# Testhandbuch für den .Net Projektworkspace

von jk-ware, Dipl.-Ing. Reinhard Jakob

Version 3.0 vom 10.06.2014

# 1 Vorwort

Nach der Entwicklung mehrerer Spielprogramme für PC und Pocket PC unter den nativen Windows-Betriebssystemen Win32 und CE, war es unser Ziel eine solide und erweiterbare Basis für die Produktentwicklung unter .Net zu schaffen. Diese Basis sollte möglichst viele Eigenschaften der vorhandenen Spielprogramme für unsere zukünftigen .Net-Projekte beinhalten.

Mit dem .Net Projektworkspace können Projekte für den PC unter dem .Net Framework und für Smart Devices unter dem .Net Compact Framework in einem mittleren Umfang realisiert werden. Hierzu werden die Anwendungen auf dem neuesten Stand der Objektorientierten Programmierung von einem umfangreichen Basisprojekt abgeleitet.

Die Projekte für Smart Devices stehen in der Professional-Version nur unter Visual Studio 2008 aber nicht mehr ab Visual Studio 2010 zur Verfügung.

Dieses Handbuch beschreibt die Projektmappe von dem .Net Projektworkspace in einer Visual C# Entwicklungsumgebung und liefert Informationen zu den darin vorhandenen Projekten und deren Komponenten. Es enthält aber keine weitere Abhandlung über die Programmiersprache C# oder über das .Net-Framework, soweit dies über die erforderlichen Erläuterungen zum Programmablauf hinausgeht.

Die Quelltexte sind nur in dem Handbuch des Endproduktes enthalten. Ein umfassendes Verständnis des Programmcodes kann aber nur innerhalb der Entwicklungsumgebung durch die kontextsensitive Hilfe erlangt werden.

Hinweis 1 zu dieser Dokumenterstellung mit Microsoft Word!

- Die aufgeführten Inhaltspunkte sind im Text als Formatvorlagen Überschrift 1, Überschrift 2 und Überschrift 3 gekennzeichnet.
- Die Hauptkapitel mit der Formatierung Überschrift 1 beginnen durch Einfügen / Manueller Umbruch / Abschnittsumbruch auf ungeraden Seiten.
- Die Kapitelnummern werden dadurch nach einer Bearbeitung der Formatvorlage für die Überschrift 1 mit Anpassung der Gliederung über Auswahl von Nummerierung unter Format automatisch erzeugt. Sie können auch manuell als Gliederung der Überschriften unter Format / Nummerierung und Aufzählungszeichen zugefügt werden.
- Das Kapitelverzeichnis und das Inhaltsverzeichnis werden durch Einfügen / Index und Verzeichnisse / Inhaltsverzeichnis erzeugt. Hierbei werden für das Kapitelverzeichnis eine Ebene und für das Inhaltsverzeichnis drei Ebenen ausgewählt.
- Über Einfügen / Beschriftung werden neue Bezeichnungen „Kodierung“ für die Quellcode-Titel mit einbezogenen Kapitelnummern und „Hinweis“ mit fortlaufender Nummerierung für weiterführende Hinweise zugefügt.
- Zum Einfügen des Quellcodes wurden unter dem Format-Menü alle Tabstopps gelöscht und danach die Position der Standardtabstopps auf 0,5cm gesetzt.
- Die Standard-Formatvorlage wurde von Times New Roman 12Pt auf Trebuchet MS 10Pt verändert.
- Unterschiedliche Seitennummerierung wird durch Abschnittswechsel über Einfügen / manueller Umbruch / Abschnittsumbruch nächste Seite möglich. Jedem Abschnitt können danach separat formatierte Seitenzahlen zugewiesen werden.
- Eine Formatvorlage innerhalb der Kopfzeile wird über Ansicht / Kopfzeile (Kopfzeile bearbeiten) mit Einfügen / Schnellbausteine / Feld Kategorie „Verknüpfungen und Verweise“ mit Feldname StyleRef und der Felderoptionen Formatvorlagen hinzugefügt. Soll nur die Nummer der Gliederungsebene (Kapitelnummer) angezeigt werden, ist zusätzlich \n mit anzugeben.
- Der Titeltext, dieser Hinweis, der Kommentar, das Inhaltsverzeichnis und weitere Hinweise innerhalb des Handbuches sind in einer einfachen Tabelle eingefügt. Über Format / Rahmen und Schattierungen kann danach die Hintergrundfarbe geändert werden.
- Alle Hinweise beziehen sich auf die Möglichkeiten und Menüauswahl von Word 2003 und 2010.
- Ab Word 2003 kann dem Dokument über Format / Design ein Hintergrundmuster zugefügt werden.
- Literaturquelle: Albrecht, Nicol: Wissenschaftliche Arbeiten Schreiben mit Word, Addison-Wesley 2002, ISBN 3-8273-1943-9, auch als kostengünstiges eBook.

# 2 Kapitelverzeichnis

# 3 Inhaltsverzeichnis

# 4 Kurzbeschreibung

Mit dem Projektworkspace für Visual C# stellt jk-ware seine Projektbasis für das .Net Framework und das .Net Compact Framework zur Verfügung. Diese Projektbasis, bestehend aus sechs Projekten für die Entwicklerumgebung Microsoft Visual Studio 2008, bildet die Grundlage zur Entwicklung eigener Programme für das Laufzeitsystem des .Net Framework und für mobile Geräte unter dem .Net Compact Framework.

