach the data in the system, also this method will sum 1 to z as a counter for array index.
@param **z** This variable will work to store data in the array depending of the index that is z
@param **expenseArray** This is a double array which will store expense values
@param **usernameArray** This is a String array which will store username values
@return **z** This return variable will be the sum in 1 of z to move the index forward

Method 3: Query-Total-Expenses-For-A-User | METHOD void
Description: This method will go through an array comparing the username stored with the indicated by the user and will sum the values to a variable, which will be displayed in the end.
@param **username** This variable will store the username indicated by the user to compare to the array username values
@param **expenseArray** This is a double array which will store expense values
@param **usernameArray** This is a String array which will store username values
@param **i** This variable will work as a counter for the cycle to go through the array
@param **totalExpensesUniqueUser** This variable will store temporarily the total expenses of an user

Method 4: Query-Average-Spending-Of-All-Users | METHOD void
Description: This method will query the average of all user expenses by going through an array and doing a sum to a variable, dividing it by the amount of times that a value was registered.
@param **averageAllUserExpenses** This variable will store temporarily the average of all user expenses
@param **expenseArray** This is a double array which will store expense values
@param **i** This variable will work as a counter for the cycle to go through the array
@param **validArrayCounter** This variable will work as a counter for an array that have a value higher than 0