Kleine Kundenverwaltung

SWE-Projekt

Johann Hofer

Johannes Hirsch

Inhaltsverzeichnis

[1 Einleitung 1](#_Toc452481092)

[1.1 Kurzbeschreibung 1](#_Toc452481093)

[1.2 Code-Konventionen 1](#_Toc452481094)

[2 Design 1](#_Toc452481095)

[2.1 Klassen 1](#_Toc452481096)

[2.2 Interface-Beschreibungen 1](#_Toc452481097)

[bool IsEmailValid(string email) 1](#_Toc452481098)

[void AddAmount(double newAmount) 2](#_Toc452481099)

[bool IsCustomerUniqe(Customer customer) 2](#_Toc452481100)

[void WriteCSV() 2](#_Toc452481101)

[void WriteLastCustomerCSV() 2](#_Toc452481102)

[List<Customer> ReadCSV() 2](#_Toc452481103)

[2.3 Use-Cases 2](#_Toc452481104)

[2.3.1 Einloggen 3](#_Toc452481105)

[2.3.2 Customer hinzufügen 3](#_Toc452481106)

[2.3.3 Customer ändern 3](#_Toc452481107)

[3 Error Handling 3](#_Toc452481108)

[3.1 Error Codes 3](#_Toc452481109)

[4 Tests 3](#_Toc452481110)

[5 Anhang 5](#_Toc452481111)

[5.1 Abbildungsverzeichnis 5](#_Toc452481112)

# Einleitung

## Kurzbeschreibung

Für die Lehrveranstaltung Software-Engineering sollte eine kleine Kundenverwaltung. In einer Windows-Forms-Anwendung sollten bestehenden Kunden angezeigt und beschränkt verändert werden können. Neue Kunden sollen angelegt werden können. Weiters sollen die Daten verschlüsselt und die Anwendung Passwortgeschützt sein.

## Code-Konventionen

Für das gesamte Projekt gelten alle Code-Konventionen, welche für die Lehrveranstaltung vereinbart wurden:

### Naming

* Variable names start with small letter
* Class names start with Capital Letter
* Function/Method names start with a Capital Letter
* CamelCase generally used (i.e. for variable, function/method and class names) - no umlauts (ä...)
* constants in all CAPITALS - with underscores if multiple words, e.g. NUM\_SERVERS
* Meaningful names, also for controls (not Button1!)

### Formatting

* control structures always with { } - even with only one command
* { } on a separate line (except with } for do-while)

### Comments

* comments for every method / function, but not standard events (Button click, etc.)
* use /// so can see when coding
* also include all exceptions (///<exception cref="FormatException">Wann wird die FormatException aufgerufen.</exception>)

### Conventions

* if an error code is used - it should be an int, and generally errors are negative values, e.g. -1 and success is greater than or equal to 0.

# Design

## Klassen

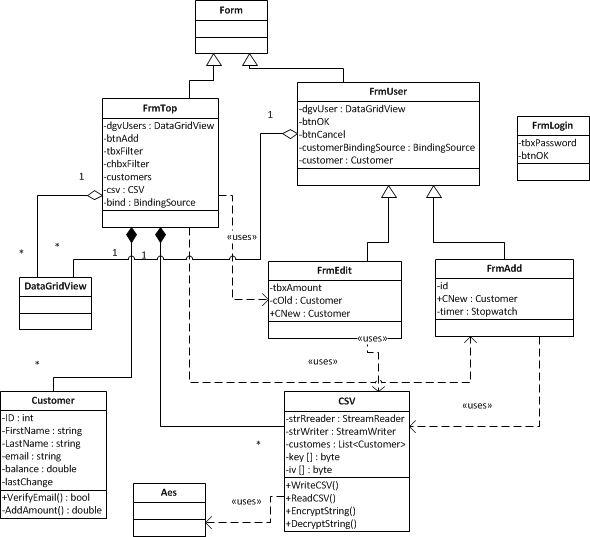


Abb. 1: Klassen

## Interface-Beschreibungen

### bool IsEmailValid(string email)

* Input: string email
* Output: bool valid
* Prozess: Überprüft alle spezifizierten Fälle einer nicht korrekten Email-Adresse. Wenn keiner zutrifft, wird true zurückgegeben.

### void AddAmount(double newAmount)

* Input: double newAmount
* Output: -
* Prozess: Addiert newAmount zum aktuellen Stand des Customers.

### bool IsCustomerUniqe(Customer customer)

* Input:
* Output:
* Prozess:

### void WriteCSV()

* Input: -
* Output: -
* Prozess: Verschlüsselt die aktuelle vollständige Liste und schreibt diese in die CSV-Datei.

### void WriteLastCustomerCSV()

* Input: -
* Output: -
* Prozess: Verschlüsselt den neuen Eintrag in der Liste und schreibt diesen in die CSV-Datei.

### List<Customer> ReadCSV()

* Input: -
* Output: List<Customer> customers
* Prozess: Liest die gesamte CSV-Datei ein, entschlüsselt diese und schreibt die Customers in die Liste.

