Universität Potsdam
Institut für Informatik
Professur Informatik II
Prof. Dr. E. Horn
Lehrveranstaltung Softwarebauelemente II

Potsdam, 31. August 2001

Praktikumsdokumentation

Aufgabe: Persistente Verwaltung der Daten einer Dokumentenmappe unter Verwendung von C++ und MFC

Autor: Stephan Brumme, Matrikelnr. 702544

Umfang: 182 Seiten und 1 CD

Seminarbetreuer: Dr. Wolfgang Schubert

Bearbeitungsvermerk:

Aufbau der Dokumentation

Inhaltsverzeichnis

ANHANG 50

Form

Als Standardschriftart wird *Times New Roman* in der Größe 10 verwendet. Der Zeilenabstand ist nicht vergrößert worden, jedoch wird jeder Absatz mit einer leicht eingerückten Zeile begonnen.

Besonders wichtige Wörter sind *kursiv* hervorgehoben. Dateinamen erkennt man an der Schriftart Arial.

In dieser Dokumentation werden Namen aus den Quelltexten in der Schriftart `Courier` gesetzt, wobei der Anhang auf die Größe `8` reduziert wurde. Der Quelltext selber wurde mit farbigen Syntax-Highlighting bearbeitet, um die Lesbarkeit zu erhöhen.

1. Aufgabenstellung

Universität Potsdam
Institut für Informatik
Professur Software Engineering
Prof. Dr. Erika Horn

Aufgaben für das Praktikum zur Lehrveranstaltung "Softwarebauelemente II"

Entwickeln Sie eine objektorientierte Lösung für die Problemstellung "persistente Verwaltung der Daten einer Dokumentenmappe" in den Programmiersprachen C++ oder Java.
Die Bausteine des Programmsystem sollen in Teilsystemen abgelegt werden.
Bausteine des Programmsystems sind mindestens die Klassen

CBankDocument	Wurzelklasse der Dokumentenhierarchie
CForm	allgemeines Formular
CStandingOrder	Dauerauftrag
CContract	Vertrag
CFolder	Mappe von Dokumenten für einen Kunden

Die Beschreibung der Klassen ist als Anhang beigefügt.

Es sind folgende Klassenbibliotheken zu benutzen

C++: MFC oder Qt
Java: JFC.

Das Programmsystem soll als MVC Applikation aufgebaut werden.
Definieren Sie geeignete Fensterstrukturen für die Präsentation der Funktionalität Ihres Programmsystems und zur Durchführung der Ein-/Ausgaben.
Bestimmen Sie die Klassen aus der von Ihnen gewählten Klassenbibliothek, die für die Nachnutzung in Ihrer programmtechnischen Lösung geeignet sind, und leiten Sie daraus die Architektur Ihres Programmsystems ab.

Prozedurparameternamen und ihre Semantik

Klasse CDate

dat	date
da	day
mo	mo
ye	year
se	second
mi	minute
hou	hour

Klasse CBankDocument

bdoc	bankdocument
aut	author
tit	title
key	key

Klasse CForm

form	form
prv	proved
sig	signed
fty	formality

Klasse CStandingOrder

sord	standing order
amo	amount
sub	subject
sou	source
rec	recipient
int	interval

Klasse CContract

cont	contract
numb	registration number
ada	alternation date
prv	proved
sig	signed
word	wording

Klasse CFolder

fold	folder
cinf	customer information
card	cardinality

Klasse CDate

```
class CDate ... classend // Date&Time
```

Beschreibung der Klasse CBankDocument

```
class CBankDocument
import CDate;

     attributes
          private Author: String,
          private DateOfFoundation: CDate,
          private Title: String,
          private Key: String;

     operations
          public +CBankDocument( ),
          // Constructs an object of 'CBankDocument'.

          public +CBankDocument(in aut: String, in tit: String, in key: String),
          // Constructs an object of 'CBankDocument' and
          // initializes the attributes.

          public +CBankDocument(in bdoc: CBankDocument),
          // Constructs an object of 'CBankDocument' and
          // initializes the attributes with the
          // attributes of bdoc.

          public virtual -CBankDocument( ),
          // Destructs the object.

          public virtual Generate(out bdoc: CBankDocument),
          // Clones an object.

          public virtual ClassInvariant(out t: Boolean),
          // Proofs the invariance. Returns "TRUE", if the invariance
          // is valid (strings are not empty), else returns "FALSE".

          public virtual Equal(in bdoc: CBankDocument, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the objects are
          // identical, else returns "FALSE".

          public virtual EqualValue(in bdoc: CBankDocument, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the attributes
          // of the Objects are equal, else returns "FALSE".

          public EqualKey(in bdoc: CBankDocument, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the keys
          // of the Objects are equal, else returns "FALSE".

          public virtual KeyOf(out key: String),
          // Delivers the Key of the object.

          public virtual Copy(in bdoc: CBankDocument, out t: Boolean),
          // Copies the object, if it is possible.

          public GetAuthor(out aut: String),

          public GetDateOfFoundation(out dat: CDate),

          public GetTitle(out tit: String),

          public GetKey(out key: String),

          public virtual Show( );
          // Shows the attributes of a BankDocument.

// Valid cases
     //
     // Case 1:
     // require: The invariance (ClassInvariant) of bdoc2 as an instance of
     // 'CBankDocument' is "TRUE".
     // The object bdoc1 is an instance of 'CBankDocument'.
     // sequence:
     // bdoc1.Copy(bdoc2, t1);
     // bdoc1.ClassInvariant(t2);
     // bdoc1.EqualValue(bdoc2, t3)
     // ensure: The values of t1, t2 and t3 are "TRUE".
     //
classend
```

Beschreibung der Klasse CForm

```
class CForm: inherit public CBankDocument;
import CDate, CBankDocument;

     attributes
          private Proved: Boolean,
          private Signed: Boolean,
          private Formality: String,
          private AlternationDate: CDate;

     operations
          public +CForm( ),
          // Constructs an object of 'CForm'.
```

```
          public +CForm(in aut: String, in tit: String, in key: String,
          in prv: Boolean, in sig: Boolean, in fty: String),
          // Constructs an object of 'CForm' and
          // initializes the attributes.

          public +CForm(in form: CForm),
          // Constructs an object of 'CForm' and
          // initializes the attributes with the
          // attributes of form.

          public virtual -CForm( ),
          // Destructs the object.

          public virtual Generate(out form: CForm),
          // Clones an object.

          public virtual ClassInvariant(out t: Boolean),
          // Proofs the invariance. Returns "TRUE", if the invariance
          // is valid (strings are not empty and the invariance of the
          // inherited class is valid), else returns "FALSE".

          public virtual Equal(in form: CForm, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the objects are
          // identical, else returns "FALSE".

          public virtual EqualValue(in form: CForm, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the attributes
          // of the Objects are equal, else returns "FALSE".

          public virtual KeyOf(out key: String),
          // Delivers the Key of the object.

          public virtual Copy(in form: CForm, out t: Boolean),
          // Copies the object, if it is possible.

          public GetProved(out prv: Boolean),

          public SetProved(in prv: Boolean),

          public GetSigned(out sig: Boolean),

          public SetSigned(in sig: Boolean),

          public GetFormality(out fty: String),

          public GetAlternationDate(out dat: CDate),

          public virtual Show( );

// Valid cases
          //
          // Case 1:
          // require: The invariance (ClassInvariant) of form2 as an instance of
          // 'CForm' is "TRUE".
          // The object form1 is an instance of 'CForm'.
          // sequence:
          // form1.Copy(form2, t1);
          // form1.ClassInvariant(t2);
          // form1.EqualValue(form2, t3)
          // ensure: The values of t1, t2 and t3 are "TRUE".
          //
classend
```

Beschreibung der Klasse CStandingOrder

```
class CStandingOrder: inherit public CForm;
import CDate, CForm;

     attributes
          private Amount: Ordinal,
          private Subject: String,
          private Source: String,
          private Recipient: String,
          private Interval: Ordinal;

     operations
          public +CStandingOrder( ),
          // Constructs an object of 'CStandingOrder'.

          public +CStandingOrder(in aut: String, in tit: String, in key: String,
          in prv: Boolean, in sig: Boolean, in fty: String,
          in amo: Ordinal, in sub: String, in sou: String,
          in rec: String, in int: Ordinal),
          // Constructs an object of 'CStandingOrder' and
          // initializes the attributes.

          public +CStandingOrder(in sord: CStandingOrder),
          // Constructs an object of 'CStandingOrder' and
          // initializes the attributes with the
          // attributes of sordi.

          public virtual -CStandingOrder( ),
          // Destructs the object.

          public virtual Generate(out sord: CStandingOrder),
          // Clones an object.

          public virtual ClassInvariant(out t: Boolean),
```

```
          // Proofs the invariance. Returns "TRUE", if the invariance
          // is valid (strings are not empty, the ordinals are not zero
          // and the invariance of the inherited class is valid),
          // else returns "FALSE".

          public virtual Equal(in sord: CStandingOrder, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the objects are
          // identical, else returns "FALSE".

          public virtual EqualValue(in sord: CStandingOrder, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the attributes
          // of the Objects are equal, else returns "FALSE".

          public virtual KeyOf(out key: String),
          // Delivers the Key of the object.

          public virtual Copy(in sord: CStandingOrder, out t: Boolean),
          // Copies the object, if it is possible.

          public GetAmount(out amo: Ordinal),

          public SetAmount(in amo: Ordinal),

          public GetSubject(out sub: String),

          public GetSource(out sou: String),

          public GetRecipient(out rec: String),

          public GetInterval(out int: Ordinal),

          public SetInterval(in int: Ordinal),

          public virtual Show( );

// Valid cases
          //
          // Case 1:
          // require: The invariance (ClassInvariant) of sord2 as an instance of
          // 'CStandingOrder' is "TRUE".
          // The object sord1 is an instance of 'CStandingOrder'.
          // sequence:
          // sord1.Copy(sord2, t1);
          // sord1.ClassInvariant(t2);
          // sord1.EqualValue(sord2, t3)
          // ensure: The values of t1, t2 and t3 are "TRUE".
          //
classend
```

Beschreibung der Klasse CContract

```
class CContract: inherit public CBankDocument;
import CDate, CBankDocument;

     attributes
          private RegistrationNumber: Ordinal,
          private AlternationDate: CDate,
          private Proved: Boolean,
          private Signed: Boolean,
          private Wording: String;

     operations
          public +CContract( ),
          // Constructs an object of 'CContract'.

          public +CContract(in aut: String, in tit: String, in key: String,
          in numb: Ordinal, in prv: Boolean, in sig: Boolean,
          in word: String),
          // Constructs an object of 'CContract' and
          // initializes the attributes.

          public +CContract(in cont: CContract),
          // Constructs an object of 'CContract' and
          // initializes the attributes with the
          // attributes of cont.

          public virtual -CContract( ),
          // Destructs the object.

          public virtual Generate(out cont: CContract),
          // Clones an object.

          public virtual ClassInvariant(out t: Boolean),
          // Proofs the invariance. Returns "TRUE", if the invariance
          // is valid (strings are not empty and the invariance of the
          // inherited class is valid), else returns "FALSE".

          public virtual Equal(in cont: CContract, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the objects are
          // identical, else returns "FALSE".

          public virtual EqualValue(in cont: CContract, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the attributes
          // of the Objects are equal, else returns "FALSE".

          public virtual KeyOf(out key: String),
          // Delivers the Key of the object.
```

```
          public virtual Copy(in cont: CContract, out t: Boolean),
          // Copies the object, if it is possible.

          public GetRegistrationNumber(out numb: Ordinal),

          public GetAlternationDate(out dat: CDate),

          public GetProved(out prv: Boolean),

          public SetProved(in prv: Boolean),

          public GetSigned(out sig: Boolean),

          public SetSigned(in sig: Boolean),

          public GetWording(out word: String),

          public SetWording(in word: String),

          public virtual Show( );

// Valid cases
          //
          // Case 1:
          // require: The invariance (ClassInvariant) of cont2 as an instance of
          // 'CContract' is "TRUE".
          // The object cont1 is an instance of CContract.
          // sequence:
          // cont1.Copy(cont2, t1);
          // cont1.ClassInvariant(t2);
          // cont1.EqualValue(cont2, t3)
          // ensure: The values of t1, t2 and t3 are "TRUE".
          //
classend
```

Beschreibung der Klasse CFolder

```
class CFolder
import CDate, CBankDocument;

     types TSetOfBankDocuments = ordered set of CBankDocument;

     attributes
          private CustomerInformation: String,
          private DateOfFoundation: CDate,
          private Set: TSetOfBankDocuments,
          private Count: Ordinal,
          private Cursor: Ordinal;

     operations
          public +CFolder( ),
          // Constructs an object of 'CFolder'.

          public +CFolder(in cinf: String),
          // Constructs an object of 'CFolder' and
          // initializes the attributes.

          public +CFolder(in fold: CFolder),
          // Constructs an object of 'CFolder' and initializes the attributes
          // with the attributes of fold.

          public virtual -CFolder( ),
          // Destructs the object.

          public virtual Generate(out fold: CFolder),
          // Clones an object.

          public virtual ClassInvariant(out t: Boolean),
          // Proofs the invariance. Returns "TRUE", if the set contains
          // each BankDocument only once and the invariance of each BankDocument
          // is "TRUE", else returns "FALSE".

          public virtual Equal(in fold: CFolder, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the objects are
          // identical, else returns "FALSE".

          public virtual EqualValue(in fold: CFolder, out t: Boolean),
          // Compares two sets. Returns "TRUE", if both sets contains
          // the same elements, else returns "FALSE".

          public EqualKey(in fold: CFolder, out t: Boolean),
          // Compares two objects. Returns "TRUE", if the keys
          // of the Objects are equal, else returns "FALSE".

          public virtual KeyOf(out key: String),
          // Delivers the Key of the object.

          public virtual Copy(in fold: CFolder, out t: Boolean),
          // Copies the object, if it is possible.

          public GetCustomerInformation(out cinf: String),
          // Delivers folders's customer information.

          public GetDateOfFoundation(out dat: CDate),
          // Delivers folders's date of foundation.

          public Card(out crd: Ordinal),
          // Delivers the cardinality of set.
```

```
          public Insert(in bdoc: CBankDocument, out t: Boolean),
          // Inserts a BankDocument in the set, if it is possible.

          public GetFirst(out bdoc: CBankDocument, out t: Boolean),
          // Gets the first BankDocument in the set, if it is possible.

          public GetNext(out bdoc: CBankDocument, out t: Boolean),
          // Gets the next BankDocument in the set, if it is possible.

          public Find(in bdoc: CBankDocument, out t: Boolean),
          // Sets the cursor on an object in the set, which value is
          // equal to per, if it is possible.

          public GetActual(out bdoc: CBankDocument, out t: Boolean),
          // Delivers the actual object in the set, if it is possible.

          public Scratch(out t: Boolean),
          // Scratches the actual object in the set, if it is possible.

          public virtual Show( );

// Valid cases
          //
          // Case 1:
          // require: The invariance (ClassInvariant) of fold2 as an instance of
          // CFolder is "TRUE".
          // The object fold1 is an instance of 'CFolder'.
          // sequence:
          // fold1.Copy(fold2, t1);
          // fold1.ClassInvariant(t2);
          // fold1.EqualValue(fold2, t3)
          // ensure: The values of t1, t2 and t3 are "TRUE".
          //
          // Case 2:
          // require: Object fold as an instance of 'CFolder' is not full
          // and invariance (ClassInvariant) of bdoc1 is "TRUE".
          // sequence:
          // fold.Insert(bdoc1, t1);
          // fold.Find(bdoc1, t2);
          // fold.GetActual(bdoc2, t3)
          // ensure: The elements bdoc1 and bdoc2 have the same value
          // (EqualValue).
          // The values of t2 and t3 are "TRUE".
          //
          // Case 3:
          // require: Object fold as an instance of CFolder contains bdoc (Find).
          // sequence:
          // fold.Find(bdoc, t1);
          // fold.Scratch(t2);
          // fold.Find(bdoc, t3)
          // ensure: The value of t2 is "TRUE".
          // The value of t3 is "FALSE".
          //
          // Case 4:
          // require: Object fold as an instance of 'CFolder' is not full.
          // It not contains bdoc (Find), which invariance
          // (ClassInvariant) is "TRUE".
          // Object fold has the cardinality (Card) of n1.
          // sequence:
          // fold.Insert(bdoc, t1);
          // fold.Card(n2)
          // ensure: The value of n2 is equal n1 + 1.
          //
          // Case 5:
          // require: Object fold as an instance of 'CFolder' contains bdoc
          // (Find)
          // and has the cardinality (Card) of n1.
          // sequence:
          // fold.Find(bdoc, t1);
          // fold.Scratch(t2);
          // fold.Card(n2)
          // ensure: The value of n2 is equal n1 - 1.
classend
```

2. Erläuterung der Aufgabenstellung

Allgemeines

Die Aufgabenstellung verfolgt das Ziel, eine Klassenstruktur zu gewinnen, die die Verwaltung von Kundendaten einer Bank erlaubt. Dazu gehört der Entwurf einer Datenhaltung, die einfache Dokumente (`CBankDocument`), Verträge (`CContract`), Formulare (`CForm`) und Daueraufträge (`CStandingOrder`) in einer Kartei (`CFolder`) verwalten kann. Die zu entwickelnde Lösung soll mit einer graphischen Benutzeroberfläche versehen sein, so dass sie ohne weitere Programmierung direkt eingesetzt werden könnte.

Die gegebene CEDL-Beschreibung ist bereits ein Modell, das 1:1 in die Praxis umsetzbar ist. Ich empfand es dennoch als günstiger, einige Erweiterungen in Form zusätzlicher Klassen und Methoden einzuführen, um die Realisierung zu vereinfachen. Damit ziele ich auf eine klarere Klassenstruktur und Zusammenfassung sich wiederholender Abläufe ab. Es ist aber sichergestellt, dass jede einzelne Funktionalität, die in der CEDL-Beschreibung gefordert wird, sich auch in meinem Projekt wiederfindet.

Eine Form der Qualitätssicherung besteht in der analytischen Überprüfung der Klassenstruktur. Zu diesem Zwecke entstand eine eigene Testumgebung, die der Programmierer interaktiv nutzen kann, um alle Klassen in beliebiger Kombination zu erstellen und mit ihnen zu arbeiten

Hinweis: Diese Dokumentation liegt auch in elektronischer Form auf dem beiliegenden Datenträger vor. Im Falle eventueller Rückfragen etc. bin ich jederzeit per Email unter mail@stephan-brumme.com erreichbar.

Darstellung des MVC-Konzeptes

Das Model-View-Controller-Konzept (MVC) baut auf der Idee auf, dass eine Anwendung aus 3 Teilsystemen besteht, die die verschiedenen Grundaspekte repräsentieren:

- Datenhaltung
- Darstellung der Daten
- Interaktion mit dem Benutzer

Ein viertes Teilsystem ist der Applikationsrahmen, der die drei genannten zur eigentlichen Anwendung vereint:

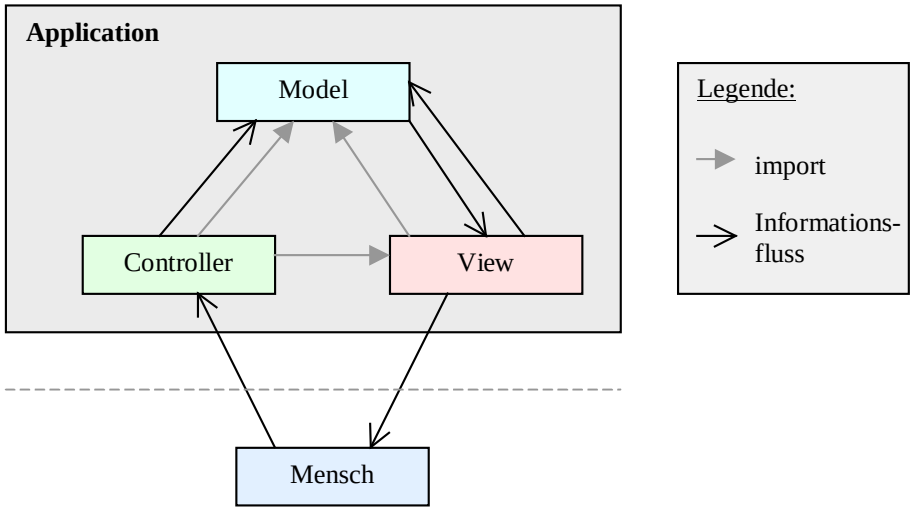

Abbildung 1: MVC-Modell

Diese idealtypische Konstruktion lässt sich nur schwer 1:1 auf ein reales Softwaresystem abbilden. Die Begründung, warum dies so ist, findet sich im Punkt 4.2.2.

Model

Das *Model* ist einzig und allein für die Datenhaltung zuständig. Deshalb muss es innerhalb der *Application* als erstes erzeugt und als letztes vernichtet werden. Der Zustand des *Models* beeinflusst den Zustand des *Views* und des *Controllers*.

In der Dokumentenmappe umfasst das *Model* die Datenstrukturen für einfache Dokumente, Verträge, Formulare und Daueraufträge sowie die Kartei, in der erstere enthalten sind.

View

Das *View* stellt die Daten des *Models* graphisch dar. Jede Änderung im *Model* muss zu einer Änderung des *Views* führen. Dazu ist eine angepasste Repräsentation einer jeden für den Benutzer relevanten Datenstruktur notwendig, was wiederum alle genannten Arten von Dokumenten sowie die Kartei sind. Es ist möglich, dass ein *Model* über verschiedene *Views* verfügbar, um jeweils bestimmte Teilaspekte optimiert darstellen zu können.

Controller

Der *Controller* übernimmt die Interaktion mit dem Benutzer. Er verarbeitet alle Eingaben und Kommandos und stellt die Ergebnisse auf dem Bildschirm dar. Um diese Tätigkeit durchführen zu können, muss er sowohl Zugriff auf das *Model* (Vornehmen von Änderungen) als auch auf das *View* (Anzeige der Änderungen) haben.

Application

Die *Application* vereinigt *Model*, *View* und *Controller* zu einem vollwertigen Programm. Hier wird die Initialisierung, der Programmlauf und die Terminierung gehandhabt. Das Betriebssystem kommuniziert nur mit der *Application*.

Nutzung der Klassenbibliotheken und daraus resultierende technische Konsequenzen

Warum Klassenbibliotheken ?

Klassenbibliotheken stellen Datenstrukturen, Konzepte und Algorithmen bereit, die die Umsetzung des MVC-Modells in ein lauffähiges Programm unterstützen. Sie sind vielfach getestet und erprobt, was mir einen Großteil der Qualitätssicherung erspart, so dass man sich auf das Wesentliche konzentrieren kann.

Allerdings schränken Klassenbibliotheken auch den Entwickler durch Vorgaben in seinen Möglichkeiten und seiner Flexibilität ein. Dieser Nachteil muss in Kauf genommen werden, da die Vorteile doch stark überwiegen.

Auswirkungen auf die Realisierung im Ganzen

Auch teilsystemübergreifende – und damit MVC-unabhängige – Konzepte werden durch Klassenbibliotheken geprägt. An erster Stelle steht dabei der Aufbau der internen Fehlererkennung und -behandlung. Darunter hat man die Sicherstellung von Invarianten und die Ausgabe von Objektzuständen zu verstehen.

Ebenso werden objektorientierte Entwurfskonzepte entscheidend beeinflusst, da viele Bibliotheken Mehrfachvererbung nicht erlauben und auch eigene Objekterzeugungsmechanismen anbieten und verlangen. Ähnlich sieht es mit Typinformationen aus, die über die in C++ standardmäßig eingebauten hinausgehen.

Nicht zu verachten sind die diversen Tools, die die Arbeit mit Klassenbibliotheken vereinfachen und beschleunigen, aber dadurch auch dem Programmierer Schranken auferlegen. So ist es nicht ohne Umwege möglich, mit Design-Werkzeugen entworfene Oberflächen zu portieren oder nachträglich entscheidend zu ändern.

Model

Das *Model* profitiert insbesondere von vorgefertigten Containerklassen und den dazugehörigen Algorithmen. In der vorliegenden Problemstellung kann die Kartei `CFolder` auf Basis eines Containers realisiert werden, welcher über separate Iteratoren angesprochen wird.

Für die Serialisierung sind für das Einlesen aus bzw. Schreiben in eine Datei Kommunikationsmechanismen erforderlich. Gerade das Einlesen ist zusätzlich auf die Fähigkeit einer dynamische Objekterzeugung angewiesen.

Um die Sicherstellung von Invarianten zu garantieren, stehen in den Klassenbibliotheken vorgefertigte Ideen bereit, die die Ausgabe eines Objektdumps ebenso unterstützen wie die Bearbeitung eines laufenden Programms im Debugger.

View

Die konkrete Darstellung der Daten auf dem Bildschirm wird durch eine Kapselung in einem *View* erreicht. Dieses kann auf Klassen zurückgreifen, die Standardelemente zur Verfügung stellen, wie z.B. Zeichenstifte, Schaltflächen oder Tabellenansichten. Idealerweise kann die Klassenbibliothek derart abstrakt sein, dass dabei die verwendete Shell des Betriebssystems oder das Betriebssystem selber für die Programmierung des *Views* irrelevant sind.

Die Klassenbibliothek sollte den Datenaustausch zwischen den Oberflächenelementen (z.B. einem Eingabefeld) und den dazugehörigen Variablen im Programm automatisiert unterstützen.

Gerade beim Entwurf des *Views* spielt die Verwendung von Design-Tools eine große Rolle. Sie werden meist als Ressourcen-Editoren bezeichnet und erlauben die Umsetzung von Fenstern, Menüs, Beschleunigertasten (Hotkeys), Bildern etc.. Problematisch ist, dass diese Elemente dynamisch zur Laufzeit nur eher umständlich verändert werden können.

Controller

Der *Controller* benötigt für die Interaktion mit dem Nutzer Zugriff auf Hardware-Ressourcen. Die Klassenbibliothek kann hier helfen, indem sie Datenfluss-Standards gewährleistet, die unabhängig von der beim Nutzer installierten Hardware sind.

Ebenso stellt sie Datenstrukturen und Klassen zur Verfügung, die die Weiterleitung von Informationen an das *View* und/oder das *Model* unterstützen. Der dazu verwendete Mechanismus wird in aller Regel vorgegeben und beruht meist auf Nachrichtenkommunikation.

Application

Schlussendlich ist auch die *Application* auf eine Klassenbibliothek angewiesen, um die Initialisierung, den Ablauf und die Beendigung der Teilsysteme *Model*, *View* und *Controller* zu gewährleisten. Dabei bedient sie sich vorgefertigter Algorithmen zur Registrierung im Betriebssystem.

Erläuterung der Begriffe sowie Lesart der Daten

Da es sich um eine objektorientierte Lösung handelt, sind für mich im folgenden die Begriffe *Modul* und *Klasse* synonym.

Ich verwende oft das Wort *Methode*, um eine *Funktion* einer Klasse zu beschreiben. Ein alternativer Ausdruck ist *Operation*, der aber für mich eine Verwechslungsgefahr mit Operator birgt.

Die Daten, d.h. Variablen einer Klasse heißen *Attribute*. Wenn ich den Unterschied zwischen einer lokalen Variablen und einem Attribut herausstellen will, dann bezeichne ich letzteres auch als *Member-Variable*.

Die graphische Notation der objektorientierten Beziehungen erfolgt mit Hilfe der *Unified Modelling Language* (UML). Sie wird im wesentlichen von der verwendeten Entwurfs-Software geprägt, die leider nicht in allen Punkten UML 1.3 entspricht.

Der Quellcode ist vollständig in englisch geschrieben und kommentiert, weil Englisch quasi *die* Sprache der Informatik ist. Deshalb fällt es mir öfters schwer, für die verwendeten englischen Begriffe deutsche Entsprechungen zu finden. Im Zweifel benutze ich daher den englischen Ausdruck, wenn es sich um einen geläufigen Standard handelt.

3. Architekturentwurf

Anforderungen an das System

Benutzungsszenarios und Dialogdatensichten

Der Benutzer muss in der Lage sein, Bankdokumente, Formulare, Verträge und Daueraufträge in einer Kartei verwalten zu können. Jede der 5 genannten Kategorien lässt sich in die Phasen *Erzeugung* und *Bearbeitung* unterteilen.

Nach dem Programmstart soll der Anwender durch einen leeren und aufgeräumten virtuellen Schreibtisch begrüßt werden. Auf diesem ist er in der Lage, mehrere Karteien gleichzeitig zu bearbeiten.

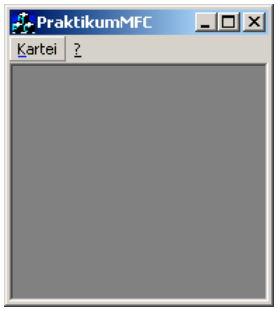

Abbildung 2: Leere Arbeitsoberfläche

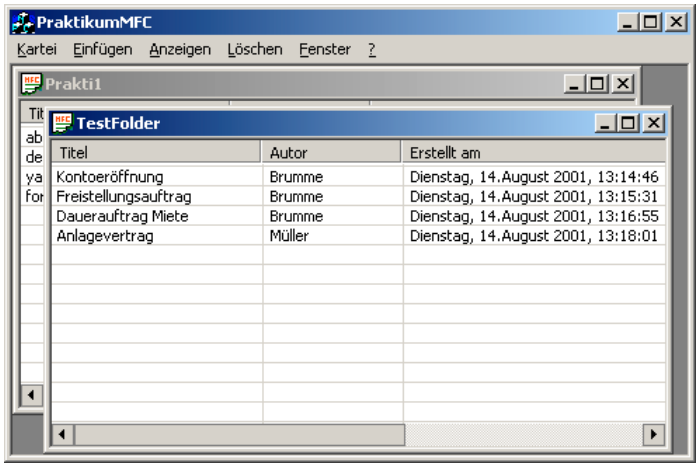

Abbildung 3: Parallele Bearbeitung mehrerer Karteien

Man kann eine Kartei auf die Arbeitsoberfläche legen, indem man sie neu erzeugt oder von einem Datenträger einliest. Letzteres kann man auf zwei Wegen lösen: Den ersten geht man, indem man im Menü *Kartei/Laden* auswählt und mit einem Standarddateidialog arbeitet. Oft ist es schneller, wenn man eine Kartei laden will, die man erst kürzlich bearbeitet hat. Sie steht dann direkt im Menü *Kartei*.

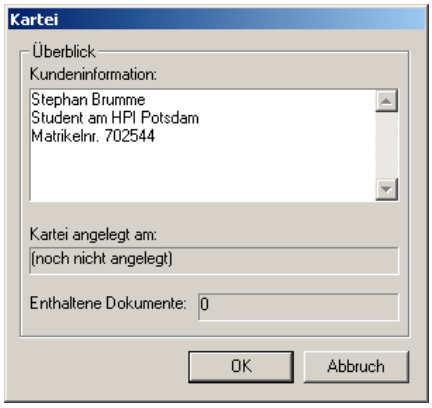

Abbildung 4: Anlegen einer neuen Kartei

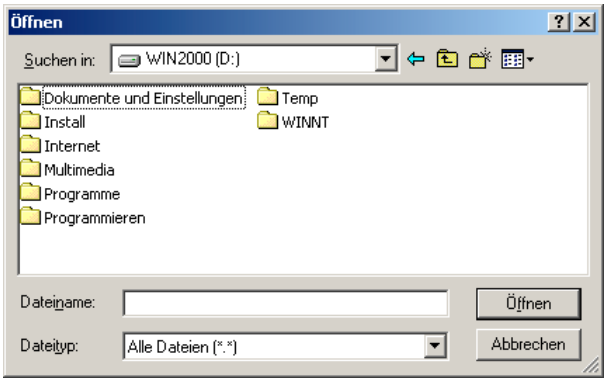

Abbildung 5: Einlesen einer Kartei vom Datenträger

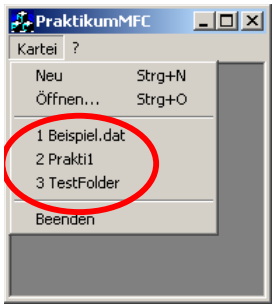

Abbildung 6: Laden einer neulich bearbeiteten Kartei

Nachdem mindestens eine Kartei auf der Arbeitsoberfläche liegt, ändert sich auch das Menü, über das nun Bankdokumente, Formulare, Verträge und Daueraufträge erzeugt und bearbeitet werden können.

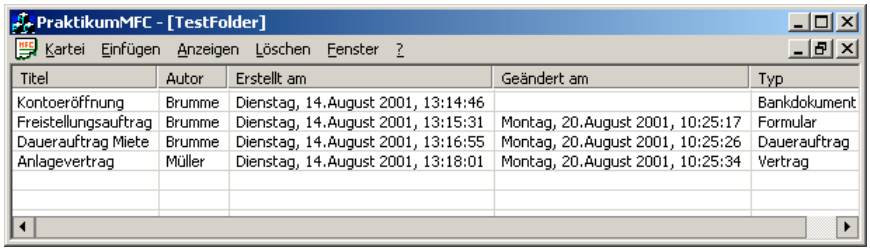

Abbildung 7: Komplettes Menü einer Kartei

Zusätzlich kann mit einem Klick der rechten Maustaste ein Kontextmenü geöffnet werden, das noch direkter einen Zugang zu den wichtigen Funktionen ermöglicht.

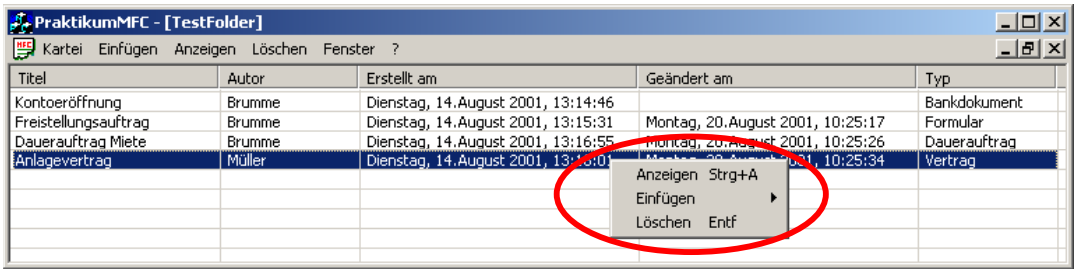

Abbildung 8: Kontextmenü

Bankdokument

Ein neues Bankdokument besteht aus einem Titel, einem Bearbeiternamen und einem Schlüssel. Für ein gültiges Bankdokument müssen alle drei vorhanden sein. Nicht direkt beeinflussbar ist das Erstellungsdatum, da es vom System automatisch verwaltet wird.

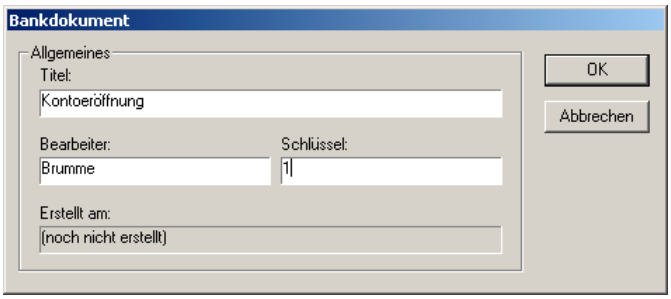

Abbildung 9: Anlegen eines neuen Bankdokumentes

Wenn bei der Eingabe Fehler unterlaufen, z.B. der Titel fehlt, so wird dies dem Benutzer mitgeteilt, damit er korrigierend eingreifen kann:

Abbildung 10: Hinweis bei unvollständiger Eingabe

Sobald ein Bankdokument angelegt wurde, kann es nicht mehr verändert werden:

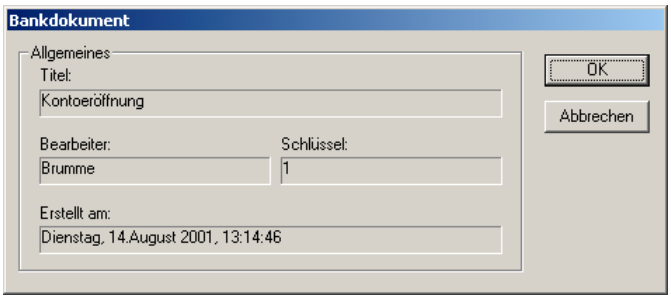

Abbildung 11: Anzeige eines Bankdokumentes

Formular

Ein Formular ist eine Erweiterung eines Bankdokumentes, daher gehe ich nur auf die Unterschiede ein.

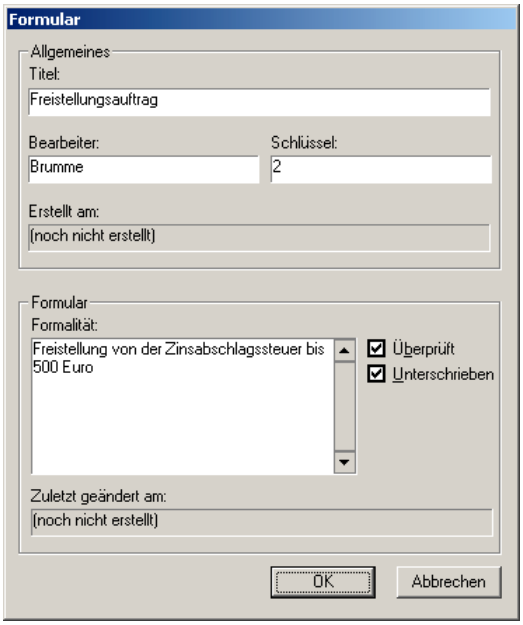

Abbildung 12: Anlegen eines neuen Formulars

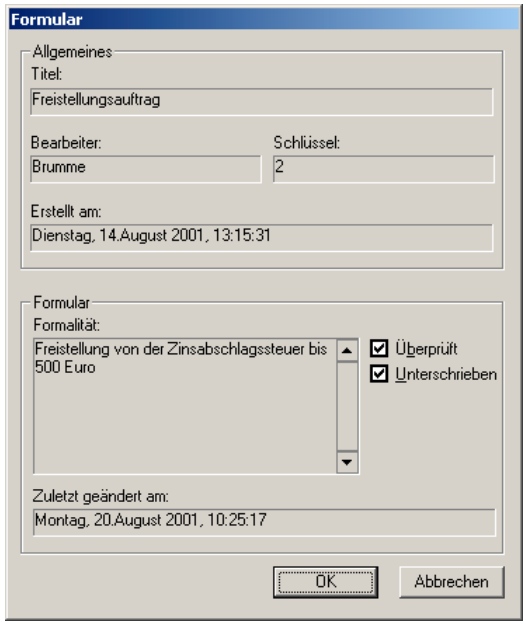

Abbildung 13: Bearbeitung eines Formulars

Neu hinzugekommen ist die Möglichkeit der Angabe einer Formalität und die Auswahl, ob das Formular unterschrieben und/oder geprüft wurde. Letzteres kann sogar jederzeit geändert werden, wobei das Datum mitprotokolliert wird.

Vertrag

Ein Vertrag ist – genau wie ein Formular – im Kern ein Bankdokument. Die Ähnlichkeit geht sogar noch weiter, es ist ebenso zur Laufzeit manipulierbar, ob der Vertrag überprüft und/oder unterschrieben wurde.

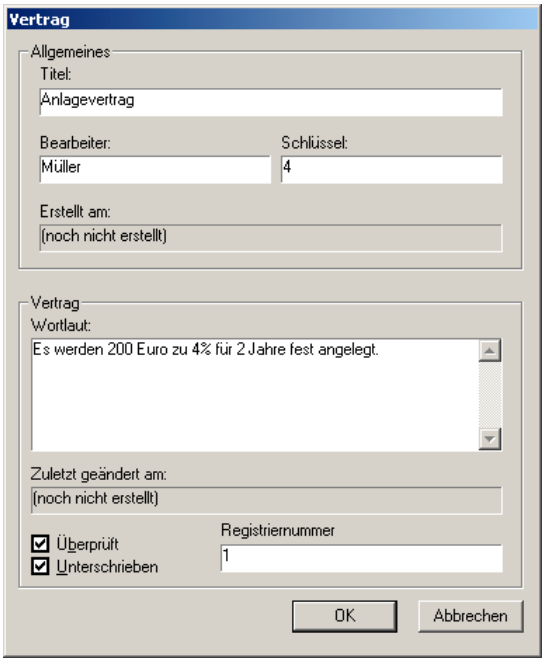

Abbildung 14: Neuer Vertrag

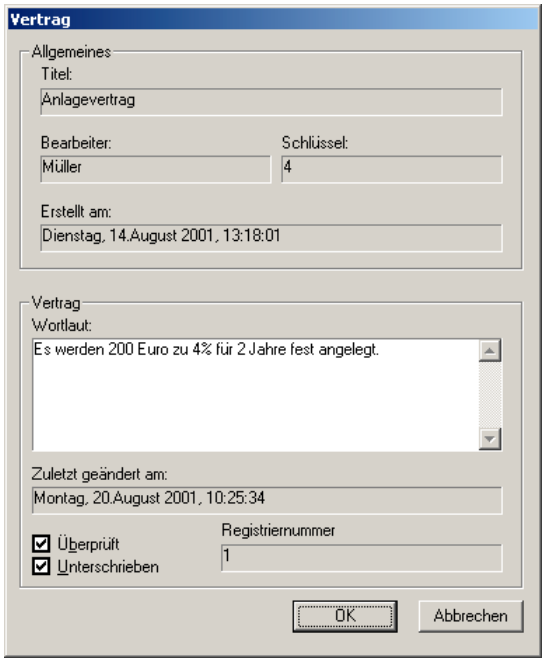

Abbildung 15: Vertrag bearbeiten

Anstatt der Formalität eines Formulars kann man nun den Wortlaut des Vertrages eingeben. Es ist jederzeit möglich, diesen Wortlaut auch nachträglich zu ändern.

Neu ist die Angabe einer Registriernummer. Sie muss – anders als der Schlüssel – eine positive Zahl sein.

Daueauftrag

Der Daueauftrag ist das komplexeste Element einer Dokumentenmappe. Er weist nicht nur alle Merkmale eines Formulars auf, sondern erlaubt zusätzlich die Eingabe von Einzahler, Empfänger, Betreff, Summe und Intervall. Die letzten beiden müssen positiven Ganzzahlen sein.

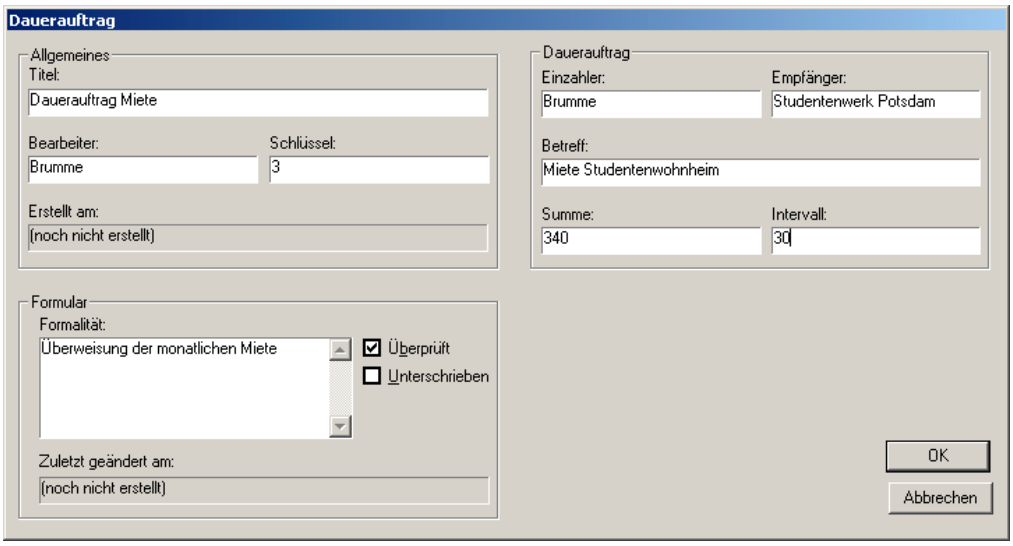

Abbildung 16: Daueauftrag anlegen

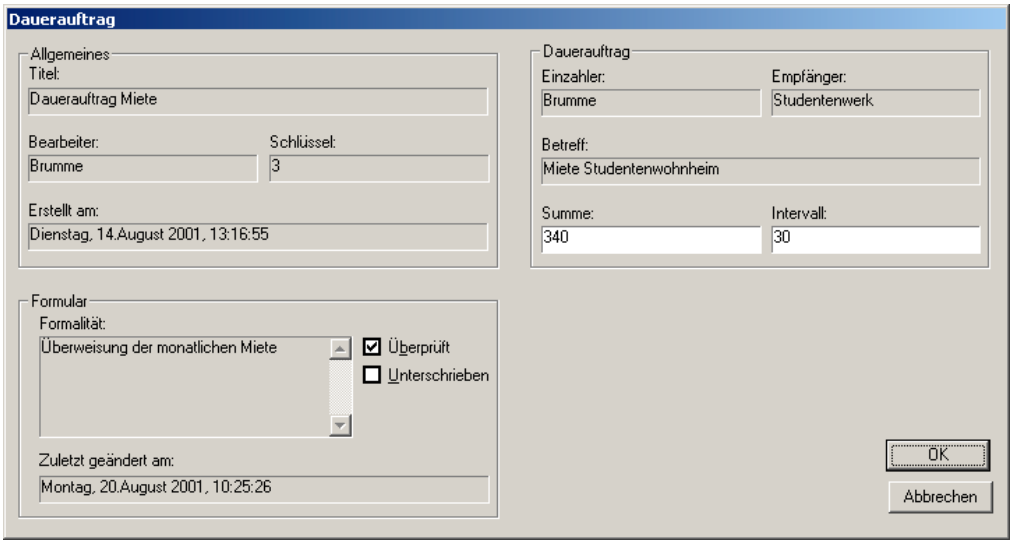

Abbildung 17: Daueauftrag bearbeiten

Anforderung an die Klassenbibliothek für das spezielle System

Als wichtigstes muss eine Klassenbibliothek über konsistente Methodiken verfügen. Dazu zählen immer wiederkehrende Modelle zur Behandlung ähnlicher Probleme (z.B. Nachrichtenprinzip in Fenster, Dialogen und Menüs), aber auch eine durchgängige und einprägsame Notation. Erst diese Faktoren ermöglichen dem Programmierer den Zugang zur Bibliothek.

