****

**FH - Studiengang für**

**Informationstechnik und System-Management**

**Salzburg**

**ITS**

**Übungen in**

**Spezielle Softwaretechnologien**

**Protokoll**

Gegenstand der Übung gemäß Anleitung:

**Softwarekomponenten – Dynamic Link Libraries**

**Version: 1**

**Datum der Übung: 06.10.2016**

**Datum der Abgabe: 20.10.2016**

**Autoren: Christopher Wieland, Martin Wieser, Stephanie Kaschnitz**

**Unterschrift des Autors / der Autorin:**

Historie

|  |  |  |  |
| --- | --- | --- | --- |
| Änderung | Datum: | Autor: | Version: |
| *Kunden&Konten Komponente:* Kundenerstellung/Änderung/Löschung, Konto anlegen/löschen wurde hinzugefügt, Funktionen wurden separiert. | 09.10.16 | CW | 0.1 |
| *Kunden&Konten Komponente:* Kontonummer, Kontostand wurden hinzugefügt, Namenskonventionen wurden geändert, Logger wurde implementiert, Textausgabe wurde durch Logger ersetzt, Funktionen wurden auf Parameterübergabe umgeschrieben, Sicherungen und Typspezifizierer wurden hinzugefügt | 10.10.16 | CW | 0.2 |
| *Kunden&Konten Komponente:* Sicherungen wurden ergänzt, Funktion Kundendatenabfrage wurde hinzugefügt | 11.10.16 | CW | 0.3 |
| *Kontofunktionen Komponente:* Funktionen Überweisung, Abheben, Einzahlen wurden hinzugefügt. | 11.10.16 | SK | 0.4 |
| *Datenspeicherungs Komponente:* externe Library für Datenspeicherung via JSON wurde hinzugefügt. | 13.10.16 | MW | 0.5 |
| *Hauptprogramm:* Komponenten wurden zusammengefügt zu einem Bank.cpp | 13.10.16 | CW | 0.5 |
| *Kontofunktionen Komponente + Hauptprogramm:* Funktion Kontoauszug wurde hinzugefügt und Pfade im Programm wurden relativiert | 14.10.16 | SK | 0.6 |
| *Datenspeicherungs Komponente:* Funktionen für Customer wurden hinzugefügt | 15.10.16 | MW | 0.7 |
| *Kunden&Konten Komponente + Hauptprogramm:* Programm wurde umstrukturiert, Funktionsbeschreibungen wurden hinzugefügt und Kundendatenänderung, Kundenerstellung und Kunden löschen Funktion wurde ergänzt mit persistenter Datenspeicherung. | 15.10.16 | CW | 0.8 |
| *Datenspeicherungs Komponente:* Funktionen für Kundenabspeicherung wurde korrigert, Funktion zur Abspeicherung von Sparkonten wurde hinzugefügt. | 16.10.16 | MW | 0.8.1 |
| *Kunden&Konten Komponente + Hauptprogramm:* Implementierung von überarbeiteten Code, Funktionsbeschreibungen wurden hinzugefügt. | 16.10.16 | CW | 0.9 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Inhaltsverzeichnis

[1 Aufgabenstellung 1](#_Toc464334936)

[2 Komponentenübersicht 2](#_Toc464334937)

[2.1 Kunden & Konten 2](#_Toc464334938)

[2.1.1 Test2.1.1 2](#_Toc464334939)

[2.2 Kontofunktionalitäten 2](#_Toc464334940)

[2.2.1 Test2.2.1 2](#_Toc464334941)

[2.3 Persistente Datenspeicherung 2](#_Toc464334942)

[2.3.1 Test2.3.1 2](#_Toc464334943)

[3 Dokumentation der Funktionalität der DLL 3](#_Toc464334944)

[3.1 Kunden & Konten 3](#_Toc464334945)

[3.1.1 Test3.1.1 3](#_Toc464334946)

[3.2 Kontofunktionalitäten 3](#_Toc464334947)

[3.2.1 Test3.2.1 3](#_Toc464334948)

[3.3 Persistente Datenspeicherung 3](#_Toc464334949)

[3.3.1 Test3.3.1 3](#_Toc464334950)

[3.3.2 Test3.3.2 3](#_Toc464334951)

[3.4 Kopplung Der einzelnen Bereiche 4](#_Toc464334952)

[4 Zusätzliche externe Komponenten 5](#_Toc464334953)

[4.1 Test4.1 5](#_Toc464334954)

[4.2 Test4.2 5](#_Toc464334955)

[4.3 Test4.3 5](#_Toc464334956)

[5 Zusammenfassung und Ausblick 6](#_Toc464334957)

[5.1 Test5.1 6](#_Toc464334958)

[5.2 Test5.1 6](#_Toc464334959)

# Aufgabenstellung

Die Aufgabenstellung bestand darin, grundlegende Funktionen einer Bank mithilfe von nachladbaren Dynamic Link Libraries (DLL) in C/C++ zu entwickeln.

