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 pageIndex, and returns the Asset list associated with the pageIndex, determined by the UserBudget objects income and expenditure Lists.

**historyBackEvent(UserBudget userBudget, Integer pageIndex)**

**: return Asset[] assetList**

This method takes in the parameters userBudget and pageIndex, and returns the Asset list associated with the pageIndex, determined by the UserBudget objects income and expenditure Lists.

**confirmAssetEvent(Asset asset, UserBudget userBudget)**

This method takes in the parameters asset and userBudget and pushes the Asset into said list.

**confirmReminderEvent(Reminder reminder, Integer index, UserBudget userBudget)**

This method takes in the parameters reminder, index and userBudget, then saves the reminder in an Asset (from either the income or expenditure lists) in userBudget.

**editAssetEvent(Integer index, Asset asset, UserBudget userBudget)**

This method takes in the parameters index, asset and userBudget, identifies the Asset to be edited via index, and then replaces said Asset in its list within userBudget.

**editReminderEvent(Integer index, Reminder reminder, UserBudget userBudget)**

This method takes in the parameters reminder, index and userBudget, identifies the Asset the reminder belongs to via index, then saves the reminder in said Asset in userBudget.

**removeAssetEvent(Integer, UserBudget)**

This method takes in the parameters index and userBudget and removes the Asset from said list.

**removeReminderEvent(Integer, UserBudget)**

This method takes in the parameters index and userBudget, identifies the Asset the reminder belongs to via index, then instantiates a blank reminder in said Asset in userBudget.

# Reminder

## Fields

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| String | name | Name for the reminder to identify it by |
| String | description | Reminder descriptor |
| Date | dueDate | Due date for the reminder |
| Integer | frequency | Integer to determine how regularly the reminder should occur. |

## Methods

**Reminder(String name, String description, Date dueDate, Integer frequency)**

Instantiates a new Reminder object based on its name, description, dueDate and frequency.

**spawnNextReminder()**

**: returns Reminder reminder**

This method returns a new reminder based on the integer frequency.

# Asset

## Fields

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| String | name | The name by which the asset is identified |
| Double | quantity | The amount the Asset is worth |
| Boolean | isIncome | A Boolean to determine if the Asset object is Expenditure or Income |
| Reminder | reminder | Reminder object for date and time based events |

## Methods

**Asset(String name, Double quantity, Boolean isIncome)**

Instantiates a new Asset object based on its name, quantity, and isIncome. The method also sets reminder to a blank state.

**setReminder(String name, String description, Date dueDate, Integer frequency)**

Calls the Reminder() method and sets reminder to the parameters listed above.

**removeReminder()**

Resets the reminder to a blank state.

# UserBudget

## Fields

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| String | userName | The user name determined by the user |
| String | password | The user password for accessing their save file |
| String | saveFileLocation | Location of the user’s save file in local storage |
| Asset[] | income | List of Asset objects where isIncome is set to true |
| Asset[] | expenditure | List of Asset objects where isIncome is set to false |

## Methods

**UserBudget(String userName, String passowrd, String filePath)**  
**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[]**