MoneyBadger Class Description Document

# Main (Class)

This class is the main running class for the application.

## Fields

|  |  |  |
| --- | --- | --- |
| Type | Field Name | Description |
| IOControl object | io | Input Output object for manipulating webpage |
| UserBudget object | userBudget | User's budget object, containing user information |
| String | localStorageFilePath | Location of user data in local storage |

## Methods

**document.onload():** Handles the instantiation of all fields listed.

**createAccount(String name, String password, String passwordConfirmation):**

On first activation of the application, The method calls

io.verifyPassword(String password, String passwordConfirmation)

to ensure the password is verified, sets

userBudget = new UserBudget(String name, String password, String filePath)

then calls

io.saveAcount(userBudget).

**resetBudgetFile():**

calls io.deleteBudgetFile(String filePath)

calls document.onload().

This resets the application to its initial first activation state.

**loadBugetFile(String name, String password):**

This method identifies the local storage file containing budget data, performs a validation based on the password parameter.

# IOControl

## Fields

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| String | userFilePath | The direct path in local storage to the user's data file |
| String | userFile | The string information contained in the user's data file |

## Methods

**IOControl(String)**

This is the constructor for the IOControl class. This initialises the field userFilePath to the user's data file path in local storage and initialises the field userFile as empty String.

**verifyPassword(String password, String passwordConfirm)**

**: returns Boolean result**

This takes the two string parameters password and passwordConfirm and compares them and returns the boolean result.  
  
**readFile()**

This method checks if a file path is stored in userFilePath, if the file path is correct, and then stores the information in said file as string in userFile.

**verifyAccount(String userName, String password)**

**: returns Boolean result**

This method, taking the parameters userName and password, determines whether the string stored in userFile matches the information giving in the parameters.  
  
**getUserBudget()**

**: return UserBudget**

This parses the string field userFile, provided it is not empty and well-formed, and stores it as a UserBudget object, which is returned.

**saveAccount(UserBudget userBudget)**

**: returns Boolean success**

This method parses the userBudget parameter as a string and stores it in userFile, then storing it in local Storage. The method then returns the success or failure of the store as a boolean.

**exportBudgetAsCSV(UserBudget userBudget)**

**:returns Boolean success**

This method, after executing the saveAccount() method, parses the information to a temporary string in the .csv format and saves it in local storage.  
  
**AddAssetEvent(Asset asset, UserBudget userBudget)**

**:returns UserBudget userBudget**

This method takes in the asset and userBudget parameters, adds the new Asset object to the UserBudget object and returns the now altered object.   
  
**historyForwardEvent(UserBudget userBudget, Integer pageIndex)**

**:return Asset[] assetList**

This method takes in the parameters userBudget and padeIndex,

**historyBackEvent(UserBudget userBudget)**

**:return Asset[] assetList**

**backButtonEvent(UserBudget userBudget)**

**confirmAssetEvent(Asset, UserBudget)**  
**confirmReminderEvent(Reminder, Integer, UserBudget)**  
**editAssetEvent(Integer, Asset, UserBudget)**  
**editReminderEvent(Integer, Asset, UserBudget)**  
**removeAssetEvent(Integer, UserBudget)**  
**removeReminderEvent(Integer, UserBudget)**

# Reminder

## Fields

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| String | name | Name for the reminder to identify it by |
| String | description | Reminder descriptor |
| Date | date | Due date for the reminder |

## Methods

Reminder( String, String, Date)  
getReminder(): Reminder  
setReminder(String, String, Date)

# Asset

## Fields

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| String | name |  |
| Double | quantity |  |
| Boolean | isIncome |  |
| Integer | Frequency |  |
| Reminder | reminder |  |
| Date | dateOfRecurrence |  |

## Methods

**Asset(String. Double. Boolean)**  
**Asset(String. Double. Boolean, Date)**  
  
**addReminder(String, String, Date)**  
**setReminder(String, String, Date)**  
**removeReminder()**  
**getReminder(): Reminder**

# UserBudget

## Fields

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| String | userName |  |
| String | password |  |
| String | saveFileLocation |  |
| Asset[] | income |  |
| Asset[] | expenditure |  |

## Methods

**UserBudget(String, String, String)**  
**addAsset(String, Double, Boolean)**  
**addAsset(String, Double, Boolean, Integer, Date)**

**setAsset(Integer, String, Double, Boolean)**  
**setAsset(Integer, String, Double, Boolean, Integer, Date)**  
**getAsset(Integer, String, Double, Boolean)**  
**getAsset(Integer, String, Double, Boolean, Integer, Date)**  
**removeIncome(index, Boolean)**  
  
**tallyIncome(): Double**  
**tallyExpenditure(): Double**  
**tallySavings(): Double**  
  
**getIncomeHistory(): Asset[]**  
**getIncomeHistory(Integer): Asset[]**  
**getExpenditureHistory():Asset[]**  
**getExpenditureHistory(Integer):Asset[]**