Die Anzahl der Komponenten sowie die Umsetzung ist selbst zu gestalten.

Um die Interoperabilität mit verschiedenen Programmiersprachen sicherzustellen, sind die Komponenten in C zu entwerfen.

D.h. alle Funktionen der DLLs müssen als C-Funktionen aus der DLL exportiert werden, um sie aus beliebigen Programmiersprachen aufrufen zu können.

Die folgenden Basisfunktionalitäten sollten beinhaltet sein:

* Kunden anlegen, löschen, umbenennen, ändern (Adressdaten, ...)
* Konten für Spar- und Kreditgeschäfte anlegen, schließen, verwalten
  + Ein Kunde kann mehrere Konten haben
  + Ein Konto kann mehrere Kontoverfüger haben
* Überweisungen von einem Konto zum anderen, Abhebung/Einzahlung, Kontoauszüge, Kontoabschlüsse
* Währungsmodul (Umrechnung, Kursverwaltung, etc.)
* Persistente Datenhaltung
* Sonstige Basis-/Hilfsfunktionen nach eigenem Ermessen

# Komponentenübersicht

Für die Umsetzung der Aufgabenstellung wurden die Aufgaben in 3 große Bereiche gespalten: Kunden & Konten, Kontofunktionalitäten und Persistenz. Jeder der Übungsteilnehmer hat sich einem Thema gewidmet.

## Kunden & Konten

Testtext

### Test2.1.1

Testtext

## Kontofunktionalitäten

Testtext

### Test2.2.1

Testtext

## Persistente Datenspeicherung

Bei der Komponente für das Persistieren der Daten geht es darum, die Kontoinformationen und die Kundendaten zu speichern. Für dieses Beispiel haben wir uns für die Datenhaltung im JSON Format entschieden. Die Kontodaten sowie die Kundendaten werden in 3 Files verarbeitet. Mit dieser Komponente ist es möglich auf die einzelnen JSON Objects zuzugreifen, sie zu verwalten und wieder abzuspeichern.

### Ausblick

Security, Erweitern der Kundendaten;

# Dokumentation der Funktionalität der DLL

Test3.0

## Kunden & Konten

Testtext

### Test3.1.1

Testtext

## Kontofunktionalitäten

Testtext

### Test3.2.1

Testtext

## Persistente Datenspeicherung (in work)

Die Komponente für die Datenspeicherung übernimmt das Speichern von Kunden und Konten und es gibt Zugriff auf einzelne Einträge davon. Diese Einträge können mit read\* gelesen und mit write\* geschrieben werden.

Da ein Kunde (besser gesagt die ID) des Kunden nur einmal vorhanden sein darf ist es nicht möglich einen weiteren User mit der gleichen ID zu persistieren. (kann vermutlich mit writeUser umgangen werden, wenn es die id schon gibt, dann wird aber alter User überschrieben!!)

Dies gilt auch für die Konten

Es gibt für jede Klasse 4 Funktionen für die Persistenz:

* Read\* (return: Element)
* Write\* (write (override) existing Element)
* Add\* (hinzufügen eines elementes das noch nicht vorhanden ist)
* Remove\* (remove mit id oder kontonr Wert)
* \*Exist ( existiert der User/das Konto schon?)

Zudem gibt es noch Funktionen die für die Auswahl einer ID oder KontoNr notwendig sind:

* readCount (liest letzte verwendete kontonr)
* writeCount (schreibt nächste kontonr +1)
* readUserCount (liest nächste verwendete userID )
* writeUserCount (schreibt nächste kontonr +1)

### Test3.3.1

Testtext

### Test3.3.2

Testtext

## Kopplung Der einzelnen Bereiche

Testtext

# Zusätzliche externe Komponenten

cJSON

<https://github.com/DaveGamble/cJSON>

http://www.json.org/

## Test4.1

Testtext

## Test4.2

Testtext

## Test4.3

Testtext

# Zusammenfassung und Ausblick

## Test5.1

Testtext

## Test5.1

Testtext