

Projektname: Bankkomponeten

Projektnummer: 0001

Kurzbeschreibung: Erzeugung von Softwarekomponente(n) als dynamisch nachladbare Bibliotheken (DLLs) in C/C++, in der die grundlegenden Funktionen einer Bank abgebildet werden

Version: 1.1.0

Vorgelegt von: Reimar Klammer, Johannes Probst, Daniel Komohorov

Author: Daniel Komohorov

Auftraggeber: DI (FH) DI Roland Graf, MSc

Erstelldatum: 20.10.2016

DOKUMENTENHISTORIE

Autor	Datum	Version	Änderung
Komohorov	20.10.2016	1.0	Erstellung der Dokumentation
•	•	•	•
•	•	•	•
•	•	•	•
•	•	•	•
•	•	•	•

I Inhaltsverzeichnis

I	Inhaltsverzeichnis	II
1	Dokumentation der Bibliothekinterfaces	1
1.1	Komponentenübersicht	1
1.2	Allgemeine Hinweise	1
1.3	Funktionsbeschreibung, Komponente Kunde (CustomerModul)	2
1.4	Funktionsbeschreibung, Komponente Konto (AccountModul)	4
1.5	Funktionsbeschreibung, Komponente Transaktion (TransactionModul)	8
1.6	Funktionsbeschreibung, Komponente Währungsumrechnung (CurrencyTranslationModul)	13
1.7	Funktionsbeschreibung, Komponente Persistenz (PersistenceModul)	15
1.8	Errorcode Liste	16
1.9	Fehlerbehandlung, allgemein mögliche Übergabeparameter	17
2	Anhang	17
3	TODO	18