Da ein MVC-Modell als Grundbedingung vorausgesetzt wurde, muss auch die Klassenbibliothek dieses Modell umsetzen. Aus verschiedensten Designentscheidungen wird dieses Modell jedoch oft nur näherungsweise oder auf Umwegen erreicht. Die einzelnen Unterschiede liste ich Punkt 4.2.2. auf.

Die Klassenbibliothek muss Schnittstellen zur Generierung einer graphischen Benutzeroberfläche (GUI) bereitstellen. Sie sollten möglichst einfach zu verstehen und zu verwenden sein, um die Konzentration des Auftragnehmers nicht vom Problemkern – der Dokumentenmappe – abzulenken.

Es wird dem Benutzer ermöglicht, dass er verschiedene Dokumente in die Kartei einfügen kann. Da der genaue Typ eines Dokumentes nicht vorher bekannt ist, muss die Kartei eine polymorphe Mengenorganisation sein. Eine Anforderung an die Klassenbibliothek besteht demzufolge darin, dass sie zumindest eine Mengenorganisation bereitstellt. Die polymorphen Eigenschaften können dann mit Mitteln der objektorientierten Programmiersprache ergänzt werden.

Zur Überprüfung der Klasseninvariante und der Ausgabe des Objektzustandes muss die Klassenbibliothek über entsprechende standardisierte Schnittstellen verfügen. Diese sollen in einer vernünftigen Umgebung nutzbar sein, z.B. mit Hilfe eines Debuggers. Idealerweise lassen sie sich dann später in einer getesteten und für fehlerfrei befundenen Version einfach und unkompliziert deaktivieren, um die erzeugte Codegröße zu verringern und die Programmgeschwindigkeit zu erhöhen.

Anforderung an die Klassenbibliothek für eine Persistenzhaltung

Das Hauptproblem bei der Persistenzhaltung ist die Polymorphie der Karteielemente. So muss es möglich sein, dass neben den Attributwerten auch zusätzliche Objektinformationen, wie seine Klassenzugehörigkeit, abgespeichert werden. Idealerweise geschieht dies nicht durch explizite Implementierung durch den Programmierer, sondern ist bereits in der Klassenbibliothek vorhanden.

Beim Einlesen der Daten ist es notwendig, dass die entsprechenden Objekte dynamisch erzeugt werden. Dazu müssen Schablonen vorhanden sein, die die schnelle Umwandlung einer „normalen“ Datenstruktur in eine serialisierbare erlauben.

Strukturierung der Architektur

Teilsysteme / Architektur

Um eine Strukturierung zu erzwingen, führe ich Namespaces ein, die *Model*, *View*, *Controller* oder *Application* heißen. Zusätzlich zu diesen zur Lösung der Problemstellung unbedingt notwendigen Teilsystemen existiert noch ein Namespace *Debug*, der die Aufgabe hat, Ablaufprotokolle und Testszenarien zu erstellen, um die korrekte Funktionalität zu gewährleisten bzw. im Fehlerfall hilfreiche Anhaltspunkte zu liefern.

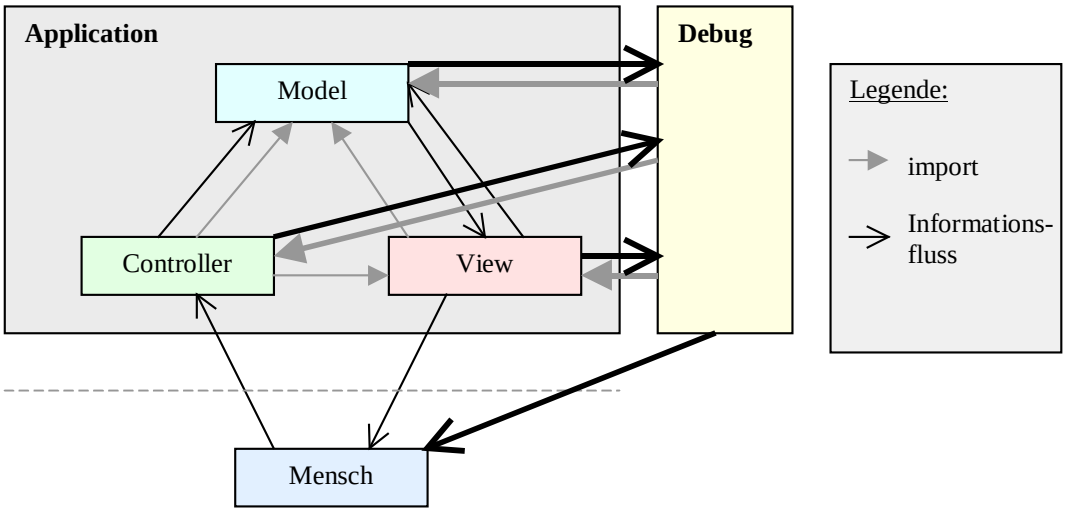

Abbildung 18: Spezielles MVC-Modell

Die im wesentlich durch *Debug* verursachten Änderungen gegenüber der Darstellung aus dem Kapitel 2.1. sind durch ihre Dicke erkennbar.

Mir fällt es schwer, die Grundstruktur der Problemlösung ohne Bezug auf die konkrete Umsetzung zu entwerfen. Der Grund liegt in konzeptionellen Zwängen, die die Klassenbibliothek MFC mir auferlegt und die dem MVC-Modell teilweise widersprechen. Die Reibungspunkte sind in den nachfolgenden Diagrammen grau unterlegt.

Weiterhin benutze ich schon die im Programm umgesetzten Klassennamen. Die Vererbungshierarchien (und damit die Einbindung ins MFC-System) sind Bestandteil des Kapitels 4.2.2.

Abbildung des speziellen MVC-Konzeptes auf logische Klassen

Model

Das *Model* beinhaltet alle Dokumente, die ich entsprechend der CEDL-Vorlage benenne: `CBankDocument`, `CForm`, `CContract`, `CStandingOrder` und `CFolder`. Sie werden im Programm durch `CPraktikumMFCDoc` gekapselt, welches allerdings auch einige Aufgaben des Views und des Controllers wahrnimmt und daher nicht der reinen MVC-Lehre entspricht (aber dem MFC-Konzept). Genauere Details finden sich in Kapitel 4.2.2.

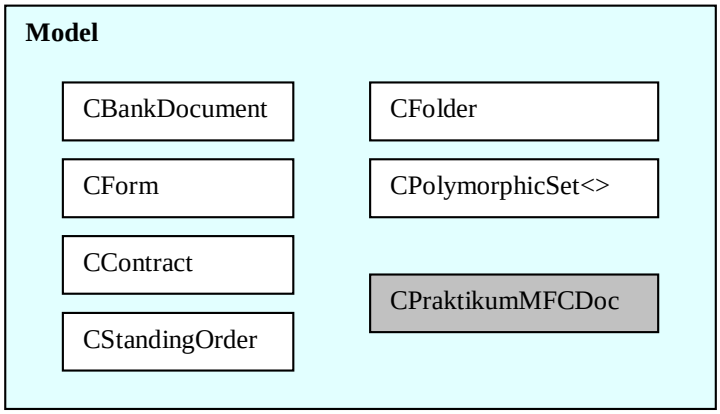

Abbildung 19: Spezielles Model

Wahrscheinlich ist aufgefallen, dass sich noch eine weitere Klasse (genauer: ein Template) namens `CPolymorphicSet<>` innerhalb des Models befindet. Ich entschließ mich dazu, weil ich die Kartei (`CFolder`) als polymorphe Mengenorganisation umsetzen wollte. Bei weiterer Überlegung kam ich zu dem Schluss, dass dies ein allgemein verwendbarer Container ist, ähnlich wie `CList`, `CArray` etc.. Er sollte daher möglich unabhängig von Einflüssen durch die Aufgabenstellung geformt sein, so dass er eher einen Teil einer benutzten (aber selbst entwickelten) Bibliothek darstellt. Somit ist CFolder selbst relativ klein ausgefallen, da der größte Teil der Arbeit von `CPolymorphicSet` bereits erledigt wird, wovon `CFolder` dann ableitet.

Die Klasse `CMyDate` führe ich nicht direkt im Model auf, da sie für mich eine Hilfsklasse mit einer nicht-existenziellen Bedeutung ist. Sie könnte genauso gut durch eine „echte“ MFC-Klasse, wie z.B. `CTime` ersetzt werden.

View

Die graphischen Repräsentationen der einzelnen Dokumente sind im Namespace `View` untergebracht. Da `CPolymorphicSet` nicht direkt verwendet wird, sondern nur als Basisklasse für `CFolder` dient, gibt es für `CPolymorphicSet` auch keine View-Entsprechung. `CAboutDialog` dient nur der Anzeige eines kleines Informationsdialoges und hat keinerlei tiefere programmtechnische Bedeutung.

Die Koordination aller graphischen Ausgaben ist Bestandteil von `CPraktikumMFCView`. Gleichzeitig übernimmt es den Großteil der Aufgaben eines Controllers. Da aber der View-Anteil überwiegt und ich die Klasse nicht auf zwei Namespaces aufteilen konnte, habe ich sie als View eingestuft.

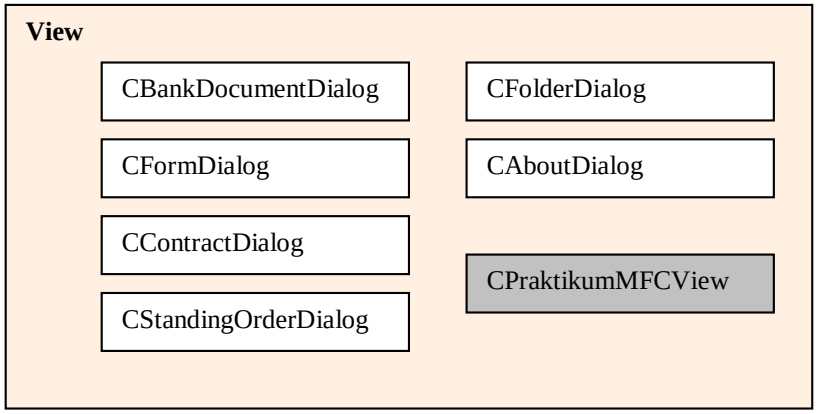

Abbildung 20: Spezielles View

Controller

Der Namespace Controller ist mehr oder weniger ein Kompromiss aus den Einschränkungen der MFC in Hinsicht auf das MVC-Modell. Normalerweise könnte man hier Anteile von `CPraktikumMFCDoc` und vor allem `CPraktikumMFCView` einbringen. Da diese aber schon in anderen Namespaces existieren, bleiben nur noch `CChildFrame` und `CMainFrame` übrig, die Aufgaben wie das Messaging-Routing und die Menüsteuerung übernehmen. Die Klassennamen sind direkt aus der MFC übernommen und können hier im grundlegenden Modell als eine Einheit angesehen werden.

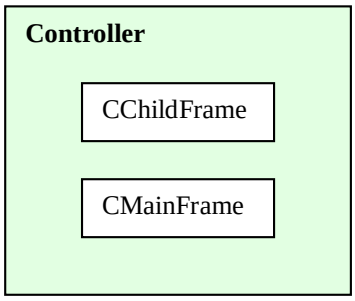

Abbildung 21: Spezieller Controller

Application

Der Application-Namespace besteht aus nur einer einzigen Klasse: `CPraktikumMFCApp`. Sie sorgt für die Instanziierung aller anderen Klassen des Modells und ist diejenige, die – als Singleton – zuerst angelegt und zuletzt zerstört wird. Neben diesen allgemeinen Verwaltungsaufgaben benutzt die Klasse noch Funktionalität aus dem Namespace *Debug*.

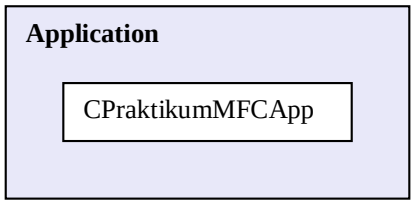

Abbildung 22: Spezielle Application

Debug

Dieser Namespace hat die Besonderheit, dass er nicht Bestandteil der Aufgabenstellung ist. Er dient nur dazu, wertvolle Architekturinformationen über das System zu sammeln und ist auch nur im Fehlerfall interessant. Die dazu notwendige Code wird nur mitkompiliert, wenn das Symbol `_DEBUG` definiert ist (sogenannte Debug-Version in Visual C++).

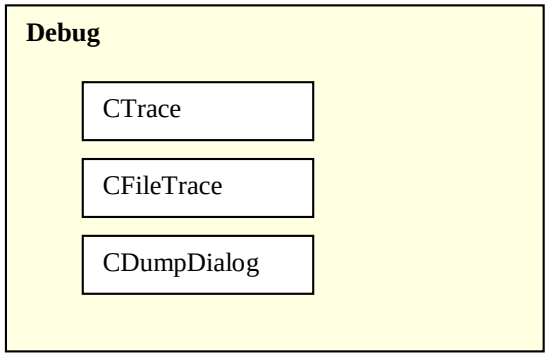

Abbildung 23: Neu eingeführter Namespace Debug

4. Realisierung des Modells

Namen und Bezeichner

Namenskonventionen

Ich habe versucht, sämtliche Klassen und Methoden entsprechend der CEDL-Vorlage umgesetzt, auf Ausnahmen gehe später ein. Bei der Benennung der Variablen ging ich einen Weg, der sich im Wesentlichen an den Microsoft Foundation Classes (MFC) orientiert, aber nicht allzu stark von der in der Vorlesung dargestellten Vorgehensweise abweicht. Die grundlegende Idee ist die, dass man bereits am Namen einer Variablen ihren Datentyp erkennen kann. Als Sprache kommt Englisch zum Zuge.

Jeder Klassenname beginnt mit `C`, jede selbstdefinierte Datenstruktur mit `T`.

Der Präfix `m_` deutet darauf hin, dass es sich um ein Attribut einer Klasse handelt. Diese bilden zusammen den Zustand eines Objektes und sind deshalb deutlich von lokalen und temporären Variablen zu unterscheiden. Weiterhin besitzt *jede* Variable einen Präfix, der eine Abkürzung eines Datentyps ist:

Datentyp	**Präfix**	**C++ Entsprechung**	**Beispiel**
natürliche Zahl	`n`	`unsigned int`	`nInterval`
Wahrheitswert	`b`	`bool`	`bSigned`
Zeiger	`p`	*Datentyp*`*`	`pClass`
Zeichenkette	`str`	`CString`	`strWording`
Datum/Zeit	`dt`	`CMyDate` (Eigenentwicklung)	`dtAlterationDate`
reelle Zahl	`f / d`	`Float / double`	`fTemperature`
Datei-Handle	`h`	*Verschieden*	`hLogFile`
Feld	`ar`	`std::vector<>`	`arPupils`

Abbildung 24: Variablenbenennung

Die Zeilen unterhalb der etwas dickeren Linie werden in den Haupt-Klassen der Praktikumsarbeit nicht gebraucht, sie dienen nur der allgemeinen Verdeutlichung des Notationsprinzips.

Die Methodennamen setzen sich i.d.R. aus zwei Teilen zusammen: einem Verb, das die auszuführende Aktion beschreibt und einem Substantiv, das das zu bearbeitende Attribut/Objekt bezeichnet. Die häufigsten Methoden folgen daher dem Muster:

Präfix-Verb	**Aktion**	**Beispiel**
`Get`	Auslesen eines Attributes	`GetDay`
`Set`	Schreiben eines Attributes	`SetYear`
`Is`	Zustandsabfrage	`IsSigned`

Abbildung 25: Methodenbenennung

Ich lege sehr großen Wert auf ausgeschriebene Namen, um so die Eindeutigkeit zu wahren und ein schnelleres Verständnis des Quelltextes zu ermöglichen. Diese Aussage bezieht sich sowohl auf Variablen- als auch auf Funktionsnamen.

Konstanten bestehen nur aus Großbuchstaben.

Namensänderungen

Um eine Konsistenz zur MFC zu wahren, entschloss ich mich zu einigen Namensänderungen.

- `ClassInvariant` wird durch `AssertValid` ersetzt. Der Rückgabewert entfällt, da `AssertValid` selbstständig den weiteren Programmablauf im Fehlerfalle dem Benutzer überlässt.

- `Show` wird durch `Dump` ersetzt. Gleichzeitig sinkt die Bedeutung drastisch, sie dient nur noch zur Ermittlung von Objektinformationen bei Auftreten eines Fehlers. Diese Änderung wird im Wesentlichen durch die neu eingeführten Dialogklassen notwendig.

- Die Variablennamen wurden – im Gegensatz zur CEDL-Beschreibung – vollständig ausgeschrieben, um die Lesbarkeit des Quellcodes zu erhöhen.

- Die Methoden `EqualValue` und `Copy` werden durch `operator==` bzw. `operator=` ersetzt. Auf den ersten Blick scheint es nur kosmetische Gründe zu geben, aber in Wahrheit spielten auch Überlegungen in Hinblick auf polymorphe Konsequenzen der Vergleichs- und Zuweisungsoperationen der verwendeten Mengenorganisation eine große Rolle.

- In der MFC existiert schon eine Klasse namens `CDate`. Meine eigene, neu entwickelte Klasse heißt daher `CMyDate`.

- Der CEDL-Datentyp `Ordinal` kennt keine exakte Entsprechung in C++. Ich benutze aus diesem Grunde den häufig verwendeten Datentyp `Integer`.

Umsetzung der Entwurfsentscheidungen

Auswahl der Realisierungsplattform

Ich implementiere die Lösung in C++, da ich darin die meiste Erfahrung besitze und ich der Sprache auch die größte Flexibilität zuspreche. Ich benutze keine compiler-spezifischen Eigenheiten, daher sollte jeder ANSI-C++-kompatible Compiler damit zurecht kommen, auch wenn die Entwicklungsplattform Visual C++ 6 von Microsoft war.

Das Programm ist keine komplette Neuentwicklung. Da bereits das Praktikum im Wintersemester 2000/2001 eine ähnliche Aufgabenstellung hatte, wollte ich Code wiederverwenden, um so bereits gewonnene Erkenntnisse nutzen zu können. Der größte Vorteil dieser Vorgehensweise liegt darin, dass mir viel Testaufwand erspart blieb und ich mich deshalb intensiver um Details und Erweiterungen kümmern konnte.

Aufgrund der Tatsache, dass über 90% aller Computer mit dem Betriebssystem Microsoft Windows (in allen Versionen) arbeiten, fiel meine Wahl der Klassenbibliothek auf die in diesem Umfeld am häufigsten verwendete: die Microsoft Foundation Classes (MFC). Erstens sind sie komplett in C++ geschrieben worden und zweitens sind sie (bei Besitz des Visual C++-Compilers) kostenfrei – auch für den kommerziellen Einsatz – erhältlich.

Da es bei jeder Software Unterschiede zwischen den einzelnen Versionen gibt, liste ich die verwendeten auf, die in der Regel dem aktuellen Stand der Technik entsprechen:

Software	**Version**
Visual Studio mit Visual C++	6.0 / Service Pack 5
Microsoft Foundation Classes	4.21
Betriebssystem I	Windows 98 / Erste Ausgabe
Betriebssystem II	Windows 2000 / Service Pack 1

Nicht unbedingt notwendig für die Realisierung, aber dennoch sehr hilfreich ist das Freeware-Quellcode-Dokumentationstool Doxygen 1.2.9 (www.doxygen.org).

Abbildung des speziellen MVC-Modells

Die Entscheidung, dass die MFC verwendet werden, ließ einige Änderungen am idealtypischen MVC-Konzept nötig werden, da zwischen den vorgefertigten Klassen oft nicht eine saubere Trennung gezogen werden kann.

Da die MFC sehr umfangreich ist und ich nicht die vollständigen Vererbungshierarchien angeben kann, beschränke ich mich auf den von modellierten Teil, so dass die Basisklasse meiner Diagramme selbst oft keine echte Basisklasse in der MFC ist.

Um dennoch einen Überblick zu bekommen, bilde ich hier die Klassenhierarchie der MFC (zitiert aus der MSDN) ab:

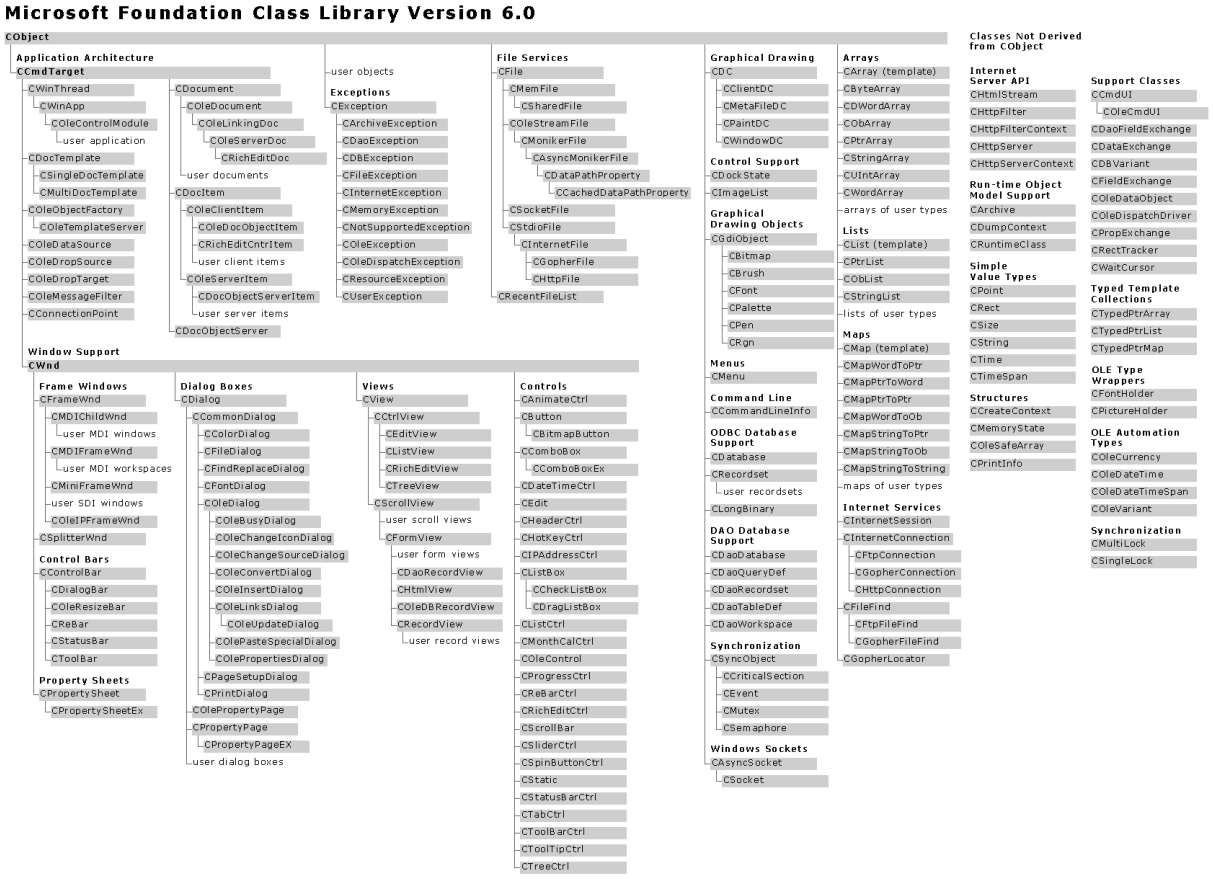

Abbildung 26: Vererbungshierarchie der MFC

Model

Das Model benötigt im Wesentlichen die Basisklasse `CObject`, um Architekturinformationen für die Persistenzhaltung (`GetRuntimeClass`, `Serialize`) und Fehleranalyse (`AssertValid`, `Dump`) bereitstellen zu können. Zwar sind die genannten Methoden dann MFC-spezifisch aufgebaut, der Rest der Klassen unterscheidet sich aber nicht wesentlich von einem allgemeinen MVC-Ansatz. Zum großen Teil konnte ich deshalb auf Wissen und Testergebnisse der letzten Praktikumsarbeit (Wintersemester 2000/2001) zurückgreifen.

Die UML-Vererbungshierarchie zeigt die Zusammenhänge zwischen den Klassen deutlicher:

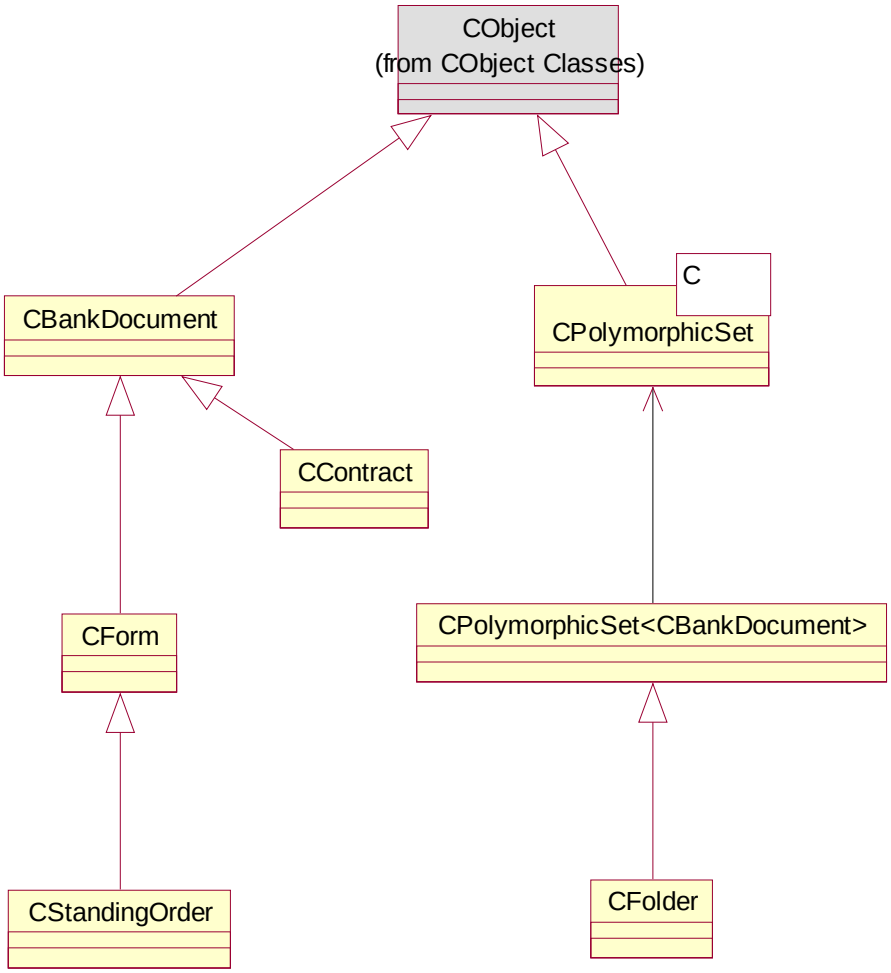

Abbildung 27: UML-Diagramm Model I

Das Dokument selbst muss gemäß MFC von `CDocument` ableiten:

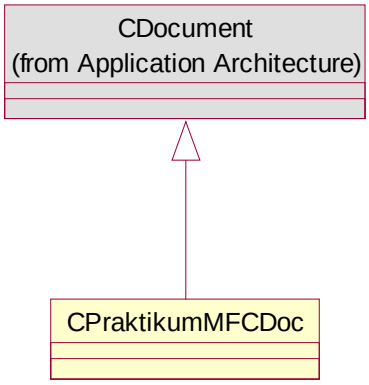

Abbildung 28: UML-Diagramm Model II

View

Das *View* ist komplett abhängig von der MFC-Architektur. Es setzt die Verwendung von Windows-Bildschirmelementen voraus und kann daher nur schwer auf andere Plattformen portiert werden. Die meisten Klassen sind Dialoge, deren Aussehen im Resourcen-Editor generiert wurde und deren Code nur zur Überprüfung der Invarianten dient. Das Haupt-*View*, das als alleinige Klasse alle anderen Bestandteile des *Views* benutzen darf, stammt von `CListView` ab, um dessen Darstellungsfähigkeiten einer Tabelle nutzen zu können.

Die Dialoge wurden mit dualer Funktionalität entworfen: sie können zur Konstruktion einer Klasse dienen (z.B. ein `CBankDocument`-Konstruktor erhält seine Daten von `CBankDocumentDialog`) oder nur zur Anzeige. Der wesentliche Unterschied besteht darin, dass ersterer Zustand die Editierung aller Attribute zulässt (einzige Ausnahme bilden die automatisch generierten Erstellungs- und Änderungszeitmarken), letzter nur die Bearbeitung weniger. Der Dialog erhält die Information, in welchem Zustand die zu bearbeitende Klasse ist, durch ein Flag bei der Konstruktion.

Trotz dieser scheinbar nahen Verwandtschaft von Dokument- und Dialogklasse wissen beide nichts von der Existenz des jeweils anderen. Diese Trennung macht Sinn, da man so die Unabhängigkeit beider wahrt. Dies äußert sich z.B. darin, dass die Dialoge *nicht* die Vererbungsstruktur der Dokumentklassen aufweisen, sondern generell direkt von `CDialog` abgeleitet werden. Jede andere Herangehensweise hätte die Nutzung der Wizards des Visual Studios erschwert und fehlerträchtige manuelle Implementierung verlangt.

In der UML-Notation ergeben sich zwei Hierarchien:

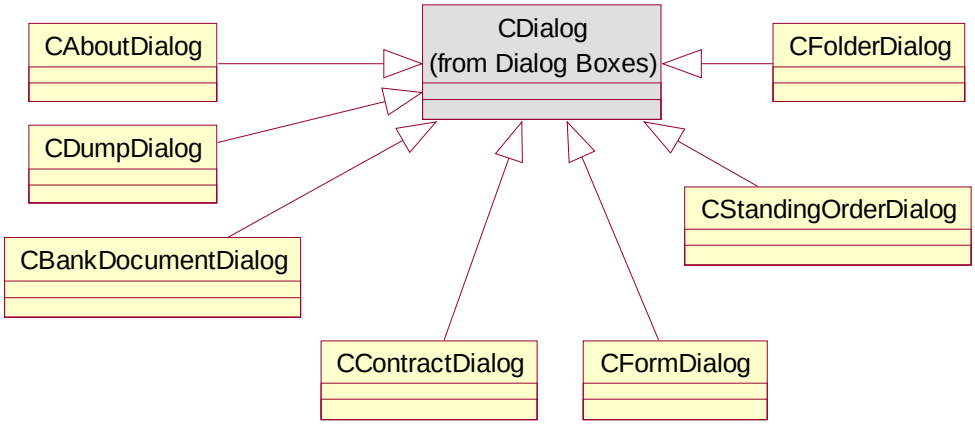

Abbildung 29: UML-Diagramm View I

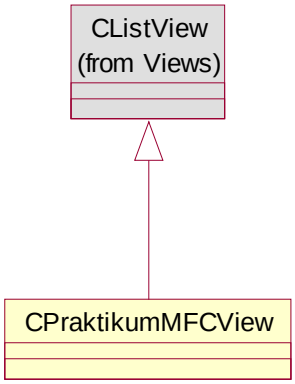

Abbildung 30: UML-Diagramm View II

Controller

Der *Controller* ist in seiner Funktionalität nur schwer innerhalb der MFC zu fassen. Viele der Aufgaben, die er inne hat, werden von Klassen des *Views* oder des *Models* übernommen. Zusätzlich kapselt die Klassenbibliothek viele Ereignisbehandlungsroutinen in ihrem Kern.

Übrig bleibt die Verwaltung der Menüs, Hotkeys und anderer Message-Dienste. Sie geschehen programmweit in `CMainFrame` oder bezogen auf das aktuelle Dokument in `CChildFrame`. Der Code ist zu weiten Teilen automatisch von den Wizards generiert worden.

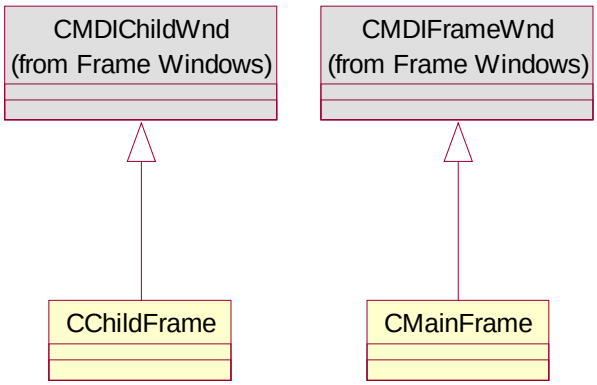

Abbildung 31: UML-Diagramm Controller

Application

Die *Application* wird – ebenso wie der *Controller* – von der MFC vorgegeben und automatisch generiert. Einzig die Integration des Debug-Teilsystem wurde von mir programmiert.

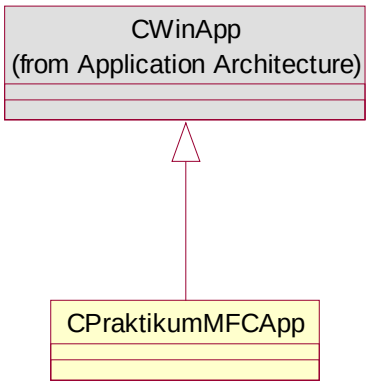

Abbildung 32: UML-Diagramm Application

Debug

Debug ist ein Sammelsurium von Klassen, die die Verwaltung von Architekturinformationen zur Laufzeit organisieren. Sie sorgen für deren Erzeugung (`CTrace`), Umleitung(`CFileTrace`) und Ausgabe (`CDumpDialog`). Zusätzlich sind Testszenarien integriert (`CScenario`).

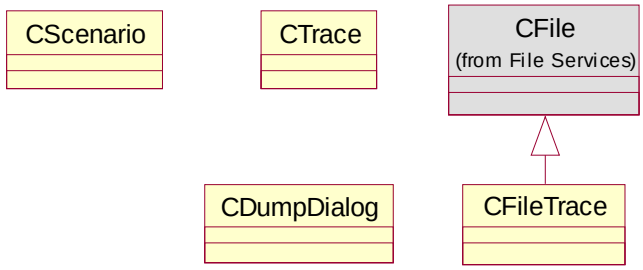

Abbildung 33: UML-Diagramm Debug

Die polymorphe Mengenorganisation

Kernstück des Modells ist die polymorphe Mengenorganisation. Sie wird in dem Template `CPolymorphicSet<>` realisiert.

Um alle polymorphen Effekte wirksam nutzen zu können, müssen die einzelnen Elemente als Pointer auf die Basisklasse gespeichert werden. Dieser Weg kann aber zu Problemen führen, wenn das referenzierte Objekt von außen gelöscht wird. Daher ist es unabdingbar, dass die Menge alle eingefügten Elemente als Kopie vorhält. Eine 1:1-Kopie einer von `CObject` abstammenden Klassen wird durch die Methode `Clone` erstellt, die die MFC-Architekturinformationen nutzt:

```
template<class C> C* CPolymorphicSet<C>::Clone(const C* element) const
{
    CTrace trace("Clone",this,element);

    // create new instance (care for correct class !)
    C* copy = static_cast<C*>(element->GetRuntimeClass()->CreateObject());
    // copy attributes
    *copy = *element;

    return copy;
}
```

Wird ein Element aus der Menge mit `GetFirst` bzw. `GetNext` ausgelesen, so wird erneut eine Kopie generiert. Um keine Speicherlecks zu erzeugen, müssen die Objekte nach Beendigung ihrer Benutzung gelöscht werden.

Realisierung der Persistenzhaltung

Das grundlegende Konzept der Abstraktion zwischen einer physikalischen Datei und der darin zu schreibenden bzw. daraus zu lesenden Daten wird durch ein Archiv repräsentiert. Dieses übernimmt die Aufgabe der korrekten Umwandlung von Datenstrukturen in einen Byte-Stream (und das in beiden Richtungen). Darum braucht man sich aber nicht weiter zu kümmern, da das Archiv bereits von der MFC in Form von `CArchiv` bereitgestellt wird.

Jede serialisierbare Klassen muss 4 Bedingungen erfüllen:

1. Die Klasse muss direkt oder indirekt von `CObject` abgeleitet sein.
2. In der Deklaration muss das Makro `DECLARE_SERIAL(Klassenname)` verwendet werden.
3. Die Implementierung muss das Makro `IMPLEMENT_SERIAL(Klassenname, KlassennameDerKlasseVonDerAbgeleitetWurde)` aufrufen.
4. Die Methode `virtual void Serialize(CArchive& ar)` ist zu überschreiben.