**string EncryptString(string original)**

* Input: string original
* Output: string encString
* Prozess: Verschlüsselt einen String.

**string DecryptString(string decrypted)**

* Input: string encrypted
* Output: string plaintext
* Prozess: Entschlüsselt einen String.

## Use-Cases

Aufgrund der geringen Komplexität des Programs werden die wenigen Use-Cases textuell beschrieben.

### Einloggen

Zu begin des Programs erscheint ein Login-Fenster. Wenn man das korrekte PW (während der Testphase “test”) eingibt erscheint das Hauptfenster und die Customer, welche sich bereits in der CSV-Datei befinden werden eingelesen und angezeigt.

### Customer hinzufügen

Klickt der Benutzer auf den Butten “Add” erscheint eine Form, in der alle nötigen Daten für einen neuen Customer eingegeben werden können. Sind alle Daten korrekt, wird ein neuer Customer angelegt und in der CSV-Datei angehängt. Wenn nicht erscheint eine MessageBox mit dem ErrorCode.

### Customer ändern

Von einem bestehenden Customer können nachträglich der Nachname (“LastName”), die Email-Adresse und der Wert “Balance” (Schuld oder Guthaben) geändert werden. Die neuen Daten werden wie beim hinzufügen des Customers geprüft.

# Datei-Struktur

Die Daten werden in einer CSV-Datei „data.csv“ gespeichert. Jede Zeile entspricht einem Kunden und alle Einträge sind mit Hilfe der Windows-Kryptographie-Klasse verschlüsselt.

# Implementierungs-Details

Die CSV-Datei wird beim Programmstart eingelesen und Entschlüsselt und in einer Liste gespeichert. Wird ein neuer Kunde hinzugefügt wird lediglich dieser am Ende der CSV angehängt. Werden Daten eines bestehenden Kunden geändert, muss die ganze Liste in die Datei geschrieben werden.

# Error Handling

In den Funktionen, in denen die häufigen Fehler auftreten geben mit „out“ einen Error-Parameter aus. Für alle anderen weniger häufigen Fehler gibt es ein generelles try-catch.

Für den Error wurde eine Struktur entworfen, welche aktuell nur einen Code als Integer enthält, aber beliebig erweitert werden kann.

## Error Codes

0: Kein Error

1: FirstName nicht OK

2: LastName nicht OK

3: Email nicht OK

4: Leere Textbox

5: Email nicht einzigartig

# Optimierung

Für den Fall, dass das Programm auf einem sehr langsamen PC trotzdem noch relativ stabil läuft, wurde es dahingehend optimiert. Hierfür wurde das Exception-Handling wenn möglich durch Error-Codes ersetzt und die Schreib- und Lese-Zugriffe reduziert und optimiert.

Gemessen wurde die Zeit, beim Anlegen des Nutzers „Johannes, Hirsch, [johannes.hirsch@aon.at](mailto:johannes.hirsch@aon.at)“.

Vor dem optimieren dauerte dies im Durchschnitt 0,0133 Sekunden, danach nur mehr 0,00681 Sekunden, was einer relativen Verbesserung von 48,8% entspricht.

# Tests

Folgende Test-Cases wurden vorab festgelegt (anhand der Requirements) und getestet. Aufgrund der geringen Komplexität wurden Fehler sofort behoben.

1. Open it with Win7 x32/x64
2. –
3. Kann bei nicht korrekten Date nein Benutzer angelegt werden? (E-mail systematisch, Namen informell vom Programmierer getestet)
4. Customer erstellen, App schließen und CSV auf Eintrag überprüfen, (Ev. Computer neu starten), App öffnen. Customer in Liste?
   1. Daten in der CSV lesbar?
   2. Login nötig?
5. Email Test-Cases:
6. test@test.at
7. test.at
8. [te@st@test.at](mailto:te@st@test.at)
9. test@at
10. [test@test.at.at](mailto:test@test.at.at)
11. test@test.a
12. test@test.atat
13. test@test.atata
14. [test@test.a&t](mailto:test@test.a&t)
15. test@test.at5
16. test@test.
17. [a@test.at](mailto:a@test.at)
18. @test.at
19. [.test@at](mailto:.test@at)
20. [Test.@test.at](mailto:Test.@test.at)
21. test@.test.at
22. [test5@test.at](mailto:test5@test.at)
23. [test@te6st.at](mailto:test@te6st.at)
24. Symbols… (see requirements)
25. ID
    1. User hat keinen Einfluss auf die ID (Anzahl der Customer + 1)
    2. Beim hinzufügen und beim Ändern eine bereits vorhandene Email eingeben. Fehler?
26. –
27. Add amounts:
28. Positiv
29. 0
30. Negativ
31. Buchstaben oder Symbole
32. Neue Balance korrekt berechnet?
33. Customers nach allen Spalten sortieren. Liste richtig sortiert?
34. Informell
35. Nachnamen und Email ändern (6,8 erneut testen)
36. -

# Anhang

## Abbildungsverzeichnis

[Abb. 1: Klassen 1](#_Toc451797683)