1 Dokumentation der Bibliothekinterfaces

1.1 Komponentenübersicht

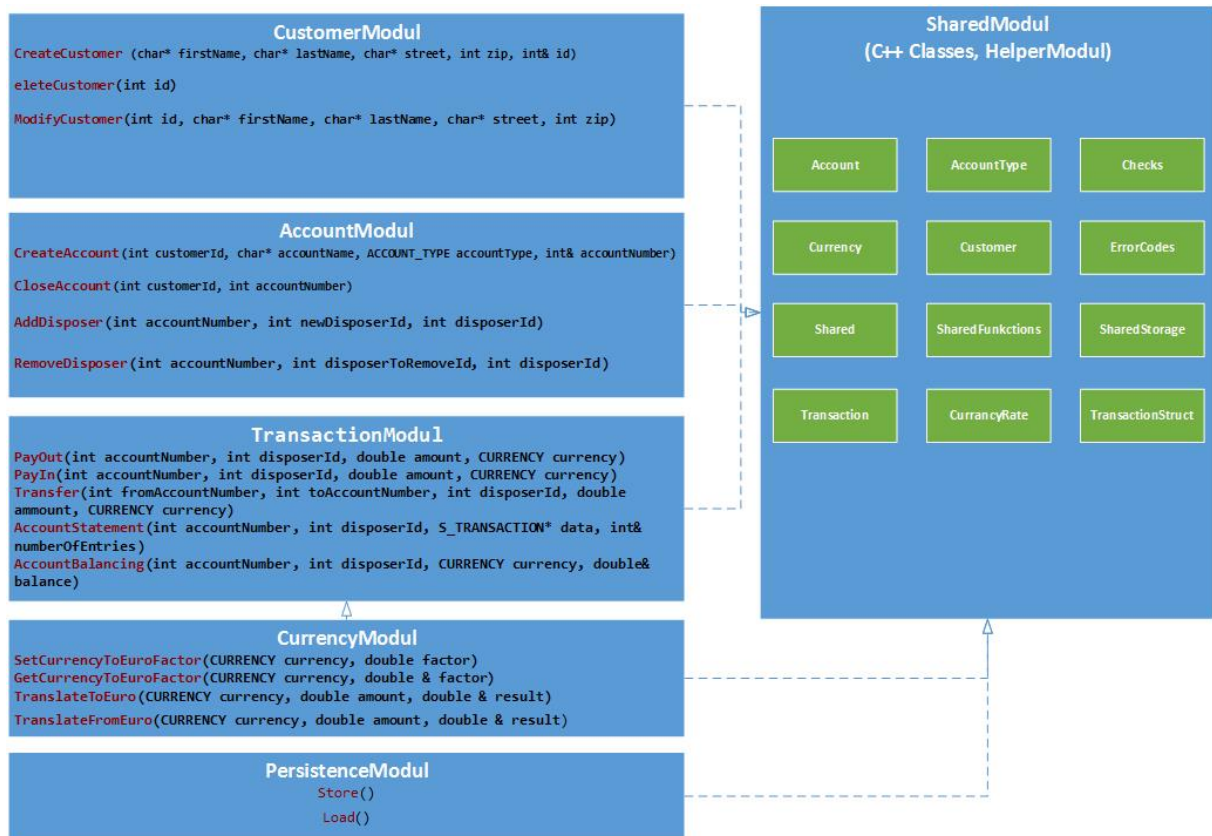


Abbildung 1: Komponentenübersicht

1.2 Allgemeine Hinweise

- Errocodes: Eine Liste mit Errocodes befindet sich im Kapitel 1.8
- Fehlerbehandlung (Checks) für wiederkehrende Fälle werden im Kapitel 1.9 erläutert
- Fehlerbehandlung für einzelne Funktionen werden in der Funktionsbeschreibungen erläutert

1.3 Funktionsbeschreibung, Komponente Kunde (CustomerModul)

Funktion:	int CreateCustomer(char* firstName, char* lastName, char* street, int zip, int& id)
Bezeichnung:	Kunden anlegen
Parameter:	char * firstName
Parameter:	char * lastName
Parameter:	char * street
Parameter:	int* zip
Parameter:	int & id
Rückgabewerte:	E_INVALID_PARAMETER und E_OK Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	1) Prüfung der Übergabeparametern. Alles Strings dürfen nicht NULL sein und zip muss ein Int-Wert von 1000 bis 9999 sein. Im Fehlerfall: Rückgabewert E_INVALID_PARAMETER 2) Nach positiver Prüfung liefert die Funktion E_OK zurück und erzeugt den Kunden
Aufrufbeispiele:	CreateCustomer("Franz", "Müller", "Hintertupfing", 5020, id)
Funktion:	int DeleteCustomer(int id)
Bezeichnung:	Kunden Löschen
Parameter:	int id
Rückgabewerte:	E_INVALID_PARAMETER, E_CUSTOMER_NOT_FOUND und E_OK Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	1) Prüfung der Übergabeparametern, ob id größer 0 ist. Im Fehlerfall: Rückgabewert E_INVALID_PARAMETER 2) Kunden id wird gesucht 3) Falls kein Kunde gefunden wird, oder Kunde ist Inaktiv, liefert die Funktion E_CUSTOMER_NOT_FOUND 4) Wenn id gefunden wurde, so wird der Kunde auf Inaktiv gesetzt und die Funktion liefert E_OK zurück
Aufrufbeispiele:	DeleteCustomer(id);

Funktion:	<code>int ModifyCustomer(int id, char* firstName, char* lastName, char* street, int* zip)</code>
Bezeichnung:	Kunden Bearbeiten
Parameter:	<code>int id</code> (Kunden- bzw. Verfüger-id)
Parameter:	<code>char * firstName</code>
Parameter:	<code>char * lastName</code>
Parameter:	<code>char * street</code>
Parameter:	<code>int zip</code>
Rückgabewerte:	<code>E_INVALID_PARAMETER</code> , <code>E_CUSTOMER_NOT_FOUND</code> und <code>E_OK</code> (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	<p>1) Prüfung der Übergabeparametern. Hier wird nur <code>id</code> geprüft (muss größer oder gleich 0 sein) und im Fehlerfall wird <code>E_INVALID_PARAMETER</code> zurückgegeben. Restliche Werte dürfen <code>NULL</code> sein, so dass z.B. nur ein Parameter geändert werden kann.</p> <p>2) Kunden <code>id</code> wird gesucht</p> <p>3) Falls kein Kunde gefunden wird, liefert die Funktion <code>E_CUSTOMER_NOT_FOUND</code> zurück</p> <p>4) Wenn alles gut geht, werden geänderte Werte übernommen und die Funktion liefert <code>E_OK</code> zurück</p>
Aufrufbeispiele:	<code>ModifyCustomer(0, "Hans", "Meier", "Daheim", &newZip);</code>