Beispielhaft für alle Dokumente folgt jetzt die Umsetzung für `CBankDocument`, die wesentlichen Zeilen sind durch ihre Schriftgröße deutlich hervorgehoben worden:

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  BankDocument.h
    \brief class CBankDocument (interface)
*/

#include "MyDate.h"
#pragma once

//! A basic bank document
/*! \li base class for all other bank-related classes

    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li each CString attribute must be non-empty
    \li m_dtDateOfFoundation must be valid
*/
class CBankDocument : public CObject
{
public:
    //! default constructor
    CBankDocument();
    //! copy constructor
    CBankDocument(const CBankDocument& bankdocument);
    //! explicitly set author, title and key
    CBankDocument(CString strAuthor, CString strTitle, CString strKey);

    //! destructor
    virtual ~CBankDocument();

    //! create a new object
    virtual CObject* Generate () const;
    //! clone this object
    virtual CObject* GenerateC() const;
    //! serialization
      virtual void Serialize(CArchive &ar);
    //! validate the folder
    virtual void AssertValid() const;
    //! dump
    virtual void Dump(CDumpContext& dc) const;

    //! copy
    virtual CBankDocument& operator = (const CBankDocument& bankdocument);
    //! compare value
    virtual bool           operator ==(const CBankDocument& bankdocument) const;
    //! compare keys
    bool EqualKey   (const CBankDocument& bankdocument) const;
    //! check for identity
```

```
    bool Equal(const CObject* object) const;

    //! deliver a unique identifier
    virtual CString KeyOf() const;

    //! get author
    CString GetAuthor() const;
    //! get title
    CString GetTitle()  const;
    //! get key
    CString GetKey()    const;
    //! get date of foundation
    CMyDate GetDateOfFoundation() const;

private:
    //! author
    CString m_strAuthor,
    //! title
            m_strTitle,
    //! key
            m_strKey;
    //! date of foundation
    CMyDate m_dtDateOfFoundation;


    DECLARE_SERIAL(CBankDocument)
};
```

In der Implementierung findet sich dann in den Zeilen 238 und 239:

```
// serializable
IMPLEMENT_SERIAL(CBankDocument, CObject, 1)
```

und in den Zeilen 90 bis 100:

```
/*!
    Loads/stores CBankDocument
    \param ar CArchive that provides data persistency
*/
void CBankDocument::Serialize(CArchive &ar)
{
    CObject::Serialize(ar);

    if (ar.IsStoring())
        ar << m_strAuthor << m_strTitle << m_strKey;
    else
        ar >> m_strAuthor >> m_strTitle >> m_strKey;

    // << and >> operators are not overloaded
    m_dtDateOfFoundation.Serialize(ar);
}
```

Die Klasse `CFolder` ist für Delegation der Serialisierung an ihre Elemente zuständig. Dazu bedient sie sich im Wesentlichen der Funktionalität von `CPolymorphicSet`. Die Implementation von `CPolymorphicSet::Serialize(CArchive &ar)` hat die Form:

```
/*!
    Loads / stores the set
    \param ar CArchive that provides access to the file system
*/
template<class C> void CPolymorphicSet<C>::Serialize(CArchive &ar)
{
    CTrace("Serialize",this);

    // do not forget to serialize CObject's data members
    CObject::Serialize(ar);

    // make use of CArchive's overloaded << and >> operators for *CObject

    if (ar.IsStoring())
    {
        // save polymorphic set
```

```
        // write cardinality
        ar << m_Set.GetCount();

        // write all elements
        POSITION pos     = m_Set.GetHeadPosition();
        int nCursorIndex = 0,
            nCount       = 0;
        while (pos != NULL)
        {
            // get cursor index
            if (m_Cursor == pos)
                nCursorIndex = nCount;
            nCount++;

            ar << m_Set.GetNext(pos);
        }

        // store cursor index
        ar << nCursorIndex;
    }
    else
    {
        // load polymorphic set

        // first, delete old set
        ScratchAll();

        int nCount;
        // load cardinality
        ar >> nCount;
        // load all elements
        for (int i=0; i<nCount; i++)
        {
            C* element;
            ar >> element;
            m_Set.AddTail(element);
        }

        // get cursor index
        int nCursorIndex;
        ar >> nCursorIndex;
        // create cursor out of the index
        m_Cursor = m_Set.FindIndex(nCursorIndex);
    }
}
```

Ich bediene mich der nützlichen Eigenart von `CArchive`, dass die Operatoren `<<` und `>>` für `CObject*` (also einen Zeiger auf `CObject`) bereits überladen sind. Der Schreibvorgang ist damit erstaunlich kurz:

```
        ar << m_Set.GetNext(pos);
```

Zu beachten ist, dass `GetNext` für mein Menge `CBankDocument*` liefert, was wiederum als Basisklasse `CObject*` hat. Auch das Einlesen kann ähnlich simpel gestaltet werden:

```
        C* element;
        ar >> element;
```

Ich halte diese Vorgehensweise für die beste, da die naive Lösung ein entscheidendes Problem hat:

```
        element.Serialize(ar);
```

umgeht die Speicherung des Schemaprüfungsmechanismus‘ der MFC. Zusätzlich fehlt der Informationsblock, so dass ein späteres Einlesen mit `operator>>` fehlschlägt, weil `GetObjectSchema` –1 liefert. Auch die umgekehrte Situation (Schreiben mit `operator<<` und Lesen mit `Serialize`) bereitet ähnliche Sorgen. Nur wenn das Programm im Voraus den Datentyp des nächstes Elementes im Archiv kennt, habe ich `Serialize` verwendet. Dies ist u.a. beim Datum der Kartei der Fall, das immer an einer festen Position in einer bekannten Anzahl gespeichert wird.

Eine kleine Schwierigkeit taucht beim Sichern des Cursors auf: Rein bit-technisch betrachtet stellt er einen Pointer dar. Mit nahezu 100%er Wahrscheinlichkeit ist die Speicherposition des aktuellen Element nach dem Laden eine andere als sie vor dem Speichern war. Aus diesem Grunde wird ein Index (nullbasiert) anstelle von POSITION bei der Serialisierung benutzt.

Qualitätssicherung

Auswahl der Teststrategie

Das Haupt-Testprinzip war der *Bottom-Up*-Test. Ich halte ihn für dieses Praktikum als besonders geeignet, da ein Teil der Klassen vom Praktikum des letzten Semesters genutzt werden konnte und bereits getestet ist. Wenn ich also eine Basisklasse weitgehend getestet habe und von ihrer Korrektheit überzeugt bin, so teste ich als nächstes alle von ihr abgeleiteten Klassen und arbeite mich auf diesem Wege zu den komplexen vor.

Der vorliegende Quellcode erlaubt mir, auf Pseudomoduln zu verzichten, weil die Klassen keinen besonders großen Umfang aufweisen und die Verschachtelungstiefe gering ist. Jede Emulation würde komplexer als das Original ausfallen.

Ich unterteile meinen Testplan ferner in zwei Kategorien: Einzeltests und Kombinationstests. Erstere sind als Funktionstests recht trivial, so dass ich sie manuell durchführte, sowohl Codeinspektion als auch Code-Walkthrough geschahen zum Zeitpunkt der initialen Programmierung. Eine Ausnahme davon bilden lediglich die Polymorphiebeziehungen, die ich eher in die Kategorie Kombinationstest einordne, da die Auswirkungen erst im Zusammenspiel mehrerer Klassen auftreten.

Eine Art von *Grenzwertanalyse* und *Äquivalenzklassenanalyse* stellen die `AssertValid`-Methoden der einzelnen Klassen dar. Die Testumgebung erlaubt aufgrund ihrer Flexibilität ein umfassendes *Error Guessing* bzw. das gezielte Testen von vermutlich kritischen Punkten. Somit kann ich auch keine starren Graphen oder ähnliches in dieser Dokumentation modellieren, da der Tester vollkommen frei in seiner konkreten Teststrategie ist. Ich gebe lediglich einen Rahmen vor. Sollte ein Fehler gefunden werden, so wird ein Dialog erzeugt:

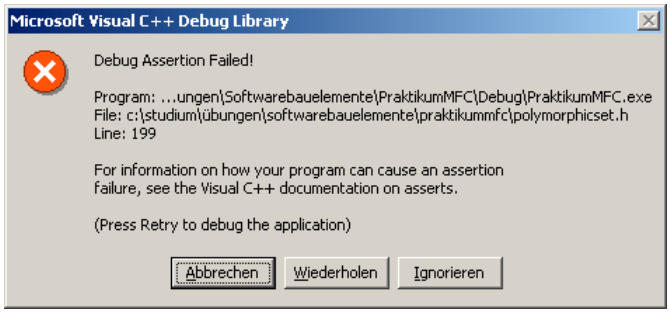

Abbildung 34: Fehlermeldung durch `AssertValid`

Der Programmierer hat dann die Wahl zwischen dem Beenden (*Abbrechen*), dem Debuggen (*Wiederholen*) oder Fortsetzen (*Ignorieren*) des Programms. Damit erhält man, was das Erzeugen von Szenarien angeht, Freiheiten und Möglichkeiten, denen keine Grenzen gesetzt sind.

In der *Debug*-Version wird zusätzlich ein Fenster geöffnet, das alle Ausgaben, die normalerweise im Ausgabebereich des Debuggers erscheinen, mitprotokolliert. Diese können abgespeichert werden, um so später als Nachweis korrekter (oder fehlerbehafteter) Funktionalität zu dienen:

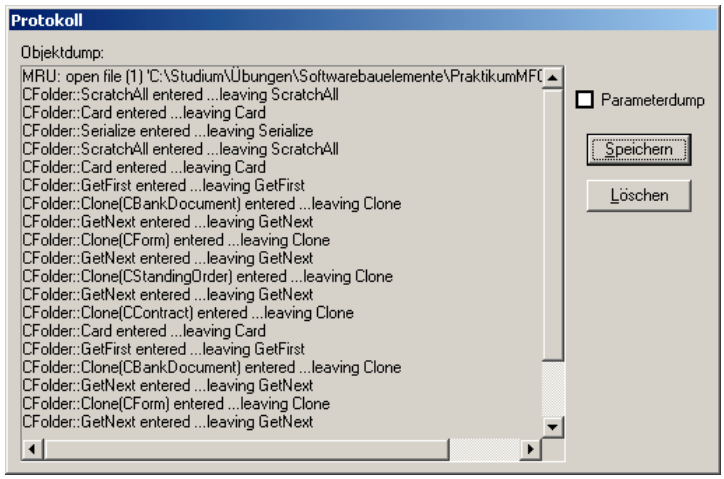

Abbildung 35: Debugfenster

Hinweis: Ich habe aufgrund ihres Umfanges darauf verzichtet, die Testprotokolle im Anhang abzudrucken. Sie befinden sich vollständig auf der beiliegenden CD und sind im ASCII-Textformat mit der Dateiendung .log abgespeichert worden.

Testszenarien und Testpläne (benutzergestützt)

Die Testszenarien dienen nur der Überprüfung des *Models*. Alle anderen Teilaspekte des Programms wurden entweder automatisch von Wizards generiert oder sind extrem kurz, was explizite Szenarien überflüssig macht.

Die implementierten Testszenarien sind *nur in der Debug-Version* aufrufbar. Sie finden sich dann im Menü *Anzeigen/Testszenarien*:

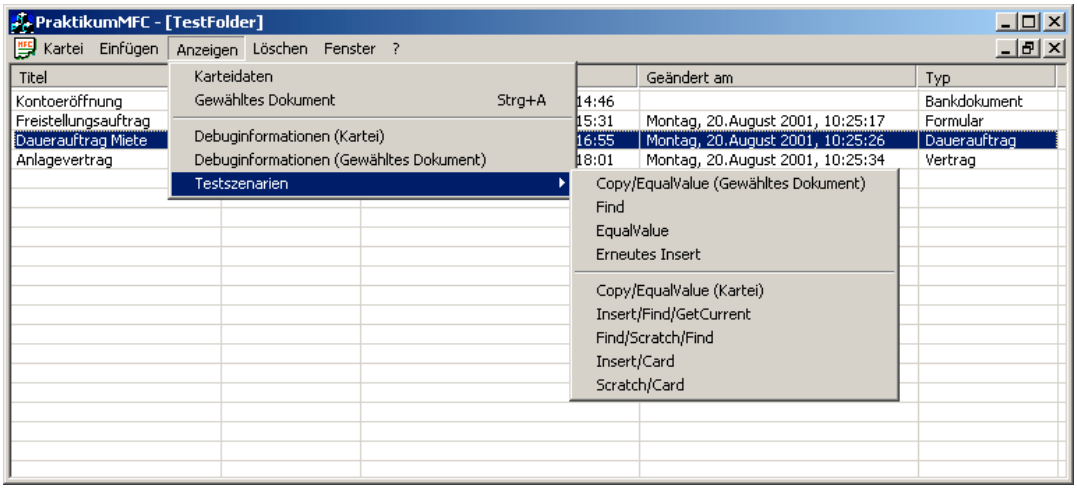

Abbildung 36: Aufruf eines Szenarios

Einige sind nur aufrufbar, wenn ein Dokument selektiert ist, es handelt sich dabei um *Copy/EqualValue*, *Find*, *EqualValue*, *Erneutes Insert*, *Find/Scratch/Find* und *Scratch/Card*.

Da beim Aufruf eines Szenarios ein Fenster erscheint, das dieses recht ausführlich erklärt, drucke ich in dieser Dokumentation einfach die Hinweisfenster ab. Ich weise nochmals darauf hin, dass einige Szenarien eine Fehlermeldung erzeugen müssen, wenn das Programm korrekt ist.

Copy/EqualValue (Dokument-basiert)

Dieses Szenario entspricht dem Fall 1 der Dokumentklassen (`CBankDocument`, `CForm`, `CStandingOrder` und `CContract`).

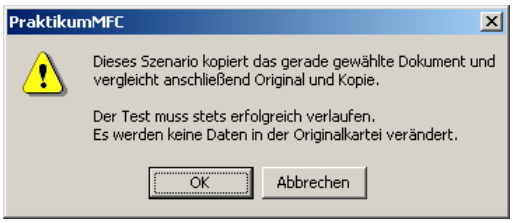

Abbildung 37: Szenario *Copy/EqualValue* (Dokument)

EqualValue

Hierbei handelt es sich um ein selbstentwickeltes Szenario, das eine Erweiterung des Dokumentenvergleiches überprüfen soll. Mir ging es darum, dass das Erzeugungsdatum keine wesentliche Eigenschaft eines Dokumentes sein darf.

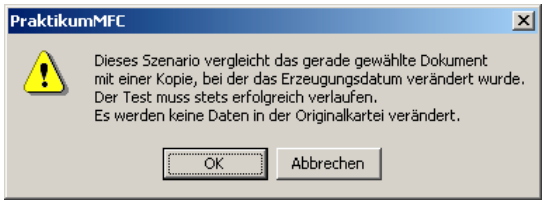

Abbildung 38: Szenario *EqualValue*

Find

Dieses Szenario dient dazu, ein bereits in der Kartei vorhandenes Dokument zu suchen. Der Suchvorgang ist u.a. auch während des Einfügens wichtig, da er die Einzigartigkeit jedes Dokumentes in der Kartei sicherstellt.

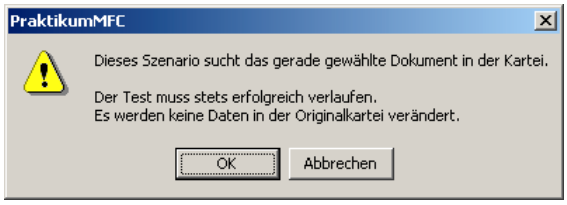

Abbildung 39: Szenario *Find*

Erneutes Insert

Wie bereits im Szenario Find angesprochen, darf kein Dokument mehrfach in der Kartei auftauchen. Dieses Szenario stellt daher lediglich eine Erweiterung des vorherigen dar.

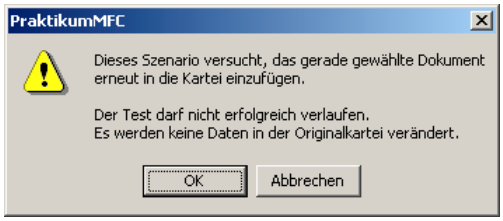

Abbildung 40: Szenario *Insert*

Copy/EqualValue (Kartei-basiert)

Hier wird der Fall 1 von `CFolder` umgesetzt, um die korrekte Überladung der Vergleichs- und Zuweisungsoperatoren sowie deren Polymorphieeigenschaften zu prüfen.

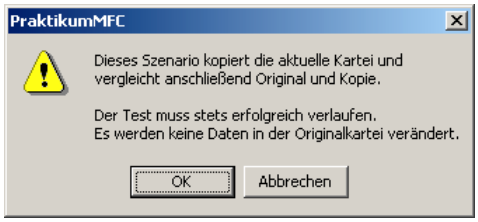

Abbildung 41: Szenario *Copy/EqualValue* (Kartei)

Insert/Find/GetCurrent

Dies entspricht dem Fall 2 von `CFolder`. Seine Aufgabe besteht in der Verifizierung des korrekten Einfügens eines Elementes nebst Änderung des Cursors.

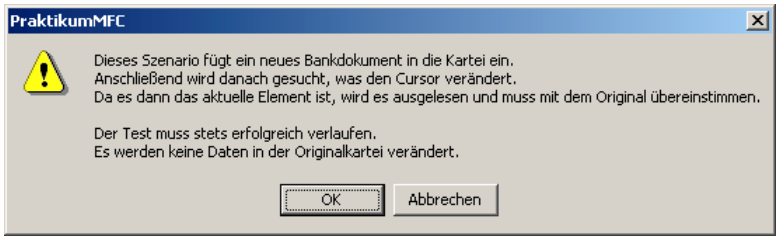

Abbildung 42: Szenario *Insert/Find/GetCurrent*

Find/Scratch/Find

Fall 3 von `CFolder` ist in etwa als das Gegenstück zu *Insert/Find/GetCurrent* zu sehen, da hier ein Element gelöscht wird. Danach darf es in der Menge nicht mehr vorhanden sein, was auch bedeutet, dass jedes Element kein Duplikat in der Menge besitzen darf (siehe *Erneutes Insert*).

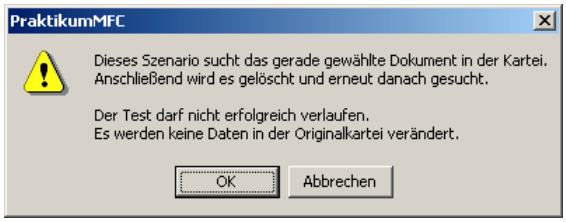

Abbildung 43: Szenario *Find/Scratch/Find*

Insert/Card

Fall 4 von `CFolder` steht in enger Beziehung zu *Insert/Find/GetCurrent*, dient aber eher dem Test der Funktion `Card`, die die Anzahl der Elemente in der Menge zurückliefert.

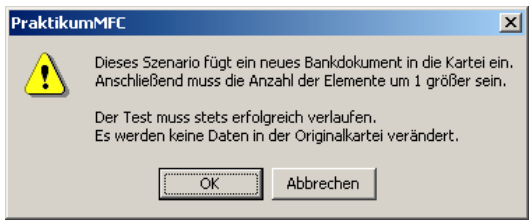

Abbildung 44: Szenario *Insert/Card*

Scratch/Card

Fall 5 von `CFolder` ist das Gegenstück zu *Insert/Card*.

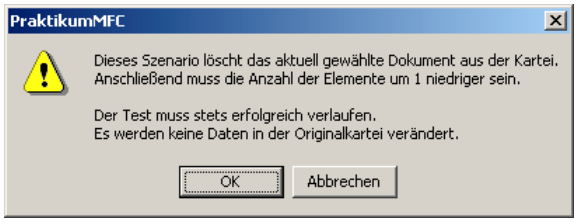

Abbildung 45: Szenario *Scratch/Card*

Codegestützte Ablaufverfolgung

Eine ganz andere Form des Testens ist im Namespace *Debug* realisiert worden. Er übernimmt die Aufgabe, zur Laufzeit ständig Informationen über die aufgerufenen Funktionen zu sammeln und in einem Fenster auszugeben.

Dazu sind 3 Komponenten notwendig:

- Sammeln der Informationen durch `CTrace`
- Abfangen der Standard-Debugausgabe in `CFileTrace`
- Anzeige in einem Fenster (`CDumpDialog`)

CTrace

`CTrace` ist technisch recht einfach aufgebaut: Ein Objekt dieser Klasse wird in den zu untersuchenden Methoden instanziiert. Dabei nutzt man die übergebenen Parameter, um den Namen der Methode, der Klasse, der sie zugehört und Informationen über den Parameter der Methode zu sammeln. Auf Wunsch wird ein Dump der Klassen erzeugt, allerdings müssen sie direkt oder indirekt von `CObject` abstammen.

```
template<class C> int CPolymorphicSet<C>::Insert(const C* element)
{
     CTrace trace("Insert",this,element);

    // don't insert an object twice
    if (Find(element))
        return -1;

    // add a copy to our set, change cursor
    m_Cursor = m_Set.AddTail(Clone(element));

    // return index in the set
    return m_Set.GetCount()-1;
}
```

Da dieses `CTrace`-Objekt auf dem Stack liegt, wird es automatisch beim Verlassen der Methode zerstört, so dass die Ausgabe abschließender Informationen möglich ist. So kann man gut verschachtelte Aufrufe erkennen, was der Ausschnitt aus einem Protokoll zeigt:

```
...
CFolder::Serialize entered ...CFolder::ScratchAll entered ...leaving
ScratchAll
leaving Serialize
CFolder::Card entered ...leaving Card
CFolder::GetFirst entered ...CFolder::Clone(CBankDocument) entered
...leaving Clone
leaving GetFirst
CFolder::GetNext entered ...CFolder::Clone(CForm) entered ...leaving Clone
leaving GetNext
CFolder::GetNext entered ...CFolder::Clone(CStandingOrder) entered
...leaving Clone
leaving GetNext
CFolder::GetNext entered ...CFolder::Clone(CContract) entered ...leaving
Clone
leaving GetNext
...
```

Man erkennt, das eine Menge geladen wird (`Serialize`), wozu erst einmal alle alten Elemente gelöscht werden (`ScratchAll`). Danach wird die Menge für die Anzeige ausgelesen, sie beinhaltet vier Elemente, die von den Typen `CBankDocument`, `CForm`, `CStandingOrder` und `CContract` sind. Intern rufen `GetFirst` bzw. `GetNext` stets die Methode `Clone` auf.

Als Ausgabefunktion wird das MFC-Makro `TRACE` verwendet, was auch den Namen der Klasse erklärt.

CFileTrace

Die Ausgaben von `TRACE` haben den entscheidenden Nachteil, dass sie nur im Debugger von Visual C++ zu sehen sind. Somit ist es unmöglich, auf einem Kundenrechner schnell mit einer speziell angepassten Programmversion (dem Debug-Build) die Ursache eines Problemes zu erkennen.

Genau hier setzt die Klasse `CFileTrace` an. Sie erlaubt das Abfangen aller `TRACE`-Ausgaben und leitet sie an benutzerdefinierte Funktionen weiter. Die Grundidee dafür stammt von Paul DiLascia und wurde in mehreren Ausgaben des Microsoft Journals (msdn.microsoft.com) besprochen. Ich erweiterte seinen Code dahingehend, dass ich nicht ein Fenster als Ziel der Umleitung definiere, sondern eine allgemeine Funktion zulasse, die bei mir `Print` genannt wird. Es erfolgen aber alle TRACE-Ausgaben auch weiterhin im Fenster des Visual Studio Debuggers.

Mit Hilfe statischer Methoden und einem versteckten Konstruktor sichere ich ab, dass zur Laufzeit höchstens eine Instanz von `CFileTrace` existiert, diese aber beim Programmstart automatisch erzeugt wird.

CDumpDialog

Nachdem nun alle TRACE-Ausgaben dem Programm zur Verfügung stehen, müssen sie auch dem Benutzer visuell zugänglich gemacht werden. Diese Aufgabe fällt `CDumpDialog` zu. Es ist in der Lage, die erzeugten Protokolle speichern zu können. Umgekehrt ist ebenso das Löschen des aktuellen Fensterinhaltes möglich. Als dritte Option erlaubt der Dialog die genaue Ausgabe eines Dumps der Parameter von `CTrace`-benutzenden Methoden. Diese sollte aber mit Vorsicht genossen werden, da die entstehenden Protokolle sehr schnell sehr lang werden.

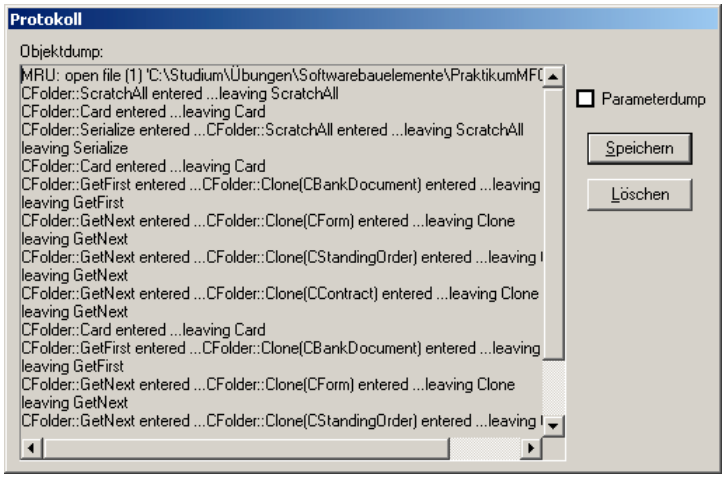

Abbildung 46: TRACE-Wiedergabe durch CDumpDialog

Bewertung der Testdurchführung

Die Testläufe waren enorm hilfreich bei der Entwicklung, da sie diverse Schwächen in der anfänglichen Implementierung der polymorphen Menge aufzeigten (insbesondere Speicherlecks). Ihre Behebung konnte derart erfolgreich durchgeführt werden, dass das Programm in der späten Phase keinerlei nicht freigegebenen Speicherblöcke oder sonstige Fehler bzw. gar Abstürze produzierte.

Einzeltests geschahen unter Verwendung der codegestützten Ablaufverfolgung, wobei die Parameter-Dump-Option aktiviert war. Kombinationstest dienten dazu, Ungereimtheiten in Bezug auf polymorphe Wechselbeziehungen in Vererbungshierarchien aufzuspüren.

Immer wieder konnte ich durch die dem Benutzer (oder in diesem Falle: dem Programmierer) gewährte Freiheit in Bezug auf die Reihenfolge und Häufigkeit der Tests ganz eigene und vielfältige Szenarien entwerfen und durchspielen, um so auch einen längeren Betrieb der Software zu simulieren. Genauso war ich in der Lage, Grenzfälle austesten, um die Bedienoberfläche sicherer zu machen. Ein Beispiel dafür ist Deaktivierung der Menüpunkte *Anzeigen* und *Löschen* wenn kein Dokument selektiert ist.

Trotz aller Vorsichtsmaßnahmen ist es möglich, dass der vorgestellte Code noch Fehler enthält, die selbst durch das aufwändige Absicherungssystem nicht auffielen. Allerdings können sie mit Hilfe der integrierten Debug-Mechanismen genau untersucht und analysiert werden. Im objektorientierten und gut dokumentierten Modell sollte es dann sehr einfach sein, den Fehler zu beheben.

5. Bewertung

Einschätzung des realisierten Modells

Die hier vorgestellte Lösung ist in der Lage, alle Anforderungen der Aufgabenstellung zu erfüllen. Da der Testplan keinerlei Fehler mehr offenbart und die Klassen zum Teil schon in mehreren Hausaufgaben erfolgreich eingesetzt wurden (insbesondere `CMyDate`, das meinem alten `CDate` entspricht) sehe ich das System als korrekt an.

Mit der Verwendung meiner Klassenstruktur ist man in der Lage, als Bank eine Kundenverwaltung mit den geforderten Fähigkeiten durchzuführen. Die Klassen wurden im vollen von der Aufgabenstellung geforderten Umfang umgesetzt, die Polymorphiebeziehungen habe ich ebenso erläutert.

Die Nutzung der MFC gestaltete sich effektiv und effizient, wenngleich auch Abstriche vom idealen MVC-Modells gemacht werden mussten. Sowohl die ansprechende Windows-Oberfläche als auch die interne Nutzung von Strukturen, Algorithmen und Konzepten sind erfolgreich mit Hilfe der MFC programmiert worden.

Der interaktive Testrahmen erlaubt eine bequeme Verifizierung, die auch problemlos von Nicht-Programmierern durchführbar ist. Eine umfassende Protokollierung rundet die Testumgebung ab.

Ausblick

Sehr wichtig ist das Ausdrucken einer Mappe bzw. von Dokumenten daraus. Diese Funktionalität kann mit verhältnismäßig geringem Aufwand integriert werden, da die MFC exzellente Vorarbeit leistet. In einer realen Bank gibt es noch wesentlich mehr Dokumente, die in einer Mappe abgelegt werden können. Darauf ist mein System durch die äußerst flexible Handlung der Polymorphiebeziehungen sehr gut vorbereitet.

Es kommt auch öfters vor, dass Dokumente von einer Mappe in eine andere übertragen werden müssen. Dazu könnte man die Kommunikationsfähigkeiten der Klassenbibliothek, z.B. die Zwischenablage, sehr gut nutzen. Mein Programm kann nur „mit sich selbst“ Daten austauschen d.h. serialisieren. Wünschenswert wäre ein offenes und standardisiertes Dateiformat. Mir schwebt XML als Idealkandidat vor.

Anhang

Anhang

Abbildungsverzeichnis

Literaturverzeichnis

[Balzert, 2001]	Helmut Balzert: *Lehrbuch der Software-Technik I und II*, 2.Auflage, Spektrum Akademischer Verlag, Heidelberg 2001, ISBN 3-8274-0480-0
[MSDN, 2001]	diverse Autoren, *Microsoft Developer Network*, http://msdn.microsoft.com
[Kruglinski, 1997]	David J. Kruglinski: *Inside Visual C++ 5*, 4.Auflage, Microsoft Press Deutschland, 1997, ISBN 3-86063-394-5
[Prosise, 1999]	Jeff Prosise: *Windows-Programmierung mit MFC*, 2.Auflage, Microsoft Press Deutschland, 1999, ISBN 3-86063-434-8
[Stroustrup, 1998]	Bjarne Stroustrup: *Die C++ Programmiersprache*, 3. Auflage, Addison-Wesley, Reading, Mass. 1998, ISBN 3-8273-1296-5
[Walnum, 1998]	Clayton Walnum: *Windows 98 Programming Secrets*, Franzis-Verlag, Poing, 1998, ISBN 3-7723-7493-X

Inhalt der beiliegenden CD

Die CD beinhaltet den gesamten Quellcode in das Programm in einer Debug- und einer Releaseversion. Zusätzlich zu der ausgedruckten Dokumentation ist auch eine Quelltext-Dokumentation vorhanden.

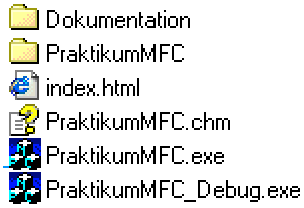

Abbildung 47: Hauptverzeichnis der CD

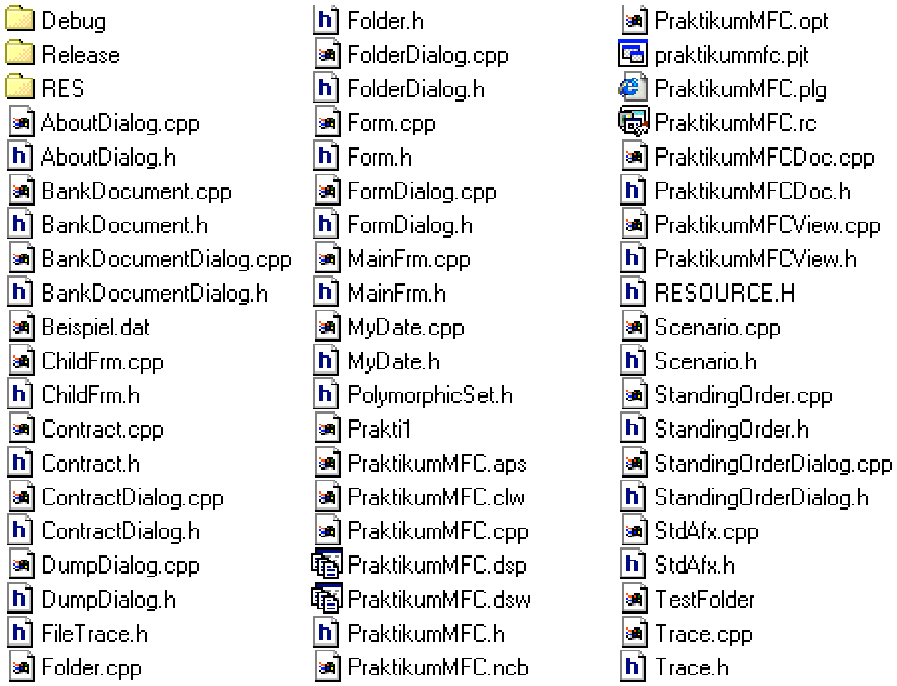

Abbildung 48: Quellcode im Verzeichnis PraktikumMFC

Quellcode

Der Quellcode wurde mit Syntax-Highlighting bearbeitet, um so eine bessere Lesbarkeit zu gewährleisten. Teilweise sind durch die verwendete Textverarbeitung unbeabsichtigt Zeilenumbrüche vorgenommen. Als endgültige Referenz dient deshalb stets der tatsächliche Quellcode auf der CD.

Ich habe das Analysetool Doxygen (www.doxygen.org) verwendet, was aus dieser Source eine sehr schöne Windows-Hilfedatei generiert hat. Sie befindet sich auf der CD.

Application

PraktikumMFC

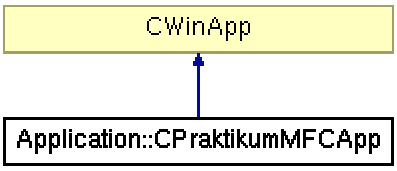

Abbildung 49: Vererbungshierachie `CPraktikumMFC`

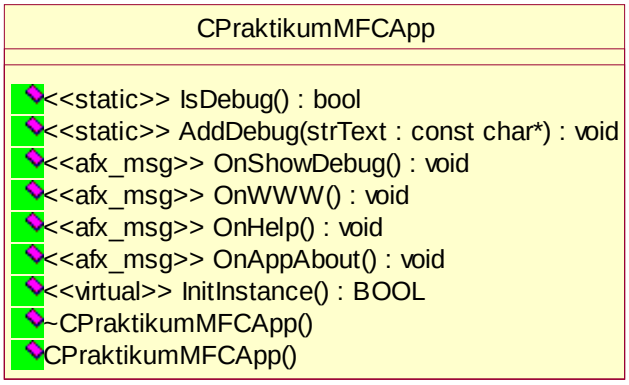

Abbildung 50: UML-Beschreibung von `CPraktikumMFC`

Interface: PraktikumMFC.h

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  PraktikumMFC.h
    \brief class CPraktikumMFCApp (interface)
*/

#pragma once

// compiler warning
#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

// load resource symbols
#include "resource.h"

#include "DumpDialog.h"

//! Application of the MVC concept
/*!
    Combines Model, View and Controller
*/
namespace Application
{
    using Debug::CDumpDialog;
```

```
//! The main class
/*! \li main class for the whole application
    \li to be used as a singleton

    \author         Stephan Brumme
    \date           August 11, 2001

    \invariant
    \li (none)
*/

//##ModelId=3B863B05010E
class CPraktikumMFCApp : public CWinApp
{
public:
    //! default constructor
    /*! Nothing left to do \see InitInstance */
    //##ModelId=3B863B050186
    CPraktikumMFCApp();

    //##ModelId=3B863B050185
    ~CPraktikumMFCApp();
    //{{AFX_VIRTUAL(CPraktikumMFCApp)
    public:
    //! initialize application (e.g. get settings)
    //##ModelId=3B863B050183
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

    //{{AFX_MSG(CPraktikumMFCApp)
    //! display some general info
    //##ModelId=3B863B050181
    afx_msg void OnAppAbout();
    //! start help system
    //##ModelId=3B863B05017F
    afx_msg void OnHelp();
    //! go to http://www.stephan-brumme.com/
    //##ModelId=3B863B05017D
    afx_msg void OnWWW();
    //! show debug dialog
    //##ModelId=3B863B050155
    afx_msg void OnShowDebug();
    //##ModelId=3B863B050152
    static void AddDebug(const char* strText);
    //! true, if debug dialog is visible
    //##ModelId=3B863B050150
    static bool IsDebug();
    //}}AFX_MSG

private:
    //##ModelId=3B863B05014D
    static CDumpDialog* m_pDumpDialog;

    DECLARE_MESSAGE_MAP()
};

} // namespace Application;
//{{AFX_INSERT_LOCATION}}
```

Implementierung (PraktikumMFC.cpp)

```
////////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  PraktikumMFC.cpp
    \brief class CPraktikumMFCApp (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"

#include "MainFrm.h"
#include "ChildFrm.h"
#include "PraktikumMFCDoc.h"
```

```
#include "PraktikumMFCView.h"
#include "AboutDialog.h"

#ifdef _DEBUG
#include "FileTrace.h"
#include "DumpDialog.h"
#endif

// use German translation for date conversion (month's name etc.)
#include <locale.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


namespace Application
{

//! the one and only application object
CPraktikumMFCApp theApp;

using Controller::CMainFrame;
using Controller::CChildFrame;
using Model::CPraktikumMFCDoc;
using View::CPraktikumMFCView;


CDumpDialog* CPraktikumMFCApp::m_pDumpDialog = NULL;


CPraktikumMFCApp::CPraktikumMFCApp()
{
#ifdef _DEBUG
    Debug::Print = AddDebug;
    m_pDumpDialog = new CDumpDialog;
#endif
}


CPraktikumMFCApp::~CPraktikumMFCApp()
{
    if (m_pDumpDialog != NULL)
    {
        m_pDumpDialog->DestroyWindow();
        delete m_pDumpDialog;
    }
}


/*!
    Initialize application
    \return true, if successful
*/
BOOL CPraktikumMFCApp::InitInstance()
{
    // use German settings
    setlocale(LC_ALL, "German");

#ifdef _AFXDLL
    // shared MFC-DLL
    Enable3dControls();
#else
    // statically linked MFC
    Enable3dControlsStatic();
#endif

    // registry key that stores the program's settings
    SetRegistryKey(_T("PraktikumMFC Stephan Brumme"));

    // load program's settings
    LoadStdProfileSettings();

    // register document templates
```

```
    CMultiDocTemplate* pDocTemplate;
    pDocTemplate = new CMultiDocTemplate(
        IDR_PRAKTITYPE,
        RUNTIME_CLASS(CPraktikumMFCDoc),
        RUNTIME_CLASS(CChildFrame),
        RUNTIME_CLASS(CPraktikumMFCView));
    AddDocTemplate(pDocTemplate);


    // build main MDI frame window
    CMainFrame* pMainFrame = new CMainFrame;
    if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
        return FALSE;
    m_pMainWnd = pMainFrame;

    // parse command line
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);

    // don't create a new file on start-up time
    if(cmdInfo.m_nShellCommand == CCommandLineInfo::FileNew)
        cmdInfo.m_nShellCommand = CCommandLineInfo::FileNothing;

    if (!ProcessShellCommand(cmdInfo))
        return FALSE;

    // show debug window
    if (m_pDumpDialog)
    {
        m_pDumpDialog->Create(IDD_DEBUG);
        m_pDumpDialog->ShowWindow(SW_SHOW);
    }

    // open new main window
    pMainFrame->ShowWindow(m_nCmdShow);
    pMainFrame->UpdateWindow();

    // done
    return TRUE;
}


/*!
    Display some general info
    \see CAboutDialog
*/
void CPraktikumMFCApp::OnAppAbout()
{
    View::CAboutDialog dlg;
    dlg.DoModal();
}


/*!
    Start CHM help system using Windows shell
*/
void CPraktikumMFCApp::OnHelp()
{
    ShellExecute(NULL, "open", "PraktikumMFC.chm", NULL, NULL, SW_SHOWNORMAL);
}


/*!
    Open web browser (http://www.stephan-brumme.com) using Windows shell
*/
void CPraktikumMFCApp::OnWWW()
{
    ShellExecute(NULL, "open", "http://www.stephan-brumme.com/", NULL, NULL, SW_SHOWNORMAL);
}


/*!
    Show object dump
*/
void CPraktikumMFCApp::OnShowDebug()
{
```

```
#ifdef _DEBUG
#else
    AfxMessageBox("Debuginformationen nicht verfügbar.", MB_ICONSTOP);
#endif
}


void CPraktikumMFCApp::AddDebug(const char* strText)
{
#ifdef _DEBUG
    if (IsDebug())
    {
        m_pDumpDialog->m_strDump += strText;
        m_pDumpDialog->m_strDump.Replace("\r\n", "\n");
        m_pDumpDialog->m_strDump.Replace("\n", "\r\n");
        m_pDumpDialog->UpdateData(FALSE);
    }
#endif
}


bool CPraktikumMFCApp::IsDebug()
{
    return (m_pDumpDialog != NULL);
}


BEGIN_MESSAGE_MAP(CPraktikumMFCApp, CWinApp)
    //{{AFX_MSG_MAP(CPraktikumMFCApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
    ON_COMMAND(ID_HELP, OnHelp)
    ON_COMMAND(ID_WWW, OnWWW)
    //}}AFX_MSG_MAP
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

} // namespace Application;
```

Symbole (resource.h)

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by PraktikumMFC.rc
//
#define IDD_ABOUTBOX                    100
#define IDR_MAINFRAME                   128
#define IDR_PRAKTITYPE                  129
#define IDD_FOLDER                      131
#define IDD_BANKDOCUMENT                132
#define IDD_FORM                        134
#define IDD_STANDINGORDER               135
#define IDD_CONTRACT                    136
#define IDR_CONTEXTMENU                 137
#define IDD_DEBUG                       138
#define IDC_EDIT_CUSTOMERINFORMATION    1001
#define IDC_EDIT_DATEOFFOUNDATION       1002
#define IDC_EDIT_CARDINALITY            1003
#define IDC_EDIT_TITLE                  1004
#define IDC_EDIT_AUTHOR                 1005
#define IDC_EDIT_KEY                    1006
#define IDC_CHECK_PROVED                1009
#define IDC_CHECK_SIGNED                1010
#define IDC_EDIT_FORMALITY              1011
#define IDC_EDIT_AMOUNT                 1013
#define IDC_EDIT_SUBJECT                1014
#define IDC_EDIT_INTERVAL               1016
#define IDC_EDIT_SOURCE                 1017
#define IDC_EDIT_RECIPIENT              1018
#define IDC_EDIT_ALTERNATIONDATE        1019
#define IDC_EDIT_WORDING                1020
#define IDC_EDIT_REGISTRATIONNUMBER     1021
#define IDC_EDIT_ALTERATIONDATE         1023
#define IDC_EDIT_DUMP                   1026
#define IDC_BUTTON_DELETE               1027
#define IDC_CHECK_OBJECTDUMP            1029
#define IDC_CHECK_PARAMETERDUMP         1030
#define ID_SHOW_CUSTOMERINFORMATION     32771
#define ID_INSERT_BANKDOCUMENT          32775
#define ID_INSERT_FORM                  32776
#define ID_INSERT_STANDINGORDER         32777
#define ID_INSERT_CONTRACT              32778
#define ID_SHOW_CURSOR                  32783
#define ID_DELETE_ALL                   32784
#define ID_DELETE_CURSOR                32785
#define ID_WWW                          32788
#define ID_SHOW_DEBUG                   32795
#define ID_WINDOW_DEBUG                 32797
#define ID_SCENARIO_COPYEQUALVALUE      32800
#define ID_SCENARIO_FIND                32802
#define ID_SCENARIO_INSERT              32803
#define ID_SCENARIO_COPYEQUALVALUE_FOLDER 32804
#define ID_SCENARIO_INSERTFINDGETCURRENT_FOLDER 32805
#define ID_SCENARIO_FINDSCRATCHFIND_FOLDER 32806
#define ID_SCENARIO_INSERTCARD_FOLDER   32807
#define ID_SCENARIO_SCRATCHCARD_FOLDER  32808
#define ID_SHOW_DEBUG_CURRENT           32809
#define ID_SCENARIO_EQUALVALUE          32811

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS                     1
#define _APS_NEXT_RESOURCE_VALUE        140
#define _APS_NEXT_COMMAND_VALUE         32812
#define _APS_NEXT_CONTROL_VALUE         1031
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Oberflächenelemente (PraktikumMFC.rc)

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

////////////////////////////////////////////////////////////////////////////
// Deutsch (Deutschland) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_DEU)
#ifdef _WIN32
LANGUAGE LANG_GERMAN, SUBLANG_GERMAN
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_SPLITTER_RESOURCES\r\n"
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_DEU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 7, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif //_WIN32\r\n"
    "#include ""res\\PraktikumMFC.rc2""  // Nicht mit Microsoft Visual C++ bearbeitete
Ressourcen\r\n"
    "#include ""l.deu\\afxres.rc""          // Standardkomponenten\r\n"
    "#endif\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME           ICON    DISCARDABLE     "res\\PraktikumMFC.ico"
IDR_PRAKTITYPE          ICON    DISCARDABLE     "res\\PraktikumMFCDoc.ico"

////////////////////////////////////////////////////////////////////////////
//
```

```
// Menu
//

IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&Kartei"
    BEGIN
        MENUITEM "&Neu\tStrg+N",                ID_FILE_NEW
        MENUITEM "Ö&ffnen...\tStrg+O",          ID_FILE_OPEN
        MENUITEM SEPARATOR
        MENUITEM "Letzte Datei",                ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "&Beenden",                    ID_APP_EXIT
    END
    POPUP "&?"
    BEGIN
        MENUITEM "Inf&o über PraktikumMFC...",  ID_APP_ABOUT
        MENUITEM SEPARATOR
        MENUITEM "&Hilfe",                      ID_HELP
        MENUITEM "www.stephan-brumme.com",      ID_WWW
    END
END

IDR_PRAKTITYPE MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&Kartei"
    BEGIN
        MENUITEM "&Neu\tStrg+N",                ID_FILE_NEW
        MENUITEM "Ö&ffnen...\tStrg+O",          ID_FILE_OPEN
        MENUITEM "&Schließen",                  ID_FILE_CLOSE
        MENUITEM "S&peichern\tStrg+S",          ID_FILE_SAVE
        MENUITEM "Speichern &unter...",         ID_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "Letzte Datei",                ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "&Beenden",                    ID_APP_EXIT
    END
    POPUP "&Einfügen"
    BEGIN
        MENUITEM "&Bankdokument\tStrg+B",       ID_INSERT_BANKDOCUMENT
        MENUITEM "&Formular\tStrg+F",           ID_INSERT_FORM
        MENUITEM "&Dauerauftrag\tStrg+D",       ID_INSERT_STANDINGORDER
        MENUITEM "&Vertrag\tStrg+V",            ID_INSERT_CONTRACT
    END
    POPUP "&Anzeigen"
    BEGIN
        MENUITEM "&Karteidaten",                ID_SHOW_CUSTOMERINFORMATION
        MENUITEM "&Gewähltes Dokument\tStrg+A", ID_SHOW_CURSOR
        MENUITEM SEPARATOR
        MENUITEM "&Debuginformationen (Kartei)", ID_SHOW_DEBUG
        MENUITEM "Debug&informationen (Gewähltes Dokument)",
                                                ID_SHOW_DEBUG_CURRENT
        POPUP "&Testszenarien"
        BEGIN
            MENUITEM "&Copy/EqualValue (Gewähltes Dokument)",
                                                    ID_SCENARIO_COPYEQUALVALUE

            MENUITEM "&Find",                       ID_SCENARIO_FIND
            MENUITEM "EqualValue",                  ID_SCENARIO_EQUALVALUE
            MENUITEM "&Erneutes Insert",            ID_SCENARIO_INSERT
            MENUITEM SEPARATOR
            MENUITEM "C&opy/&EqualValue (Kartei)",  ID_SCENARIO_COPYEQUALVALUE_FOLDER

            MENUITEM "I&nsert/Find/&GetCurrent",    ID_SCENARIO_INSERTFINDGETCURRENT_FOLDER

            MENUITEM "Find/&Scratch/Find",          ID_SCENARIO_FINDSCRATCHFIND_FOLDER

            MENUITEM "Insert/C&ard",                ID_SCENARIO_INSERTCARD_FOLDER

            MENUITEM "Scratch/Ca&rd",               ID_SCENARIO_SCRATCHCARD_FOLDER

        END
    END
    POPUP "&Löschen"
    BEGIN
        MENUITEM "&Gesamte Kartei",             ID_DELETE_ALL
```

```
        MENUITEM "Gewähltes Dokument\tEntf",    ID_DELETE_CURSOR
    END
    POPUP "&Fenster"
    BEGIN
        MENUITEM "Neues &Fenster",              ID_WINDOW_NEW
        MENUITEM SEPARATOR
        MENUITEM "Über&lappend",                ID_WINDOW_CASCADE
        MENUITEM "&Nebeneinander",              ID_WINDOW_TILE_HORZ
        MENUITEM "&Symbole anordnen",           ID_WINDOW_ARRANGE
        MENUITEM SEPARATOR
        MENUITEM "&Debugfenster",               ID_WINDOW_DEBUG
    END
    POPUP "&?"
    BEGIN
        MENUITEM "&Hilfe\tF1",                  ID_HELP
        MENUITEM SEPARATOR
        MENUITEM "Inf&o über PraktikumMFC...",  ID_APP_ABOUT
        MENUITEM "www.stephan-brumme.com",      ID_WWW
    END
END

IDR_CONTEXTMENU MENU DISCARDABLE
BEGIN
    POPUP "Kontextmenü"
    BEGIN
        MENUITEM "&Anzeigen\tStrg+A",           ID_SHOW_CURSOR
        POPUP "&Einfügen"
        BEGIN
            MENUITEM "&Bankdokument\tStrg+B",       ID_INSERT_BANKDOCUMENT
            MENUITEM "&Formular\tStrg+F",           ID_INSERT_FORM
            MENUITEM "&Dauerauftrag\tStrg+D",       ID_INSERT_STANDINGORDER
            MENUITEM "&Vertrag\tStrg+V",            ID_INSERT_CONTRACT
        END
        MENUITEM "&Löschen\tEntf",              ID_DELETE_CURSOR
    END
END


//////////////////////////////////////////////////////////////////////////////
//
// Accelerator
//

