Interface description

# Class Customer

## Name

Customer

## Input:

### Constructor:

* FirstName: string
* LastName: string
* EMailAdress: string
* CustomerNumber: int
* Balance: double

### Properties:

* FirstName: string (read/write); argumentException in case of string.Empty()
* LastName: string (read/write); argumentException in case of string.Empty()
* EMailAdress: string (read/write); argumentException in case of string.Empty()
* Balancing: double (read/write)
* DateLastChange: datetime (read/write); argumentException in case of date is older than stored one (change to read only?)
* CustomerNumber: int (read only)

## Output:

### Static Methods:

* Int ValidateEMailAdress(List<Customer> customerList, string eMailAdress){checks for validity and twins in customerList};

Return Values for ValidateEMailAdress:

|  |  |
| --- | --- |
| Return value | Description |
| 0 | E-Mail-Adress is valid |
| -1 | Does not contain exactly one "@" |
| -2 | There is no "." after "@" |
| -3 | The final part (after last ".") is not 2-4 characters long |
| -4 | The final part (after last ".") does not contain only letters |
| -5 | There is no character before the "@" |
| -6 | There is a "." at the start, end or next to the "@" |
| -7 | Includes invalid characters |
| -8 | E-Mail-Adress is already used |

### Dynamic Methods

* ToString()

### Properties:

See Input/Properties;

## Process

The Constructor creates a new instance. The validation of the data has to be done before construction. A new E-Mail-Adress can be checked by the static method ValidateEMailAdress. Data of any instance can be read or written using the properties. The properties do not check validity but the properties of type string raise argumentException if the value for writing is string.Empty(). The Property DateLastChange raises an argument Exception if the value for writing is older than the actual value.

# Main Window

## Name

FrmMainWindow

## Input:

This is the root Window; Only GUI.

## Output:

Only GUI

## Process

The Window is the root Window. While start up the Login Form is called modal. Dialogresult Ok allows the user to use Main Window. Dialogresult exit ends the program. After successful Login the database is loaded. The path of the database is hardcoded and it is in the same directory as the program is. If there is no database a new one is created. A list of customers has to be provided. The list can be searched by any argument the user wants. The type of the customers is class customer The user has the option to create a new customer. In this case the Edit Window is called modal in mode "New". The user can also select a customer and choose different the options "Edit" or "Balancing" for this customer. In this cases the Edit Window is called modal in the chosen mode. At Closing of the program the list of customers has to be stored/updated in the database.

# Edit Window

## Name:

FrmEdit

## Input:

* Reference for Customerlist
* Mode (New, Edit, Balancing)
* Customer\_ID (only in case of Mode is Edit or Balancing)

## Output:

* Dialogresult (OK,Cancel)
* In Case of Dialogresult OK the Data is stored on the input Reference

## Process:

The window is called modal. Dialogresult Cancel is always enabled. In case of Dialogresult Cancel the data at the Reference of the Customer must not be different than before the call. Validation is done by the Edit Window. Depending on the calling mode:

### New:

An Empty Form is showed to the user to fill in the data of the new user. Dialogresult Ok is only possible if a new Object of type Customer can be instantiated. Balancing is initiated with 0. In case of Dialogresult Ok the new instance of type Customer is created at the reference for the customer.

### Edit:

The Customer data is shown to the user. The user could change the user data but not balancing. Dialogresult Ok is only possible if the data of the Customer is valid. In case of Dialogresult Ok the data at the reference for the customer is updated.

### Balancing:

The Customer data is only shown but could not be changed. The balancing is showed. The user could add or subtract some money. Dialogresult Ok is always enabled. Dialogresult Ok updates the balancing.

# Login Window

## Name:

FrmLogin

## Input:

* None

## Output:

* Dialog Result OK
* Dialog Result Exit

## Process:

The Login Form is a modal Window. The password can be entered in a masked Textbox.

The password is saved hardcoded in the program code. The click-event for the button “ok” is checking the password and either prompts a message box to inform the user about a wrong password or terminates the window with the dialog result ok if the password is correct. The button “Exit” terminates the window with the dialog result exit.

Test cases

# Unit test

Testing for each part of the application according to the requirements given.

Responsible person for the GUI are responsible for catching incorrect inputs. For example when new customer is added, the GUI programmer have to check if the mail address is correct with the method provided in the class customer.

**Class Customer**

**Main Window**

**Edit Window**

**Login Window**

# System test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test number | Precondition | Input | Expected output | What is tested |
| 1. | * App. not running | Start app. and login correctly | Main window is operating | * Correct startup * Connection Login-,Main window |
| 2. | * App. running * Main window operating * Database does not exist | Add a new customer | Database is created | * Correct initialization * Creating new database * Customer is listed in Main window and database * Correct startup of the Edit window * Connection Main-, Edit window |
| 3. | * App. running * Edit window operating * Database does exist and contains at least one customer * Customer has a positive balance | Lend money, save and close window, reopen the edit window with the same customer pay money and cancel | Balance of the customer changes once in the database and the main window but not twice | * Connection edit window, database, main window * Correct save and cancel * Correct editing the database * Correct calculation |