1.4 Funktionsbeschreibung, Komponente Konto (AccountModul)

Funktion:	<code>int CreateAccount(int disposerId, char* accountName, ACCOUNT_TYPE accountType, int accountNumber)</code>
Bezeichnung:	Konto eröffnen
Parameter:	<code>int disposerId</code>
Parameter:	<code>char* accountName</code>
Parameter:	<code>ACCOUNT_TYPE accountType</code> (ein ENUM: SavingsAccount (Sparkonto) = 1; LoanAccount (Girokonto) = 2)
Parameter:	<code>int accountNumber</code>
Rückgabewerte:	<code>E_INVALID_PARAMETER</code> , <code>E_CUSTOMER_NOT_FOUND</code> (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	1) Prüfung der Übergabeparametern. Kunden- bzw. Verfügernummer muss größer oder gleich 0 und Kontoname darf nicht NULL sein. im Fehlerfall: Rückgabewert <code>E_INVALID_PARAMETER</code> 2) Prüfung ob die Kunden ID existiert, wenn nicht, liefert die Funktion <code>E_CUSTOMER_NOT_FOUND</code> 3) Wenn alles gut geht, wird ein Konto erzeugt und dem übergebenen Kunden-ID zugewiesen. Die Funktion liefert <code>E_OK</code>
Aufrufbeispiele:	<code>ModifyCustomer(0,"Hans", "Meier", "Daheim", &newZip);</code>

Funktion:	int CloseAccount(int disposerId, int accountNumber)
Bezeichnung:	Konto schließen
Parameter:	int disposerId
Parameter:	int accountNumber
Rückgabewerte:	E_INVALID_PARAMETER, E_REMOVE_DISPOSER_NOT_FOUND, E_ACCOUNT_NOT_FOUND, E_CUSTOMER_NOT_FOUND, E_UNAUTHORIZED, E_OK (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	<p>1) Prüfung der Übergabeparametern. Kunden- bzw. Verfügernummer und Kontonummer muss größer oder gleich 0 sein. Im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>2) Suche nach Konto und dazugehörige Verfüger (einen oder mehreren)</p> <p>2.1) Prüfung der Übergabeparametern, im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>2.2) Prüfung ob das Konto existiert oder inaktiv, wenn eins von beiden zutreffen, liefert die Funktion E_CUSTOMER_NOT_FOUND zurück</p> <p>2.3) Prüfung ob Verfüger (ein oder mehrere) für das Konto autorisiert sind. Falls nicht, liefert die Funktion E_UNAUTHORIZED zurück</p> <p>2.4) Falls obere Prüfungen gut gegangen sind, wird E_OK zurückgeliefert und es geht mit 3) weiter</p> <p>3) Konto wird auf Inaktiv gesetzt, Zuordnung Verfüger zum Konto wird entfernt und die Funktion liefert E_OK zurück</p>
Aufrufbeispiele:	<pre>CreateAccount(disposerId, "FirstAccount", LoanAccount, accountId); CreateAccount(0, "FirstAccount", LoanAccount, accountId);</pre>