IDR_MAINFRAME ACCELERATORS PRELOAD MOVEABLE PURE
BEGIN
    "A",            ID_SHOW_CURSOR,         VIRTKEY, CONTROL, NOINVERT
    "B",            ID_INSERT_BANKDOCUMENT, VIRTKEY, CONTROL, NOINVERT
    "C",            ID_EDIT_COPY,           VIRTKEY, CONTROL, NOINVERT
    "D",            ID_INSERT_STANDINGORDER, VIRTKEY, CONTROL, NOINVERT
    "F",            ID_INSERT_FORM,         VIRTKEY, CONTROL, NOINVERT
    "N",            ID_FILE_NEW,            VIRTKEY, CONTROL, NOINVERT
    "O",            ID_FILE_OPEN,           VIRTKEY, CONTROL, NOINVERT
    "S",            ID_FILE_SAVE,           VIRTKEY, CONTROL, NOINVERT
    "V",            ID_EDIT_PASTE,          VIRTKEY, CONTROL, NOINVERT
    "V",            ID_INSERT_CONTRACT,     VIRTKEY, CONTROL, NOINVERT
    VK_BACK,        ID_EDIT_UNDO,           VIRTKEY, ALT, NOINVERT
    VK_DELETE,      ID_DELETE_CURSOR,       VIRTKEY, NOINVERT
    VK_DELETE,      ID_EDIT_CUT,            VIRTKEY, SHIFT, NOINVERT
    VK_F6,          ID_NEXT_PANE,           VIRTKEY, NOINVERT
    VK_F6,          ID_PREV_PANE,           VIRTKEY, SHIFT, NOINVERT
    VK_INSERT,      ID_EDIT_COPY,           VIRTKEY, CONTROL, NOINVERT
    VK_INSERT,      ID_EDIT_PASTE,          VIRTKEY, SHIFT, NOINVERT
    "X",            ID_EDIT_CUT,            VIRTKEY, CONTROL, NOINVERT
    "Z",            ID_EDIT_UNDO,           VIRTKEY, CONTROL, NOINVERT
END


//////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTBOX DIALOGEX 0, 0, 177, 143
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Info über Praktikum/MFC"
```

```
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,63,125,50,14,WS_GROUP
    LTEXT           "Stephan Brumme, Matrikelnr. 702544\nStudent am Hasso-Plattner-Institut
Potsdam",
                    IDC_STATIC,15,100,140,16
    CTEXT           "Simulation einer Bank\n\nPraktikumsbeleg für das Sommersemester 2001",
                    IDC_STATIC,5,5,165,25
    LTEXT           "Prof. Erika Horn\nLehrstuhl für Software Engineering\nUniversität
Potsdam",
                    IDC_STATIC,15,55,125,25
    GROUPBOX        "Auftraggeber",IDC_STATIC,5,45,165,40
    GROUPBOX        "Auftragnehmer",IDC_STATIC,5,90,165,30
END

IDD_FOLDER DIALOGEX 0, 0, 197, 161
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Kartei"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,85,140,50,14
    EDITTEXT        IDC_EDIT_CUSTOMERINFORMATION,10,25,175,50,ES_MULTILINE |
                    ES_READONLY | ES_WANTRETURN | NOT WS_BORDER | WS_VSCROLL,
                    WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_DATEOFFOUNDATION,10,95,175,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_CARDINALITY,90,115,95,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    GROUPBOX        "Überblick",IDC_STATIC,5,5,185,130
    LTEXT           "Kundeninformation:",IDC_STATIC,10,15,62,8
    LTEXT           "Kartei angelegt am:",IDC_STATIC,10,85,62,8
    LTEXT           "Enthaltene Dokumente:",IDC_STATIC,10,115,76,8
    PUSHBUTTON      "Abbruch",IDCANCEL,140,140,50,14
END

IDD_BANKDOCUMENT DIALOGEX 0, 0, 312, 113
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Bankdokument"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,255,10,50,14
    EDITTEXT        IDC_EDIT_TITLE,15,25,220,12,ES_AUTOHSCROLL | ES_READONLY |
                    NOT WS_BORDER,WS_EX_STATICEDGE
    LTEXT           "Titel:",IDC_STATIC,15,15,16,8
    LTEXT           "Bearbeiter:",IDC_STATIC,15,45,35,8
    EDITTEXT        IDC_EDIT_AUTHOR,15,55,110,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    LTEXT           "Schlüssel:",IDC_STATIC,130,45,33,8
    EDITTEXT        IDC_EDIT_KEY,130,55,105,12,ES_AUTOHSCROLL | ES_READONLY |
                    NOT WS_BORDER,WS_EX_STATICEDGE
    LTEXT           "Erstellt am:",IDC_STATIC,15,75,35,8
    EDITTEXT        IDC_EDIT_DATEOFFOUNDATION,15,85,220,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    GROUPBOX        "Allgemeines",IDC_STATIC,5,5,240,100
    PUSHBUTTON      "Abbrechen",IDCANCEL,255,30,50,14
END

IDD_FORM DIALOGEX 0, 0, 242, 257
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Formular"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,125,235,50,14
    PUSHBUTTON      "Abbrechen",IDCANCEL,185,235,50,14
    CONTROL         "Ü&berprüft",IDC_CHECK_PROVED,"Button",BS_AUTOCHECKBOX |
                    BS_FLAT | WS_TABSTOP,171,135,45,10
    CONTROL         "&Unterschrieben",IDC_CHECK_SIGNED,"Button",
                    BS_AUTOCHECKBOX | BS_FLAT | WS_TABSTOP,171,146,60,10
    EDITTEXT        IDC_EDIT_TITLE,10,25,220,12,ES_AUTOHSCROLL | ES_READONLY |
                    NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_AUTHOR,10,55,110,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_KEY,125,55,105,12,ES_AUTOHSCROLL | ES_READONLY |
```

```
                    NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_FORMALITY,11,135,155,60,ES_MULTILINE |
                    ES_READONLY | ES_WANTRETURN | NOT WS_BORDER | WS_VSCROLL,
                    WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_DATEOFFOUNDATION,10,85,220,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_ALTERATIONDATE,11,210,220,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    LTEXT           "Titel:",IDC_STATIC,10,15,16,8
    LTEXT           "Bearbeiter:",IDC_STATIC,10,45,35,8
    LTEXT           "Schlüssel:",IDC_STATIC,125,45,33,8
    LTEXT           "Erstellt am:",IDC_STATIC,10,75,35,8
    LTEXT           "Formalität:",IDC_STATIC,11,125,33,8
    LTEXT           "Zuletzt geändert am:",IDC_STATIC,11,200,66,8
    GROUPBOX        "Allgemeines",IDC_STATIC,5,5,230,100
    GROUPBOX        "Formular",IDC_STATIC,5,115,230,115
END

IDD_STANDINGORDER DIALOGEX 0, 0, 477, 223
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Dauerauftrag"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,420,180,50,14
    PUSHBUTTON      "Abbrechen",IDCANCEL,421,199,50,14
    CONTROL         "Ü&berprüft",IDC_CHECK_PROVED,"Button",BS_AUTOCHECKBOX |
                    BS_FLAT | WS_TABSTOP,170,135,45,10
    CONTROL         "&Unterschrieben",IDC_CHECK_SIGNED,"Button",
                    BS_AUTOCHECKBOX | BS_FLAT | WS_TABSTOP,170,147,64,10
    EDITTEXT        IDC_EDIT_AMOUNT,255,86,105,12,ES_AUTOHSCROLL | ES_NUMBER |
                    NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_INTERVAL,365,86,100,12,ES_AUTOHSCROLL |
                    ES_NUMBER | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_TITLE,10,25,220,12,ES_AUTOHSCROLL | ES_READONLY |
                    NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_AUTHOR,10,55,110,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_KEY,125,55,105,12,ES_AUTOHSCROLL | ES_READONLY |
                    NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_FORMALITY,15,135,150,45,ES_MULTILINE |
                    ES_READONLY | ES_WANTRETURN | NOT WS_BORDER | WS_VSCROLL,
                    WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_SOURCE,255,26,105,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_RECIPIENT,365,26,100,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_SUBJECT,255,56,210,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_DATEOFFOUNDATION,10,85,220,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_ALTERATIONDATE,15,196,215,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    LTEXT           "Titel:",IDC_STATIC,10,15,16,8
    LTEXT           "Bearbeiter:",IDC_STATIC,10,45,35,8
    LTEXT           "Schlüssel:",IDC_STATIC,125,45,33,8
    LTEXT           "Erstellt am:",IDC_STATIC,10,75,35,8
    LTEXT           "Formalität:",IDC_STATIC,15,125,33,8
    LTEXT           "Zuletzt geändert am:",IDC_STATIC,15,185,66,8
    LTEXT           "Summe:",IDC_STATIC,255,76,26,8
    LTEXT           "Betreff:",IDC_STATIC,255,46,24,8
    LTEXT           "Intervall:",IDC_STATIC,365,76,28,8
    LTEXT           "Einzahler:",IDC_STATIC,255,16,32,8
    LTEXT           "Empfänger:",IDC_STATIC,365,16,37,8
    GROUPBOX        "Allgemeines",IDC_STATIC,5,6,230,99
    GROUPBOX        "Dauerauftrag",IDC_STATIC,250,5,220,100
    GROUPBOX        "Formular",IDC_STATIC,5,115,230,100
END

IDD_CONTRACT DIALOGEX 0, 0, 252, 273
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Vertrag"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,135,250,50,14
```

```
    PUSHBUTTON      "Abbrechen",IDCANCEL,195,250,50,14
    EDITTEXT        IDC_EDIT_WORDING,11,135,224,50,ES_MULTILINE |
                    ES_WANTRETURN | NOT WS_BORDER | WS_VSCROLL,
                    WS_EX_STATICEDGE
    CONTROL         "Ü&berprüft",IDC_CHECK_PROVED,"Button",BS_AUTOCHECKBOX |
                    BS_FLAT | WS_TABSTOP,11,220,45,10
    CONTROL         "&Unterschrieben",IDC_CHECK_SIGNED,"Button",
                    BS_AUTOCHECKBOX | BS_FLAT | WS_TABSTOP,11,230,64,10
    EDITTEXT        IDC_EDIT_TITLE,15,25,220,12,ES_AUTOHSCROLL | ES_READONLY |
                    NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_AUTHOR,15,55,110,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_KEY,130,55,105,12,ES_AUTOHSCROLL | ES_READONLY |
                    NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_REGISTRATIONNUMBER,101,225,134,12,
                    ES_AUTOHSCROLL | ES_READONLY | ES_NUMBER | NOT WS_BORDER,
                    WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_DATEOFFOUNDATION,15,85,220,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    EDITTEXT        IDC_EDIT_ALTERATIONDATE,11,200,224,12,ES_AUTOHSCROLL |
                    ES_READONLY | NOT WS_BORDER,WS_EX_STATICEDGE
    LTEXT           "Titel:",IDC_STATIC,15,15,16,8
    LTEXT           "Bearbeiter:",IDC_STATIC,15,45,35,8
    LTEXT           "Schlüssel:",IDC_STATIC,130,45,33,8
    LTEXT           "Erstellt am:",IDC_STATIC,15,75,35,8
    LTEXT           "Zuletzt geändert am:",IDC_STATIC,11,190,66,8
    LTEXT           "Wortlaut:",IDC_STATIC,11,125,30,8
    LTEXT           "Registriernummer",IDC_STATIC,101,215,55,8
    GROUPBOX        "Allgemeines",IDC_STATIC,5,5,240,100
    GROUPBOX        "Vertrag",IDC_STATIC,5,115,240,130
END

IDD_DEBUG DIALOGEX 0, 0, 337, 193
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION
EXSTYLE WS_EX_TOOLWINDOW
CAPTION "Protokoll"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "&Speichern",IDOK,275,45,50,14
    PUSHBUTTON      "&Löschen",IDC_BUTTON_DELETE,275,65,50,14
    EDITTEXT        IDC_EDIT_DUMP,5,15,260,175,ES_MULTILINE | ES_OEMCONVERT |
                    ES_READONLY | NOT WS_BORDER | WS_VSCROLL | WS_HSCROLL,
                    WS_EX_STATICEDGE
    CONTROL         "Parameterdump",IDC_CHECK_PARAMETERDUMP,"Button",
                    BS_AUTOCHECKBOX | BS_FLAT | WS_TABSTOP,270,25,65,10
    LTEXT           "Objektdump:",IDC_STATIC,5,5,41,8
    CONTROL         "Objektdump",IDC_CHECK_OBJECTDUMP,"Button",
                    BS_AUTOCHECKBOX | BS_FLAT | NOT WS_VISIBLE | WS_TABSTOP,
                    270,15,54,10
END


#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040704b0"
        BEGIN
            VALUE "Comments", "http://www.stephan-brumme.com\0"
```

```
            VALUE "CompanyName", "Stephan Brumme\0"
            VALUE "FileDescription", "Praktikum/MFC für Softwarebauelemente II (SS 2001)\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "PraktikumMFC\0"
            VALUE "LegalCopyright", "Copyright (C) 2001 Stephan Brumme\0"
            VALUE "LegalTrademarks", "(none)\0"
            VALUE "OriginalFilename", "PraktikumMFC.EXE\0"
            VALUE "PrivateBuild", "\0"
            VALUE "ProductName", "Praktikum/MFC für Softwarebauelemente II (SS 2001)\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
            VALUE "SpecialBuild", "\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x407, 1200
    END
END

#endif    // !_MAC


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 170
        TOPMARGIN, 7
        BOTTOMMARGIN, 136
    END

    IDD_FOLDER, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 190
        TOPMARGIN, 7
        BOTTOMMARGIN, 154
    END

    IDD_BANKDOCUMENT, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 305
        TOPMARGIN, 7
        BOTTOMMARGIN, 106
    END

    IDD_FORM, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 235
        TOPMARGIN, 7
        BOTTOMMARGIN, 250
    END

    IDD_STANDINGORDER, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 470
        TOPMARGIN, 7
        BOTTOMMARGIN, 216
    END

    IDD_CONTRACT, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 245
        TOPMARGIN, 7
        BOTTOMMARGIN, 266
```

```
    END

    IDD_DEBUG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 330
        TOPMARGIN, 7
        BOTTOMMARGIN, 186
    END
END
#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE PRELOAD DISCARDABLE
BEGIN
    IDR_MAINFRAME           "PraktikumMFC"
    IDR_PRAKTITYPE          "\nPrakti\nPrakti\n\n\nPraktikumMFC.Document\nPrakti Document"
END

STRINGTABLE PRELOAD DISCARDABLE
BEGIN
    AFX_IDS_APP_TITLE       "PraktikumMFC"
    AFX_IDS_IDLEMESSAGE     "Bereit"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_INDICATOR_EXT        "ER"
    ID_INDICATOR_CAPS       "UF"
    ID_INDICATOR_NUM        "NUM"
    ID_INDICATOR_SCRL       "RF"
    ID_INDICATOR_OVR        "ÜB"
    ID_INDICATOR_REC        "MA"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_FILE_NEW             "Erstellt ein neues Dokument.\nNeu"
    ID_FILE_OPEN            "Öffnet ein bestehendes Dokument.\nÖffnen"
    ID_FILE_CLOSE           "Schließt das aktive Dokument.\nSchließen"
    ID_FILE_SAVE            "Speichert das aktive Dokument.\nSpeichern"
    ID_FILE_SAVE_AS         "Speichert das aktive Dokument unter neuem Namen.\nSpeichern
unter"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_APP_ABOUT            "Zeigt Programm-Informationen, Versionsnummer und Copyright
an.\nInfo"
    ID_APP_EXIT             "Verläßt die Anwendung; fragt, ob Dokumente gespeichert werden
sollen.\nBeenden"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_FILE_MRU_FILE1       "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE2       "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE3       "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE4       "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE5       "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE6       "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE7       "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE8       "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE9       "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE10      "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE11      "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE12      "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE13      "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE14      "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE15      "Öffnet dieses Dokument."
    ID_FILE_MRU_FILE16      "Öffnet dieses Dokument."
```

```
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_NEXT_PANE            "Wechselt zum nächsten Fensterausschnitt.\nNächster Ausschnitt"
    ID_PREV_PANE            "Springt zum vorherigen Fensterausschnitt zurück.\nVorheriger
Ausschnitt"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_WINDOW_NEW           "Öffnet ein weiteres Fenster für das aktive Dokument.\nNeues
Fenster"
    ID_WINDOW_ARRANGE       "Ordnet die Symbole im unteren Bereich des Fensters an.\nSymbole
anordnen"
    ID_WINDOW_CASCADE       "Ordnet die Fenster überlappend an.\nÜberlappende Fenster"
    ID_WINDOW_TILE_HORZ     "Ordnet die Fenster nebeneinander an.\nFenster nebeneinander"
    ID_WINDOW_TILE_VERT     "Ordnet die Fenster nebeneinander an.\nFenster nebeneinander"
    ID_WINDOW_SPLIT         "Teilt das aktive Fenster in Ausschnitte.\nTeilen"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_EDIT_CLEAR           "Löscht die Markierung.\nLöschen"
    ID_EDIT_CLEAR_ALL       "Löscht alles.\nAlles löschen"
    ID_EDIT_COPY            "Kopiert die Markierung und überträgt sie in die
Zwischenablage.\nKopieren"
    ID_EDIT_CUT             "Entfernt die Markierung und überträgt sie in die
Zwischenablage.\nAusschneiden"
    ID_EDIT_FIND            "Sucht den angegebenen Text.\nSuchen"
    ID_EDIT_PASTE           "Fügt den Inhalt der Zwischenablage ein.\nEinfügen"
    ID_EDIT_REPEAT          "Wiederholt die letzte Aktion.\nWiederholen"
    ID_EDIT_REPLACE         "Ersetzt einen bestimmten Text durch einen anderen.\nErsetzen"
    ID_EDIT_SELECT_ALL      "Markiert das gesamte Dokument.\nAlles markieren"
    ID_EDIT_UNDO            "Macht die letzte Aktion rückgängig.\nRückgängig"
    ID_EDIT_REDO            "Wiederholt die vorher rückgängig gemachte
Aktion.\nWiederherstellen"
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCSIZE          "Ändert die Fenstergröße."
    AFX_IDS_SCMOVE          "Ändert die Position des Fensters."
    AFX_IDS_SCMINIMIZE      "Verkleinert das Fenster zu einem Symbol."
    AFX_IDS_SCMAXIMIZE      "Vergrößert das Fenster zu voller Größe."
    AFX_IDS_SCNEXTWINDOW    "Wechselt zum nächsten Dokumentfenster."
    AFX_IDS_SCPREVWINDOW    "Wechselt zum vorherigen Dokumentfenster."
    AFX_IDS_SCCLOSE         "Schließt das aktive Fenster und fordert zur Dokumentspeicherung
auf."
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCRESTORE       "Stellt das Fenster in seiner normalen Größe wieder her."
    AFX_IDS_SCTASKLIST      "Aktiviert die Task-Liste."
    AFX_IDS_MDICHILD        "Aktiviert dieses Fenster."
END

#endif    // Deutsch (Deutschland) resources
//////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_SPLITTER_RESOURCES
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_DEU)
#ifdef _WIN32
LANGUAGE 7, 1
```

```
#pragma code_page(1252)
#endif //_WIN32
#include "res\PraktikumMFC.rc2"  // Nicht mit Microsoft Visual C++ bearbeitete Ressourcen
#include "l.deu\afxres.rc"          // Standardkomponenten
#endif

/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

Vorkompilierte Dateien der MFC

Interface (stdafx.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  StdAfx.h
    \brief MFC default includes
*/

#pragma once

// don‘t include rarely used header files
#define VC_EXTRALEAN
// MFC core
#include <afxwin.h>
// CListView
#include <afxcview.h>
// CFileDialog
#include <afxdlgs.h>


/*!
    \author         MFC wizard (automatically)
    \date           August 21, 2001
*/


//{{AFX_INSERT_LOCATION}}
```

Implementierung (stdafx.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  StdAfx.cpp
    \brief build pre-compiled headers out of MFC default includes
*/

#include "stdafx.h"
```

Model

PraktikumMFCDoc

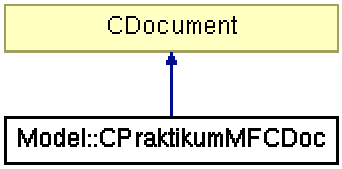

Abbildung 51: Vererbungshierarchie `CPraktikumMFCDoc`

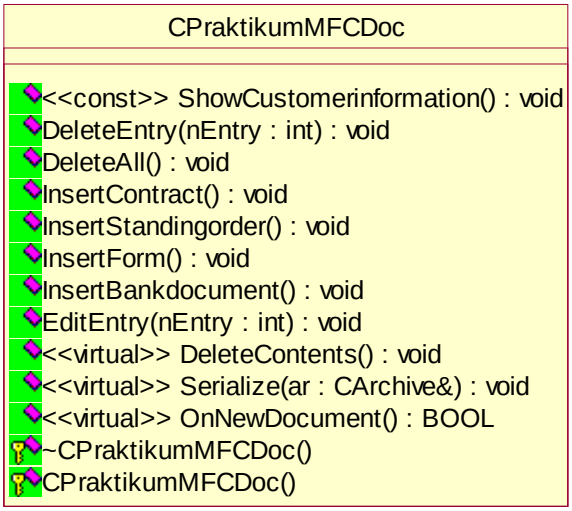

Abbildung 52: UML-Beschreibung von `CPraktikumMFCDoc`

Interface (PraktikumMfcDoc.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  PraktikumMFCDoc.h
    \brief class CPraktikumMFCDoc (interface)
*/

#pragma once
#include "Folder.h"
#include "DumpDialog.h"

namespace Model
{

//! The main document
/*! \li document of the application

    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li (none)
*/

//##ModelId=3B863AFF0190
class CPraktikumMFCDoc : public CDocument
{
protected:
    //! default constructor
    //##ModelId=3B863AFF0202
    CPraktikumMFCDoc();
    //! destructor
    //##ModelId=3B863AFF0201
```

```
    ~CPraktikumMFCDoc();

public:
    //! one and only data member
    //##ModelId=3B863AFF0194
    Model::CFolder m_Folder;

    //{{AFX_VIRTUAL(CPraktikumMFCDoc)
    public:
    //! create a new document
    //##ModelId=3B863AFF01FF
    virtual BOOL OnNewDocument();
    //! serialize the folder
    //##ModelId=3B863AFF01CF
    virtual void Serialize(CArchive& ar);
    //! delete the folder's content
    //##ModelId=3B863AFF01CD
    virtual void DeleteContents();
    //}}AFX_VIRTUAL

    //! edit a folder's element
    //##ModelId=3B863AFF01CB
    void EditEntry(int nEntry);
    //! insert a new bank document
    //##ModelId=3B863AFF01CA
    void InsertBankdocument();
    //! insert a new form
    //##ModelId=3B863AFF01C9
    void InsertForm();
    //! insert a new standing order
    //##ModelId=3B863AFF01C8
    void InsertStandingorder();
    //! insert a new contract
    //##ModelId=3B863AFF01C7
    void InsertContract();
    //! delete all elements
    //##ModelId=3B863AFF01C6
    void DeleteAll();
    //! delete a single element
    //##ModelId=3B863AFF01C4
    void DeleteEntry(int nEntry);
    //! show folder's properties
    //##ModelId=3B863AFF01C2
    void ShowCustomerinformation() const;

protected:
    //{{AFX_MSG(CPraktikumMFCDoc)
    //}}AFX_MSG
    DECLARE_DYNCREATE(CPraktikumMFCDoc)
    DECLARE_MESSAGE_MAP()
};

} // namespace Model

//{{AFX_INSERT_LOCATION}}
```

Implementierung (PraktikumMfcDoc.cpp)

```
////////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  PraktikumMFCDoc.cpp
    \brief class CPraktikumMFCDoc (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"
#include "PraktikumMFCDoc.h"

// used classes
#include "MyDate.h"
#include "Folder.h"
#include "FolderDialog.h"
#include "BankDocument.h"
```

```
#include "BankDocumentDialog.h"
#include "Form.h"
#include "FormDialog.h"
#include "StandingOrder.h"
#include "StandingOrderDialog.h"
#include "Contract.h"
#include "ContractDialog.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


namespace Model
{

/*!
    No debug dialog on startup
*/
CPraktikumMFCDoc::CPraktikumMFCDoc()
{
}


CPraktikumMFCDoc::~CPraktikumMFCDoc()
{
}


/*!
    Ask for folder's properties
    \return true, if successful
    \see CFolderDialog
    \invariant see CFolderDialog
*/
BOOL CPraktikumMFCDoc::OnNewDocument()
{
    // the usual stuff
    if (!CDocument::OnNewDocument())
        return FALSE;

    // ask for customer information
    View::CFolderDialog dlg(true);
    dlg.m_strDateOfFoundation = "(noch nicht angelegt)";
    if (dlg.DoModal() == IDOK)
    {
        // OK pressed, so create a new one
        CFolder myFolder(dlg.m_strCustomerInformation);
        m_Folder = myFolder;
        return TRUE;
    }
    else
        // cancel
        return FALSE;
}


/*!
    Load/store the document
    \param ar CArchive to be used
    \see CFolder::Serialize
*/
void CPraktikumMFCDoc::Serialize(CArchive& ar)
{
    m_Folder.Serialize(ar);

    UpdateAllViews(NULL);
}


/*!
    Reset folder's set
*/
```

```
void CPraktikumMFCDoc::DeleteContents()
{
    // delete all elements in the set
    m_Folder.ScratchAll();

    CDocument::DeleteContents();
    UpdateAllViews(NULL);
}


/*!
    Show folder's properties
    \see CFolderDialog
    \invariant see CFolderDialog
*/
void CPraktikumMFCDoc::ShowCustomerinformation() const
{
    View::CFolderDialog dlg;
    dlg.m_nCardinality = m_Folder.Card();
    dlg.m_strDateOfFoundation = m_Folder.GetDateOfFoundation().Format("%A, %d.%B, %Y,
%H:%M:%S");
    dlg.m_strCustomerInformation = m_Folder.GetCustomerInformation();

    dlg.DoModal();
}


/*!
    Insert a new bank document
    \see CBankDocument, CBankDocumentDialog
    \invariant see CBankDocumentDialog
*/
void CPraktikumMFCDoc::InsertBankdocument()
{
    // dialog is used for construction
    View::CBankDocumentDialog dlg(true);

    dlg.m_strDateOfFoundation = "(noch nicht erstellt)";

    if (dlg.DoModal() == IDOK)
    {
        CBankDocument bankdocument(dlg.m_strAuthor, dlg.m_strTitle, dlg.m_strKey);
        m_Folder.Insert(&bankdocument);
    }

    UpdateAllViews(NULL);
}


/*!
    Insert a new form
    \see CForm, CFormDialog
    \invariant see CFormDialog
*/
void CPraktikumMFCDoc::InsertForm()
{
    // dialog is used for construction
    View::CFormDialog dlg(true);

    dlg.m_strDateOfFoundation = "(noch nicht erstellt)";
    dlg.m_strAlterationDate   = "(noch nicht erstellt)";
    dlg.m_bProved = dlg.m_bSigned = FALSE;

    if (dlg.DoModal() == IDOK)
    {
        // convert BOOL to bool
        bool bProved = (dlg.m_bProved != 0);
        bool bSigned = (dlg.m_bSigned != 0);

        CForm form(dlg.m_strAuthor, dlg.m_strTitle, dlg.m_strKey,
                   bProved, bSigned, dlg.m_strFormality);
        m_Folder.Insert(&form);
    }

    UpdateAllViews(NULL);
}
```

```
/*!
    Insert a new standing order
    \see CStandingOrder, CStandingOrderDialog
    \invariant see CStandingOrderDialog
*/
void CPraktikumMFCDoc::InsertStandingorder()
{
    // dialog is used for construction
    View::CStandingOrderDialog dlg(true);

    dlg.m_strDateOfFoundation = "(noch nicht erstellt)";
    dlg.m_strAlterationDate   = "(noch nicht erstellt)";
    dlg.m_bProved = dlg.m_bSigned = FALSE;
    dlg.m_nAmount = dlg.m_nInterval = 0;

    if (dlg.DoModal() == IDOK)
    {
        // convert BOOL to bool
        bool bProved = (dlg.m_bProved != 0);
        bool bSigned = (dlg.m_bSigned != 0);

        CStandingOrder order(dlg.m_strAuthor, dlg.m_strTitle, dlg.m_strKey,
                             bProved, bSigned, dlg.m_strFormality,
                             dlg.m_nAmount, dlg.m_strSubject, dlg.m_strSource,
                             dlg.m_strRecipient, dlg.m_nInterval);
        m_Folder.Insert(&order);
    }

    UpdateAllViews(NULL);
}


/*!
    Insert a new contract
    \see CContract, CContractDialog
    \invariant see CContractDialog
*/
void CPraktikumMFCDoc::InsertContract()
{
    // dialog is used for construction
    View::CContractDialog dlg(true);

    dlg.m_strDateOfFoundation = "(noch nicht erstellt)";
    dlg.m_strAlterationDate   = "(noch nicht erstellt)";
    dlg.m_bProved = dlg.m_bSigned = FALSE;

    if (dlg.DoModal() == IDOK)
    {
        // convert BOOL to bool
        bool bProved = (dlg.m_bProved != 0);
        bool bSigned = (dlg.m_bSigned != 0);

        CContract contract(dlg.m_strAuthor, dlg.m_strTitle, dlg.m_strKey,
                           dlg.m_nRegistrationNumber, bProved, bSigned, dlg.m_strWording);
        m_Folder.Insert(&contract);
    }

    UpdateAllViews(NULL);
}


/*!
    Edit a folder's element, care for polymorphic behaviour of the set
    \param nEntry index of the edited element
    \see CBankDocument, CForm, CStandingOrder, CContract,
         CBankDocumentDialog, CFormDialog, CStandingOrderDialog, CContractDialog,
         CObject::IsKindOf
*/
void CPraktikumMFCDoc::EditEntry(int nEntry)
{
    // move cursor to the selected element
    m_Folder.SetCursor(nEntry);
    // get it
    CBankDocument* pObject = m_Folder.GetCurrent();
```

```
    // IsKindOf is true, if class(a)=class(b) AND if class(a)=derived from class(b)
    // therefore, we have to look for derived classes first

    // CContract ?
    if (pObject->IsKindOf(RUNTIME_CLASS(CContract)))
    {
        CContract* pContract = static_cast<CContract*>(pObject);

        // fill in dialog's controls
        View::CContractDialog dlg;
        dlg.m_strDateOfFoundation = pContract->GetDateOfFoundation();
        dlg.m_strAlterationDate   = pContract->GetAlterationDate();
        dlg.m_strWording          = pContract->GetWording();
        dlg.m_nRegistrationNumber = pContract->GetRegistrationNumber();
        dlg.m_strAuthor = pContract->GetAuthor();
        dlg.m_strKey    = pContract->GetKey();
        dlg.m_strTitle  = pContract->GetTitle();
        dlg.m_bProved   = pContract->GetProved();
        dlg.m_bSigned   = pContract->GetSigned();

        // only update if user pressed OK
        if (dlg.DoModal() == IDOK)
        {
            pContract->SetSigned(dlg.m_bProved != 0);
            pContract->SetProved(dlg.m_bSigned != 0);
            pContract->SetWording(dlg.m_strWording);

            // delete old entry
            m_Folder.Scratch();
            // add the updated one
            m_Folder.Insert(pContract);

            SetModifiedFlag();
        }
    }
    else
    // CStandingOrder ?
    if (pObject->IsKindOf(RUNTIME_CLASS(CStandingOrder)))
    {
        CStandingOrder* pOrder = static_cast<CStandingOrder*>(pObject);

        // fill in dialog's controls
        View::CStandingOrderDialog dlg;
        dlg.m_strDateOfFoundation = pOrder->GetDateOfFoundation();
        dlg.m_strAlterationDate   = pOrder->GetAlterationDate();
        dlg.m_strFormality        = pOrder->GetFormality();
        dlg.m_strAuthor = pOrder->GetAuthor();
        dlg.m_strKey    = pOrder->GetKey();
        dlg.m_strTitle  = pOrder->GetTitle();
        dlg.m_bProved   = pOrder->GetProved();
        dlg.m_bSigned   = pOrder->GetSigned();
        dlg.m_nAmount   = pOrder->GetAmount();
        dlg.m_nInterval = pOrder->GetInterval();
        dlg.m_strSource = pOrder->GetSource();
        dlg.m_strRecipient = pOrder->GetRecipient();
        dlg.m_strSubject   = pOrder->GetSubject();

        // only update if user pressed OK
        if (dlg.DoModal() == IDOK)
        {
            pOrder->SetSigned(dlg.m_bProved != 0);
            pOrder->SetProved(dlg.m_bSigned != 0);
            pOrder->SetAmount(dlg.m_nAmount);
            pOrder->SetInterval(dlg.m_nInterval);

            // delete old entry
            m_Folder.Scratch();
            // add the updated one
            m_Folder.Insert(pOrder);

            SetModifiedFlag();
        }
    }
    else
    // CForm ?
```

```
    if (pObject->IsKindOf(RUNTIME_CLASS(CForm)))
    {
        CForm* pForm = static_cast<CForm*>(pObject);

        // fill in dialog's controls
        View::CFormDialog dlg;
        dlg.m_strDateOfFoundation = pForm->GetDateOfFoundation();
        dlg.m_strAlterationDate   = pForm->GetAlterationDate();
        dlg.m_strFormality        = pForm->GetFormality();
        dlg.m_strAuthor = pForm->GetAuthor();
        dlg.m_strKey    = pForm->GetKey();
        dlg.m_strTitle  = pForm->GetTitle();
        dlg.m_bProved   = pForm->GetProved();
        dlg.m_bSigned   = pForm->GetSigned();

        // only update if user pressed OK
        if (dlg.DoModal() == IDOK)
        {
            pForm->SetSigned(dlg.m_bProved != 0);
            pForm->SetProved(dlg.m_bSigned != 0);

            // delete old entry
            m_Folder.Scratch();
            // add the updated one
            m_Folder.Insert(pForm);

            SetModifiedFlag();
        }
    }
    else
    // CBankDocument ?
    if (pObject->IsKindOf(RUNTIME_CLASS(CBankDocument)))
    {
        // fill in dialog's controls
        View::CBankDocumentDialog dlg;
        dlg.m_strAuthor = pObject->GetAuthor();
        dlg.m_strDateOfFoundation = pObject->GetDateOfFoundation();
        dlg.m_strKey    = pObject->GetKey();
        dlg.m_strTitle  = pObject->GetTitle();

        // show dialog
        dlg.DoModal();
    }
    else
        // RTTI failed
        TRACE("Cannot identify class %s.\n", pObject->GetRuntimeClass()->m_lpszClassName);

    // delete the copy
    delete pObject;
    UpdateAllViews(NULL);
}


/*!
    Delete the whole folder
*/
void CPraktikumMFCDoc::DeleteAll()
{
    m_Folder.ScratchAll();
    UpdateAllViews(NULL);
}


/*!
    Delete current element
    \param nEntry index of the selected element
*/
void CPraktikumMFCDoc::DeleteEntry(int nEntry)
{
    m_Folder.SetCursor(nEntry);
    m_Folder.Scratch();
    UpdateAllViews(NULL);
}


IMPLEMENT_DYNCREATE(CPraktikumMFCDoc, CDocument)
```

```
BEGIN_MESSAGE_MAP(CPraktikumMFCDoc, CDocument)
    //{{AFX_MSG_MAP(CPraktikumMFCDoc)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

} // namespace Model
```

Polymorphe Mengenorganisation

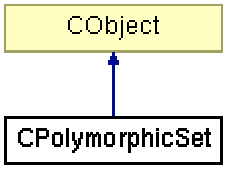

Abbildung 53: Vererbungshierarchie `CPolymorphicSet<>`

Interface und Implementierung (PolymorphicSet.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  PolymorphicSet.h
    \brief template CPolymorphicSet<C> (interface + implementation)
*/

#pragma once
#include "stdafx.h"
// MFC's CList template is used
#include <afxtempl.h>
// add CTrace
#include "Trace.h"


//! Namespace that contains all model classes
/*!
    \li Model component of the MVC concept
*/
namespace Model
{
    using Debug::CTrace;

//! Template of a polymorphic set
/*! \li The only parameter is the base class of all elements (at least CObject)
    \li All elements are copied by the interface methods
        to avoid unexpected behaviour (e.g. external deletion)
    \li fully serializable

    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li If the set is empty then m_Cursor must be NULL else it must point to a element
    \li No element is allowed to appear more than once
*/


//##ModelId=3B863AFA0154
template<class C> class CPolymorphicSet : public CObject
{
public:
    //! default constructor
    CPolymorphicSet();
    //! copy constructor
    CPolymorphicSet(const CPolymorphicSet<C>& set);

    //! destructor
    virtual ~CPolymorphicSet();

    //! serialization
    virtual void Serialize(CArchive &ar);
    //! invariance
    virtual void AssertValid() const;
    //! dump
    virtual void Dump(CDumpContext& dc) const;

    //! copy the whole set
    virtual CPolymorphicSet<C>& operator=(const CPolymorphicSet<C>& set);
    //! compare two sets
```

```
    //! value identity
    virtual bool operator==(const CPolymorphicSet<C>& set) const;
    //! object identity
            bool Equal     (const CPolymorphicSet<C>* set) const;

    //! returns number of stored elements
    int Card() const;

    //! first stored element
    C* GetFirst();
    //! next element
    C* GetNext();
    //! gets the element the cursor points to
    C* GetCurrent() const;
    //! set the cursor to the element specified by an index
    void SetCursor(int nIndex);

    //! inserts an element
    int  Insert(const C* element);
    //! deletes current element
    void Scratch();
    //! deletes the whole set
    void ScratchAll();

    //! finds an element
    bool Find(const C* element);

protected:
    //! creates a new instance and copies all attributes
    C* Clone(const C* element) const;

    //! stores all elements in a list
    //##ModelId=3B863AFA0199
    CList<C*, C*> m_Set;
    //! cursor, may be altered at any time
    //##ModelId=3B863AFA0194
    POSITION m_Cursor;
};


//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////


/*!
    Constructs an empty set
*/
template<class C> CPolymorphicSet<C>::CPolymorphicSet()
{
    // empty set
    m_Cursor = NULL;
}


/*!
    Copies an existing set
    \param set reference to the set that should be copied
    \see operator=
*/
template<class C> CPolymorphicSet<C>::CPolymorphicSet(const CPolymorphicSet<C>& set)
{
    operator=(set);
}


/*!
    Cleans up memory
    \see ScratchAll
*/
template<class C> CPolymorphicSet<C>::~CPolymorphicSet()
{
    ScratchAll();
}
```

```
/*!
    Loads / stores the set
    \param ar CArchive that provides access to the file system
*/
template<class C> void CPolymorphicSet<C>::Serialize(CArchive &ar)
{
    CTrace trace("Serialize",this);

    // do not forget to serialize CObject's data members
    CObject::Serialize(ar);

    // make use of CArchive's overloaded << and >> operators for *CObject

    if (ar.IsStoring())
    {
        // save polymorphic set

        // write cardinality
        ar << m_Set.GetCount();

        // write all elements
        POSITION pos     = m_Set.GetHeadPosition();
        int nCursorIndex = 0,
            nCount       = 0;
        while (pos != NULL)
        {
            // get cursor index
            if (m_Cursor == pos)
                nCursorIndex = nCount;
            nCount++;

            ar << m_Set.GetNext(pos);
        }

        // store cursor index
        ar << nCursorIndex;
    }
    else
    {
        // load polymorphic set

        // first, delete old set
        ScratchAll();

        int nCount;
        // load cardinality
        ar >> nCount;
        // load all elements
        for (int i=0; i<nCount; i++)
        {
            C* element;
            ar >> element;
            m_Set.AddTail(element);
        }

        // get cursor index
        int nCursorIndex;
        ar >> nCursorIndex;
        // create cursor out of the index
        m_Cursor = m_Set.FindIndex(nCursorIndex);
    }
}


/*!
    Determines whether the set is valid, \b invariant describes further details (see above)
*/
template<class C> void CPolymorphicSet<C>::AssertValid() const
{
    CObject::AssertValid();

    // if the set is empty then cursor must be NULL
    if (m_Set.IsEmpty())
    {
        ASSERT(m_Cursor == NULL);
        return;
```

```
    }

    CString strClassName = GetRuntimeClass()->m_lpszClassName;

    // each element may be present in set at most once
    // cursor must point to an element (or NULL)
    POSITION pos = m_Set.GetHeadPosition();
    bool bValidCursor = false;
    if (m_Cursor == NULL)
        bValidCursor = true;

    while (pos != NULL)
    {
        // element that the cursor points to was found ?
        if (pos == m_Cursor)
            bValidCursor = true;

        // get element
        C* element = m_Set.GetNext(pos);
        POSITION posfind = pos;
        // compare to the remaining elements
        while (posfind != NULL)
        {
            C* compare = m_Set.GetNext(posfind);
            // no two equal elements are allowed

            // same class ?
            if (strClassName == compare->GetRuntimeClass()->m_lpszClassName)
                // equal values ?
                ASSERT(!(*element == *compare));
        }
    }

    // cursor successfully validated ?
    ASSERT(bValidCursor);
}


/*!
    Create a dump of the class (only DEBUG version)
    \param dc output dump stream
*/
template<class C> void CPolymorphicSet<C>::Dump(CDumpContext& dc) const
{
    CObject::Dump(dc);

    // some general infos
    dc << Card() << " elements, cursor points to " << m_Cursor << "\n"
       << "// the following elements are separated by a line (----------)\n";

    // dump all elements
    POSITION pos = m_Set.GetHeadPosition();
    while (pos != NULL)
    {
        m_Set.GetNext(pos)->Dump(dc);
        dc << "\n----------\n";
    }
}


/*!
    Copies all attributes with respect to cloning all elements and adjusting the cursor
    \param  set set that will be copied
    \return reference to the current set
    \see    CPolymorphicSet(const CPolymorphicSet<C>& set)
*/
template<class C> CPolymorphicSet<C>& CPolymorphicSet<C>::operator=(const CPolymorphicSet<C>&
set)
{
    CTrace trace("operator=",this,&set);

    // delete old elements
    ScratchAll();

    POSITION pos = set.m_Set.GetHeadPosition();
    while (pos != NULL)
```

```
    {
        // save current position
        POSITION cursor = pos;

        // clone each element
        C* element = set.m_Set.GetNext(pos);
        C* copy    = Clone(element);
        m_Set.AddTail(copy);

        // set cursor
        // we can't use set.m_Cursor directly because we copied all elements
        // and therefore lost their old memory address
        if (cursor == set.m_Cursor)
            m_Cursor = m_Set.GetTailPosition();
    }

    return *this;
}


/*!
    Compares all elements of two sets by value regardless their positions within the set.
    The cursors are not compared and may differ therefore
    \param  set set to compare to
    \return true, if both set are equal
    \pre    parameter "set" must be valid
*/
template<class C> bool CPolymorphicSet<C>::operator==(const CPolymorphicSet<C>& set) const
{
    CTrace trace("operator==",this,&set);

    ASSERT_VALID(&set);

    // same number of elements is required
    if (m_Set.GetCount() != set.m_Set.GetCount())
        return false;

    // each element of this set must be found in the other set
    POSITION pos = m_Set.GetHeadPosition();
    while (pos != NULL)
    {
        C* element = m_Set.GetNext(pos);
        if (set.m_Set.Find(element))
            return false;
    }

    // all elements were found, sets must be equal
    return true;
}


/*!
    Check whether two to sets are not only equal but also identical
    \param  set set to compare to
    \return true, if both sets are identical
*/
template<class C> bool CPolymorphicSet<C>::Equal(const CPolymorphicSet<C>* set) const
{
    CTrace trace("Equal",this,set);

    return (set == this);
}


/*!
    Determine the size of the polymorphic set
    \return number of stored elements
*/
template<class C> int CPolymorphicSet<C>::Card() const
{
    CTrace trace("Card",this);

    return m_Set.GetCount();
}
```

```
/*!
    Set cursor to the first element and return a copy of it
    \return copy of the first element
    \see    GetNext, GetCurrent
    \pre    set must be non-empty
*/
template<class C> C* CPolymorphicSet<C>::GetFirst()
{
    CTrace trace("GetFirst",this);

    // set must not be empty
    ASSERT(!m_Set.IsEmpty());

    // set cursor
    m_Cursor = m_Set.GetHeadPosition();

    // generate copy
    C* copy = Clone(m_Set.GetHead());
    return copy;
}


/*!
    Iterate cursor to the next element and return a copy of it
    \return copy of the next element
    \see    GetFirst, GetCurrent
    \pre    set must be non-empty and cursor must be valid
*/
template<class C> C* CPolymorphicSet<C>::GetNext()
{
    CTrace trace("GetNext",this);

    // set must not be empty
    ASSERT(!m_Set.IsEmpty());
    // cursor must not be at the end of the set
    ASSERT(m_Cursor != NULL);

    // set cursor
    m_Set.GetNext(m_Cursor);

    // generate copy
    C* copy = Clone(m_Set.GetAt(m_Cursor));
    return copy;
}


/*!
    Return a copy of the element the cursor points to
    \return copy of the current element
    \see    GetFirst, GetNext
    \pre    cursor must be valid
*/
template<class C> C* CPolymorphicSet<C>::GetCurrent() const
{
    CTrace trace("GetCurrent",this);

    // cursor must not be at the end of the set
    ASSERT(m_Cursor != NULL);

    // generate copy
    C* copy = Clone(m_Set.GetAt(m_Cursor));
    return copy;
}


/*!
    Set the cursor to an element specified by an index
    \param nIndex zero-based index
*/
template<class C> void CPolymorphicSet<C>::SetCursor(int nIndex)
{
    CTrace trace("SetCursor",this);

    m_Cursor = m_Set.FindIndex(nIndex);
}
```

```
/*!
    Insert a new element into the set and return its position
    \param  element element to be inserted
    \return zero-based index, -1 if failed (element already exists)
    \see Clone
*/
template<class C> int CPolymorphicSet<C>::Insert(const C* element)
{
    CTrace trace("Insert",this,element);

    // don't insert an object twice
    if (Find(element))
        return -1;

    // add a copy to our set, change cursor
    m_Cursor = m_Set.AddTail(Clone(element));

    // return index in the set
    return m_Set.GetCount()-1;
}


/*!
    Delete the element the cursor currently points to and adjust cursor
    \see ScratchAll
    \pre cursor must be valid (and the set non-empty)
*/
template<class C> void CPolymorphicSet<C>::Scratch()
{
    CTrace trace("Scratch",this);

    // cursor must not be at the end of the set
    ASSERT(m_Cursor != NULL);

    // store current cursor
    POSITION removepos = m_Cursor;
    // go on to the next element (else we lose it after erasing this element)
    m_Set.GetNext(m_Cursor);

    // remove element that the cursor previously pointed to
    delete m_Set.GetAt(removepos);
    m_Set.RemoveAt(removepos);
}


/*!
    Delete the whole set including all stored elements
    \see Scratch
*/
template<class C> void CPolymorphicSet<C>::ScratchAll()
{
    CTrace trace("ScratchAll",this);

    // front to back
    while (!m_Set.IsEmpty())
    {
        // delete element
        delete m_Set.GetHead();
        // remove the correspending pointer from the set
        m_Set.RemoveHead();
    }

    // reset cursor
    m_Cursor = NULL;
}


/*!
    Search through the whole set and set m_Cursor if found
    \param  element element to look for
    \return true, if element was found
*/
template<class C> bool CPolymorphicSet<C>::Find(const C* element)
{
    CTrace trace("Find",this,element);
```

```
    // only non-empty sets
    if (m_Set.IsEmpty())
        return false;

    CString strClassName = element->GetRuntimeClass()->m_lpszClassName;

    // iterate through the whole list
    POSITION pos = m_Set.GetHeadPosition();
    while (pos != NULL)
    {
        // store position in case we find a match
        POSITION storepos = pos;

        // get an element we compare to
        C* compare  = m_Set.GetNext(pos);

        // same class ?
        if (strClassName == compare->GetRuntimeClass()->m_lpszClassName)
            // equal values ?
            if (*element == *compare)
            {
                // yes, set cursor and return "success !"
                m_Cursor = storepos;
                return true;
            }
    }

    // element is not part of the set
    return false;
}


/*!
    Copy an object using MFC's own RTTI mechanismus to prevent polymorphism
    \param  element element to be cloned
    \return copy of element
*/
template<class C> C* CPolymorphicSet<C>::Clone(const C* element) const
{
    CTrace trace("Clone",this,element);

    // create new instance (care for correct class !)
    C* copy = static_cast<C*>(element->GetRuntimeClass()->CreateObject());
    // copy attributes
    *copy = *element;

    return copy;
}

} // namespace Model
```

