****

**FH - Studiengang für**

**Informationstechnik und System-Management**

**Salzburg**

**ITS**

**Übungen in**

**Spezielle Softwaretechnologien**

**Protokoll**

Gegenstand der Übung gemäß Anleitung:

**Softwarekomponenten – Wrapper/Adaptoren/Bridges als C# Assemblies File**

**Version: 1**

**Datum der Übung: 20.10.2016**

**Datum der Abgabe: 01.11.2016**

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

**Unterschrift des Autors / der Autorin:**

Historie

|  |  |  |  |
| --- | --- | --- | --- |
| Änderung | Datum: | Autor: | Version: |
| Implementierung FremdWrapper | 28.10.16 | CW | 0.1 |
| Implementierung EigenWrapper | 28.10.16 | SK | 0.2 |
| Absprache mit anderer Gruppe für FremdDll | 27.10.16 | CW | x |
| Implementierung Programm | 30.10.16 | SK | x |
| Fehlerbehandlung & Bugfixes Eigendll | 11.10.16 | MW, SK | 0.3 |
| UserFunktionen hinzufügen/fixen bei unserem dll | 24.10.16 | MW | x |
| Tests | 31.10.16 | MW | x |
| Definition des Interfaces (IBankManagement) | 25.10.16 | CW,SK | x |
| Fehlerbehandlung & Fixes FremdDll | 1.11.16 | CW,MW | x |
| Tests und Abgabe | 2.11.16 | MW | x |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Inhaltsverzeichnis

[1 Aufgabenstellung 1](#_Toc465761197)

[2 Komponentenübersicht 2](#_Toc465761198)

[2.1 Interface 2](#_Toc465761199)

[2.2 Unsere Komponente (bank.dll) 2](#_Toc465761200)

[2.2.1 Minimum Requirements 2](#_Toc465761201)

[2.3 Fremde Komponente (XMLController.dll) 2](#_Toc465761202)

[2.3.1 Minimum Requirements 2](#_Toc465761203)

[3 Funktionalität des Programms 2](#_Toc465761204)

[3.1 Account Management 2](#_Toc465761205)

[3.2 Customer-Management 2](#_Toc465761206)

[4 Zusammenfassung und Ausblick 3](#_Toc465761207)

# Aufgabenstellung

Entwerfen Sie nun auf Basis des/der von einem anderen Team übernommen DLLs und

der eigenen DLLs ein gemeinsames C#/.NET Interface und darauf basierend .NET Assemblies, die – diesmal objektorientiert unter C# und .NET – dem eigenen DLL und

dem übernommen DLL als Wrapper/Adaptor/Bridge dienen. Sie sollten die volle bzw.

geforderte Funktionalität der DLLs anbieten, ohne jedoch Details der DLLs und deren

API kennen zu müssen.

Letztlich sollten Sie mehrere .NET Assemblies implementieren, welche eine möglichst

einfache Verwendung der DLLs in C#/.NET Anwendungen und – da wir ja Komponenten

haben – den Austausch der verwendeten DLLs erlaubt. Das Ziel ist, dass Ihre zukünftige

Applikation durch den Austausch von Assemblies und der DLLs sowohl mit den eigenen

DLLs, als auch mit den DLLs des anderen Teams arbeiten kann (nicht gleichzeitig), da die

Schnittstellen beider Assembly-Gruppen in beiden Fällen ja gleich sind.

Einschränkung: Ein dynamisches Laden/Entladen der Assemblies bzw. DLLs während der

Runtime wird nicht benötigt.

Die Interfaces, Klassen, Typen,.. und das Fehlerhandling sollte so ausgelegt sein, dass die

DLLs für die Anwender völlig transparent und ohne Detailwissen über die DLLs

angebunden werden können. Aus der Nutzung der Assemblies darf in keiner Art und

Weise ersichtlich werden, dass es sich um native Dynamic Link Libraries handelt, die

lediglich ein primitives C-API anbieten.

# Komponentenübersicht

Komponentendiagramm einfügen

Ein C# Assembly ist eine dll für ein c# Programm

## Interface

Beide komponenten implementieren das interface IBankManagement somit können diese ausgetauscht werden.

## Unsere Komponente (bank.dll)

Minimum Requirements

* Bank.dll
* Eigene\_Bank\_DLL\_Assembly.dll
* Bankanwendung.exe

## Fremde Komponente (XMLController.dll)

Hier sind die imports.vom der XMLController.dll Komponenten enthalten und werden adaptiert.

D.h. Andere-Bank\_Dll\_Assembly.dll ist ein Adapter vom XMLController.dll der das interface IBankManagement impementiert.

Minimum Requirements

* Andere\_Bank\_DLL\_Assembly.dll
* Logging.dll
* XMLController.dll

# Funktionalität des Programms

Chart -> verzweigungen

## Account Management

## Customer-Management

# Zusammenfassung und Ausblick

Das Programm kann durch das Interface IBankManagement mit beiden Assemblys arbeiten. Die zwei Assemblys sind Adaptoren die die primitiven C-API wegkapseln.

Da das dynamische laden der Assemblys nicht notwendig war wird im Programmcode angegeben welche Komponente verwendet werden soll.