Funktion:	AddDisposer(int disposerId, int accountNumber, int newDisposerId)
Bezeichnung:	Verfüger zuweisen
Parameter:	int disposerId (aktueller Kunde bzw. Verfüger)
Parameter:	int accountNumber (Kontonummer vom aktuellen Kunden)
Parameter:	int newDisposerId (neuer Kunde bzw. Verfüger)
Rückgabewerte:	E_INVALID_PARAMETER, E_NEW_DISPOSER_NOT_FOUND, E_ACCOUNT_NOT_FOUND, E_CUSTOMER_NOT_FOUND, E_UNAUTHORIZED, E_OK (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	<p>1) Prüfung der Übergabeparametern. Alle drei Parameter werden müssen größer gleich 0 sein. Im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>2) Suche nach Konto und dazugehörige Verfüger (einen oder mehreren)</p> <p>2.1) Prüfung der Übergabeparametern, im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>2.2) Prüfung ob Konto existiert oder inaktiv, wenn eins von beiden zutreffen liefert die Funktion E_CUSTOMER_NOT_FOUND</p> <p>2.3) Prüfung ob Verfüger (ein oder mehrere) für das Konto autorisiert sind. Falls nicht, liefert die Funktion E_UNAUTHORIZED zurück</p> <p>2.4) Falls obere Prüfungen gut gegangen sind, wird E_OK zurückgeliefert und es geht mit 3) weiter</p> <p>3) Prüfung ob neue Verfüger bereits existiert, im Fehlerfall liefert die Funktion E_NEW_DISPOSER_NOT_FOUND</p> <p>4) Wenn alles gut geht, wird ein neuer Verfüger zu dem übergeben Konto zugewiesen und die Funktion liefert E_OK</p>
Aufrufbeispiele:	AddDisposer(0, 0, 33)

Funktion:	int RemoveDisposer(int disposerId, int accountNumber, int disposerToRemoveId)
Bezeichnung:	Verfüger löschen
Parameter:	int disposerId (Kunde bzw. Verfüger des Konotos)
Parameter:	int accountNumber (Konto von den Kunden)
Parameter:	int disposerToRemoveId (Verfüger der nicht mehr für das Konto berechtigt sein darf)
Rückgabewerte:	E_INVALID_PARAMETER, E_CANNOT_REMOVE_SELF, E_REMOVE_DISPOSER_NOT_FOUND, E_ACCOUNT_NOT_FOUND, E_CUSTOMER_NOT_FOUND, E_UNAUTHORIZED, E_OK (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	<p>1) Prüfung der Übergabeparametern. Alle drei Parameter werden müssen größer gleich 0 sein. Im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>2) Prüfung ob die übergeben Verfüger-ID die gleiche wie zu löschende Verfüger-ID, falls ja, liefert die Funktion Fehlercode E_CANNOT_REMOVE_SELF zurück</p> <p>3) Suche nach Konto und dazugehörige Verfüger (einen oder mehreren)</p> <p>3.1) Prüfung der Übergabeparametern, im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>3.2) Prüfung ob Konto existiert oder inaktiv, wenn eins von beiden zutreffen liefert die Funktion E_CUSTOMER_NOT_FOUND</p> <p>3.3) Prüfung ob Verfüger (ein oder mehrere) für das Konto autorisiert sind. Falls nicht, liefert die Funktion E_UNAUTHORIZED zurück</p> <p>3.4) Falls obere Prüfungen gut gegangen sind, wird E_OK zurückgeliefert und es geht mit 3) weiter</p> <p>4) Prüfung ob zu löschender Verfüger existiert, wenn nicht liefert die Funktion einen Fehlercode E_REMOVE_DISPOSER_NOT_FOUND zurück</p> <p>5) Wenn alles gut geht, wird Zuweisung zwischen Konto und Verfüger und umgekehrt dem Verfüger das Konto entfernt. Die Funktion liefert E_OK zurück</p>
Aufrufbeispiele:	RemoveDisposer(0, 0, 1);