Kartei

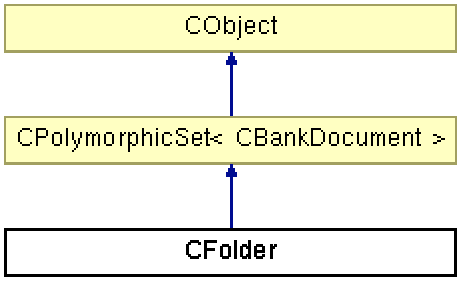

Abbildung 54: Vererbungshierarchie `CFolder`

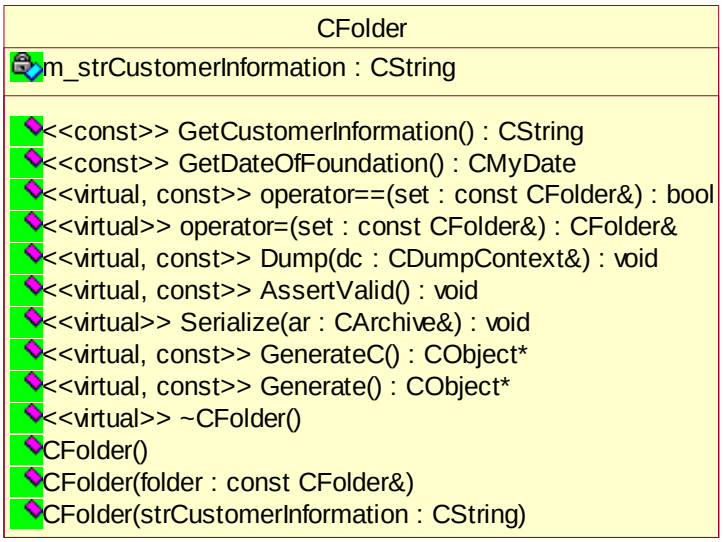

Abbildung 55: UML-Beschreibung von `CFolder`

Interface (Folder.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Folder.h
    \brief class CFolder (interface)
*/

#pragma once
#include "PolymorphicSet.h"
#include "BankDocument.h"

namespace Model
{

//! A folder
/*!
    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li date of foundation must be valid
    \li customer information must be non-empty
    \li parent class must be valid
*/

//##ModelId=3B863B000294
class CFolder : public CPolymorphicSet<CBankDocument>
{
```

```
public:
    //! default constructor
    //##ModelId=3B863B000342
    CFolder();
    //! copy constructor
    //##ModelId=3B863B000343
    CFolder(const CFolder& folder);
    //! explicitly set attributes
    //##ModelId=3B863B000345
    CFolder(CString strCustomerInformation);

    //! destructor
    //##ModelId=3B863B000340
    virtual ~CFolder();

    //! create a new object
    //##ModelId=3B863B00033E
    virtual CObject* Generate () const;
    //! clone this object
    //##ModelId=3B863B00033C
    virtual CObject* GenerateC() const;
    //! serialization
    //##ModelId=3B863B000339
    virtual void Serialize(CArchive &ar);
    //! validate the form
    //##ModelId=3B863B000337
    virtual void AssertValid() const;
    //! dump
    //##ModelId=3B863B000334
    virtual void Dump(CDumpContext& dc) const;

    //! copy the whole set
    //##ModelId=3B863B00030F
    virtual CFolder& operator=(const CFolder& set);
    //! compare two sets
    //##ModelId=3B863B00030C
    virtual bool operator==(const CFolder& set) const;

    //! get date of foundation
    //##ModelId=3B863B00030A
    CMyDate GetDateOfFoundation()    const;
    //! get customer information
    //##ModelId=3B863B000308
    CString GetCustomerInformation() const;

private:
    //! customer information
    //##ModelId=3B863B000307
    CString m_strCustomerInformation;
    //! date of foundation
    //##ModelId=3B863B000304
    CMyDate m_dtDateOfFoundation;


    DECLARE_SERIAL(CFolder)
};

} // namespace Model
```

Implementation (Folder.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Folder.cpp
    \brief class CFolder (implementation)
*/

#include "stdafx.h"

#include "MyDate.h"
#include "Folder.h"

#ifdef _DEBUG
```

```
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif


namespace Model
{

/*!
    Default constructor
*/
CFolder::CFolder()
{
    m_strCustomerInformation = "(neu)";
}


/*!
    Copy constructor
    \param folder CFolder to be copied
*/
CFolder::CFolder(const CFolder &folder)
{
    operator=(folder);
}


/*!
    Initialize using customer information
    \param strCustomerInformation customer information
*/
CFolder::CFolder(CString strCustomerInformation)
{
    m_strCustomerInformation = strCustomerInformation;
}


/*!
    Nothing left to do
*/
CFolder::~CFolder()
{
}


/*!
    Create a new non-initialized object
    \return new instance of CFolder
*/
CObject* CFolder::Generate () const
{
    return new CFolder;
}


/*!
    Clone this object
    \return exact copy
*/
CObject* CFolder::GenerateC() const
{
    return new CFolder(*this);
}


/*!
    Loads/stores CFolder
    \param ar CArchive that provides data persistency
*/
void CFolder::Serialize(CArchive &ar)
{
    // serialize parent
    CPolymorphicSet<CBankDocument>::Serialize(ar);
```

```
    if (ar.IsStoring())
        ar << m_strCustomerInformation;
    else
        ar >> m_strCustomerInformation;

    m_dtDateOfFoundation.Serialize(ar);
}


/*!
    Determines whether the foolder is valid,
    \b invariant describes further details (see above)
*/
void CFolder::AssertValid() const
{
    CPolymorphicSet<CBankDocument>::AssertValid();

    ASSERT(!m_strCustomerInformation.IsEmpty());
    ASSERT_VALID(&m_dtDateOfFoundation);
}


/*!
    Create a dump of the class (only DEBUG version)
    \param dc output dump stream
*/
void CFolder::Dump(CDumpContext& dc) const
{
    CPolymorphicSet<CBankDocument>::Dump(dc);

    dc << "customer information: " << m_strCustomerInformation << "\n"
       << "date of foundation: ";

    m_dtDateOfFoundation.Dump(dc);
}


/*!
    Copies all attributes
    \param  set set that will be copied
    \return reference to the current set
*/
CFolder& CFolder::operator=(const CFolder& set)
{
    m_dtDateOfFoundation     = set.m_dtDateOfFoundation;
    m_strCustomerInformation = set.m_strCustomerInformation;

    CPolymorphicSet<CBankDocument>::operator=(set);

    return *this;
}


/*!
    Compares all elements of two sets by value regardless their positions within the set.
    The cursors are not compared and may differ therefore
    \param  set set to compare to
    \return true, if both set are equal
    \pre    parameter "set" must be valid
*/
bool CFolder::operator==(const CFolder& set) const
{
   return (CPolymorphicSet<CBankDocument>::operator==(set) &&
           m_strCustomerInformation == set.m_strCustomerInformation);
}


/*!
    Get date of foundation
    \return date of foundation
*/
CMyDate CFolder::GetDateOfFoundation() const
{
    return m_dtDateOfFoundation;
}
```

```
/*!
    Get customer information
    \return customer information
*/
CString CFolder::GetCustomerInformation() const
{
    return m_strCustomerInformation;
}


// serializable
IMPLEMENT_SERIAL(CFolder, CObject, 1)

} // namespace Model
```

BankDokument

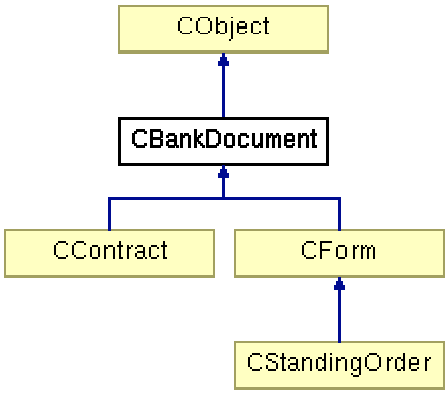

Abbildung 56: Vererbungshierarchie `CBankDocument`

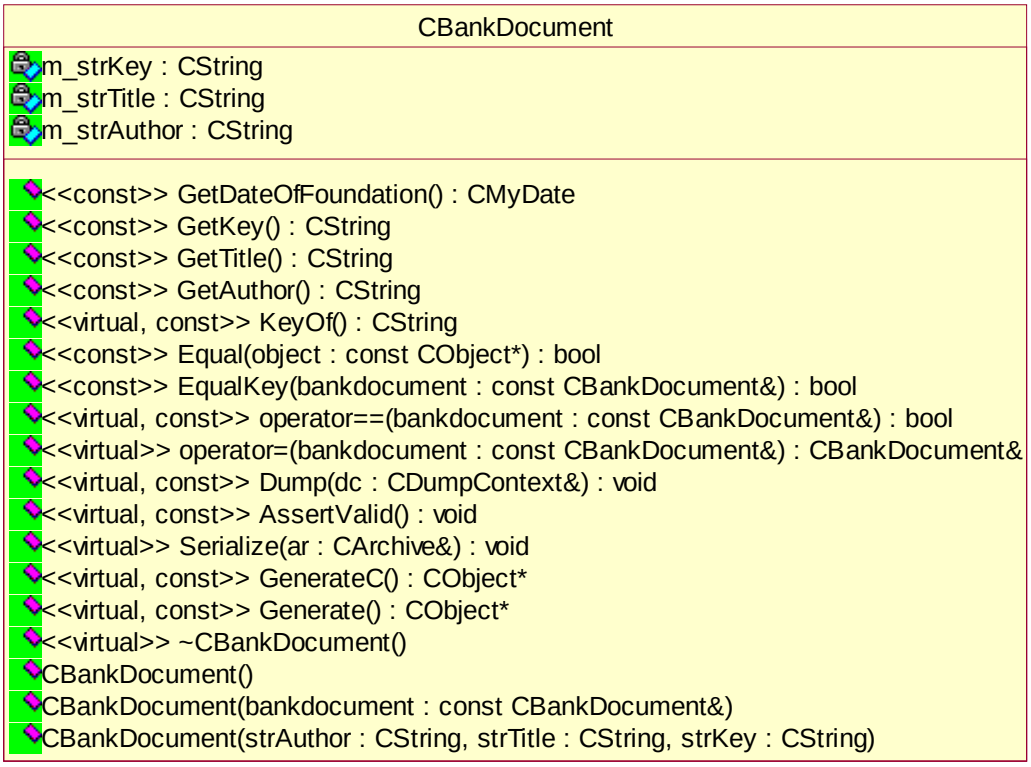

Abbildung 57: UML-Beschreibung von `CBankDocument`

Interface (BankDocument.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  BankDocument.h
    \brief class CBankDocument (interface)
*/

#include "MyDate.h"
#pragma once

namespace Model
{
```

```
//! A basic bank document
/*! \li base class for all other bank-related classes

    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li each CString attribute must be non-empty
    \li m_dtDateOfFoundation must be valid
*/

//##ModelId=3B863B040230
class CBankDocument : public CObject
{
public:
    //! default constructor
    //##ModelId=3B863B0402D4
    CBankDocument();
    //! copy constructor
    //##ModelId=3B863B0402D5
    CBankDocument(const CBankDocument& bankdocument);
    //! explicitly set author, title and key
    //##ModelId=3B863B0402D7
    CBankDocument(CString strAuthor, CString strTitle, CString strKey);

    //! destructor
    //##ModelId=3B863B0402D2
    virtual ~CBankDocument();

    //! create a new object
    //##ModelId=3B863B0402D0
    virtual CObject* Generate () const;
    //! clone this object
    //##ModelId=3B863B0402AD
    virtual CObject* GenerateC() const;
    //! serialization
    //##ModelId=3B863B0402AA
    virtual void Serialize(CArchive &ar);
    //! validate the folder
    //##ModelId=3B863B0402A8
    virtual void AssertValid() const;
    //! dump
    //##ModelId=3B863B0402A5
    virtual void Dump(CDumpContext& dc) const;

    //! copy
    //##ModelId=3B863B0402A2
    virtual CBankDocument& operator = (const CBankDocument& bankdocument);
    //! compare value
    //##ModelId=3B863B04029F
    virtual bool           operator ==(const CBankDocument& bankdocument) const;
    //! compare keys
    //##ModelId=3B863B04026F
    bool EqualKey   (const CBankDocument& bankdocument) const;
    //! check for identity
    //##ModelId=3B863B04026C
    bool Equal(const CObject* object) const;

    //! deliver a unique identifier
    //##ModelId=3B863B04026A
    virtual CString KeyOf() const;

    //! get author
    //##ModelId=3B863B040268
    CString GetAuthor() const;
    //! get title
    //##ModelId=3B863B040266
    CString GetTitle()  const;
    //! get key
    //##ModelId=3B863B040264
    CString GetKey()    const;
    //! get date of foundation
    //##ModelId=3B863B040262
    CMyDate GetDateOfFoundation() const;
```

```
private:
    //! author
    //##ModelId=3B88C0FF0078
    CString m_strAuthor;
    //! title
    //##ModelId=3B88C0FF003D
    CString m_strTitle;
    //! key
    //##ModelId=3B88C0FF003C
    CString m_strKey;
    //! date of foundation
    //##ModelId=3B863B040234
    CMyDate m_dtDateOfFoundation;


    DECLARE_SERIAL(CBankDocument)
};

} // namespace Model
```

Implementation (BankDocument.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  BankDocument.cpp
    \brief class CBankDocument (implementation)
*/

#include "stdafx.h"
#include "BankDocument.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif



namespace Model
{

/*!
    Constructs a new bank document with the current date/time
    \see CMyDate()
*/
CBankDocument::CBankDocument()
{
    m_dtDateOfFoundation = CMyDate();
}


/*!
    Copies an existing CBankDocument
    \param bankdocument bank document to be copied
    \see operator=
*/
CBankDocument::CBankDocument(const CBankDocument& bankdocument)
{
    operator=(bankdocument);
}


/*!
    Constructs a new bank document using given author, title and key
    \param strAuthor author
    \param strTitle  title
    \param strKey    key
*/
CBankDocument::CBankDocument(CString strAuthor, CString strTitle, CString strKey)
{
    // copy parameters
    m_strAuthor = strAuthor;
```

```
    m_strTitle  = strTitle;
    m_strKey    = strKey;
    // refresh date
    m_dtDateOfFoundation = CMyDate();
}


/*!
    No clean-up required
*/
CBankDocument::~CBankDocument()
{
}


/*!
    Create a new non-initialized object
    \return new instance of CBankDocument
*/
CObject* CBankDocument::Generate() const
{
    return new CBankDocument;
}


/*!
    Clone this object
    \return exact copy
*/
CObject* CBankDocument::GenerateC() const
{
    return new CBankDocument(*this);
}


/*!
    Loads/stores CBankDocument
    \param ar CArchive that provides data persistency
*/
void CBankDocument::Serialize(CArchive &ar)
{
    CObject::Serialize(ar);

    if (ar.IsStoring())
        ar << m_strAuthor << m_strTitle << m_strKey;
    else
        ar >> m_strAuthor >> m_strTitle >> m_strKey;

    // << and >> operators are not overloaded
    m_dtDateOfFoundation.Serialize(ar);
}


/*!
    Determines whether the bank document is valid,
    \b invariant describes further details (see above)
*/
void CBankDocument::AssertValid() const
{
    CObject::AssertValid();

    ASSERT(!m_strAuthor.IsEmpty());
    ASSERT(!m_strTitle .IsEmpty());
    ASSERT(!m_strKey   .IsEmpty());

    ASSERT_VALID(&m_dtDateOfFoundation);
}


/*!
    Create a dump of the class (only DEBUG version)
    \param dc output dump stream
*/
void CBankDocument::Dump(CDumpContext& dc) const
{
    CObject::Dump(dc);
```

```
    dc << "author: " << m_strAuthor << "\n"
       << "title:  " << m_strTitle  << "\n"
       << "key:    " << m_strKey    << "\n"
       << "date:   ";
    // << operator is not overloaded
    m_dtDateOfFoundation.Dump(dc);
}


/*!
    Copies all attributes
    \param  bankdocument CBankDocument that should be copied
    \return reference to the copied bank document
*/
CBankDocument& CBankDocument::operator = (const CBankDocument& bankdocument)
{
    // copy attributes
    m_strAuthor = bankdocument.m_strAuthor;
    m_strTitle  = bankdocument.m_strTitle;
    m_strKey    = bankdocument.m_strKey;
    m_dtDateOfFoundation = bankdocument.m_dtDateOfFoundation;

    return *this;
}


/*!
    Compare attributes (except key) of two bank documents
    \param  bankdocument CBankDocument that we compare to
    \return true, if all attributes are equal
*/
bool CBankDocument::operator ==(const CBankDocument& bankdocument) const
{
    return (m_strAuthor == bankdocument.m_strAuthor &&
            m_strTitle  == bankdocument.m_strTitle  &&
            m_strKey    == bankdocument.m_strKey    );
//            m_dtDateOfFoundation == bankdocument.m_dtDateOfFoundation);
}


/*!
    Compare keys
    \param bankdocument CBankDocument that we compare to
    \return true, if keys are equal
*/
bool CBankDocument::EqualKey(const CBankDocument& bankdocument) const
{
    return (KeyOf() == bankdocument.KeyOf());
}


/*!
    Check for object identity
    \param object object that we compare to
    \return true, if both objects are identical
*/
bool CBankDocument::Equal(const CObject* object) const
{
    return (this == object);
}


/*!
    Responsible for the generation of a unique key based on the object's attributes
    \return CString that contains the key
*/
CString CBankDocument::KeyOf() const
{
    CString strKey;
    strKey.Format("[%s|%s|%s|%s|%p]", m_strAuthor, m_strTitle,  m_strKey,
                                      m_dtDateOfFoundation.KeyOf(),
                                      this);
    return strKey;
}
```

```
/*!
    Get author
    \return author
*/
CString CBankDocument::GetAuthor() const
{
    return m_strAuthor;
}
/*!
    Get title
    \return title
*/
CString CBankDocument::GetTitle() const
{
    return m_strTitle;
}
/*!
    Get key
    \return key
*/
CString CBankDocument::GetKey() const
{
    return m_strKey;
}
/*!
    Get date of foundation
    \return date of foundation
*/
CMyDate CBankDocument::GetDateOfFoundation() const
{
    return m_dtDateOfFoundation;
}


// serializable
IMPLEMENT_SERIAL(CBankDocument, CObject, 1)

}
```

Formular

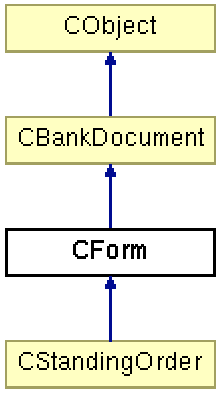

Abbildung 58: Vererbungshierarchie `CForm`

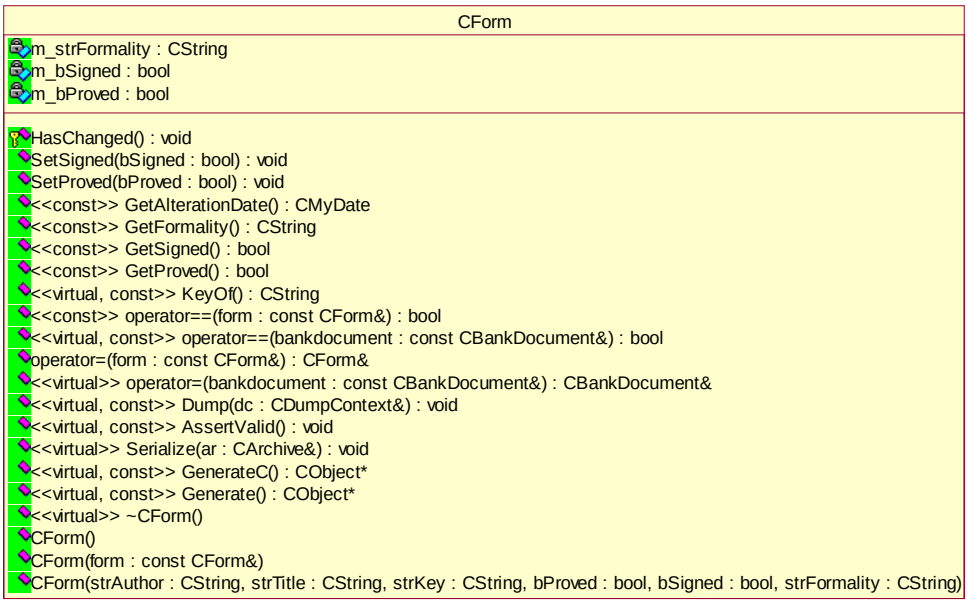

Abbildung 59: UML-Beschreibung von `CForm`

Interface (Form.h)

```
//////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Form.h
    \brief class CForm (interface)
*/

#include "BankDocument.h"
#pragma once

namespace Model
{

//! A form
/*!
    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li m_strFormality must be non-empty
    \li m_dtAlterationDate must be valid
```

```
    \li parent class must be valid
*/

//##ModelId=3B863AF702F8
class CForm : public CBankDocument
{
public:
    //! default constructor
    //##ModelId=3B863AF703E0
    CForm();
    //! copy constructor
    //##ModelId=3B863AF703E1
    CForm(const CForm& form);
    //! explicitly set attributes
    //##ModelId=3B863AF703E3
    CForm(CString strAuthor, CString strTitle, CString strKey,
          bool    bProved,   bool    bSigned,  CString strFormality);

    //! destructor
    //##ModelId=3B863AF703DE
    virtual ~CForm();

    //! create a new object
    //##ModelId=3B863AF703DC
    virtual CObject* Generate () const;
    //! clone this object
    //##ModelId=3B863AF703DA
    virtual CObject* GenerateC() const;
    //! serialization
    //##ModelId=3B863AF703D7
    virtual void Serialize(CArchive &ar);
    //! validate the form
    //##ModelId=3B863AF703D5
    virtual void AssertValid() const;
    //! dump
    //##ModelId=3B863AF703B2
    virtual void Dump(CDumpContext& dc) const;

    //! copy
    //##ModelId=3B863AF703AD
            CForm&         operator = (const CForm& form);
    //! polymorphic copy
    //##ModelId=3B863AF703AF
    virtual CBankDocument& operator = (const CBankDocument& bankdocument);
    //! comparison
    //##ModelId=3B863AF703A7
            bool operator ==(const CForm& form) const;
    //! polymorphic comparison
    //##ModelId=3B863AF703AA
    virtual bool operator ==(const CBankDocument& bankdocument) const;

    //! deliver a unique identifier
    //##ModelId=3B863AF703A5
    virtual CString KeyOf() const;

    //! determine whether form is proved
    //##ModelId=3B863AF703A3
    bool    GetProved()         const;
    //! determine whether form is signed
    //##ModelId=3B863AF70373
    bool    GetSigned()         const;
    //! get formality
    //##ModelId=3B863AF70371
    CString GetFormality()      const;
    //! get alteration date
    //##ModelId=3B863AF7036F
    CMyDate GetAlterationDate() const;

    //! set "proved" property
    //##ModelId=3B863AF7036D
    void    SetProved(bool bProved);
    //! set "signed" property
    //##ModelId=3B863AF7036B
    void    SetSigned(bool bSigned);

protected:
```

```
    //! at least one attribute has changed
    //##ModelId=3B863AF7036A
    void HasChanged();

private:
    //! form is proved
    //##ModelId=3B88C0F20398
    bool    m_bProved;
    //! form is signed
    //##ModelId=3B88C0F20366
    bool    m_bSigned;
    //! formality
    //##ModelId=3B863AF70369
    CString m_strFormality;
    //! alteration date
    //##ModelId=3B863AF70366
    CMyDate m_dtAlterationDate;


    DECLARE_SERIAL(CForm)
};

} // namespace Model
```

Implementation (Form.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Form.cpp
    \brief class CForm (implementation)
*/

#include "stdafx.h"
#include "Form.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif


namespace Model
{

/*!
    Constructs a new unproved and unsigned form
*/
CForm::CForm()
{
    m_bProved = m_bSigned = false;
}


/*!
    Copies an existing CForm
    \param form form to be copied
    \see operator=
*/
CForm::CForm(const CForm& form)
{
    operator=(form);
}


/*!
    Constructs a new form using given attributes
    \param strAuthor    author
    \param strTitle     title
    \param strKey       key
    \param bProved      is it proved ?
    \param bSigned      is it signed ?
    \param strFormality formality
```

```
*/
CForm::CForm(CString strAuthor, CString strTitle, CString strKey,
             bool    bProved,   bool    bSigned,  CString strFormality)
     : CBankDocument(strAuthor, strTitle, strKey)
{
    m_bProved = bProved;
    m_bSigned = bSigned;

    m_strFormality = strFormality;
    HasChanged();
}


/*!
    No clean-up required
*/
CForm::~CForm()
{
}


/*!
    Create a new non-initialized object
    \return new instance of CForm
*/
CObject* CForm::Generate () const
{
    return new CForm;
}


/*!
    Clone this object
    \return exact copy
*/
CObject* CForm::GenerateC() const
{
    return new CForm(*this);
}


/*!
    Loads/stores CForm
    \param ar CArchive that provides data persistency
*/
void CForm::Serialize(CArchive &ar)
{
    // serialize parent
    CBankDocument::Serialize(ar);

    // CArchive can't handle "bool" so I have to use "BOOL" (which is "int" ...)
    BOOL bProved = m_bProved,
         bSigned = m_bSigned;

    if (ar.IsStoring())
        ar << bProved << bSigned << m_strFormality;
    else
        ar >> bProved >> bSigned >> m_strFormality;

    // neccessary because of MFC's "bool!=BOOL story"
    m_bProved = (bProved == TRUE);
    m_bSigned = (bSigned == TRUE);

    // << and >> operators are not overloaded
    m_dtAlterationDate.Serialize(ar);
}


/*!
    Determines whether the form is valid,
    \b invariant describes further details (see above)
*/
void CForm::AssertValid() const
{
    CBankDocument::AssertValid();
```

```
    ASSERT(!m_strFormality.IsEmpty());
    ASSERT_VALID(&m_dtAlterationDate);
}


/*!
    Create a dump of the class (only DEBUG version)
    \param dc output dump stream
*/
void CForm::Dump(CDumpContext& dc) const
{
    CBankDocument::Dump(dc);

    dc << "\n"
       << "proved: " << m_bProved << "\n"
       << "signed: " << m_bSigned << "\n"
       << "formality: " << m_strFormality << "\n"
       << "alternation date: ";

    m_dtAlterationDate.Dump(dc);
}


/*!
    Copies all attributes for polymorphic behaviour
    \param  form CForm that should be copied
    \return reference to the copied form
*/
CForm& CForm::operator = (const CForm& form)
{
    // copy inherited attributes
    CBankDocument::operator =(form);

    // copy CForm's attributes
    m_bProved = form.m_bProved;
    m_bSigned = form.m_bSigned;
    m_dtAlterationDate = form.m_dtAlterationDate;
    m_strFormality     = form.m_strFormality;

    return *this;
}


/*!
    Copies all attributes for polymorphic behaviour
    \param  bankdocument CForm (casted as CBankDocument) that should be copied
    \return reference to the copied form
*/
CBankDocument& CForm::operator = (const CBankDocument& bankdocument)
{
    try
    {
        // copy all attributes
        operator =(dynamic_cast<const CForm&>(bankdocument));
    }
    catch(...)
    {
        // invalid class
#ifdef _DEBUG
        TRACE("Copy denied because of invalid cast (needed a CForm)\n");
        Dump(afxDump);
        bankdocument.Dump(afxDump);
#endif
    }

    return *this;
}


/*!
    Compare attributes (except key) of two forms
    \param  form CForm that we compare to
    \return true, if all attributes are equal
*/
bool CForm::operator ==(const CForm& form) const
{
```

```
    return (CBankDocument::operator ==(form) &&
            m_bProved == form.m_bProved &&
            m_bSigned == form.m_bSigned &&
//            m_dtAlterationDate == form.m_dtAlterationDate &&
            m_strFormality     == form.m_strFormality);
}


/*!
    Compares all attributes for polymorphic behaviour
    \param  bankdocument CForm (casted as CBankDocument) that should be compared to
    \return true, if all attributes are equal
*/
bool CForm::operator ==(const CBankDocument& bankdocument) const
{
    try
    {
        // compare all attributes
        return operator ==(dynamic_cast<const CForm&>(bankdocument));
    }
    catch(...)
    {
        // invalid class
        TRACE("Comparison denied because of invalid cast\n"
              "(needed a CForm and got %s)\n", bankdocument.GetRuntimeClass()-
>m_lpszClassName);
#ifdef _DEBUG
        Dump(afxDump);
        bankdocument.Dump(afxDump);
#endif
    }

    return false;
}


/*!
    Responsible for the generation of a unique key based on the object's attributes
    \return CString that contains the key
*/
CString CForm::KeyOf() const
{
    CString strKey;
    strKey.Format("[%s|%d|%d|%s|%s|%p]", m_strFormality, m_bProved,  m_bSigned,
                                         m_dtAlterationDate.KeyOf(),
                                         CBankDocument::KeyOf(),
                                         this);
    return strKey;
}


/*!
    Get "proved" property
    \return true, if form is proved
*/
bool CForm::GetProved() const
{
    return m_bProved;
}
/*!
    Get "signed" property
    \return true, if form is signed
*/
bool CForm::GetSigned() const
{
    return m_bSigned;
}
/*!
    Get formality
    \return formality
*/
CString CForm::GetFormality() const
{
    return m_strFormality;
}
/*!
```

```
    Get alteration date
    \return alteration date
*/
CMyDate CForm::GetAlterationDate() const
{
    return m_dtAlterationDate;
}


/*!
    Set "proved" property
    \param bProved true, if form is proved
*/
void CForm::SetProved(bool bProved)
{
    m_bProved = bProved;
    // update alteration date
    HasChanged();
}
/*!
    Set "signed" property
    \param bSigned true, if form is signed
*/
void CForm::SetSigned(bool bSigned)
{
    m_bSigned = bSigned;
    // update alteration date
    HasChanged();
}


/*!
    Update date of alteration
*/
void CForm::HasChanged()
{
    m_dtAlterationDate = CMyDate();
}



// serializable
IMPLEMENT_SERIAL(CForm, CBankDocument, 1)

} // namespace Model
```

Dauerauftrag

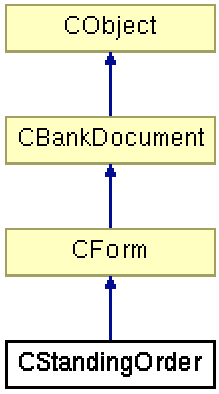

Abbildung 60: Vererbungshierarchie `CStandingOrder`

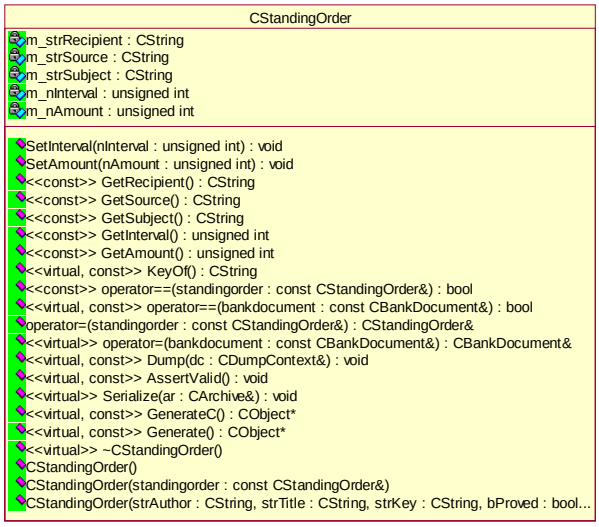

Abbildung 61: UML-Beschreibung `CStandingOrder`

Interface (StandingOrder.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  StandingOrder.h
    \brief class CStandingOrder (interface)
*/

#include "Form.h"
#pragma once

namespace Model
{

//! A standing order
/*!
    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li all CString attributes must be non-empty
    \li m_nAmount and m_nInterval must be positive (greater than zero)
    \li parent class must be valid
*/
```

```
//##ModelId=3B863B0200C8
class CStandingOrder : public CForm
{
public:
    //! default constructor
    //##ModelId=3B863B020147
    CStandingOrder();
    //! copy constructor
    //##ModelId=3B863B020148
    CStandingOrder(const CStandingOrder& standingorder);
    //! explicitly set attributes
    //##ModelId=3B863B020168
    CStandingOrder(CString strAuthor, CString strTitle, CString strKey,
                   bool    bProved,   bool    bSigned,  CString strFormality,
                   unsigned int nAmount, CString strSubject, CString strSource,
                   CString strRecipient, unsigned int nInterval);

    //! destructor
    //##ModelId=3B863B020145
    virtual ~CStandingOrder();

    //! create a new object
    //##ModelId=3B863B020143
    virtual CObject* Generate () const;
    //! clone this object
    //##ModelId=3B863B020141
    virtual CObject* GenerateC() const;
    //! serialization
    //##ModelId=3B863B02013E
    virtual void Serialize(CArchive &ar);
    //! validate
    //##ModelId=3B863B02013C
    virtual void AssertValid() const;
    //! dump
    //##ModelId=3B863B020139
    virtual void Dump(CDumpContext& dc) const;

    //! copy
    //##ModelId=3B863B02010C
            CStandingOrder& operator = (const CStandingOrder& standingorder);
    //! polymorphic copy
    //##ModelId=3B863B020136
    virtual CBankDocument& operator = (const CBankDocument& bankdocument);
    //! comparison
    //##ModelId=3B863B020106
            bool           operator ==(const CStandingOrder& standingorder) const;
    //! polymorphic comparison
    //##ModelId=3B863B020109
    virtual bool operator ==(const CBankDocument& bankdocument) const;

    //! deliver a unique identifier
    //##ModelId=3B863B020104
    virtual CString KeyOf() const;

    //! get amount
    //##ModelId=3B863B020102
    unsigned int GetAmount()    const;
    //! get interval
    //##ModelId=3B863B020100
    unsigned int GetInterval()  const;
    //! get subject
    //##ModelId=3B863B0200FE
    CString      GetSubject()   const;
    //! get source
    //##ModelId=3B863B0200FC
    CString      GetSource()    const;
    //! get recipient
    //##ModelId=3B863B0200FA
    CString      GetRecipient() const;

    //! set amount
    //##ModelId=3B863B0200CC
    void SetAmount  (unsigned int nAmount);
    //! set interval
    //##ModelId=3B863B0200CA
    void SetInterval(unsigned int nInterval);
```

```
private:
    //! amount each term
    //##ModelId=3B88C1AD01C6
    unsigned int m_nAmount;
    //! number of terms
    //##ModelId=3B88C1AD01C5
    unsigned int m_nInterval;
    //! subject to pay for
    //##ModelId=3B88C1AD01C4
    CString m_strSubject;
    //! person who pays
    //##ModelId=3B88C1AD01C3
    CString m_strSource;
    //! person that gets paid
    //##ModelId=3B88C1AD01C2
    CString m_strRecipient;


    DECLARE_SERIAL(CStandingOrder)
};

} // namespace Model
```

Implementation (StandingOrder.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  StandingOrder.cpp
    \brief class CStandingOrder (implementation)
*/

#include "stdafx.h"
#include "StandingOrder.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif


namespace Model
{

/*!
    Constructs a new and empty standing order
*/
CStandingOrder::CStandingOrder()
{
    m_nAmount = m_nInterval = 0;
}


/*!
    Copies an existing CStandingOrder
    \param standingorder standing order to be copied
    \see operator=
*/
CStandingOrder::CStandingOrder(const CStandingOrder& standingorder)
{
    operator=(standingorder);
}


/*!
    Constructs a new standing order using given attributes
    \param strAuthor    author
    \param strTitle     title
    \param strKey       key
    \param bProved      is it proved ?
    \param bSigned      is it signed ?
```

```
    \param strFormality formality
    \param nAmount      amount to be paid
    \param nInterval    numbers of terms
    \param strSubject   subject
    \param strSource    source
    \param strRecipient recipient
*/
CStandingOrder::CStandingOrder(CString strAuthor, CString strTitle, CString strKey,
                bool    bProved,   bool    bSigned,  CString strFormality,
                unsigned int nAmount, CString strSubject, CString strSource,
                CString strRecipient, unsigned int nInterval)
              : CForm(strAuthor, strTitle, strKey, bProved, bSigned, strFormality)
{
    m_nAmount      = nAmount;
    m_nInterval    = nInterval;
    m_strSubject   = strSubject;
    m_strSource    = strSource;
    m_strRecipient = strRecipient;
}


/*!
    No clean-up required
*/
CStandingOrder::~CStandingOrder()
{
}


/*!
    Create a new non-initialized object
    \return new instance of CStandingOrder
*/
CObject* CStandingOrder::Generate() const
{
    return new CStandingOrder;
}


/*!
    Clone this object
    \return exact copy
*/
CObject* CStandingOrder::GenerateC() const
{
    return new CStandingOrder(*this);
}


/*!
    Loads/stores CStandingOrder
    \param ar CArchive that provides data persistency
*/
void CStandingOrder::Serialize(CArchive &ar)
{
    // serialize parent
    CForm::Serialize(ar);

    if (ar.IsStoring())
        ar << m_nAmount << m_nInterval << m_strSubject << m_strSource << m_strRecipient;
    else
        ar >> m_nAmount >> m_nInterval >> m_strSubject >> m_strSource >> m_strRecipient;
}


/*!
    Determines whether the standing order is valid,
    \b invariant describes further details (see above)
*/
void CStandingOrder::AssertValid() const
{
    CForm::AssertValid();

    ASSERT(m_nAmount   > 0);
    ASSERT(m_nInterval > 0);
```

```
    ASSERT(!m_strSubject.  IsEmpty());
    ASSERT(!m_strSource.   IsEmpty());
    ASSERT(!m_strRecipient.IsEmpty());
}


/*!
    Create a dump of the class (only DEBUG version)
    \param dc output dump stream
*/
void CStandingOrder::Dump(CDumpContext& dc) const
{
    CForm::Dump(dc);

    CString strOutput;
    strOutput.Format("\namount: %d\ninterval: %d\nsubject: %s\nsource: %s\nrecipient: %s",
                     m_nAmount, m_nInterval, m_strSubject, m_strSource, m_strRecipient);

    dc << strOutput;
}


/*!
    Copies all attributes
    \param  standingorder CStandingOrder that should be copied
    \return reference to the copied standing order
*/
CStandingOrder& CStandingOrder::operator = (const CStandingOrder& standingorder)
{
    // copy inherited attributes
    CForm::operator =(standingorder);

    // copy CStandingOrder's attributes
    m_nAmount      = standingorder.m_nAmount;
    m_nInterval    = standingorder.m_nInterval;
    m_strSubject   = standingorder.m_strSubject;
    m_strSource    = standingorder.m_strSource;
    m_strRecipient = standingorder.m_strRecipient;

    return *this;
}


/*!
    Copies all attributes for polymorphic behaviour
    \param  bankdocument CStandingOrder (casted as CBankDocument) that should be copied
    \return reference to the copied form
*/
CBankDocument& CStandingOrder::operator = (const CBankDocument& bankdocument)
{
    try
    {
        // copy all attributes
        operator =(dynamic_cast<const CStandingOrder&>(bankdocument));
    }
    catch(...)
    {
        // invalid class
#ifdef _DEBUG
        TRACE("Copy denied because of invalid cast (needed a CStandingOrder)\n");
        Dump(afxDump);
        bankdocument.Dump(afxDump);
#endif
    }

    return *this;
}


/*!
    Compare attributes (except key) of two standing orders
    \param  form CStandingOrder that we compare to
    \return true, if all attributes are equal
*/
bool CStandingOrder::operator ==(const CStandingOrder& standingorder) const
{
```

```
    return (CForm::operator ==(standingorder) &&
            m_nAmount       == standingorder.m_nAmount &&
            m_nInterval     == standingorder.m_nInterval &&
            m_strSubject    == standingorder.m_strSubject &&
            m_strSource     == standingorder.m_strSource &&
            m_strRecipient  == standingorder.m_strRecipient);
}