1.5 Funktionsbeschreibung, Komponente Transaktion (TransactionModul)

Funktion:	int PayOut(int disposerId, int accountNumber, double amount, CURRENCY currency)
Bezeichnung:	Bargeld abheben
Parameter:	int disposerId
Parameter:	int accountNumber
Parameter:	double amount
Parameter:	CURRENCY currency (ein Enum mit allen gängigen Währungen)
Rückgabewerte:	E_INVALID_PARAMETER, E_ACCOUNT_NOT_FOUND, E_CUSTOMER_NOT_FOUND, E_UNAUTHORIZED, E_CURRENCY_FACTOR_NOT_STORED, E_INSUFFICIENT_FUNDS, E_OK (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	<p>1) Prüfung der Übergabeparametern. Kunden- und Kontonummer müssen größer oder gleich 0 und Amount größer 0 sein. Im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>2) Suche nach autorisierten Verfüger. E_OK falls gut gegangen, sonst:</p> <ul style="list-style-type: none">- E_INVALID_PARAMETER, falls die Daten für das Konto oder Verfüger falsch übergeben worden sind- E_ACCOUNT_NOT_FOUND, falls ungültiges Konto (NULL, oder inaktiv)- E_CUSTOMER_NOT_FOUND, falls Verfüger (Kunde) nicht gefunden (NULL, oder inaktiv)- E_UNAUTHORIZED, falls der Verfüger für das Konto nicht autorisiert ist <p>3) Kontostandprüfung. Rückgabe E_OK, falls gut gegangen, sonst:</p> <ul style="list-style-type: none">- E_INVALID_PARAMETER, falls die Daten für das Konto oder Verfüger falsch übergeben worden sind- Rückgabewerte aus 2), da Prüfung auf autorisierten Verfüger <p>4) Umrechnung der übergebenen Währung in Euro. E_OK falls gut gegangen, sonst E_CURRENCY_FACTOR_NOT_STORED, falls kein Umrechnungsfaktor für die Währung gespeichert ist</p>

	5) Falls alles gut geht, wird die Auszahlung gespeichert und die Funktion liefert E_OK
Aufrufbeispiele:	PayOut(0, 0, 150, EUR); PayOut(0, 0, 150, USD);
Funktion:	PayIn(int disposerId, int accountNumber, double amount, CURRENCY currency)
Bezeichnung:	Bargeld einzahlen
Parameter:	int disposerId
Parameter:	int accountNumber
Parameter:	double amount
Parameter:	CURRENCY currency (ein Enum mit allen gängigen Währungen)
Rückgabewerte:	E_INVALID_PARAMETER, E_ACCOUNT_NOT_FOUND, E_CUSTOMER_NOT_FOUND, E_UNAUTHORIZED, E_CURRENCY_FACTOR_NOT_STORED, E_OK (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	<p>1) Prüfung der Übergabeparametern. Kunden- und Kontonummer müssen größer oder gleich 0 und Amount größer 0 sein. Im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>2) Suche nach autorisierten Verfüger. E_OK falls gut gegangen, sonst:</p> <ul style="list-style-type: none">- E_INVALID_PARAMETER, falls die Daten für das Konto oder Verfüger falsch übergeben worden sind- E_ACCOUNT_NOT_FOUND, falls ungültiges Konto (NULL, oder inaktiv)- E_CUSTOMER_NOT_FOUND, falls Verfüger (Kunde) nicht gefunden (NULL, oder inaktiv)- E_UNAUTHORIZED, falls der Verfüger für das Konto nicht autorisiert ist <p>3) Ermittlung des Faktors für die übergeben Währung. E_OK, falls gut gegangen, sonst E_CURRENCY_FACTOR_NOT_STORED, wenn der Faktor für die Währung nicht gespeichert ist</p> <p>4) Falls alles gut geht, wird die Einzahlung gespeichert und die Funktion liefert E_OK zurück</p>
Aufrufbeispiele:	PayIn(0, 0, 150, EUR); PayIn(0, 0, 150, USD);

Funktion:	int Transfer(int disposerId, int fromAccountNumber, int toAccountNumber, double amount, CURRENCY currency)
Bezeichnung:	Überweisung
Parameter:	int disposerId (Kunde der die Überweisung tätigt)
Parameter:	int fromAccountNumber (Kunden seine Kontonummer)
Parameter:	int toAccountNumber(Zielkonto)
Parameter:	double amount
Parameter:	CURRENCY currency (ein Enum mit allen gängigen Währungen)
Rückgabewerte:	E_INVALID_PARAMETER, E_ACCOUNT_NOT_FOUND, E_CUSTOMER_NOT_FOUND, E_UNAUTHORIZED, E_TARGET_ACCOUNT_NOT_FOUND, E_CURRENCY_FACTOR_NOT_STORED, E_INSUFFICIENT_FUNDS, E_OK (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	<p>1) Prüfung der Übergabeparametern. Kunden- und seine Kontonummer müssen größer oder gleich 0 und Amount größer 0 sein. Im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>2) Suche nach autorisierten Verfüger. E_OK falls gut gegangen, sonst:</p> <ul style="list-style-type: none">- E_INVALID_PARAMETER, falls die Daten für das Konto oder Verfüger falsch übergeben worden sind- E_ACCOUNT_NOT_FOUND, falls ungültiges Konto (NULL, oder inaktiv)- E_CUSTOMER_NOT_FOUND, falls Verfüger (Kunde) nicht gefunden (NULL, oder inaktiv)- E_UNAUTHORIZED, falls der Verfüger für das Konto nicht autorisiert ist <p>3) Prüfung des Zielkontos. E_OK falls gut gegangen, sonst E_TARGET_ACCOUNT_NOT_FOUND</p> <p>4) Kontostandprüfung. Rückgabe E_OK, falls gut gegangen, sonst:</p> <ul style="list-style-type: none">- E_INVALID_PARAMETER, falls die Daten für das Konto oder Verfüger falsch übergeben worden sind- Rückgabewerte aus 2), da Prüfung auf autorisierten Verfüger <p>5) Umrechnung der übergebenen Währung in Euro. E_OK falls gut gegangen, sonst E_CURRENCY_FACTOR_NOT_STORED, falls kein Umrechnungsfaktor für die Währung gespeichert ist</p>

6) Falls das Konto von dem die Überweisung getätigt wird ein Sparkonto ist, und der Betrag kleiner als Kontostand, liefert die Funktion E_INSUFFICIENT_FUNDS zurück

7) Falls alles gut geht, wird die Überweisung abgespeichert und die Funktion liefert E_OK zurück

Funktion:	int AccountStatement(int disposerId, int accountNumber, S_TRANSACTION*** data, int& numberOfEntries)
Bezeichnung:	Kontoauszug
Parameter:	int disposerId
Parameter:	int accountNumber
Parameter:	S_TRANSACTION*** data (Eine Struktur mit Pointer auf Array von Transaktionen das wiederum auf die Daten zeigt)
Parameter:	int& numberOfEntries
Rückgabewerte:	E_INVALID_PARAMETER, E_ACCOUNT_NOT_FOUND, E_CUSTOMER_NOT_FOUND, E_UNAUTHORIZED, E_OK (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	<p>1) Prüfung der Übergabeparametern. Kunden- und seine Kontonummer müssen größer oder gleich 0 und der Pointer nicht NULL. im Fehlerfall: Rückgabewert E_INVALID_PARAMETER</p> <p>2) Suche nach autorisierten Verfüger. E_OK falls gut gegangen, sonst:</p> <ul style="list-style-type: none"> - E_INVALID_PARAMETER, falls die Daten für das Konto oder Verfüger falsch übergeben worden sind - E_ACCOUNT_NOT_FOUND, falls ungültiges Konto (NULL, oder inaktiv) - E_CUSTOMER_NOT_FOUND, falls Verfüger (Kunde) nicht gefunden (NULL, oder inaktiv) - E_UNAUTHORIZED, falls der Verfüger für das Konto nicht autorisiert ist <p>3) Falls alles gut geht, liefert die Funktion E_OK, und ermöglicht einen Zugriff auf die S_TRANSACTION</p>
Aufrufbeispiele:	AccountStatement(0, 0, &transactions, numOfEntries);