/*!
    Compares all attributes for polymorphic behaviour
    \param  bankdocument CStandingOrder (casted as CBankDocument) that should be compared to
    \return true, if all attributes are equal
*/
bool CStandingOrder::operator ==(const CBankDocument& bankdocument) const
{
    try
    {
        // compare all attributes
        return operator ==(dynamic_cast<const CStandingOrder&>(bankdocument));
    }
    catch(...)
    {
        // invalid class
        TRACE("Comparison denied because of invalid cast\n"
              "(needed a CStandingOrder and got %s)\n", bankdocument.GetRuntimeClass()-
>m_lpszClassName);
#ifdef _DEBUG
        Dump(afxDump);
        bankdocument.Dump(afxDump);
#endif
    }

    return false;
}


/*!
    Responsible for the generation of a unique key based on the object's attributes
    \return CString that contains the key
*/
CString CStandingOrder::KeyOf() const
{
    CString strKey;
    strKey.Format("[%d|%d|%s|%s|%s|%s|%p]", m_nAmount, m_nInterval,
                                            m_strSubject, m_strSource, m_strRecipient,
                                            CForm::KeyOf(),
                                            this);
    return strKey;
}


/*!
    Get amount
    \return amount
*/
unsigned int CStandingOrder::GetAmount() const
{
    return m_nAmount;
}
/*!
    Get interval
    \return interval
*/
unsigned int CStandingOrder::GetInterval() const
{
    return m_nInterval;
}
/*!
    Get subject
    \return subject
*/
CString CStandingOrder::GetSubject() const
{
    return m_strSubject;
}
```

```
/*!
    Get source
    \return source
*/
CString CStandingOrder::GetSource() const
{
    return m_strSource;
}
/*!
    Get recipient
    \return recipient
*/
CString CStandingOrder::GetRecipient() const
{
    return m_strRecipient;
}


/*!
    Set amount
    \param nAmount amount
*/
void CStandingOrder::SetAmount(unsigned int nAmount)
{
    m_nAmount = nAmount;
    // update alteration date
    HasChanged();
}
/*!
    Set interval
    \param nInterval interval
*/
void CStandingOrder::SetInterval(unsigned int nInterval)
{
    m_nInterval = nInterval;
    // update alteration date
    HasChanged();
}



// serializable
IMPLEMENT_SERIAL(CStandingOrder, CForm, 1)

}
```

Vertrag

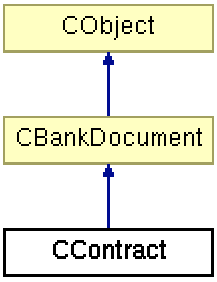

Abbildung 62: Vererbungshierarchie `CContract`

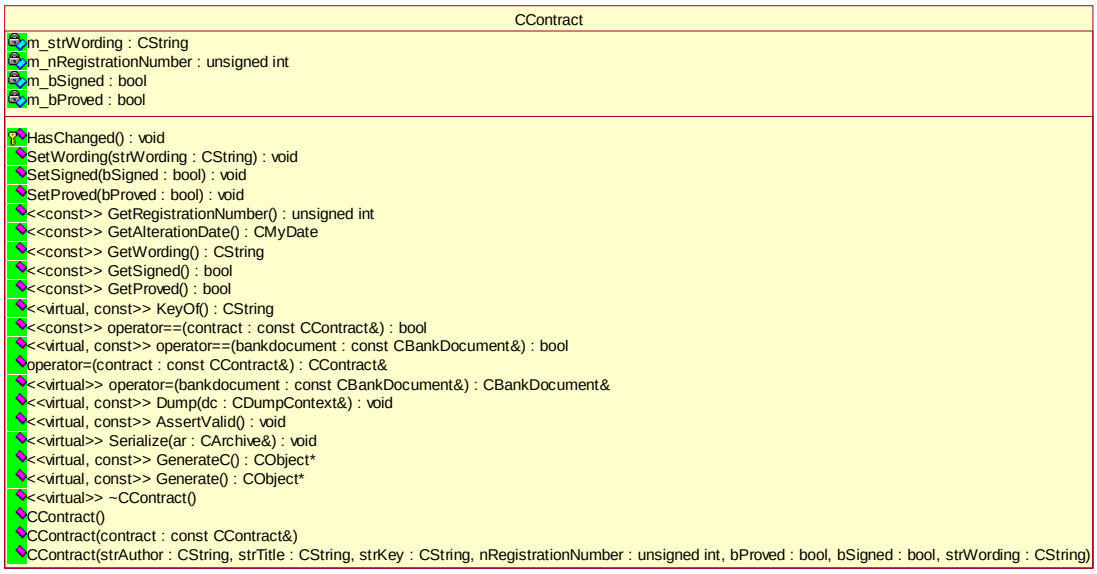

Abbildung 63: UML-Beschreibung von `CContract`

Interface (Contract.h)

```
////////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Contract.h
    \brief class CContract (interface)
*/

#include "BankDocument.h"
#pragma once

namespace Model
{

//! A contract
/*!
    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li m_nRegistrationNumber is positive
    \li m_dtAlterationDate must be valid
    \li m_strWording must be non-empty
    \li parent class must be valid
*/

//##ModelId=3B863AFB0370
class CContract : public CBankDocument
```

```
{
public:
    //! default constructor
    //##ModelId=3B863AFC0042
    CContract();
    //! copy constructor
    //##ModelId=3B863AFC0043
    CContract(const CContract& contract);
    //! explicitly set attributes
    //##ModelId=3B863AFC0065
    CContract(CString strAuthor, CString strTitle, CString strKey,
              unsigned int nRegistrationNumber, bool bProved, bool bSigned,
              CString strWording);

    //! destructor
    //##ModelId=3B863AFC0040
    virtual ~CContract();

    //! create a new object
    //##ModelId=3B863AFC003E
    virtual CObject* Generate () const;
    //! clone this object
    //##ModelId=3B863AFC003C
    virtual CObject* GenerateC() const;
    //! serialization
    //##ModelId=3B863AFC0039
    virtual void Serialize(CArchive &ar);
    //! validate the contract
    //##ModelId=3B863AFC0037
    virtual void AssertValid() const;
    //! dump
    //##ModelId=3B863AFC0034
    virtual void Dump(CDumpContext& dc) const;

    //! copy
    //##ModelId=3B863AFB03ED
            CContract&     operator = (const CContract& contract);
    //! polymorphic copy
    //##ModelId=3B863AFB03EF
    virtual CBankDocument& operator = (const CBankDocument& bankdocument);
    //! comparison
    //##ModelId=3B863AFB03E7
            bool operator ==(const CContract& contract) const;
    //! polymorphic comparison
    //##ModelId=3B863AFB03EA
    virtual bool operator ==(const CBankDocument& bankdocument) const;


    //! deliver a unique identifier
    //##ModelId=3B863AFB03E5
    virtual CString KeyOf() const;

    //! determine whether form is proved
    //##ModelId=3B863AFB03E3
    bool    GetProved()         const;
    //! determine whether form is signed
    //##ModelId=3B863AFB03E1
    bool    GetSigned()         const;
    //! get formality
    //##ModelId=3B863AFB03DF
    CString GetWording()        const;
    //! get alteration date
    //##ModelId=3B863AFB03B6
    CMyDate GetAlterationDate() const;
    //! get registration number
    //##ModelId=3B863AFB03B4
    unsigned int GetRegistrationNumber() const;

    //! set "proved" property
    //##ModelId=3B863AFB03B2
    void    SetProved(bool bProved);
    //! set "signed" property
    //##ModelId=3B863AFB03B0
    void    SetSigned(bool bSigned);
    //! set wording
    //##ModelId=3B863AFB03AE
```

```
    void    SetWording(CString strWording);

protected:
    //! at least one attribute has changed
    //##ModelId=3B863AFB03AD
    void HasChanged();

private:
    //! registration number
    //##ModelId=3B863AFB03AC
    unsigned int m_nRegistrationNumber;
    //! alteration date
    //##ModelId=3B863AFB0375
    CMyDate      m_dtAlterationDate;
    //! "proved" property
    //##ModelId=3B88C0F60303
    bool         m_bProved;
    //! "signed" property
    //##ModelId=3B88C0F60302
    bool         m_bSigned;
    //! wording
    //##ModelId=3B863AFB0372
    CString      m_strWording;



    DECLARE_SERIAL(CContract)
};

} // namespace Model
```

Implementation (Contract.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Contract.cpp
    \brief class CContract (implementation)
*/

#include "stdafx.h"
#include "Contract.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif


namespace Model
{

/*!
    Constructs a new unproved and unsigned contract
*/
CContract::CContract()
{
    m_bProved = m_bSigned = false;
    m_nRegistrationNumber = 0;
}


/*!
    Copies an existing CContract
    \param contract contract to be copied
    \see operator=
*/
CContract::CContract(const CContract& contract)
{
    operator=(contract);
}
```

```
//! explicitly set attributes
CContract::CContract(CString strAuthor, CString strTitle, CString strKey,
           unsigned int nRegistrationNumber, bool bProved, bool bSigned,
           CString strWording)
         : CBankDocument(strAuthor, strTitle, strKey)
{
    m_nRegistrationNumber = nRegistrationNumber;
    m_bProved = bProved;
    m_bSigned = bSigned;
    m_strWording = strWording;
    m_dtAlterationDate = CMyDate();
}


/*!
    No clean-up required
*/
CContract::~CContract()
{
}


/*!
    Create a new non-initialized object
    \return new instance of CContract
*/
CObject* CContract::Generate () const
{
    return new CContract;
}


/*!
    Clone this object
    \return exact copy
*/
CObject* CContract::GenerateC() const
{
    return new CContract(*this);
}


/*!
    Loads/stores CContract
    \param ar CArchive that provides data persistency
*/
void CContract::Serialize(CArchive &ar)
{
    // serialize parent
    CBankDocument::Serialize(ar);

    // CArchive can't handle "bool" so I have to use "BOOL" (which is "int" ...)
    BOOL bProved = m_bProved,
         bSigned = m_bSigned;

    if (ar.IsStoring())
        ar << bProved << bSigned << m_nRegistrationNumber << m_strWording;
    else
        ar >> bProved >> bSigned >> m_nRegistrationNumber >> m_strWording;

    // neccessary because of MFC's "bool!=BOOL story"
    m_bProved = (bProved == TRUE);
    m_bSigned = (bSigned == TRUE);

    // << and >> operators are not overloaded
    m_dtAlterationDate.Serialize(ar);
}


/*!
    Determines whether the contract is valid,
    \b invariant describes further details (see above)
*/
void CContract::AssertValid() const
{
    CBankDocument::AssertValid();
```

```
    ASSERT(!m_strWording.IsEmpty());
    ASSERT(m_nRegistrationNumber > 0);
    ASSERT_VALID(&m_dtAlterationDate);
}


/*!
    Create a dump of the class (only DEBUG version)
    \param dc output dump stream
*/
void CContract::Dump(CDumpContext& dc) const
{
    CBankDocument::Dump(dc);

    dc << "\n"
       << "registration number: " << m_nRegistrationNumber << "\n"
       << "proved: " << m_bProved << "\n"
       << "signed: " << m_bSigned << "\n"
       << "wording: " << m_strWording << "\n"
       << "alternation date: ";

    m_dtAlterationDate.Dump(dc);
}


/*!
    Copies all attributes
    \param  contract CContract that should be copied
    \return reference to the copied form
*/
CContract& CContract::operator = (const CContract& contract)
{
    // copy inherited attributes
    CBankDocument::operator =(contract);

    // copy CContract's attributes
    m_bProved = contract.m_bProved;
    m_bSigned = contract.m_bSigned;
    m_dtAlterationDate    = contract.m_dtAlterationDate;
    m_strWording          = contract.m_strWording;
    m_nRegistrationNumber = contract.m_nRegistrationNumber;

    return *this;
}


/*!
    Copies all attributes for polymorphic behaviour
    \param  bankdocument CContract (casted as CBankDocument) that should be copied
    \return reference to the copied form
*/
CBankDocument& CContract::operator = (const CBankDocument& bankdocument)
{
    try
    {
        // copy all attributes
        operator =(dynamic_cast<const CContract&>(bankdocument));
    }
    catch(...)
    {
        // invalid class
#ifdef _DEBUG
        TRACE("Copy denied because of invalid cast (needed a CContract)\n");
        Dump(afxDump);
        bankdocument.Dump(afxDump);
#endif
    }

    return *this;
}


/*!
    Compare attributes (except key) of two contracts
    \param  form CContract that we compare to
```

```
    \return true, if all attributes are equal
*/
bool CContract::operator ==(const CContract& contract) const
{
    return (CBankDocument::operator ==(contract) &&
            m_bProved == contract.m_bProved &&
            m_bSigned == contract.m_bSigned &&
//            m_dtAlterationDate    == contract.m_dtAlterationDate &&
            m_strWording          == contract.m_strWording &&
            m_nRegistrationNumber == contract.m_nRegistrationNumber);
}


/*!
    Compares all attributes for polymorphic behaviour
    \param  bankdocument CContract (casted as CBankDocument) that should be compared to
    \return true, if all attributes are equal
*/
bool CContract::operator ==(const CBankDocument& bankdocument) const
{
    try
    {
        // compare all attributes
        return operator ==(dynamic_cast<const CContract&>(bankdocument));
    }
    catch(...)
    {
        // invalid class
        TRACE("Comparison denied because of invalid cast\n"
              "(needed a CContract and got %s)\n", bankdocument.GetRuntimeClass()-
>m_lpszClassName);
#ifdef _DEBUG
        Dump(afxDump);
        bankdocument.Dump(afxDump);
#endif
    }

    return false;
}


/*!
    Responsible for the generation of a unique key based on the object's attributes
    \return CString that contains the key
*/
CString CContract::KeyOf() const
{
    CString strKey;
    strKey.Format("[%d|%s|%d|%d|%s|%s|%p]", m_nRegistrationNumber,
                                            m_strWording, m_bProved,  m_bSigned,
                                            m_dtAlterationDate.KeyOf(),
                                            CBankDocument::KeyOf(),
                                            this);
    return strKey;
}


/*!
    Get "proved" property
    \return true, if form is proved
*/
bool CContract::GetProved() const
{
    return m_bProved;
}
/*!
    Get "signed" property
    \return true, if form is signed
*/
bool CContract::GetSigned() const
{
    return m_bSigned;
}
/*!
    Get wording
    \return wording
```

```
*/
CString CContract::GetWording() const
{
    return m_strWording;
}
/*!
    Get alteration date
    \return alteration date
*/
CMyDate CContract::GetAlterationDate() const
{
    return m_dtAlterationDate;
}
/*!
    Get registration number
    \return registration number
*/
unsigned int CContract::GetRegistrationNumber() const
{
    return m_nRegistrationNumber;
}


/*!
    Set "proved" property
    \param bProved true, if form is proved
*/
void CContract::SetProved(bool bProved)
{
    m_bProved = bProved;
    // update alteration date
    HasChanged();
}
/*!
    Set "signed" property
    \param bSigned true, if form is signed
*/
void CContract::SetSigned(bool bSigned)
{
    m_bSigned = bSigned;
    // update alteration date
    HasChanged();
}
/*!
    Set wording
    \param strWording wording
*/
void CContract::SetWording(CString strWording)
{
    m_strWording = strWording;
    // update alteration date
    HasChanged();
}


/*!
    Update date of alteration
*/
void CContract::HasChanged()
{
    m_dtAlterationDate = CMyDate();
}


// serializable
IMPLEMENT_SERIAL(CContract, CBankDocument, 1)

} // namespace Model
```

View

CPraktikumMFCView

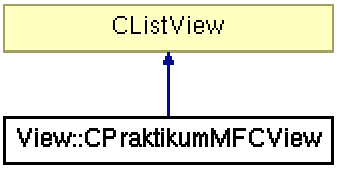

Abbildung 64: Vererbungshierarchie `CPraktikumMFCView`

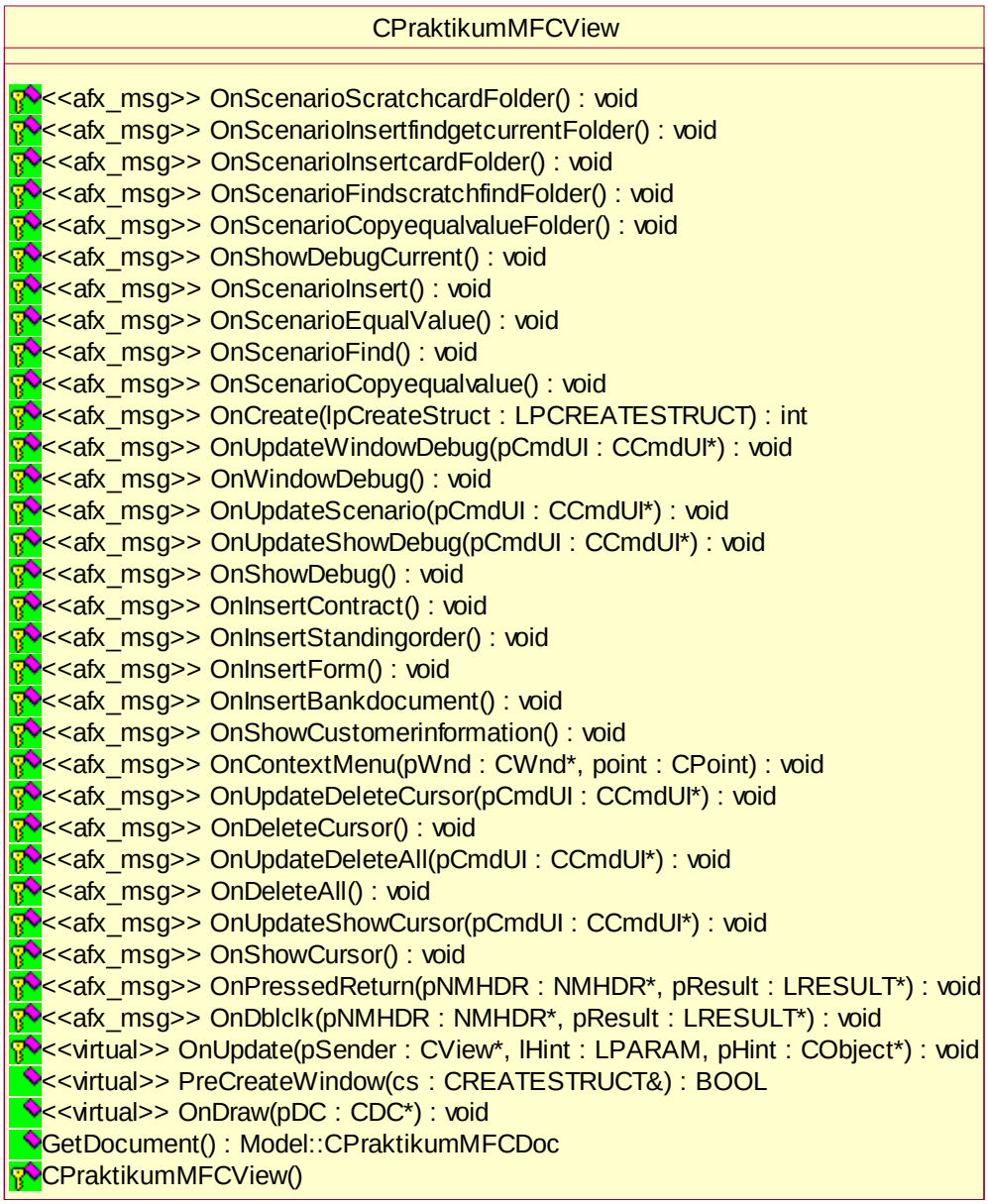

Abbildung 65: UML-Beschreibung von `CPraktikumMFCView`

Interface (CPraktikumMFCView.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  PraktikumMFCView.h
    \brief class CPraktikumMFCView (interface)
*/

#pragma once

//! View of the MVC concept
/*!
    Displays all neccessary parts of the Model
*/
namespace View
{

//! The main view
/*! \li view of the application

    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li (none)
*/

//##ModelId=3B863AFE01CC
class CPraktikumMFCView : public CListView
{
protected:
    //! default constructor
    /*! Nothing left to do */
    //##ModelId=3B863AFE02EA
    CPraktikumMFCView() {};

public:
    //! get attached document
    //##ModelId=3B863AFE02E9
    Model::CPraktikumMFCDoc* GetDocument();

    //{{AFX_VIRTUAL(CPraktikumMFCView)
    public:
    //##ModelId=3B863AFE02E6
    virtual void OnDraw(CDC* pDC);
    //##ModelId=3B863AFE02BE
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    //##ModelId=3B863AFE02B9
    virtual void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);
    //}}AFX_VIRTUAL

protected:

    //{{AFX_MSG(CPraktikumMFCView)
    //##ModelId=3B863AFE02B5
    afx_msg void OnDblclk(NMHDR* pNMHDR, LRESULT* pResult);
    //##ModelId=3B863AFE02B1
    afx_msg void OnPressedReturn(NMHDR* pNMHDR, LRESULT* pResult);
    //##ModelId=3B863AFE02AF
    afx_msg void OnShowCursor();
    //##ModelId=3B863AFE02AC
    afx_msg void OnUpdateShowCursor(CCmdUI* pCmdUI);
    //##ModelId=3B863AFE02AA
    afx_msg void OnDeleteAll();
    //##ModelId=3B863AFE0289
    afx_msg void OnUpdateDeleteAll(CCmdUI* pCmdUI);
    //##ModelId=3B863AFE0287
    afx_msg void OnDeleteCursor();
    //##ModelId=3B863AFE0284
    afx_msg void OnUpdateDeleteCursor(CCmdUI* pCmdUI);
    //##ModelId=3B863AFE0280
    afx_msg void OnContextMenu(CWnd* pWnd, CPoint point);
```

```
    //##ModelId=3B863AFE027E
    afx_msg void OnShowCustomerinformation();
    //##ModelId=3B863AFE027C
    afx_msg void OnInsertBankdocument();
    //##ModelId=3B863AFE027A
    afx_msg void OnInsertForm();
    //##ModelId=3B863AFE0278
    afx_msg void OnInsertStandingorder();
    //##ModelId=3B863AFE0276
    afx_msg void OnInsertContract();
    //##ModelId=3B863AFE024C
    afx_msg void OnShowDebug();
    //##ModelId=3B863AFE0249
    afx_msg void OnUpdateShowDebug(CCmdUI* pCmdUI);
    //##ModelId=3B863AFE0246
    afx_msg void OnUpdateScenario(CCmdUI* pCmdUI);
    //##ModelId=3B863AFE0244
    afx_msg void OnWindowDebug();
    //##ModelId=3B863AFE0241
    afx_msg void OnUpdateWindowDebug(CCmdUI* pCmdUI);
    //##ModelId=3B863AFE023E
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //##ModelId=3B863AFE023C
    afx_msg void OnScenarioCopyequalvalue();
    //##ModelId=3B863AFE023A
    afx_msg void OnScenarioFind();
    //##ModelId=3B863AFE0217
    afx_msg void OnScenarioEqualValue();
    //##ModelId=3B863AFE0215
    afx_msg void OnScenarioInsert();
    //##ModelId=3B863AFE0213
    afx_msg void OnShowDebugCurrent();
    //##ModelId=3B863AFE0211
    afx_msg void OnScenarioCopyequalvalueFolder();
    //##ModelId=3B863AFE020F
    afx_msg void OnScenarioFindscratchfindFolder();
    //##ModelId=3B863AFE020D
    afx_msg void OnScenarioInsertcardFolder();
    //##ModelId=3B863AFE020B
    afx_msg void OnScenarioInsertfindgetcurrentFolder();
    //##ModelId=3B863AFE0209
    afx_msg void OnScenarioScratchcardFolder();
    //}}AFX_MSG

    DECLARE_DYNCREATE(CPraktikumMFCView)
    DECLARE_MESSAGE_MAP()
};

/*!
    Return the document that is attached to current view
    \return pointer to the document
*/
inline Model::CPraktikumMFCDoc* CPraktikumMFCView::GetDocument()
{ return (Model::CPraktikumMFCDoc*)m_pDocument; }


} // namespace View
//{{AFX_INSERT_LOCATION}}
```

Implementierung (CPraktikumMFCView.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  PraktikumMFCView.cpp
    \brief class CPraktikumMFCView (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"
#include "PraktikumMFCDoc.h"
#include "PraktikumMFCView.h"

// access CFolder
```

```
#include "Folder.h"
// folder's elements are derived from CBankDocument
#include "BankDocument.h"
#include "Form.h"
#include "StandingOrder.h"
#include "Contract.h"

#include "Scenario.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


namespace View
{
    using namespace Model;

/*!
    Set default window style (report that is maximized)
    \param cs current window style
    \return true, if successful
    \see OnCreate
*/
BOOL CPraktikumMFCView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CListView::PreCreateWindow(cs))
        return FALSE;

    // change style to a list view
    cs.style &= ~LVS_TYPEMASK;
    cs.style |= LVS_REPORT;

    return TRUE;
}


/*!
    Set CListView's style and define columns
    \param lpCreateStruct passed to CWnd::OnCreate
    \return 0, if successful
    \see PreCreateWindow
*/
int CPraktikumMFCView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CListView::OnCreate(lpCreateStruct) == -1)
        return -1;

    // get list control of our list view
    CListCtrl& list = GetListCtrl();

    // set style
    list.ModifyStyle(0, LVS_SINGLESEL | LVS_SHOWSELALWAYS | LVS_NOSORTHEADER);
    list.SetExtendedStyle(list.GetExtendedStyle() |
                          LVS_EX_FULLROWSELECT | LVS_EX_GRIDLINES |
                          LVS_EX_FLATSB        | LVS_EX_TWOCLICKACTIVATE |
                          /*LVS_EX_LABELTIP*/ 0x00004000);

    list.GetHeaderCtrl()->ModifyStyle(HDS_BUTTONS, 0);

    // add needed columns
    list.InsertColumn(0, "Titel",    LVCFMT_LEFT, 150);
    list.InsertColumn(1, "Autor",    LVCFMT_LEFT, 100);
    list.InsertColumn(2, "Erstellt am",   LVCFMT_LEFT, 200);
    list.InsertColumn(3, "Geändert am",   LVCFMT_LEFT, 200);
    list.InsertColumn(4, "Typ",      LVCFMT_LEFT, 100);

    return 0;
}


/*!
    Redraw view (CListView draws itself properly, see OnUpdate)
    \param pDC device context that needs to be redrawn
```

```
    \see OnUpdate
*/
void CPraktikumMFCView::OnDraw(CDC* pDC)
{
    CPraktikumMFCDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
}


/*!
    Update view because of changed document
    \param pSender message source (not used)
    \param lHint parameter (not used)
    \param pHint parameter (not used)
*/
void CPraktikumMFCView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
{
    // call parent class
    CListView::OnUpdate(pSender, lHint, pHint);

    // get list control of our list view
    CListCtrl& list = GetListCtrl();
    // delete all entries
    list.DeleteAllItems();

    // get folder
    CFolder* pFolder = &(GetDocument()->m_Folder);
    // determine size
    int nCount = pFolder->Card();
    // disable and exit if empty
    if (nCount == 0)
    {
        list.EnableWindow(FALSE);
        return;
    }

    // enable window (white background instead of grey)
    list.EnableWindow(TRUE);

    // prepare control that we insert some items
    list.SetItemCount(nCount);

    // process all elements of the set
    for (int i=0; i<nCount; i++)
    {
        // get current element
        const CBankDocument* pBankDocument;
        if (i==0)
            pBankDocument = pFolder->GetFirst();
        else
            pBankDocument = pFolder->GetNext();

        // show title
        list.InsertItem(i, pBankDocument->GetTitle());
        // show author
        list.SetItemText(i, 1, pBankDocument->GetAuthor());
        // show date of foundation
        CString a = pBankDocument->GetDateOfFoundation();
        list.SetItemText(i, 2, (CString)pBankDocument->GetDateOfFoundation());

        // show type
        // first, determine class name
        CString strClassname = pBankDocument->GetRuntimeClass()->m_lpszClassName;
        if (strClassname == "CBankDocument")
            strClassname = "Bankdokument";
        else
            if (strClassname == "CForm")
            {
                strClassname = "Formular";
                list.SetItemText(i, 3,
                    (CString)((CForm*)pBankDocument)->GetAlterationDate());
            }
            else
                if (strClassname == "CStandingOrder")
                {
                    strClassname = "Dauerauftrag";
```

```
                    list.SetItemText(i, 3,
                        (CString)((CStandingOrder*)pBankDocument)->GetAlterationDate());
                }
                else
                    if (strClassname == "CContract")
                    {
                        strClassname = "Vertrag";
                        list.SetItemText(i, 3,
                            (CString)((CContract*)pBankDocument)->GetAlterationDate());
                    }
                    else
                        strClassname = "unbekannt ("+strClassname+")";

            // actually show type
            list.SetItemText(i, 4, strClassname);

            // delete copy fo the set‘s element
            delete pBankDocument;
        }

}


/*!
    Called on double-clicks
    \param pNMHDR (not used)
    \param pResult (see return value)
    \return NULL if successful
    \see CPraktikumMFCDoc::EditEntry
*/
void CPraktikumMFCView::OnDblclk(NMHDR* pNMHDR, LRESULT* pResult)
{
    // code taken from Jeff Prosise‘s "MFC Programming" book, p. 545 (German edition)

    // get mouse position
    DWORD dwPos = ::GetMessagePos();
    CPoint point ((int) LOWORD(dwPos), (int) HIWORD(dwPos));
    // convert from screen to window coordinates
    GetListCtrl().ScreenToClient(&point);

    // determine correspending list entry
    int nIndex = GetListCtrl().HitTest(point);

    // schnipp ! now my own code !
    if (nIndex != -1)
    {
        // edit list entry
        TRACE("%d clicked\n", nIndex);
        GetDocument()->EditEntry(nIndex);
    }

    // okay, this very last line was written by Prosise, too
    *pResult = 0;
}


/*!
    Show current entry (called if user pressed "Return")
    \param pNMHDR (not used)
    \param pResult (see return value)
    \return NULL if successful
    \see OnShowCursor, OnDblclk
*/
void CPraktikumMFCView::OnPressedReturn(NMHDR* pNMHDR, LRESULT* pResult)
{
    OnShowCursor();
    *pResult = 0;
}


/*!
    Show current entry
    \see OnPressedReturn, OnDblclk
*/
void CPraktikumMFCView::OnShowCursor()
{
```

```
    // cursor position
    int nSelected = GetListCtrl().GetSelectionMark();

    // only -1 if no document is selected
    if (nSelected != -1)
        GetDocument()->EditEntry(nSelected);
}


/*!
    Control menu entry that fires "OnShowCursor".
    Enable only if set is non-empty and list selection is valid
    \param pCmdUI menu entry
    \see OnShowCursor
*/
void CPraktikumMFCView::OnUpdateShowCursor(CCmdUI* pCmdUI)
{
    if (GetDocument()->m_Folder.Card() == 0 ||
        GetListCtrl().GetSelectionMark() == -1)
        pCmdUI->Enable(FALSE);
    else
        pCmdUI->Enable(TRUE);
}


/*!
    Delete the whole set
    \see OnUpdateDeleteAll, CPraktikumMFCDoc::DeleteAll
*/
void CPraktikumMFCView::OnDeleteAll()
{
    if (AfxMessageBox("Sind Sie sicher, dass Sie alle Dokumente löschen wollen ?",
                      MB_YESNO|MB_ICONQUESTION) == IDYES)
        GetDocument()->DeleteAll();
}


/*!
    Control menu entry that fires "OnDeleteAll".
    Enable only if set is non-empty
    \param pCmdUI menu entry
    \see OnDeleteAll
*/
void CPraktikumMFCView::OnUpdateDeleteAll(CCmdUI* pCmdUI)
{
    if (GetDocument()->m_Folder.Card() == 0)
        pCmdUI->Enable(FALSE);
    else
        pCmdUI->Enable(TRUE);
}


/*!
    Delete the current element
    \see OnUpdateDeleteCursor, CPraktikumMFCDoc::DeleteEntry
*/
void CPraktikumMFCView::OnDeleteCursor()
{
    int nSelected = GetListCtrl().GetSelectionMark();

    if (nSelected != -1)
        if (AfxMessageBox("Sind Sie sicher, dass Sie das gewählte Dokument löschen wollen ?",
                          MB_YESNO|MB_ICONQUESTION) == IDYES)
            GetDocument()->DeleteEntry(nSelected);
}


/*!
    Control menu entry that fires "OnDeleteCursor".
    Enable only if list selection is valid
    \param pCmdUI menu entry
    \see OnDeleteCursor
*/
void CPraktikumMFCView::OnUpdateDeleteCursor(CCmdUI* pCmdUI)
{
    if (GetListCtrl().GetSelectionMark() == -1)
```

```
        pCmdUI->Enable(FALSE);
    else
        pCmdUI->Enable(TRUE);
}


/*!
    Show context menu on right mouse button click
    \param pWnd window that requests the context menu
    \param point mouse position
*/
void CPraktikumMFCView::OnContextMenu(CWnd* pWnd, CPoint point)
{
    // call parent class
    CWnd::OnContextMenu(pWnd, point);

    // load menu
    CMenu menu;
    menu.LoadMenu(IDR_CONTEXTMENU);

    // extract context menu
    CMenu* pContextMenu = menu.GetSubMenu(0);
    // show it
    pContextMenu->TrackPopupMenu(TPM_LEFTALIGN | TPM_RIGHTBUTTON,
                                 point.x, point.y, AfxGetMainWnd());
}


/*!
    Insert a new bank document
    \see CPraktikumMFCDoc::InsertBankdocument
*/
void CPraktikumMFCView::OnInsertBankdocument()
{
    GetDocument()->InsertBankdocument();
}

/*!
    Insert a new form
    \see CPraktikumMFCDoc::InsertForm
*/
void CPraktikumMFCView::OnInsertForm()
{
    GetDocument()->InsertForm();
}

/*!
    Insert a new standing order
    \see CPraktikumMFCDoc::InsertStandingorder
*/
void CPraktikumMFCView::OnInsertStandingorder()
{
    GetDocument()->InsertStandingorder();
}

/*!
    Insert a new contract
    \see CPraktikumMFCDoc::InsertContract
*/
void CPraktikumMFCView::OnInsertContract()
{
    GetDocument()->InsertContract();
}

/*!
    Show the folder's properties
    \see CPraktikumMFCDoc::ShowCustomerinformation
*/
void CPraktikumMFCView::OnShowCustomerinformation()
{
    GetDocument()->ShowCustomerinformation();
}


/*!
    Create dump of the whole folder
```

```
    \see CPraktikumMFCDoc::ShowDebug
*/
void CPraktikumMFCView::OnShowDebug()
{
#ifdef _DEBUG
    GetDocument()->m_Folder.Dump(afxDump);
#endif
}


/*!
    Create dump of the currently selectly document
    \see OnShowDebug
*/
void CPraktikumMFCView::OnShowDebugCurrent()
{
#ifdef _DEBUG
    GetDocument()->m_Folder.SetCursor(GetListCtrl().GetSelectionMark());

    CBankDocument* bankdocument = GetDocument()->m_Folder.GetCurrent();
    bankdocument->Dump(afxDump);
    delete bankdocument;
#endif
}


/*!
    Enable/disable menu entry for OnShowDebug
    \param pCmdUI menu entry that should be modified
    \see OnShowDebug
*/
void CPraktikumMFCView::OnUpdateShowDebug(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(((Application::CPraktikumMFCApp*)AfxGetApp())->IsDebug());
}


/*!
    Open debug dialog
    \see CPraktikumMFCApp::OnShowDebug
*/
void CPraktikumMFCView::OnWindowDebug()
{
    ((Application::CPraktikumMFCApp*)AfxGetApp())->OnShowDebug();
}


/*!
    Enable/disable menu entry for OnWindowDebug
    \param pCmdUI menu entry that should be modified
    \see OnWindowDebug
*/
void CPraktikumMFCView::OnUpdateWindowDebug(CCmdUI* pCmdUI)
{
    pCmdUI->SetCheck(((Application::CPraktikumMFCApp*)AfxGetApp())->IsDebug());
}


/*!
    Run scenario Copy/EqualValue (document)
    \see Debug::CScenario::Copyequalvalue
*/
void CPraktikumMFCView::OnScenarioCopyequalvalue()
{
#ifdef _DEBUG
    Debug::CScenario::Copyequalvalue(GetDocument()->m_Folder,
GetListCtrl().GetSelectionMark());
#endif
}


/*!
    Run scenario Find
    \see Debug::CScenario::Find
*/
void CPraktikumMFCView::OnScenarioFind()
```

```
{
#ifdef _DEBUG
    Debug::CScenario::Find(GetDocument()->m_Folder, GetListCtrl().GetSelectionMark());
#endif
}


/*!
    Run scenario EqualValue
    \see Debug::CScenario::EqualValue
*/
void CPraktikumMFCView::OnScenarioEqualValue()
{
#ifdef _DEBUG
    Debug::CScenario::EqualValue(GetDocument()->m_Folder, GetListCtrl().GetSelectionMark());
#endif
}


/*!
    Run scenario Insert
    \see Debug::CScenario::Insert
*/
void CPraktikumMFCView::OnScenarioInsert()
{
#ifdef _DEBUG
    Debug::CScenario::Insert(GetDocument()->m_Folder, GetListCtrl().GetSelectionMark());
#endif
}


/*!
    Run scenario Copy/EqualValue (CFolder)
    \see Debug::CScenario::CopyequalvalueFolder
*/
void CPraktikumMFCView::OnScenarioCopyequalvalueFolder()
{
#ifdef _DEBUG
    Debug::CScenario::CopyequalvalueFolder(GetDocument()->m_Folder);
#endif
}


/*!
    Run scenario Find/Scratch/Find
    \see Debug::CScenario::FindscratchfindFolder
*/
void CPraktikumMFCView::OnScenarioFindscratchfindFolder()
{
#ifdef _DEBUG
    Debug::CScenario::FindscratchfindFolder(GetDocument()->m_Folder,
GetListCtrl().GetSelectionMark());
#endif
}


/*!
    Run scenario Insert/Card
    \see Debug::CScenario::InsertcardFolder
*/
void CPraktikumMFCView::OnScenarioInsertcardFolder()
{
#ifdef _DEBUG
    Debug::CScenario::InsertcardFolder(GetDocument()->m_Folder,
GetListCtrl().GetSelectionMark());
#endif
}


/*!
    Run scenario Insert/Find/GetCurrent
    \see Debug::CScenario::InsertfindgetcurrentFolder
*/
void CPraktikumMFCView::OnScenarioInsertfindgetcurrentFolder()
{
#ifdef _DEBUG
```

```
    Debug::CScenario::Insertfindgetcurrentfolder(GetDocument()->m_Folder,
GetListCtrl().GetSelectionMark());
#endif
}


/*!
    Run scenario Scratch/Card
    \see Debug::CScenario::ScratchcardFolder
*/
void CPraktikumMFCView::OnScenarioScratchcardFolder()
{
#ifdef _DEBUG
    Debug::CScenario::ScratchcardFolder(GetDocument()->m_Folder,
GetListCtrl().GetSelectionMark());
#endif
}


/*!
    Enable/disable menu entry for all scenarios
    \param pCmdUI menu entry that should be modified
    \see Debug::CScenario
*/
void CPraktikumMFCView::OnUpdateScenario(CCmdUI* pCmdUI)
{
#ifndef _DEBUG
    pCmdUI->Enable(FALSE);
#else
    pCmdUI->Enable(((Application::CPraktikumMFCApp*)AfxGetApp())->IsDebug() &&
                   GetListCtrl().GetSelectionMark() != -1);
#endif
}



BEGIN_MESSAGE_MAP(CPraktikumMFCView, CListView)
    //{{AFX_MSG_MAP(CPraktikumMFCView)
    ON_NOTIFY_REFLECT(NM_DBLCLK, OnDblclk)
    ON_NOTIFY_REFLECT(NM_RETURN, OnPressedReturn)
    ON_COMMAND(ID_SHOW_CURSOR, OnShowCursor)
    ON_UPDATE_COMMAND_UI(ID_SHOW_CURSOR, OnUpdateShowCursor)
    ON_COMMAND(ID_DELETE_ALL, OnDeleteAll)
    ON_UPDATE_COMMAND_UI(ID_DELETE_ALL, OnUpdateDeleteAll)
    ON_COMMAND(ID_DELETE_CURSOR, OnDeleteCursor)
    ON_UPDATE_COMMAND_UI(ID_DELETE_CURSOR, OnUpdateDeleteCursor)
    ON_WM_CONTEXTMENU()
    ON_COMMAND(ID_SHOW_CUSTOMERINFORMATION, OnShowCustomerinformation)
    ON_COMMAND(ID_INSERT_BANKDOCUMENT, OnInsertBankdocument)
    ON_COMMAND(ID_INSERT_FORM, OnInsertForm)
    ON_COMMAND(ID_INSERT_STANDINGORDER, OnInsertStandingorder)
    ON_COMMAND(ID_INSERT_CONTRACT, OnInsertContract)
    ON_COMMAND(ID_SHOW_DEBUG, OnShowDebug)
    ON_UPDATE_COMMAND_UI(ID_SHOW_DEBUG, OnUpdateShowDebug)
    ON_UPDATE_COMMAND_UI(ID_SCENARIO_COPYEQUALVALUE, OnUpdateScenario)
    ON_COMMAND(ID_WINDOW_DEBUG, OnWindowDebug)
    ON_UPDATE_COMMAND_UI(ID_WINDOW_DEBUG, OnUpdateWindowDebug)
    ON_WM_CREATE()
    ON_COMMAND(ID_SCENARIO_COPYEQUALVALUE, OnScenarioCopyequalvalue)
    ON_COMMAND(ID_SCENARIO_FIND, OnScenarioFind)
    ON_COMMAND(ID_SCENARIO_EQUALVALUE, OnScenarioEqualValue)
    ON_COMMAND(ID_SCENARIO_INSERT, OnScenarioInsert)
    ON_COMMAND(ID_SHOW_DEBUG_CURRENT, OnShowDebugCurrent)
    ON_UPDATE_COMMAND_UI(ID_SCENARIO_FIND, OnUpdateScenario)
    ON_UPDATE_COMMAND_UI(ID_SCENARIO_EQUALVALUE, OnUpdateScenario)
    ON_UPDATE_COMMAND_UI(ID_SCENARIO_INSERT, OnUpdateScenario)
    ON_UPDATE_COMMAND_UI(ID_SHOW_DEBUG_CURRENT, OnUpdateScenario)
    ON_UPDATE_COMMAND_UI(ID_SCENARIO_COPYEQUALVALUE_FOLDER, OnUpdateShowDebug)
    ON_UPDATE_COMMAND_UI(ID_SCENARIO_FINDSCRATCHFIND_FOLDER, OnUpdateScenario)
    ON_UPDATE_COMMAND_UI(ID_SCENARIO_INSERTFINDGETCURRENT_FOLDER, OnUpdateShowDebug)
    ON_UPDATE_COMMAND_UI(ID_SCENARIO_INSERTCARD_FOLDER, OnUpdateShowDebug)
    ON_UPDATE_COMMAND_UI(ID_SCENARIO_SCRATCHCARD_FOLDER, OnUpdateScenario)
    ON_COMMAND(ID_SCENARIO_COPYEQUALVALUE_FOLDER, OnScenarioCopyequalvalueFolder)
    ON_COMMAND(ID_SCENARIO_FINDSCRATCHFIND_FOLDER, OnScenarioFindscratchfindFolder)
    ON_COMMAND(ID_SCENARIO_INSERTCARD_FOLDER, OnScenarioInsertcardFolder)
    ON_COMMAND(ID_SCENARIO_INSERTFINDGETCURRENT_FOLDER, OnScenarioInsertfindgetcurrentFolder)
```

```
    ON_COMMAND(ID_SCENARIO_SCRATCHCARD_FOLDER, OnScenarioScratchcardFolder)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()


IMPLEMENT_DYNCREATE(CPraktikumMFCView, CListView)

} // namespace View
```

Karteidialog

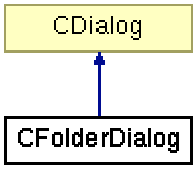

Abbildung 66: Vererbungshierarchie `CFolderDialog`

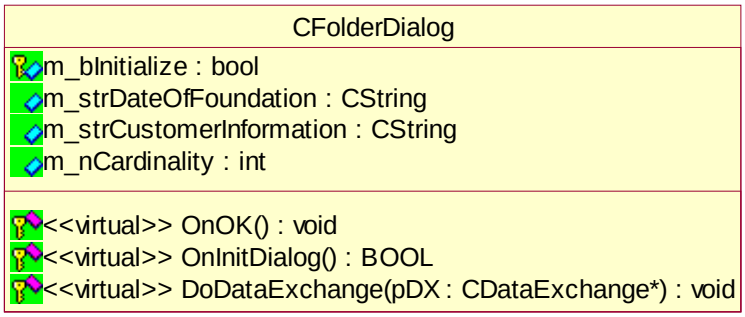

Abbildung 67: UML-Beschreibung von `CFolderDialog`

Interface (FolderDialog.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  FolderDialog.h
    \brief class CFolderDialog (interface)
*/

#pragma once

namespace View
{

//! A dialog to create/display CFolder
/*! \author         Stephan Brumme
    \date           August 11, 2001

    \invariant
    \li customer information must be non-empty
    \li invariant is verified by OnOK
*/

//##ModelId=3B863B020212
class CFolderDialog : public CDialog
{
public:
    //! constructor
    CFolderDialog(bool bInitialize = false, CWnd* pParent = NULL);

    //{{AFX_DATA(CFolderDialog)
    //! ID of the used dialog resource
    enum { IDD = IDD_FOLDER };
    //! value of "cardinality" control
    //##ModelId=3B863B020217
    int     m_nCardinality;
    //! value of "customer information" edit control
    //##ModelId=3B863B020216
    CString m_strCustomerInformation;
    //! value of "date of foundation" control
    //##ModelId=3B863B020215
    CString m_strDateOfFoundation;
    //}}AFX_DATA

    //{{AFX_VIRTUAL(CFolderDialog)
```

```
    protected:
    //! data exchange
    //##ModelId=3B863B020248
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL

protected:

    //##ModelId=3B863B020214
    bool m_bInitialize;

    //{{AFX_MSG(CFolderDialog)
    //! initialize dialog controls
    //##ModelId=3B863B020246
    virtual BOOL OnInitDialog();
    //! validate controls and close dialog
    //##ModelId=3B863B020244
    virtual void OnOK();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

} // namespace View

//{{AFX_INSERT_LOCATION}}
```

Implementierung (FolderDialog.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  FolderDialog.cpp
    \brief class CFolderDialog (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"
#include "FolderDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

namespace View
{

/*!
    Initialize all dialog controls and get dialog mode (create or display CFolder)
    \param bInitialize true, if used to create a new CFolder
    \param pParent handle of the parent window (not used)
*/
CFolderDialog::CFolderDialog(bool bInitialize, CWnd* pParent /*=NULL*/)
    : CDialog(CFolderDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CFolderDialog)
    m_nCardinality = 0;
    m_strCustomerInformation = _T("");
    m_strDateOfFoundation = _T("");
    //}}AFX_DATA_INIT

    m_bInitialize = bInitialize;
}


/*!
    Transfer data from the dialog controls to this class' member variables (and vice versa).
    Whole code is generated by wizards
    \param pDX data exchange object
    \see OnOK
*/
void CFolderDialog::DoDataExchange(CDataExchange* pDX)
{
```

```
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CFolderDialog)
    DDX_Text(pDX, IDC_EDIT_CARDINALITY, m_nCardinality);
    DDX_Text(pDX, IDC_EDIT_CUSTOMERINFORMATION, m_strCustomerInformation);
    DDX_Text(pDX, IDC_EDIT_DATEOFFOUNDATION, m_strDateOfFoundation);
    //}}AFX_DATA_MAP
}


/*!
    Enable/disable dialog controls to differ between creation and display mode
    \see CFolderDialog()
*/
BOOL CFolderDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    // controls are only writeable if used for construction of a new object
    if (m_bInitialize)
    {
        ((CEdit*)GetDlgItem(IDC_EDIT_CUSTOMERINFORMATION))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_CUSTOMERINFORMATION))->SetFocus();

        // disallow MFC to change the focus we set
        return FALSE;
    }

    // done, MFC may automatically set focus according to tab order
    return TRUE;
}


/*!
    Verify control‘s values and exit if all are valid
    \see DoDataExchange
*/
void CFolderDialog::OnOK()
{
    // data exchange
    UpdateData();

    // validate m_strCustomerInformation
    if (!m_strCustomerInformation.IsEmpty())
        CDialog::OnOK();
    else
        AfxMessageBox("Die Kundeninformation darf nicht leer sein.", MB_ICONSTOP);
}



BEGIN_MESSAGE_MAP(CFolderDialog, CDialog)
    //{{AFX_MSG_MAP(CFolderDialog)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

} // namespace View
```

Bankdokumentdialog

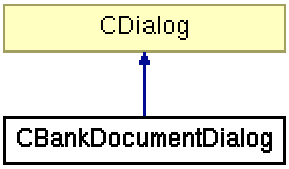

Abbildung 68: Vererbungshierarchie `CBankDocumentDialog`

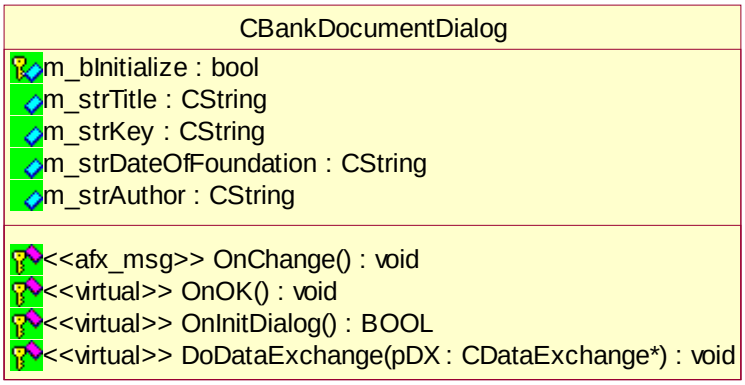

Abbildung 69: UML-Beschreibung von `CBankDocumentDialog`

Interface (BankDocumentDialog.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  BankDocumentDialog.h
    \brief class CBankDocumentDialog (interface)
*/

#pragma once

namespace View
{

//! A dialog to create/display CBankDocument
/*! \author         Stephan Brumme
    \date           August 11, 2001

    \invariant
    \li each CString attribute must be non-empty
    \li invariant is verified by OnOK
*/

//##ModelId=3B863B0202EE
class CBankDocumentDialog : public CDialog
{
public:
    //! constructor
    CBankDocumentDialog(bool bInitialize = false, CWnd* pParent = NULL);

    //{{AFX_DATA(CBankDocumentDialog)
    //! ID of the used dialog resource
    enum { IDD = IDD_BANKDOCUMENT };
    //! value of "author" edit control
    //##ModelId=3B863B020325
    CString m_strAuthor;
    //! value of "date of foundation" control
    //##ModelId=3B863B020324
    CString m_strDateOfFoundation;
    //! value of "key" edit control
    //##ModelId=3B863B020323
    CString m_strKey;
```

```
    //! value of "title" edit control
    //##ModelId=3B863B020322
    CString m_strTitle;
    //}}AFX_DATA

    //{{AFX_VIRTUAL(CBankDocumentDialog)
    protected:
    //! data exchange
    //##ModelId=3B863B02032C
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL

protected:

    //! true, if used to construct a new CBankDocument (enable more controls)
    //##ModelId=3B863B020321
    bool m_bInitialize;

    //{{AFX_MSG(CBankDocumentDialog)
    //##ModelId=3B863B02032A
    virtual BOOL OnInitDialog();
    //##ModelId=3B863B020328
    virtual void OnOK();
    //##ModelId=3B863B020326
    afx_msg void OnChange();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

} // namespace View
//{{AFX_INSERT_LOCATION}}
```

Implementierung (BankDocumentDialog.cpp)

```
////////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  BankDocumentDialog.cpp
    \brief class CBankDocumentDialog (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"
#include "BankDocumentDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

namespace View
{

/*!
    Initialize all dialog controls and get dialog mode (create or display CBankDocument)
    \param bInitialize true, if used to create a new CBankDocument
    \param pParent handle of the parent window (not used)
*/
CBankDocumentDialog::CBankDocumentDialog(bool bInitialize, CWnd* pParent /*=NULL*/)
    : CDialog(CBankDocumentDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CBankDocumentDialog)
    m_strAuthor = _T("");
    m_strDateOfFoundation = _T("");
    m_strKey = _T("");
    m_strTitle = _T("");
    //}}AFX_DATA_INIT

    m_bInitialize = bInitialize;
}


/*!
```

```
    Transfer data from the dialog controls to this class' member variables (and vice versa).
    Whole code is generated by wizards
    \param pDX data exchange object
    \see OnOK
*/
void CBankDocumentDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CBankDocumentDialog)
    DDX_Text(pDX, IDC_EDIT_AUTHOR, m_strAuthor);
    DDX_Text(pDX, IDC_EDIT_DATEOFFOUNDATION, m_strDateOfFoundation);
    DDX_Text(pDX, IDC_EDIT_KEY, m_strKey);
    DDX_Text(pDX, IDC_EDIT_TITLE, m_strTitle);
    //}}AFX_DATA_MAP
}


/*!
    Enable/disable dialog controls to differ between creation and display mode
    \see CBankDocumentDialog()
*/
BOOL CBankDocumentDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    // controls are only writeable if used for construction of a new object
    if (m_bInitialize)
    {
        ((CEdit*)GetDlgItem(IDC_EDIT_TITLE ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_AUTHOR))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_KEY   ))->SetReadOnly(FALSE);

        // set cursor to title edit control
        ((CEdit*)GetDlgItem(IDC_EDIT_TITLE ))->SetFocus();

        // disallow MFC to change the focus we set
        return FALSE;
    }

    // done, MFC may automatically set focus according to tab order
    return TRUE;
}


/*!
    Verify control's values and exit if all are valid
    \see DoDataExchange
*/
void CBankDocumentDialog::OnOK()
{
    // data exchange
    UpdateData();

    // validate m_strTitle
    if (m_strTitle.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Titel ein.");
        return;
    }

    // validate m_strAuthor
    if (m_strAuthor.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Bearbeiter ein.");
        return;
    }

    // validate m_strKey
    if (m_strKey.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Schlüssel ein.");
        return;
    }

    // done
    CDialog::OnOK();
```

```
}


/*!
    No user defined messages defined
*/
BEGIN_MESSAGE_MAP(CBankDocumentDialog, CDialog)
    //{{AFX_MSG_MAP(CBankDocumentDialog)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

} // namespace View
```

Formulardialog

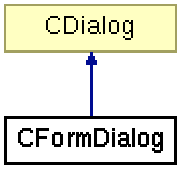

Abbildung 70: Vererbungshierarchie `CFormDialog`

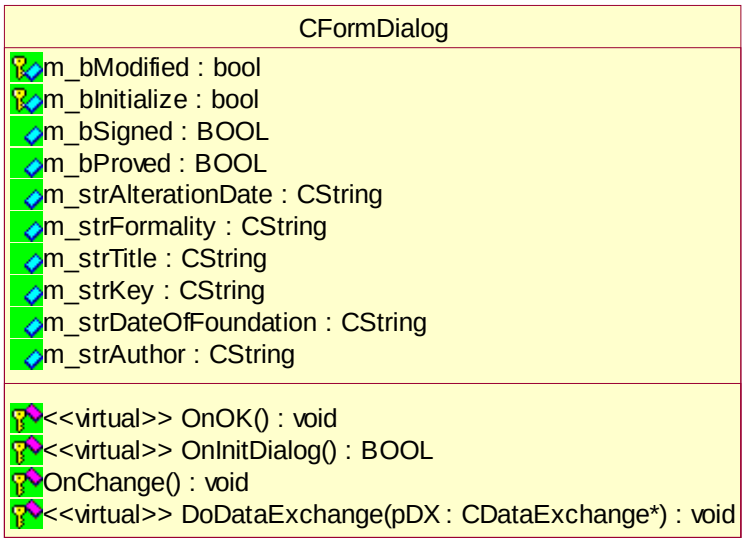

Abbildung 71: UML-Beschreibung von `CFormDialog`

Interface (FormDialog.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  FormDialog.h
    \brief class CFormDialog (interface)
*/

#pragma once

namespace View
{

//! A dialog to create/edit CForm
/*! \author         Stephan Brumme
    \date           August 11, 2001

    \invariant
    \li each CString attribute must be non-empty
    \li invariant is verified by OnOK
*/

//##ModelId=3B863B030050
class CFormDialog : public CDialog
{
public:
    //! constructor
    CFormDialog(bool bInitialize = false, CWnd* pParent = NULL);

    //{{AFX_DATA(CFormDialog)
    //! ID of the used dialog resource
    enum { IDD = IDD_FORM };
    //! value of "author" edit control
    //##ModelId=3B863B03008A
```

```
    CString m_strAuthor;
    //! value of "date of foundation"  control
    //##ModelId=3B863B030089
    CString m_strDateOfFoundation;
    //! value of "key" edit control
    //##ModelId=3B863B030088
    CString m_strKey;
    //! value of "title" edit control
    //##ModelId=3B863B030087
    CString m_strTitle;
    //! value of "formality" edit control
    //##ModelId=3B863B030086
    CString m_strFormality;
    //! value of "alteration date" control
    //##ModelId=3B863B030085
    CString m_strAlterationDate;
    //! value of "proved" checkbox control
    //##ModelId=3B863B030084
    BOOL    m_bProved;
    //! value of "signed" checkbox control
    //##ModelId=3B863B030083
    BOOL    m_bSigned;
    //}}AFX_DATA


    //{{AFX_VIRTUAL(CFormDialog)
    protected:
    //! data exchange
    //##ModelId=3B863B0300BE
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL

protected:

    //! true, if used to construct a new CForm (enable more controls)
    //##ModelId=3B863B030082
    bool m_bInitialize;
    //##ModelId=3B863B030052
    bool m_bModified;
    //##ModelId=3B863B03008F
    void OnChange();

    //{{AFX_MSG(CFormDialog)
    //! initialize dialog controls
    //##ModelId=3B863B03008D
    virtual BOOL OnInitDialog();
    //! validate controls and close dialog
    //##ModelId=3B863B03008B
    virtual void OnOK();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};


} // namespace View
//{{AFX_INSERT_LOCATION}}
```

Implementierung (FormDialog.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  FormDialog.cpp
    \brief class CFormDialog (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"
#include "FormDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
```

```
#endif


namespace View
{

/*!
    Initialize all dialog controls and get dialog mode (create or edit CForm)
    \param bInitialize true, if used to create a new CForm
    \param pParent handle of the parent window (not used)
*/
CFormDialog::CFormDialog(bool bInitialize, CWnd* pParent /*=NULL*/)
    : CDialog(CFormDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CFormDialog)
    m_strAuthor = _T("");
    m_strDateOfFoundation = _T("");
    m_strKey = _T("");
    m_strTitle = _T("");
    m_strFormality = _T("");
    m_strAlterationDate = _T("");
    m_bProved = FALSE;
    m_bSigned = FALSE;
    //}}AFX_DATA_INIT

    m_bInitialize = bInitialize;
    m_bModified   = false;
}


/*!
    Transfer data from the dialog controls to this class' member variables (and vice versa).
    Whole code is generated by wizards
    \param pDX data exchange object
    \see OnOK
*/
void CFormDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CFormDialog)
    DDX_Text(pDX, IDC_EDIT_AUTHOR, m_strAuthor);
    DDX_Text(pDX, IDC_EDIT_DATEOFFOUNDATION, m_strDateOfFoundation);
    DDX_Text(pDX, IDC_EDIT_KEY, m_strKey);
    DDX_Text(pDX, IDC_EDIT_TITLE, m_strTitle);
    DDX_Text(pDX, IDC_EDIT_FORMALITY, m_strFormality);
    DDX_Text(pDX, IDC_EDIT_ALTERATIONDATE, m_strAlterationDate);
    DDX_Check(pDX, IDC_CHECK_PROVED, m_bProved);
    DDX_Check(pDX, IDC_CHECK_SIGNED, m_bSigned);
    //}}AFX_DATA_MAP
}


/*!
    Enable/disable dialog controls to differ between creation and edit mode
    \see CFormDialog()
*/
BOOL CFormDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    InitializeFlatSB      (GetDlgItem(IDC_EDIT_FORMALITY)->m_hWnd);
    FlatSB_EnableScrollBar(GetDlgItem(IDC_EDIT_FORMALITY)->m_hWnd, SB_VERT, 0);

    // controls are only writeable if used for construction of a new object
    if (m_bInitialize)
    {
        ((CEdit*)GetDlgItem(IDC_EDIT_TITLE    ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_AUTHOR   ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_KEY      ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_FORMALITY))->SetReadOnly(FALSE);

        // set cursor to title edit control
        ((CEdit*)GetDlgItem(IDC_EDIT_TITLE ))->SetFocus();

        // disallow MFC to change the focus we set
        return FALSE;
```

```
    }

    // done, MFC may automatically set focus according to tab order
    return TRUE;
}


/*!
    Verify control's values and exit if all are valid
    \see DoDataExchange
*/
void CFormDialog::OnOK()
{
    // data exchange
    UpdateData();

    // validate m_strTitle
    if (m_strTitle.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Titel ein.");
        return;
    }

    // validate m_strAuthor
    if (m_strAuthor.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Bearbeiter ein.");
        return;
    }

    // validate m_strKey
    if (m_strKey.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Schlüssel ein.");
        return;
    }

    // validate m_strFormality
    if (m_strFormality.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie eine Formalität ein.");
        return;
    }

    // done
    if (m_bModified)
        CDialog::OnOK();
    else
        CDialog::OnCancel();
}

void CFormDialog::OnChange()
{
    m_bModified = true;
}


BEGIN_MESSAGE_MAP(CFormDialog, CDialog)
    //{{AFX_MSG_MAP(CFormDialog)
    ON_BN_CLICKED(IDC_CHECK_PROVED, OnChange)
    ON_BN_CLICKED(IDC_CHECK_SIGNED, OnChange)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

} // namespace View
```

Dauerauftragdialog

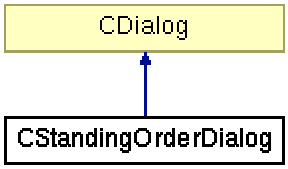

Abbildung 72: Vererbungshierarchie `CStandingOrderDialog`

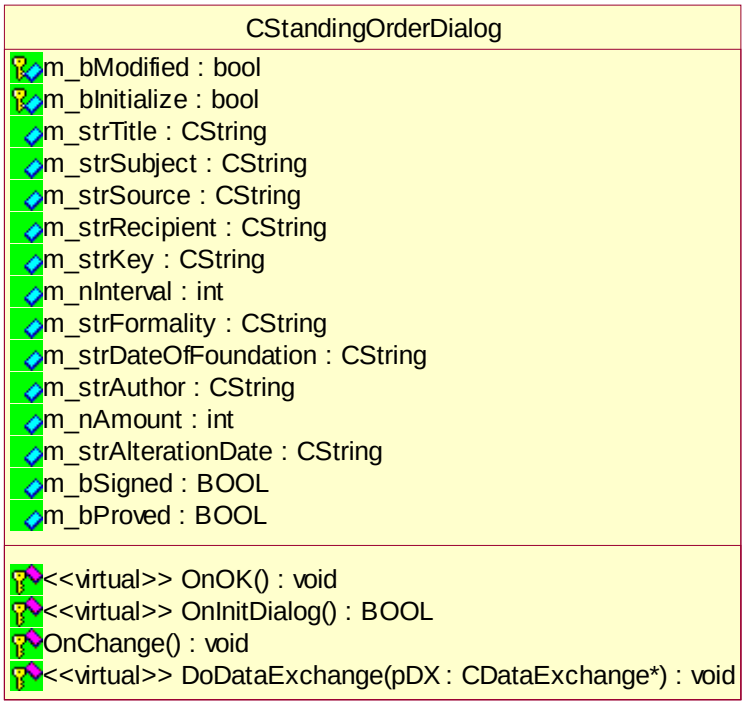

Abbildung 73: UML-Beschreibung von `CStandingOrderDialog`

Interface (StandingOrderDialog.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  StandingOrderDialog.h
    \brief class CStandingOrderDialog (interface)
*/

#pragma once

namespace View
{

//! A dialog to create/edit CStandingOrder
/*! \author         Stephan Brumme
    \date           August 11, 2001

    \invariant
    \li each CString attribute must be non-empty
    \li m_nAmount and m_nInterval must be positive
    \li invariant is verified by OnOK
*/

//##ModelId=3B863B0502C6
class CStandingOrderDialog : public CDialog
{
public:
```

```
    //! constructor
    CStandingOrderDialog(bool bInitialize = false, CWnd* pParent = NULL);

    //{{AFX_DATA(CStandingOrderDialog)
    //! ID of the used dialog resource
    enum { IDD = IDD_STANDINGORDER };
    //! value of "proved" checkbox control
    //##ModelId=3B863B050335
    BOOL    m_bProved;
    //! value of "signed" checkbox control
    //##ModelId=3B863B050334
    BOOL    m_bSigned;
    //! value of "alteration date" control
    //##ModelId=3B863B05030D
    CString m_strAlterationDate;
    //! value of "amount" edit control
    //##ModelId=3B863B05030C
    int     m_nAmount;
    //! value of "author" edit control
    //##ModelId=3B863B05030B
    CString m_strAuthor;
    //! value of "date of foundation"  control
    //##ModelId=3B863B05030A
    CString m_strDateOfFoundation;
    //! value of "formality" edit control
    //##ModelId=3B863B050309
    CString m_strFormality;
    //! value of "interval" edit control
    //##ModelId=3B863B050308
    int     m_nInterval;
    //! value of "key" edit control
    //##ModelId=3B863B050307
    CString m_strKey;
    //! value of "recipient" edit control
    //##ModelId=3B863B050306
    CString m_strRecipient;
    //! value of "source" edit control
    //##ModelId=3B863B050305
    CString m_strSource;
    //! value of "subject" edit control
    //##ModelId=3B863B050304
    CString m_strSubject;
    //! value of "title" edit control
    //##ModelId=3B863B050303
    CString m_strTitle;
    //}}AFX_DATA


    //{{AFX_VIRTUAL(CStandingOrderDialog)
    protected:
    //! data exchange
    //##ModelId=3B863B05033B
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL

protected:

    //! true, if used to construct a new CStandingOrder (enable more controls)
    //##ModelId=3B863B050302
    bool m_bInitialize;
    //##ModelId=3B863B0502C8
    bool m_bModified;
    //##ModelId=3B863B05033A
    void OnChange();

    //{{AFX_MSG(CStandingOrderDialog)
    //! initialize dialog controls
    //##ModelId=3B863B050338
    virtual BOOL OnInitDialog();
    //! validate controls and close dialog
    //##ModelId=3B863B050336
    virtual void OnOK();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

```
} // namespace View
//{{AFX_INSERT_LOCATION}}
```

Implementierung (StandingOrderDialog.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  StandingOrderDialog.cpp
    \brief class CStandingOrderDialog (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"
#include "StandingOrderDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

namespace View
{

/*!
    Initialize all dialog controls and get dialog mode (create or edit CStandingOrder)
    \param bInitialize true, if used to create a new CStandingOrder
    \param pParent handle of the parent window (not used)
*/
CStandingOrderDialog::CStandingOrderDialog(bool bInitialize, CWnd* pParent /*=NULL*/)
    : CDialog(CStandingOrderDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CStandingOrderDialog)
    m_bProved = FALSE;
    m_bSigned = FALSE;
    m_strAlterationDate = _T("");
    m_nAmount = 0;
    m_strAuthor = _T("");
    m_strDateOfFoundation = _T("");
    m_strFormality = _T("");
    m_nInterval = 0;
    m_strKey = _T("");
    m_strRecipient = _T("");
    m_strSource = _T("");
    m_strSubject = _T("");
    m_strTitle = _T("");
    //}}AFX_DATA_INIT

    m_bInitialize = bInitialize;
    m_bModified   = false;
}


/*!
    Transfer data from the dialog controls to this class' member variables (and vice versa).
    Whole code is generated by wizards
    \param pDX data exchange object
    \see OnOK
*/
void CStandingOrderDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CStandingOrderDialog)
    DDX_Check(pDX, IDC_CHECK_PROVED, m_bProved);
    DDX_Check(pDX, IDC_CHECK_SIGNED, m_bSigned);
    DDX_Text(pDX, IDC_EDIT_ALTERATIONDATE, m_strAlterationDate);
    DDX_Text(pDX, IDC_EDIT_AMOUNT, m_nAmount);
    DDX_Text(pDX, IDC_EDIT_AUTHOR, m_strAuthor);
    DDX_Text(pDX, IDC_EDIT_DATEOFFOUNDATION, m_strDateOfFoundation);
    DDX_Text(pDX, IDC_EDIT_FORMALITY, m_strFormality);
    DDX_Text(pDX, IDC_EDIT_INTERVAL, m_nInterval);
    DDX_Text(pDX, IDC_EDIT_KEY, m_strKey);
    DDX_Text(pDX, IDC_EDIT_RECIPIENT, m_strRecipient);
```

```
    DDX_Text(pDX, IDC_EDIT_SOURCE, m_strSource);
    DDX_Text(pDX, IDC_EDIT_SUBJECT, m_strSubject);
    DDX_Text(pDX, IDC_EDIT_TITLE, m_strTitle);
    //}}AFX_DATA_MAP
}



/*!
    Enable/disable dialog controls to differ between creation and edit mode
    \see CStandingOrderDialog()
*/
BOOL CStandingOrderDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    // controls are only writeable if used for construction of a new object
    if (m_bInitialize)
    {
        ((CEdit*)GetDlgItem(IDC_EDIT_TITLE    ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_AUTHOR   ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_KEY      ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_FORMALITY))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_SOURCE   ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_RECIPIENT))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_SUBJECT  ))->SetReadOnly(FALSE);

        // set cursor to title edit control
        ((CEdit*)GetDlgItem(IDC_EDIT_TITLE ))->SetFocus();

        // disallow MFC to change the focus we set
        return FALSE;
    }

    // done, MFC may automatically set focus according to tab order
    return TRUE;
}


/*!
    Verify control's values and exit if all are valid
    \see DoDataExchange
*/
void CStandingOrderDialog::OnOK()
{
    // data exchange
    UpdateData();

    // validate m_strTitle
    if (m_strTitle.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Titel ein.");
        return;
    }

    // validate m_strAuthor
    if (m_strAuthor.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Bearbeiter ein.");
        return;
    }

    // validate m_strKey
    if (m_strKey.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Schlüssel ein.");
        return;
    }

    // validate m_strFormality
    if (m_strFormality.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie eine Formalität ein.");
        return;
    }
```

```
    // validate m_strSource
    if (m_strSource.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Einzahler ein.");
        return;
    }

    // validate m_strRecipient
    if (m_strRecipient.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Empfänger ein.");
        return;
    }

    // validate m_strSubject
    if (m_strSubject.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Betreff ein.");
        return;
    }

    // validate m_nAmount
    if (m_nAmount <= 0)
    {
        AfxMessageBox("Bitte geben Sie eine positive Summe ein.");
        return;
    }

    // validate m_nInterval
    if (m_nInterval <= 0)
    {
        AfxMessageBox("Bitte geben Sie einen positiven Intervall ein.");
        return;
    }

    // done
    if (m_bModified)
        CDialog::OnOK();
    else
        CDialog::OnCancel();
}

void CStandingOrderDialog::OnChange()
{
    m_bModified = true;
}


BEGIN_MESSAGE_MAP(CStandingOrderDialog, CDialog)
    //{{AFX_MSG_MAP(CStandingOrderDialog)
    ON_BN_CLICKED(IDC_CHECK_PROVED, OnChange)
    ON_BN_CLICKED(IDC_CHECK_SIGNED, OnChange)
    ON_EN_CHANGE(IDC_EDIT_AMOUNT, OnChange)
    ON_EN_CHANGE(IDC_EDIT_INTERVAL, OnChange)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

} // namespace View
```

Vertragsdialog

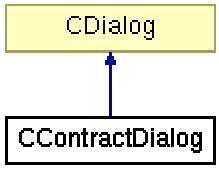

Abbildung 74: Vererbungshierarchie `CContractDialog`

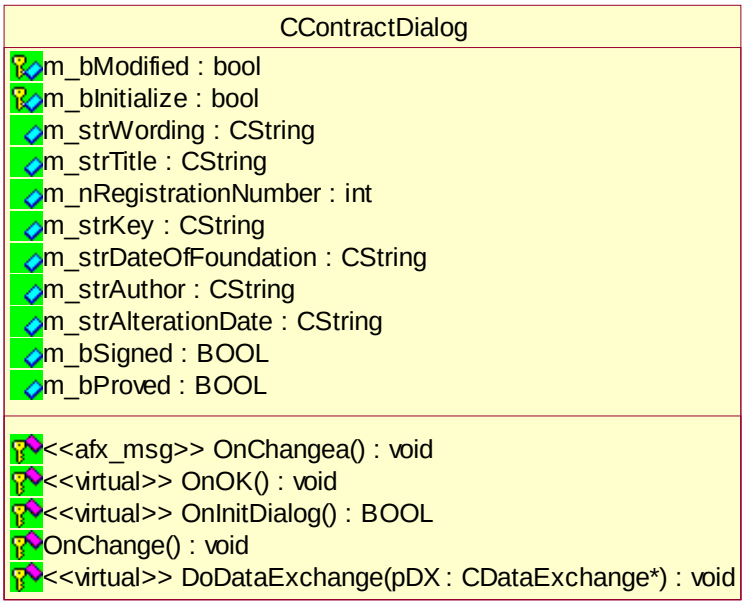

Abbildung 75: UML-Beschreibung von `CContractDialog`

Interface (ContractDialog.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  ContractDialog.h
    \brief class CContractDialog (interface)
*/

#pragma once

namespace View
{

//! A dialog to create/edit CContract
/*! \author         Stephan Brumme
    \date           August 11, 2001

    \invariant
    \li each CString attribute must be non-empty
    \li registration number must be positive
    \li invariant is verified by OnOK
*/

//##ModelId=3B863B0301CC
class CContractDialog : public CDialog
{
public:
    //! constructor
    CContractDialog(bool bInitialize = false, CWnd* pParent = NULL);

    //{{AFX_DATA(CContractDialog)
```

```
    //! ID of the used dialog resource
    enum { IDD = IDD_CONTRACT };
    //! value of "proved" checkbox control
    //##ModelId=3B863B03020D
    BOOL    m_bProved;
    //! value of "signed" checkbox control
    //##ModelId=3B863B03020C
    BOOL    m_bSigned;
    //! value of "alteration date" control
    //##ModelId=3B863B03020B
    CString m_strAlterationDate;
    //! value of "author" edit control
    //##ModelId=3B863B03020A
    CString m_strAuthor;
    //! value of "date of foundation" control
    //##ModelId=3B863B030209
    CString m_strDateOfFoundation;
    //! value of "key" edit control
    //##ModelId=3B863B030208
    CString m_strKey;
    //! value of "registration" edit control
    //##ModelId=3B863B0301D2
    int     m_nRegistrationNumber;
    //! value of "title" edit control
    //##ModelId=3B863B0301D1
    CString m_strTitle;
    //! value of "wording" edit control
    //##ModelId=3B863B0301D0
    CString m_strWording;
    //}}AFX_DATA


    //{{AFX_VIRTUAL(CContractDialog)
    protected:
    //! data exchange
    //##ModelId=3B863B030215
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL

protected:

    //! true, if used to construct a new CContract (enable more controls)
    //##ModelId=3B863B0301CF
    bool m_bInitialize;
    //##ModelId=3B863B0301CE
    bool m_bModified;
    //##ModelId=3B863B030214
    void OnChange();

    //{{AFX_MSG(CContractDialog)
    //##ModelId=3B863B030212
    virtual BOOL OnInitDialog();
    //##ModelId=3B863B030210
    virtual void OnOK();
    //##ModelId=3B863B03020E
    afx_msg void OnChangea();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

} // namespace View

//{{AFX_INSERT_LOCATION}}
```

Implementierung (ContractDialog.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  ContractDialog.cpp
    \brief class CContractDialog (implementation)
*/

#include "stdafx.h"
```

```
#include "PraktikumMFC.h"
#include "ContractDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

namespace View
{

/*!
    Initialize all dialog controls and get dialog mode (create or edit CContract)
    \param bInitialize true, if used to create a new CContract
    \param pParent handle of the parent window (not used)
*/
CContractDialog::CContractDialog(bool bInitialize, CWnd* pParent /*=NULL*/)
    : CDialog(CContractDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CContractDialog)
    m_bProved = FALSE;
    m_bSigned = FALSE;
    m_strAlterationDate = _T("");
    m_strAuthor = _T("");
    m_strDateOfFoundation = _T("");
    m_strKey = _T("");
    m_nRegistrationNumber = 0;
    m_strTitle = _T("");
    m_strWording = _T("");
    //}}AFX_DATA_INIT

    m_bInitialize = bInitialize;
    m_bModified   = false;
}


/*!
    Transfer data from the dialog controls to this class' member variables (and vice versa).
    Whole code is generated by wizards
    \param pDX data exchange object
    \see OnOK
*/
void CContractDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CContractDialog)
    DDX_Check(pDX, IDC_CHECK_PROVED, m_bProved);
    DDX_Check(pDX, IDC_CHECK_SIGNED, m_bSigned);
    DDX_Text(pDX, IDC_EDIT_ALTERATIONDATE, m_strAlterationDate);
    DDX_Text(pDX, IDC_EDIT_AUTHOR, m_strAuthor);
    DDX_Text(pDX, IDC_EDIT_DATEOFFOUNDATION, m_strDateOfFoundation);
    DDX_Text(pDX, IDC_EDIT_KEY, m_strKey);
    DDX_Text(pDX, IDC_EDIT_REGISTRATIONNUMBER, m_nRegistrationNumber);
    DDX_Text(pDX, IDC_EDIT_TITLE, m_strTitle);
    DDX_Text(pDX, IDC_EDIT_WORDING, m_strWording);
    //}}AFX_DATA_MAP
}


/*!
    Enable/disable dialog controls to differ between creation and edit mode
    \see CContractDialog()
*/
BOOL CContractDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    // controls are only writeable if used for construction of a new object
    if (m_bInitialize)
    {
        ((CEdit*)GetDlgItem(IDC_EDIT_TITLE    ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_AUTHOR   ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_KEY      ))->SetReadOnly(FALSE);
        ((CEdit*)GetDlgItem(IDC_EDIT_REGISTRATIONNUMBER))->SetReadOnly(FALSE);
```

```
        // set cursor to title edit control
        ((CEdit*)GetDlgItem(IDC_EDIT_TITLE ))->SetFocus();

        // disallow MFC to change the focus we set
        return FALSE;
    }

    // done, MFC may automatically set focus according to tab order
    return TRUE;
}


/*!
    Verify control's values and exit if all are valid
    \see DoDataExchange
*/
void CContractDialog::OnOK()
{
    // data exchange
    UpdateData();

    // validate m_strTitle
    if (m_strTitle.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Titel ein.");
        return;
    }

    // validate m_strAuthor
    if (m_strAuthor.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Bearbeiter ein.");
        return;
    }

    // validate m_strKey
    if (m_strKey.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie einen Schlüssel ein.");
        return;
    }

    // validate m_strWording
    if (m_strWording.IsEmpty())
    {
        AfxMessageBox("Bitte geben Sie eine Formalität ein.");
        return;
    }

    // validate m_nRegistrationNumber
    if (m_nRegistrationNumber <= 0)
    {
        AfxMessageBox("Bitte geben Sie eine positive Registeriernummer ein.");
        return;
    }

    // done
    if (m_bModified)
        CDialog::OnOK();
    else
        CDialog::OnCancel();
}

void CContractDialog::OnChange()
{
    m_bModified = true;
}


BEGIN_MESSAGE_MAP(CContractDialog, CDialog)
    //{{AFX_MSG_MAP(CContractDialog)
    ON_BN_CLICKED(IDC_CHECK_PROVED, OnChange)
    ON_BN_CLICKED(IDC_CHECK_SIGNED, OnChange)
    ON_EN_CHANGE(IDC_EDIT_WORDING, OnChange)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
} // namespace View
```

Infodialog

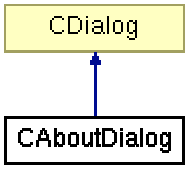

Abbildung 76: Vererbungshierarchie `CAboutDialog`

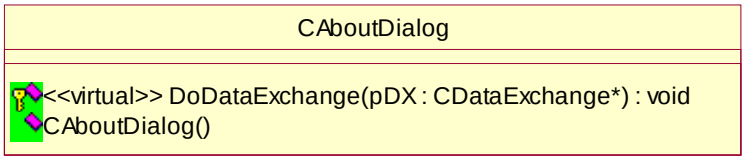

Abbildung 77: UML-Beschreibung von `CAboutDialog`

Interface (AboutDialog.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  AboutDialog.h
    \brief class CAboutDialog (interface)
*/

#pragma once

namespace View
{

//! A dialog that displays some general info
/*! \author         Stephan Brumme
    \date           August 11, 2001

    \invariant
    \li (none)
*/

//##ModelId=3B863B04037A
class CAboutDialog : public CDialog
{
public:
    //! constructor
    //##ModelId=3B863B04037F
    CAboutDialog();

    //{{AFX_DATA(CAboutDialog)
    //! ID of the used dialog resource
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    //{{AFX_VIRTUAL(CAboutDialog)
    protected:
    //! data exchange
    //##ModelId=3B863B04037C
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL

protected:
    //{{AFX_MSG(CAboutDialog)
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

} // namespace View
```

Implementierung (AboutDialog.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  AboutDialog.cpp
    \brief class CAboutDialog (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"
#include "AboutDialog.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

namespace View
{

/*!
    Route dialog ID down to CDialog
*/
CAboutDialog::CAboutDialog() : CDialog(CAboutDialog::IDD)
{
    //{{AFX_DATA_INIT(CAboutDialog)
    //}}AFX_DATA_INIT
}


/*
    Transfer data from the dialog controls to this class' member variables (and vice versa).
    Whole code is generated by wizards. Not actually required because no controls were defined
*/
void CAboutDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDialog)
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CAboutDialog, CDialog)
    //{{AFX_MSG_MAP(CAboutDialog)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

} // namespace View
```

Controller

Programmcontroller

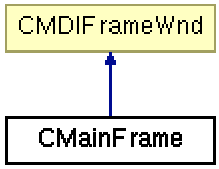

Abbildung 78: Vererbungshierarchie `CMainFrame`

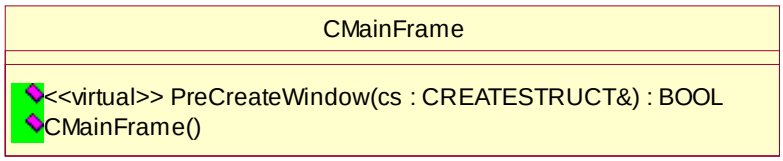

Abbildung 79: UML-Beschreibung von `CMainFrame`

Interface (MainFrm.h)

```
/////////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  MainFrm.h
    \brief class CMainFrame (interface)
*/

#pragma once

//! Controller of the MVC concept
/*!
    Coordinates data flow between user, Model and View
*/
namespace Controller
{

//! The MDI main frame
/*! \author         Visual Studio Wizard
    \date           August 4, 2001
*/

//##ModelId=3B863AF8005A
class CMainFrame : public CMDIFrameWnd
{
public:
    //! default constructor
    //##ModelId=3B863AF80096
    CMainFrame() {};

    //{{AFX_VIRTUAL(CMainFrame)
    public:
    //##ModelId=3B863AF8005C
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

protected:
    //{{AFX_MSG(CMainFrame)
    //}}AFX_MSG

    DECLARE_DYNAMIC(CMainFrame)
    DECLARE_MESSAGE_MAP()
};


} // namespace Controller

//{{AFX_INSERT_LOCATION}}
```

Implementierung (MainFrm.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  MainFrm.cpp
    \brief class CMainFrame (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

namespace Controller
{

/*!
    Change visual appearance, remove file name from window title
    \param cs Windows style
    \return true, if successful
*/
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CMDIFrameWnd::PreCreateWindow(cs) )
        return FALSE;

    cs.style &= ~ FWS_ADDTOTITLE;
    cs.style |= FWS_PREFIXTITLE;

    return TRUE;
}


IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

} // namespace Controller
```

Karteicontroller

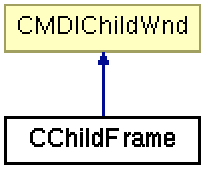

Abbildung 80: Vererbungshierarchie `CChildFrame`

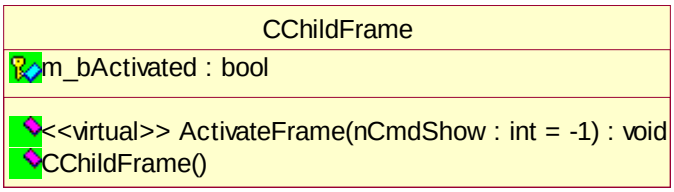

Abbildung 81: UML-Beschreibung von `CChildFrame`

Interface (ChildFrm.h)

```
/////////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  ChildFrm.h
    \brief class CChildFrame (interface)
*/

#pragma once

namespace Controller
{

//! The MDI child frame
/*! \author         Visual Studio Wizard
    \date           August 23, 2001
*/

//##ModelId=3B863AFF026C
class CChildFrame : public CMDIChildWnd
{
public:
    //! default constructor
    /*! nothing left to do */
    //##ModelId=3B863AFF02A6
    CChildFrame() ;

    //{{AFX_VIRTUAL(CChildFrame)
    public:
    //##ModelId=3B863AFF02A3
    virtual void ActivateFrame(int nCmdShow = -1);
    //}}AFX_VIRTUAL

protected:
    //{{AFX_MSG(CChildFrame)
    //}}AFX_MSG

    //! true, if window is newly created
    //##ModelId=3B863AFF029F
    bool m_bActivated;

    DECLARE_DYNCREATE(CChildFrame)
    DECLARE_MESSAGE_MAP()
};


} // namespace Controller

//{{AFX_INSERT_LOCATION}}
```

Implementierung (ChildFrm.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  ChildFrm.cpp
    \brief class CChildFrame (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"

#include "ChildFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


namespace Controller
{

/*!
    Window must be created
*/
CChildFrame::CChildFrame()
{
    m_bActivated = false;
}


/*!
    activate child window and maximize if possible
    \param nCmdShow window style on activation
*/
void CChildFrame::ActivateFrame(int nCmdShow)
{
    if (!m_bActivated)
    {
        m_bActivated = true;
        nCmdShow = SW_SHOWMAXIMIZED;
    }

    CMDIChildWnd::ActivateFrame(nCmdShow);
}


IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)

BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
    //{{AFX_MSG_MAP(CChildFrame)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

} // namespace Controller
```

Debug

Szenarien

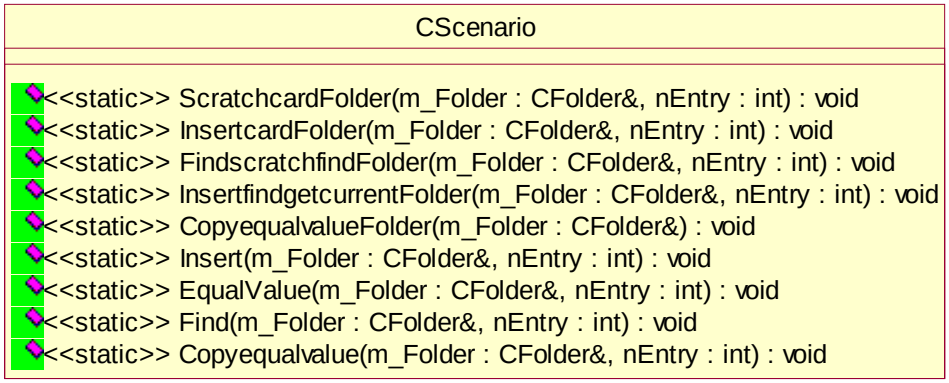

Abbildung 82: UML-Beschreibung von `CScenario`

Interface (Scenario.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Scenario.h
    \brief class Debug::CScenario (interface)
*/

#pragma once

namespace Debug
{

#ifdef _DEBUG
    using Model::CFolder;

//! Generate scenarios
/*! \li

    \author         Stephan Brumme
    \date           August 23, 2001

    \invariant
    \li (none)
*/

//##ModelId=3B863AF60071
class CScenario
{
public:
    //! Copy/EqualValue (document, case 1)
    //##ModelId=3B863AF6011A
    static void Copyequalvalue(CFolder& m_Folder, int nEntry);
    //! Find a given document
    //##ModelId=3B863AF600F1
    static void Find(CFolder& m_Folder, int nEntry);
    //! Compare two slightly different documents
    //##ModelId=3B863AF600ED
    static void EqualValue(CFolder& m_Folder, int nEntry);
    //! Insert an already existing document
    //##ModelId=3B863AF600E9
    static void Insert(CFolder& m_Folder, int nEntry);
    //! Copy/EqualValue (folder, case 1)
    //##ModelId=3B863AF600E6
    static void CopyequalvalueFolder(CFolder& m_Folder);
    //! Insert/Find/GetCurrent (case 2);
    //##ModelId=3B863AF600E2
    static void InsertfindgetcurrentFolder(CFolder& m_Folder, int nEntry);
    //! Find/Scratch/Find (case 3)
```