Funktion:	<code>int AccountBalancing(int disposerId, int accountNumber, CURRENCY currency, double& balance)</code>
Bezeichnung:	Kontostand
Parameter:	<code>int disposerId</code>
Parameter:	<code>int accountNumber</code>
Parameter:	<code>CURRENCY currency</code> (ein Enum mit allen gängigen Währungen)
Parameter:	<code>double& balance</code>
Rückgabewerte:	<code>E_INVALID_PARAMETER</code> , <code>E_ACCOUNT_NOT_FOUND</code> , <code>E_CUSTOMER_NOT_FOUND</code> , <code>E_UNAUTHORIZED</code> , <code>E_CURRENCY_FACTOR_NOT_STORED</code> , <code>E_OK</code> (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	<p>1) Prüfung der Übergabeparametern. Kunden- und seine Kontonummer müssen größer oder gleich 0 sein. Im Fehlerfall: Rückgabewert <code>E_INVALID_PARAMETER</code></p> <p>2) Suche nach autorisierten Verfüger. <code>E_OK</code> falls gut gegangen, sonst:</p> <ul style="list-style-type: none">- <code>E_INVALID_PARAMETER</code>, falls die Daten für das Konto oder Verfüger falsch übergeben worden sind- <code>E_ACCOUNT_NOT_FOUND</code>, falls ungültiges Konto (NULL, oder inaktiv)- <code>E_CUSTOMER_NOT_FOUND</code>, falls Verfüger (Kunde) nicht gefunden (NULL, oder inaktiv)- <code>E_UNAUTHORIZED</code>, falls der Verfüger für das Konto nicht autorisiert ist <p>2) Kontostand wird berechnet und in die Übergebene Währung umgerechnet (weil alle Transaktionen werden in Euro gespeichert). Wenn alles gute geht, liefert die Funktion <code>E_OK</code>, sonst <code>E_CURRENCY_FACTOR_NOT_STORED</code>, falls Umrechnungsfaktor nicht gespeichert ist</p>
Aufrufbeispiele:	<code>AccountBalancing(0, 0, EUR, amount);</code>

1.6 Funktionsbeschreibung, Komponente Währungsumrechnung (CurrencyTranslationModul)

Das Währungsmodul hängt mit dem Transaktionsmodul zusammen. Sobald eine Transaktion durchgeführt wird, wird auch das Währungsmodul eingebunden.

Funktion:	int SetCurrencyToEuroFactor(CURRENCY currency, double factor)
Bezeichnung:	Faktor für Währung zu Euro setzten
Parameter:	CURRENCY currency (Enum)
Parameter:	double factor
Rückgabewerte:	E_INVALID_PARAMETER, E_OK (Siehe Kapitel 1.8 Error-code liste)
Funktionsbeschreibung:	Prüfung der Übergabeparametern. Währungsfaktor darf nicht 0 sein und Währung darf nicht Eur sein. Im Fehlerfall: Rückgabewert E_INVALID_PARAMETER
Aufrufbeispiele:	SetCurrencyToEuroFactor(USD, 1);
Funktion:	GetCurrencyToEuroFactor(CURRENCY currency, double & factor)
Bezeichnung:	Faktor Währung zu Euro ermitteln
Parameter:	CURRENCY currency (ein Enum mit allen gängigen Währungen)
Parameter:	double & factor
Rückgabewerte:	E_OK, E_CURRENCY_FACTOR_NOT_STORED (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	falls der Faktor für die angegebene Währung nicht ermittelt werden kann, liefert die Funktion E_CURRENCY_FACTOR_NOT_STORED und sonst E_OK
Aufrufbeispiele:	GetCurrencyToEuroFactor(EUR, factor);

Funktion:	TranslateToEuro(CURRENCY currency, double amount, double & result)
Bezeichnung:	Auf Euro umrechnen
Parameter:	CURRENCY currency (ein Enum mit allen gängigen Währungen)
Parameter:	double amount
Parameter:	double & result
Rückgabewerte:	E_OK, E_CURRENCY_FACTOR_NOT_STORED (Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	falls der Faktor für die angegebene Währung nicht ermittelt werden kann, liefert die Funktion E_CURRENCY_FACTOR_NOT_STORED und sonst wird Umgerechnet und E_OK zurückgeliefert
Aufrufbeispiele:	TranslateToEuro(USD, 1, result);
Funktion:	TranslateFromEuro(CURRENCY currency, double amount, result) double
Bezeichnung:	Von Euro aus umrechnen
Parameter:	CURRENCY currency (ein Enum mit allen gängigen Währungen)
Parameter:	double amount
Parameter:	double & result
Rückgabewerte:	E_OK, E_CURRENCY_FACTOR_NOT_STORED(Siehe Kapitel 1.8 Errorcode liste)
Funktionsbeschreibung:	falls der Faktor für die angegebene Währung nicht ermittelt werden kann, liefert die Funktion E_CURRENCY_FACTOR_NOT_STORED und sonst wird Umgerechnet und E_OK zurückgeliefert
Aufrufbeispiele:	TranslateFromEuro(USD, 1, result);

1.7 Funktionsbeschreibung, Komponente Persistenz (PersistenceModul)

Funktion: int Store()

Bezeichnung: **Daten speichern**

Rückgabewerte: E_OK (Siehe Kapitel 1.8 Errorcode liste)

Funktionsbeschreibung: Der Aufruf speichert die Daten die shared memory (Pfad, wo sich die Komponenten befinden)

Funktion: int Load()

Bezeichnung: **Daten laden**

Rückgabewerte: E_OK (Siehe Kapitel 1.8 Errorcode liste)

Funktionsbeschreibung: der Aufruf ruft Daten, die auf shared memory zu Verfügung stehen auf und bilden eine DQL-Lite Datenbank auf

1.8 Errorcode Liste

pragma region OK

define E_OK 0

pragma endregion

pragma region Unexpected

define E_NOT_EXPECTED -1

pragma endregion Unexpected

pragma region Authorization

define E_UNAUTHORIZED -10

pragma endregion Unauthorized

pragma region Initialize

define E_NOT_INITIALIZED -20

pragma endregion Initialize

pragma region Parameter

define E_INVALID_PARAMETER -30

pragma endregion Parameter

pragma region Customer

define E_CUSTOMER_NOT_FOUND -41

pragma endregion Customer

pragma region Account

define E_ACCOUNT_NOT_FOUND -51

define E_NEW_DISPOSER_NOT_FOUND -52

define E_REMOVE_DISPOSER_NOT_FOUND -53

define E_CANNOT_REMOVE_SELF -54

pragma endregion Account

pragma region CurrencyTranslation

define E_CURRENCY_FACTOR_NOT_STORED -60

pragma endregion CurrencyTranslation

```
pragma region Transaction
define E_INSUFFICIENT_FUNDS -70
define E_TARGET_ACCOUNT_NOT_FOUND -71
pragma endregion Transaction
```

1.9 Fehlerbehandlung, allgemein mögliche Übergabeparameter

Check String - ein String darf kein nullpointer sein

Check Zip - Es dürfen Werte zwischen 1000 und 9999 eingegeben werden

Check Id - die Id muss größer/gleich 0 sein

Check Currency Factor - Währungsfaktor muss größer 0 sein

Check Amount - Ein Betrag ist immer Positiv also größer 0

2 Anhang

A) Source Files

B) Doku

C) Header Files

3 TODO

(Alle TODO's erledigt. Topic bleibt trotzdem, falls es weitere TODO's dazukommen)