```
    //##ModelId=3B863AF600DE
    static void FindscratchfindFolder(CFolder& m_Folder, int nEntry);
    //! Insert/Card (case 4)
    //##ModelId=3B863AF600AE
    static void InsertcardFolder(CFolder& m_Folder, int nEntry);
    //! Scratch/Card (case 5)
    //##ModelId=3B863AF600AA
    static void ScratchcardFolder(CFolder& m_Folder, int nEntry);
};

#endif
}
```

Implementierung (Scenario.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Scenario.cpp
    \brief class Debug::CScenario (implementation)
*/

#include "stdafx.h"
#include "Folder.h"
#include "BankDocument.h"
#include "Scenario.h"


#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif



namespace Debug
{

#ifdef _DEBUG

    using Model::CBankDocument;

/*!
    \test  Case 1 of all documents: copy a document and compare copy/original document
    \attention always true !
    \param m_Folder used folder
    \param nEntry   used document
*/
void CScenario::Copyequalvalue(CFolder& m_Folder, int nEntry)
{
    if (AfxMessageBox("Dieses Szenario kopiert das gerade gewählte Dokument und\n"
                      "vergleicht anschließend Original und Kopie.\n\n"
                      "Der Test muss stets erfolgreich verlaufen.\n"
                      "Es werden keine Daten in der Originalkartei verändert.\n",
                      MB_OKCANCEL) == IDOK)
    {
        // move cursor to the selected element
        m_Folder.SetCursor(nEntry);
        // get it
        CBankDocument* pObject = m_Folder.GetCurrent();
        // generate copy
        CBankDocument* pObject2 = (CBankDocument*)pObject->Generate();

        // copy
        TRACE("\nScenario Copy/EqualValue (document-based)\n");
        pObject->Dump(afxDump);
        TRACE("Now copying ...\n");
        *pObject2 = *pObject;

        // compare
        TRACE("\n\nAnd comparing ...\n");
        ASSERT (*pObject == *pObject2);
```

```
        pObject2->Dump(afxDump);

        // free allocated memory
        delete pObject;
        delete pObject2;

        TRACE("\nThis scenario was successfully completed if no window popped up !\n\n");
    }
}


/*!
    \test Find an already existing document of the set
    \attention always true !
    \param m_Folder used folder
    \param nEntry   used document
*/
void CScenario::Find(CFolder& m_Folder, int nEntry)
{
    if (AfxMessageBox("Dieses Szenario sucht das gerade gewählte Dokument in der Kartei.\n\n"
                      "Der Test muss stets erfolgreich verlaufen.\n"
                      "Es werden keine Daten in der Originalkartei verändert.\n",
                      MB_OKCANCEL) == IDOK)
    {
        // move cursor to the selected element
        m_Folder.SetCursor(nEntry);
        // get it
        CBankDocument* pObject = m_Folder.GetCurrent();

        TRACE("\nScenario Find\n");
        pObject->Dump(afxDump);

        // find
        TRACE("Searching ...\n");
        ASSERT(m_Folder.Find(pObject));

        // free allocated memory
        delete pObject;

        TRACE("\nThis scenario was successfully completed if no window popped up !\n\n");
    }
}


/*!
    \test  Create a copy and slightly change date of foundation
    \attention always true, because date of foundation is not used for comparisons !
    \param m_Folder used folder
    \param nEntry   used document
*/
void CScenario::EqualValue(CFolder& m_Folder, int nEntry)
{
    if (AfxMessageBox("Dieses Szenario vergleicht das gerade gewählte Dokument\n"
                      "mit einer Kopie, bei der das Erzeugungsdatum verändert wurde.\n"
                      "Der Test muss stets erfolgreich verlaufen.\n"
                      "Es werden keine Daten in der Originalkartei verändert.\n",
                      MB_OKCANCEL) == IDOK)
    {
        // move cursor to the selected element
        m_Folder.SetCursor(nEntry);
        // get it (only used as CBankDocument)
        CBankDocument* pObject = m_Folder.GetCurrent();

        // create a copy, modify date
        CBankDocument* pObject2 = new CBankDocument(pObject->GetAuthor(), pObject->GetTitle(),
                                                    pObject->GetKey());

        TRACE("\nScenario Find\n");
        pObject->Dump(afxDump);

        // compare (as CBankDocuments !!!)
        TRACE("Comparing ...\n");
        ASSERT(*pObject2 == *pObject);

        // free allocated memory
        delete pObject2;
```

```
        delete pObject;

        TRACE("\nThis scenario was successfully completed if no window popped up !\n\n");
    }
}


/*!
    \test  Insert an already document into the set
    \attention fails always !
    \param m_Folder used folder
    \param nEntry   used document
*/
void CScenario::Insert(CFolder& m_Folder, int nEntry)
{
    if (AfxMessageBox("Dieses Szenario versucht, das gerade gewählte Dokument\n"
                      "erneut in die Kartei einzufügen.\n\n"
                      "Der Test darf nicht erfolgreich verlaufen.\n"
                      "Es werden keine Daten in der Originalkartei verändert.\n",
                      MB_OKCANCEL) == IDOK)
    {
        // move cursor to the selected element
        m_Folder.SetCursor(nEntry);
        // get it
        CBankDocument* pObject = m_Folder.GetCurrent();

        TRACE("\nScenario Find\n");
        pObject->Dump(afxDump);

        // insert
        TRACE("Inserting ...\n");
        ASSERT(m_Folder.Insert(pObject) != -1);

        // free allocated memory
        delete pObject;

        TRACE("\nThis scenario was successfully completed if a window popped up !\n\n");
    }
}


/*!
    \test  Case 1 (CFolder): copy a folder and compare copy/original folder
    \attention always true !
    \param m_Folder used folder
*/
void CScenario::CopyequalvalueFolder(CFolder& m_Folder)
{
    if (AfxMessageBox("Dieses Szenario kopiert die aktuelle Kartei und\n"
                      "vergleicht anschließend Original und Kopie.\n\n"
                      "Der Test muss stets erfolgreich verlaufen.\n"
                      "Es werden keine Daten in der Originalkartei verändert.\n",
                      MB_OKCANCEL) == IDOK)
    {
        // create new folder
        CFolder* pFolder = new CFolder();

        TRACE("\nScenario Copy/EqualValue (folder-based)\n");
        m_Folder.Dump(afxDump);

        // copy
        TRACE("Now copying ...\n");
        *pFolder = m_Folder;

        // compare
        TRACE("\n\nAnd comparing ...\n");
        ASSERT (m_Folder == *pFolder);

        // free allocated memory
        delete pFolder;

        TRACE("\nThis scenario was successfully completed if no window popped up !\n\n");
    }
}
```

```
/*!
    \test  Case 2 (CFolder): Insert a document into the set, find and retrieve it, compare it
to the original
    \attention always true !
    \param m_Folder used folder
    \param nEntry   used document
*/
void CScenario::InsertfindgetcurrentFolder(CFolder& m_Folder, int nEntry)
{
    if (AfxMessageBox("Dieses Szenario fügt ein neues Bankdokument in die Kartei ein.\n"
                      "Anschließend wird danach gesucht, was den Cursor verändert.\n"
                      "Da es dann das aktuelle Element ist, wird es ausgelesen und muss mit
dem Original übereinstimmen.\n\n"
                      "Der Test muss stets erfolgreich verlaufen.\n"
                      "Es werden keine Daten in der Originalkartei verändert.\n",
                      MB_OKCANCEL) == IDOK)
    {
        // new bankdocument
        CBankDocument bankdocument("Testautor_", "Testtitel_", "Testkey_");
        TRACE("\nScenario Insert/Find/GetCurrent\n");
        bankdocument.Dump(afxDump);

        // copy folder (to avoid modifactions to the original one)
        CFolder* pFolder = new CFolder(m_Folder);

        // insert
        TRACE("Inserting ...\n");
        ASSERT(pFolder->Insert(&bankdocument));

        // find
        TRACE("Looking up ...\n");
        ASSERT(pFolder->Find(&bankdocument));

        // get current
        TRACE("Getting current document ...\n");
        CBankDocument *pBankDocument = pFolder->GetCurrent();

        // compare
        TRACE("Comparing ...\n");
        ASSERT(bankdocument == *pBankDocument);

        // free allocated memory
        delete pBankDocument;
        delete pFolder;

        TRACE("\nThis scenario was successfully completed if no window popped up !\n\n");
    }
}


/*!
    \test  Case 3 (CFolder): Find a document, scratch it and search for it again
    \attention fails always !
    \param m_Folder used folder
    \param nEntry   used document
*/
void CScenario::FindscratchfindFolder(CFolder& m_Folder, int nEntry)
{
    if (AfxMessageBox("Dieses Szenario sucht das gerade gewählte Dokument in der Kartei.\n"
                      "Anschließend wird es gelöscht und erneut danach gesucht.\n\n"
                      "Der Test darf nicht erfolgreich verlaufen.\n"
                      "Es werden keine Daten in der Originalkartei verändert.\n",
                      MB_OKCANCEL) == IDOK)
    {
        // move cursor to the selected element
        m_Folder.SetCursor(nEntry);
        // get it
        CBankDocument* pObject = m_Folder.GetCurrent();

        TRACE("\nScenario Find/Scratch/Find\n");
        pObject->Dump(afxDump);

        // copy folder (to avoid modifactions to the original one)
        CFolder* pFolder = new CFolder(m_Folder);

        // find
```

```
        TRACE("Looking up document ...\n");
        ASSERT(pFolder->Find(pObject));

        // scratch
        TRACE("Scratch it ...\n");
        pFolder->Scratch();

        // find again
        TRACE("Looking up document for a second time ...\n");
        ASSERT(pFolder->Find(pObject));

        // free allocated memory
        delete pObject;
        delete pFolder;

        TRACE("\nThis scenario was successfully completed if a window popped up !\n\n");
    }
}


/*!
    \test  Case 4 (CFolder): Insert a new document - cardinality must increase by 1
    \attention always true !
    \param m_Folder used folder
    \param nEntry   used document
*/
void CScenario::InsertcardFolder(CFolder& m_Folder, int nEntry)
{
    if (AfxMessageBox("Dieses Szenario fügt ein neues Bankdokument in die Kartei ein.\n"
                      "Anschließend muss die Anzahl der Elemente um 1 größer sein.\n\n"
                      "Der Test muss stets erfolgreich verlaufen.\n"
                      "Es werden keine Daten in der Originalkartei verändert.\n",
                      MB_OKCANCEL) == IDOK)
    {
        // new bankdocument
        CBankDocument bankdocument("Testautor_", "Testtitel_", "Testkey_");
        TRACE("\nScenario Insert/Card\n");
        bankdocument.Dump(afxDump);

        // copy folder (to avoid modifactions to the original one)
        CFolder* pFolder = new CFolder(m_Folder);

        // cardinality
        int nOldCard = pFolder->Card();
        TRACE("Old Card: %d\n", nOldCard);

        // insert
        TRACE("Inserting ...\n");
        ASSERT(pFolder->Insert(&bankdocument));

        // cardinality for the second time
        int nNewCard = pFolder->Card();
        TRACE("New Card: %d\n", nNewCard);

        // verify condition
        ASSERT(nOldCard+1==nNewCard);

        // free allocated memory
        delete pFolder;

        TRACE("\nThis scenario was successfully completed if no window popped up !\n\n");
    }
}


/*!
    \test  Case 5 (CFolder): Scratch current document - cardinality must decrease by 1
    \attention always true !
    \param m_Folder used folder
    \param nEntry   used document
*/
void CScenario::ScratchcardFolder(CFolder& m_Folder, int nEntry)
{
    if (AfxMessageBox("Dieses Szenario löscht das aktuell gewählte Dokument aus der Kartei.\n"
                      "Anschließend muss die Anzahl der Elemente um 1 niedriger sein.\n\n"
                      "Der Test muss stets erfolgreich verlaufen.\n"
```

```
                    "Es werden keine Daten in der Originalkartei verändert.\n",
                    MB_OKCANCEL) == IDOK)
    {
        // move cursor to the selected element
        m_Folder.SetCursor(nEntry);
        // get it
        CBankDocument* pObject = m_Folder.GetCurrent();

        TRACE("\nScenario Scratch/Card\n");

        // copy folder (to avoid modifactions to the original one)
        CFolder* pFolder = new CFolder(m_Folder);

        // cardinality
        int nOldCard = pFolder->Card();
        TRACE("Old Card: %d\n", nOldCard);

        // find
        TRACE("Looking up ...\n");
        ASSERT(pFolder->Find(pObject));

        // scratch
        TRACE("Scratching ...\n");
        pFolder->Scratch();

        // cardinality for the second time
        int nNewCard = pFolder->Card();
        TRACE("New Card: %d\n", nNewCard);

        // verify condition
        ASSERT(nOldCard-1==nNewCard);

        // free allocated memory
        delete pObject;
        delete pFolder;

        TRACE("\nThis scenario was successfully completed if no window popped up !\n\n");
    }
}

#endif
} // namespace Debug
```

Ausgabe von Ablaufinformationen

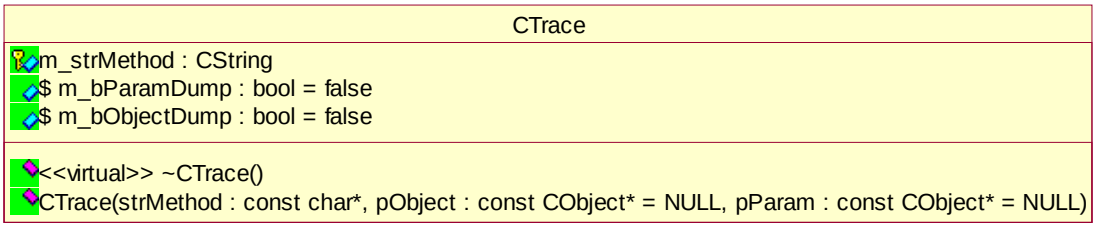

Abbildung 83: UML-Beschreibung von `CTrace`

Interface (Trace.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Trace.h
    \brief class Debug::CTrace (interface)
*/

#pragma once

//! Gathers all Debug information
namespace Debug
{

//! Generic debug info class
/*! \li encapsulates TRACE

    \author         Stephan Brumme
    \date           August 21, 2001

    \invariant
    \li (none)
*/

//##ModelId=3B863AF80104
class CTrace
{
public:
    //! constructor
    //##ModelId=3B863AF8013E
    CTrace(const char* strMethod, const CObject* pObject = NULL,
           const CObject* pParam  = NULL);
    //! destructor
    //##ModelId=3B863AF8013C
    virtual ~CTrace();

    //! dump the whole object each time
    //##ModelId=3B863AF8013B
    static bool m_bObjectDump;
    //! dump the parameter in the constructor
    //##ModelId=3B863AF8013A
    static bool m_bParamDump;

protected:
    //! name of the calling method
    //##ModelId=3B863AF80139
    CString        m_strMethod;
    //! calling object
    //##ModelId=3B863AF80136
    const CObject* m_pObject;
};

}
```

Implementierung (Trace.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  Trace.cpp
    \brief class Debug::CTrace (implementation)
*/

#include "stdafx.h"
#include "Trace.h"


namespace Debug
{
    // default: no dumps
    bool CTrace::m_bObjectDump = false;
    bool CTrace::m_bParamDump  = false;


/*!
    Initial TRACE
    \param strMethod name of the calling method
    \param pObject   pointer to the calling object
    \param pParam    parameter of the calling method (only for dumps)
*/
CTrace::CTrace(const char* strMethod, const CObject* pObject, const CObject* pParam)
{
    // only compiled in DEBUG build
#ifdef _DEBUG
    // copy parameters
    m_strMethod = strMethod;
    m_pObject   = pObject;

    // valid object ?
    if (m_pObject)
    {
        // display classname
        TRACE("%s::", m_pObject->GetRuntimeClass()->m_lpszClassName);
        // verify object assertions
        ASSERT_VALID(m_pObject);

        // dump if neccessary
        if (m_bObjectDump)
            //m_pObject->Dump(afxDump);
            TRACE("OBJECT DUMP DISABLED TO AVOID MEMORY OVERFLOW !!!\n");
    }

    // display method name
    TRACE("%s", m_strMethod);

    // TRACE and ASSERT parameter
    if (pParam)
    {
        TRACE("(%s)", pParam->GetRuntimeClass()->m_lpszClassName);
        ASSERT_VALID(pParam);

        // dump if neccessary
        if (m_bParamDump)
            pParam->Dump(afxDump);
    }

    TRACE(" entered ...");
#endif
}


/*!
    Finishing TRACE and ASSERT
*/
CTrace::~CTrace()
{
    // only compiled in DEBUG build
#ifdef _DEBUG
    TRACE("leaving %s\n", m_strMethod);
    if (m_pObject)
```

```
        ASSERT_VALID(m_pObject);
#endif
}

} // namespace Debug
```

Umleitung der Debug-Ausgabe

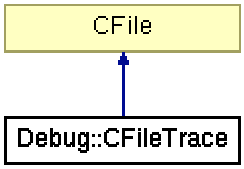

Abbildung 84: Vererbungshierarchie `CFileTrace`

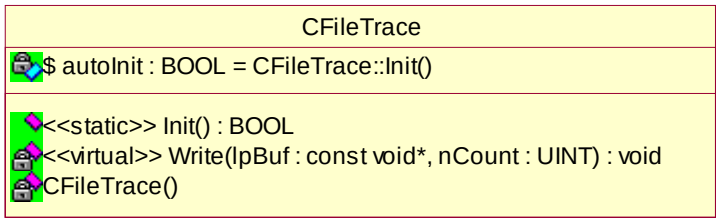

Abbildung 85: UML-Beschreibung von `CFileTrace`

Interface und Implementierung (FileTrace.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  FileTrace.h
    \brief class Debug::CFileTrace (interface+implementation)

    \author         Stephan Brumme and Paul DiLascia
    \date           August 18, 2001
*/

////////////////////////////////////////////////////////////////
//
// minor changes by Stephan Brumme
//
// now the original copyright:
//
// TraceWin Copyright 1995-1999 Paul DiLascia
// If this program works, it was written by Paul DiLascia.
// If not, I don't know who wrote it.
//
// ************************************************************************
// TraceWin is a tool that displays MFC diagnostic (afxDump, TRACE) output.
//
// To use TraceWin, you must #include this file somewhere in your main program
// file (typically where you implement your CWinApp). Since this file contains
// code, you should #include it in only once--i.e. NOT in StdAfx.h--or you'll
// get multiply-defined symbol errors in the linker. This file contains an
// auto-initializing static variable that works in most cases; but you may miss
// some TRACE output from constructors of static objects.
//
// ************************************************************************
//

#pragma once

// only in Debug build
#ifdef _DEBUG

namespace Debug
{

//! function pointer that gets all Debug output
static void (*Print)(const char*);

//! CFileTrace is a CFile that "writes" to the trace window
//##ModelId=3B863AFF037A
class CFileTrace : public CFile {
private:
```

```
    //! default constructor
    //##ModelId=3B863AFF03B6
    CFileTrace();

    //! true, if CFileTrace is successfully invoked
    //##ModelId=3B863AFF037C
    static BOOL autoInit;
    //! overwrite output writer
    //##ModelId=3B863AFF037F
    virtual void Write(const void* lpBuf, UINT nCount);

public:
    //! initialize this class as a singleton
    //##ModelId=3B863AFF037D
    static BOOL Init();

    DECLARE_DYNAMIC(CFileTrace)
};


/*!
    Set attributes
*/
CFileTrace::CFileTrace()
{
    // stream name
    m_strFileName = _T("CFileTrace");
    // no output function defined
    Print = NULL;
}

/*!
    Override to write to debug window instead of file.
    \param lpBuf pointer to the debug output
    \param nCount number of bytes in lpBuf
    \see CFile
*/
void CFileTrace::Write(const void* lpBuf, UINT nCount)
{
    // MFC tracing not enabled ?
    if (!afxTraceEnabled)
        return;

    // only redirect if user defined output function is defined
    if (Print)
    {
        static char mybuf[2048];

        // copy buffer
        memcpy(mybuf, lpBuf, nCount);

        // create a CString
        CString strDebug(mybuf, nCount);
        // and print it
        Print(strDebug);
    }

    // Also do normal debug thing
    ::OutputDebugString((LPCTSTR)lpBuf);
}


/*!
    Initialize tracing. Replace global afxDump.m_pFile with CFileTrace object.
*/
BOOL CFileTrace::Init()
{
    if (afxDump.m_pFile==NULL) {
        // Initialize tracing: replace afxDump.m_pFile w/ my own CFile object
        //
        // It's important to allocate with "new" here, not a static object,
        // because CFileTrace is virtual--i.e., called through a pointer in
        // the object's vtbl--and the compiler will zero out the virtual
        // function table with a NOP function when a static object
        // goes out of scope. But I want my CFileTrace to stay around to
        // display memory leak diagnostics even after all static objects
```

```
        // have been destructed. So I allocate the object with new and
        // never delete it. I don't want this allocation to itself generate
        // a reported memory leak, so I turn off memory tracking before I
        // allocate, then on again after.
        //
        BOOL bEnable = AfxEnableMemoryTracking(FALSE);
        afxDump.m_pFile = new CFileTrace;
        AfxEnableMemoryTracking(bEnable);
        return TRUE;
    }
    return FALSE;
}


//////////////////
// This object does nothing but call CFileTrace::Init, so all you have to
// do is #include this file. Because afxDump is defined in a module with
//
// #pragma init_seg(lib)
//
// afxDump gets initialized before the "user" segment which is where your
// app (and autoInit) is by default. If you need to use init_seg(lib),
// or you have other objects whose constructors call TRACE that get
// initialized before CFileTrace::bInitialized, you will have to call
// CFileTrace::Init yourself, before your first TRACE statement.
//
BOOL CFileTrace::autoInit = CFileTrace::Init();

IMPLEMENT_DYNAMIC(CFileTrace, CFile)

} // namespace Debug

#endif
```

Debugausgabe-Dialog

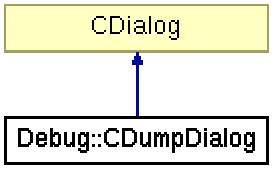

Abbildung 86: Vererbungshierarchie `CDumpDialog`

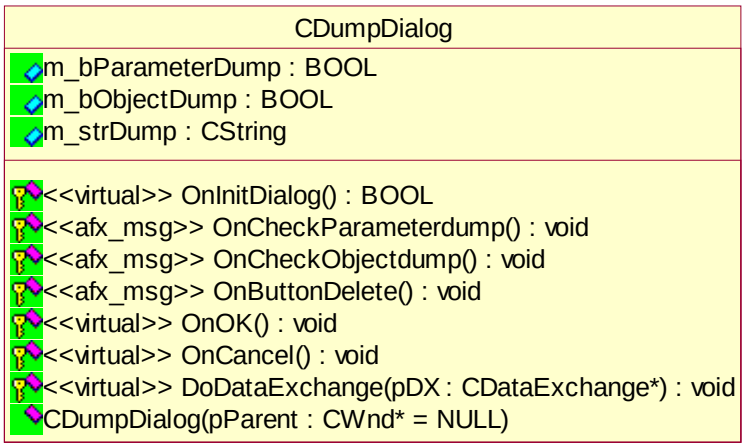

Abbildung 87: UML-Beschreibung von `CDumpDialog`

Interface (DumpDialog.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  DumpDialog.h
    \brief class Debug::CDumpDialog (interface)
*/

#pragma once
#include "Folder.h"

namespace Debug
{

//! A modeless dialog that displays DEBUG output
/*! \author         Stephan Brumme
    \date           August 21, 2001

    \invariant
    \li (none)
*/

//##ModelId=3B863AFC01EA
class CDumpDialog : public CDialog
{
public:
    //! constructor
    //##ModelId=3B863AFC0224
    CDumpDialog(CWnd* pParent = NULL);
    //{{AFX_DATA(CDumpDialog)
    enum { IDD = IDD_DEBUG };
    //##ModelId=3B863AFC01EE
    CString m_strDump;
    //##ModelId=3B863AFC01ED
    BOOL    m_bObjectDump;
    //##ModelId=3B863AFC01EC
    BOOL    m_bParameterDump;
    //}}AFX_DATA
```

```
    /*! \var m_strDump whole dump information */

    //{{AFX_VIRTUAL(CDumpDialog)
    protected:
    //##ModelId=3B863AFC0221
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL

protected:

    //{{AFX_MSG(CDumpDialog)
    //##ModelId=3B863AFC021F
    virtual void OnCancel();
    //##ModelId=3B863AFC021D
    virtual void OnOK();
    //##ModelId=3B863AFC01F5
    afx_msg void OnButtonDelete();
    //##ModelId=3B863AFC01F3
    afx_msg void OnCheckObjectdump();
    //##ModelId=3B863AFC01F1
    afx_msg void OnCheckParameterdump();
    //##ModelId=3B863AFC01EF
    virtual BOOL OnInitDialog();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

} // namespace Debug

//{{AFX_INSERT_LOCATION}}
```

Implementierung (DumpDialog.cpp)

```
//////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  DumpDialog.cpp
    \brief class CDumpDialog (implementation)
*/

#include "stdafx.h"
#include "PraktikumMFC.h"
#include "DumpDialog.h"
#include "Trace.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


namespace Debug
{

/*!
    Initialize Dialog
    \param pParent handle of the parent window (not used)
*/
CDumpDialog::CDumpDialog(CWnd* pParent /*=NULL*/)
    : CDialog(CDumpDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CDumpDialog)
    m_strDump = _T("");
    m_bObjectDump = CTrace::m_bObjectDump;
    m_bParameterDump = CTrace::m_bParamDump;
    //}}AFX_DATA_INIT
}


/*!
    Set window style attributes (transparency)
*/
```

```
BOOL CDumpDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    // code taken from "Windows 2000 Systemprogrammierung" p. 87 by Ron Nanko
//  SetWindowLong(m_hWnd, GWL_EXSTYLE, GetWindowLong(m_hWnd, GWL_EXSTYLE) | WS_EX_LAYERED);
//  SetLayeredWindowAttributes(m_hWnd, 0, (255*90)/100, LWA_ALPHA);

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX-Eigenschaftenseiten sollten FALSE zurückgeben
}


/*!
    Transfer data from the dialog controls to this class' member variables (and vice versa).
    Whole code is generated by wizards
    \param pDX data exchange object
*/
void CDumpDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CDumpDialog)
    DDX_Text(pDX, IDC_EDIT_DUMP, m_strDump);
    DDX_Check(pDX, IDC_CHECK_OBJECTDUMP, m_bObjectDump);
    DDX_Check(pDX, IDC_CHECK_PARAMETERDUMP, m_bParameterDump);
    //}}AFX_DATA_MAP
}


/*!
    Overridden to avoid closing the dialog
*/
void CDumpDialog::OnCancel()
{
}


/*!
    Save whole protocol to disc
*/
void CDumpDialog::OnOK()
{
    // accepted file extensions
    const char strFilter[] = "Protokolldateien (*.log)|*.log|Alle Dateien (*.*)|*.*|";
    // "save" dialog
    CFileDialog dlg(FALSE, ".log", "*.log", NULL, strFilter, NULL);

    if (dlg.DoModal() == IDOK)
    {
        // text file, overwrite if neccessary
        CStdioFile fFile(dlg.GetPathName(), CFile::modeCreate | CFile::modeWrite);

        // windows-style CRLF
        CString strNoCRLF(m_strDump);
        strNoCRLF.Replace("\r", "");

        // write it to disc
        fFile.WriteString(strNoCRLF);

        // user feedback
        AfxMessageBox("Protokoll gespeichert.", MB_ICONINFORMATION);
    }
}


/*!
    Clear protocol
*/
void CDumpDialog::OnButtonDelete()
{
    m_strDump = "";
    UpdateData(FALSE);
}


void CDumpDialog::OnCheckObjectdump()
```

```
{
    UpdateData(TRUE);
    CTrace::m_bObjectDump = (m_bObjectDump == TRUE);
}

void CDumpDialog::OnCheckParameterdump()
{
    UpdateData(TRUE);
    CTrace::m_bParamDump = (m_bParameterDump == TRUE);
}


BEGIN_MESSAGE_MAP(CDumpDialog, CDialog)
    //{{AFX_MSG_MAP(CDumpDialog)
    ON_BN_CLICKED(IDC_BUTTON_DELETE, OnButtonDelete)
    ON_BN_CLICKED(IDC_CHECK_OBJECTDUMP, OnCheckObjectdump)
    ON_BN_CLICKED(IDC_CHECK_PARAMETERDUMP, OnCheckParameterdump)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

} // namespace Debug
```

Hilfsklassen

Uhrzeit/Datum

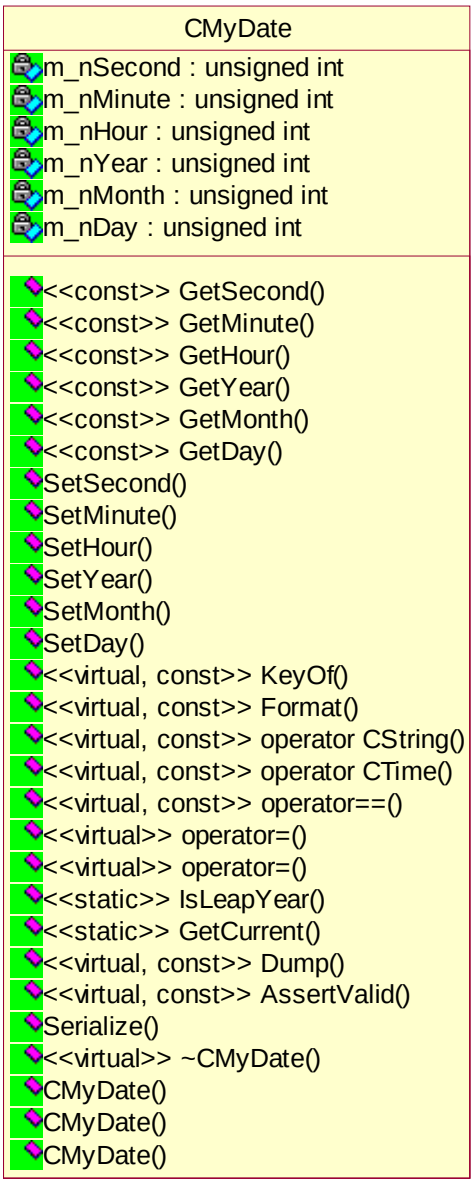

Abbildung 88: UML-Beschreibung von `CMyDate`

Hinweis: Da die Signaturen in der UML-Beschreibung sehr lang sind, habe ich sie aus Platzgründen weggelassen.

Interface (MyDate.h)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  MyDate.h
    \brief class CMyDate (interface)
*/

#pragma once


//! Stores date and time
/*!
    \li 24h-based, no AM/PM support
    \li interface inspired by MFC's CTime
    \li conversion operator for CTime overloaded

    \author         Stephan Brumme
    \date           August 5, 2001

    \invariant
    \li restricted to the usual Gregorian calender (recognizes leap years)
*/

//##ModelId=3B863AFA000A
class CMyDate : public CObject
{
public:
    //! default constructor
    //##ModelId=3B863AFA00EA
    CMyDate();
    //! copy constructor
    //##ModelId=3B863AFA00EB
    CMyDate(const CMyDate& date);
    //! set user defined date at construction time
    //##ModelId=3B863AFA00ED
    CMyDate(unsigned int nDay,    unsigned int nMonth,    unsigned int nYear,
            unsigned int nHour=0, unsigned int nMinute=0, unsigned int nSecond=0);

    //! destructor
    //##ModelId=3B863AFA00E8
    virtual ~CMyDate();

    //! serialization
    //##ModelId=3B863AFA00E6
    void Serialize(CArchive &ar);
    //! validate
    //##ModelId=3B863AFA00C9
    virtual void AssertValid() const;
    //! dump
    //##ModelId=3B863AFA00C6
    virtual void Dump(CDumpContext& dc) const;

    //! return current date and time
    //##ModelId=3B863AFA00BE
    static void GetCurrent(unsigned int& nDay,  unsigned int& nMonth,  unsigned int& nYear,
                           unsigned int& nHour, unsigned int& nMinute, unsigned int& nSecond);
    //! true, if it is a leap year
    //##ModelId=3B863AFA00BB
    static bool IsLeapYear(unsigned int nYear);

    //! copy
    //##ModelId=3B863AFA00B5
    virtual CMyDate& operator = (const CMyDate& date);
    //! to enhance compatibility to CTime
    //##ModelId=3B863AFA00B8
    virtual CMyDate& operator = (const CTime& time);
    //! compare two dates
    //##ModelId=3B863AFA008A
    virtual bool     operator ==(const CMyDate& date) const;

    //! convert to MFC's CTime
    //##ModelId=3B863AFA0088
```

```
    virtual operator CTime() const;
    //! formatted string representation
    //##ModelId=3B863AFA0086
    virtual operator CString() const;
    //! formatted string representation
    //##ModelId=3B863AFA0083
    virtual CString  Format(CString strFormat) const;

    //! generate a key
    //##ModelId=3B863AFA0081
    virtual CString KeyOf() const;

    //! set day
    //##ModelId=3B863AFA007F
    void SetDay   (unsigned int nDay);
    //! set month
    //##ModelId=3B863AFA007D
    void SetMonth (unsigned int nMonth);
    //! set year
    //##ModelId=3B863AFA007B
    void SetYear  (unsigned int nYear);
    //! set hour
    //##ModelId=3B863AFA0079
    void SetHour  (unsigned int nHour);
    //! set minute
    //##ModelId=3B863AFA0056
    void SetMinute(unsigned int nMinute);
    //! set second
    //##ModelId=3B863AFA0054
    void SetSecond(unsigned int nSecond);

    //! return day
    //##ModelId=3B863AFA0052
    unsigned int GetDay   () const;
    //! return month
    //##ModelId=3B863AFA0050
    unsigned int GetMonth () const;
    //! return year
    //##ModelId=3B863AFA004E
    unsigned int GetYear  () const;
    //! return hour
    //##ModelId=3B863AFA004C
    unsigned int GetHour  () const;
    //! return minute
    //##ModelId=3B863AFA004A
    unsigned int GetMinute() const;
    //! return second
    //##ModelId=3B863AFA0048
    unsigned int GetSecond() const;

    //! constants for month‘s names
    /*! The constants were defined for better readability and aren‘t urgently required
    */
    enum { JANUARY   =  1, FEBRUARY =  2, MARCH    =  3, APRIL    =  4,
           MAY       =  5, JUNE     =  6, JULY     =  7, AUGUST   =  8,
           SEPTEMBER =  9, OCTOBER  = 10, NOVEMBER = 11, DECEMBER = 12 };

private:
    //! day
    //##ModelId=3B863AFA0047
    unsigned int m_nDay;
    //! month
    //##ModelId=3B863AFA0046
    unsigned int m_nMonth;
    //! year
    //##ModelId=3B863AFA000F
    unsigned int m_nYear;

    //! hour
    //##ModelId=3B863AFA000E
    unsigned int m_nHour;
    //! minute
    //##ModelId=3B863AFA000D
    unsigned int m_nMinute;
    //! second
    //##ModelId=3B863AFA000C
```

```
    unsigned int m_nSecond;


    DECLARE_SERIAL(CMyDate)
};
```

Implementierung (MyDate.cpp)

```
////////////////////////////////////////////////////////
//  Softwarebauelemente II, Praktikumsbeleg/MFC

/*! \file  MyDate.cpp
    \brief class CMyDate (implementation)
*/


#include "stdafx.h"
#include "MyDate.h"
// we are using OS-specific date/time operations
#include <time.h>

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif


/*!
    Constructs a new object that contains current date and time
    \see GetCurrent
*/
CMyDate::CMyDate()
{
    GetCurrent(m_nDay, m_nMonth, m_nYear, m_nHour, m_nMinute, m_nSecond);
}


/*!
    Copies an existing CMyDate
    \param date date to be copied
    \see operator=
*/
CMyDate::CMyDate(const CMyDate& date)
{
    operator=(date);
}


/*!
    Constructs a new CMyDate using given date/time
    \param nDay    day
    \param nMonth  month
    \param nYear   year
    \param nHour   hour
    \param nMinute minute
    \param nSecond second
*/
CMyDate::CMyDate(unsigned int nDay,  unsigned int nMonth,  unsigned int nYear,
                 unsigned int nHour, unsigned int nMinute, unsigned int nSecond)
{
    // store date
    m_nDay    = nDay;
    m_nMonth  = nMonth;
    m_nYear   = nYear;

    // store time
    m_nHour   = nHour;
    m_nMinute = nMinute;
    m_nSecond = nSecond;
}


/*!
```

```
    No deeper meaning right now
*/
CMyDate::~CMyDate()
{
}


/*!
    Loads/stores CMyDate
    \param ar CArchive that provides data persistency
*/
void CMyDate::Serialize(CArchive &ar)
{
    // do not forget to serialize CObject's data members
    CObject::Serialize(ar);

    if (ar.IsStoring())
        ar << m_nYear << m_nMonth << m_nDay << m_nHour << m_nMinute << m_nSecond;
    else
        ar >> m_nYear >> m_nMonth >> m_nDay >> m_nHour >> m_nMinute >> m_nSecond;
}


/*!
    Determines whether the date is valid,
    \b invariant describes further details (see above)
*/
void CMyDate::AssertValid() const
{
    CObject::AssertValid();

    // validate month
    ASSERT(m_nMonth >= JANUARY && m_nMonth <= DECEMBER);

    // days per month, february may vary, note that array starts with 0 (JANUARY=1 !)
    unsigned int nDaysPerMonth[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    // adjust days of february
    if (IsLeapYear(m_nYear))
        nDaysPerMonth[FEBRUARY]++;

    // validate day
    ASSERT(m_nDay > 0 && m_nDay <= nDaysPerMonth[m_nMonth]);
    // day and month are valid

    // now check the time
    ASSERT(m_nHour   >= 0 && m_nHour   <= 23);
    ASSERT(m_nMinute >= 0 && m_nMinute <= 59);
    ASSERT(m_nSecond >= 0 && m_nSecond <= 59);
}


/*!
    Create a dump of the class (only DEBUG version)
    \param dc output dump stream
*/
void CMyDate::Dump(CDumpContext& dc) const
{
    CObject::Dump(dc);

    CString strOutput;
    strOutput.Format("%02d.%02d.%04d - %02d:%02d:%02d",
                     m_nDay, m_nMonth, m_nYear, m_nHour, m_nMinute, m_nSecond);
    dc << strOutput;
}


/*!
    Static method that returns current date/time
*/
void CMyDate::GetCurrent(unsigned int& nDay,  unsigned int& nMonth,  unsigned int& nYear,
                         unsigned int& nHour, unsigned int& nMinute, unsigned int& nSecond)
{
    // the following code is basically taken from MSDN

    // use system functions to get the current date as UTC
```

```
    time_t secondsSince1970;
    time(&secondsSince1970);

    // convert UTC to local time zone
    struct tm *localTime;
    localTime = localtime(&secondsSince1970);

    // store retrieved date
    nDay    = localTime->tm_mday;
    nMonth  = localTime->tm_mon + 1;
    nYear   = localTime->tm_year + 1900;
    // store time
    nHour   = localTime->tm_hour;
    nMinute = localTime->tm_min;
    nSecond = localTime->tm_sec;
}


/*!
    Static method to determine whether it is a leap year
    \param  nYear year to examine
    \return true, if nYear is a leap year
*/
bool CMyDate::IsLeapYear(unsigned int nYear)
{
    // used to speed up code, may be deleted
    if (nYear % 4   != 0)
        return false;

    // algorithm taken from MSDN, just converted from VBA to C++
    if (nYear % 400 == 0)
        return true;
    if (nYear % 100 == 0)
        return false;
    if (nYear % 4   == 0)
        return true;

    // this line won't be executed because of optimization (see above)
    return false;
}


/*!
    Copies all attributes
    \param  date CMyDate that should be copied
    \return reference to the copied date
    \see    CMyDate(const CMyDate& date)
    \see    Copy
    \see    operator=(const CTime& time)
*/
CMyDate& CMyDate::operator =(const CMyDate& date)
{
    // date
    m_nDay    = date.m_nDay;
    m_nMonth  = date.m_nMonth;
    m_nYear   = date.m_nYear;
    // time
    m_nHour   = date.m_nHour;
    m_nMinute = date.m_nMinute;
    m_nSecond = date.m_nSecond;

    return *this;
}


/*!
    Copies date/time specific attributes of MFC's CTime
    \param  time CTime that should be copied
    \return reference to the copied date
    \see    CMyDate(const CMyDate& date)
    \see    operator=(const CMyDate& date)
*/
CMyDate& CMyDate::operator =(const CTime& time)
{
    // date
    m_nDay    = time.GetDay();
```

```
    m_nMonth  = time.GetMonth();
    m_nYear   = time.GetYear();
    // time
    m_nHour   = time.GetHour();
    m_nMinute = time.GetMinute();
    m_nSecond = time.GetSecond();

    return *this;
}


/*!
    Compare attributes of two dates
    \param  date CMyDate that we compare to
    \return true, if all attributes are equal
*/
bool CMyDate::operator ==(const CMyDate &date) const
{
    // compare date and time attributes
    return (m_nDay    == date.m_nDay    &&
            m_nMonth  == date.m_nMonth  &&
            m_nYear   == date.m_nYear   &&
            m_nHour   == date.m_nHour   &&
            m_nMinute == date.m_nMinute &&
            m_nSecond == date.m_nSecond);
}


/*!
    Overloaded convertion operator for CTime
    \return CMyDate that is casted to CTime
*/
CMyDate::operator CTime() const
{
    return CTime(m_nYear, m_nMonth, m_nDay, m_nHour, m_nMinute, m_nSecond);
}


/*!
    Overloaded convertion operator for CString
    \return CMyDate that is casted to CString
    \see Format
*/
CMyDate::operator CString() const
{
    return Format("%A, %d.%B %Y, %H:%M:%S");
}


/*
    Formatted string representation similiar to printf
    \return a formatted string
*/
CString CMyDate::Format(CString strFormat) const
{
    return CTime(*this).Format(strFormat);
}


/*!
    Responsible for the generation of a unique key based on the object's attributes
    \return CString that contains the key
*/
CString CMyDate::KeyOf() const
{
    CString strKey;
    strKey.Format("[%d|%d|%d|%d|%d|%d|%p]", m_nDay,  m_nMonth,  m_nYear,
                                            m_nHour, m_nMinute, m_nSecond,
                                            this);
    return strKey;
}


/*!
    Set day
    \param nDay day
```

```
*/
void CMyDate::SetDay(unsigned int nDay)
{
    m_nDay = nDay;
}
/*!
    Set month
    \param nMonth month
*/
void CMyDate::SetMonth(unsigned int nMonth)
{
    m_nMonth = nMonth;
}
/*!
    Set year
    \param nYear year
*/
void CMyDate::SetYear(unsigned int nYear)
{
    m_nYear = nYear;
}
/*!
    Set hour
    \param nHour hour
*/
void CMyDate::SetHour(unsigned int nHour)
{
    m_nHour = nHour;
}
/*!
    Set minute
    \param nMinute minute
*/
void CMyDate::SetMinute(unsigned int nMinute)
{
    m_nMinute = nMinute;
}
/*!
    Set second
    \param nSecond second
*/
void CMyDate::SetSecond(unsigned int nSecond)
{
    m_nSecond = nSecond;
}

/*!
    Get day
    \return day
*/
unsigned int CMyDate::GetDay() const
{
    return m_nDay;
}
/*!
    Get month
    \return month
*/
unsigned int CMyDate::GetMonth() const
{
    return m_nMonth;
}
/*!
    Get year
    \return year
*/
unsigned int CMyDate::GetYear() const
{
    return m_nYear;
}
/*!
    Get hour
    \return hour
*/
unsigned int CMyDate::GetHour() const
{
```

```
    return m_nHour;
}
/*!
    Get minute
    \return minute
*/
unsigned int CMyDate::GetMinute() const
{
    return m_nMinute;
}
/*!
    Get second
    \return second
*/
unsigned int CMyDate::GetSecond() const
{
    return m_nSecond;
}


// serializable
IMPLEMENT_SERIAL(CMyDate, CObject, 1)
```