Der Projektworkspace besteht aus einem

- Basisprojekt das ein Basisprogramm erstellt
- Anwendungs-Projekt mit einem Beispielprogramm
- Basisprojekt für das .Net Compact Framework das ein Basisprogramm erstellt
- Anwendungs-Projekt .Net Compact Framework mit einem Beispielprogramm
- Projekt zur Erstellung eines kryptographischen Aktivierungsschlüssels
- Projekt zur Einbindung eines erweiterten Lizenzschutzes

## 4.1 Das Basisprojekt

Das Basisprojekt stellt eine Grundlage für die schnelle Portierung und Erweiterung unserer Produkte von der nativen Win32 auf die .Net-Entwicklungsplattform zur Verfügung. Die zu realisierenden Anwendungen werden alle von diesem Basisprojekt abgeleitet und erben dadurch grundlegende Eigenschaften.

Das aus dem Basisprojekt durch den Compiler erzeugte Basisprogramm realisiert ein Ausgabeformular (Form) mit einem Auswahl- und einem Kontextmenü. Die Grafik des Basisprogramms gibt einfache geometrische Figuren in eine Basisbitmap und zur Grafikfläche der Form aus.

Über das Auswahlmenü kann über das „Basis“-Menüelement

- eine modale Dialogform mit drei Seiten aufgerufen werden:
  - „Basis“ enthält grundlegende Einstellungen für das Basisprogramm wie Hauptmenü- und Statusleiste ein- oder ausblenden, die Sichtbarkeit der Form und transparenter Hintergrund.
  - „Grafik“ mit grundlegenden Einstellungen für die Grafikausgabe wie SmoothingMode und PixelOffsetMode.
  - „Sprache“ beinhaltet die Sprachauswahl für die Programmbedienung und das Hilfesystem.
- die Form-Grafik geladen, ausgedruckt und abgespeichert sowie
- die Statusleiste ein- und ausgeblendet werden.

Über das „Hilfe“-Menü kann HTML-Hilfe für das Programm und die Menüauswahl aufgerufen werden und die Sprache für die Programmbedienung und das Hilfesystem ausgewählt werden. Zur Sprachauswahl stehen Deutsch, Englisch, Französisch, Italienisch, Spanisch und Portugiesisch.

## 4.2 Verwenden des Basisprojekts

Neue Projekte werden von der BASISPROJEKT-Klasse abgeleitet.

Der Konstruktor kann dreifach aufgerufen werden: ist der erste Parameter ein string, wird ein Kontrollablauf zur Abfrage eines Passwortes oder einer Lizenzdatei durchgeführt. Für die Abfrage einer Lizenzdatei ist der zweite Parameter true zu setzen.

Durch eine weitere Ableitung mit dem ersten Parameter true, wird der Licence Protector von Mirage-Systems eingebunden. Hierzu muss für das Basisprojekt unter Symbole für bedingte Kompilierung auch die Stringkonstante MirageLP zugefügt werden.

Zur Realisierung der nachfolgenden Punkte werden die Quellcode-Dateien des Basisprojekts (Basisprojekt.cs, Basis_Globals.cs, Basis_Procs.cs, Basis_Dialog.cs und Basis_Dialog_Texte.cs) als vorhandenes Element mit

dem abgeleiteten Projekt unter dem neu erstellten Ordner Basisprojekt verknüpft. Hierzu müssen die Dateien über den rechten Pfeil der Taste „Öffnen“ mit „als Verknüpfung hinzufügen“ ausgewählt werden.

In BASISPROJEKT.Basis wird die Methode Programm ständig aufgerufen. In ihr wird die Anwendung realisiert. Zur Abarbeitung anstehender Ereignisse wird Programm ständig verlassen. Programm ist virtuell definiert und kann somit durch eine gleichnamige Methode der Anwendung überschrieben werden.

Das Basisprogramm erstellt ein Hauptmenü mit den Punkten „Programm“, „Basis“ und „Hilfe“ und ein Kontextmenü, das auf Betätigung der rechten Maustaste angezeigt wird.

Die Anwendung erweitert das Basismenü durch Überschreiben der virtuellen Methode ErzeugeFormMenues um neue Punkte oder verändert bestehende Punkte beider Menüs. Hierzu wird vorab die überschriebene Methode des Basisprojekts aufgerufen.

Danach können die einzelnen Menüelemente über die öffentlichen Arraytexte direkt verändert und erweitert werden. Hierzu müssen auch die Dimensionen der Textarrays angepasst werden.

Der Titeltext wird in der gleichnamigen virtuellen Methode ermittelt und gesetzt. Der Aufruf erfolgt in ErzeugeFormMenues. Hierdurch wird er auch nach einer Sprachauswahl korrekt verändert. Die Anwendung überschreibt diese virtuelle Methode, wodurch der korrekte Text für alle Versionen der Anwendungen polymorph ermittelt wird.

Zum direkten Zeichnen auf dem Display wird ein GDI+ Graphics-Objekt in der Größe des Panels (wenn vorhanden und sichtbar) oder der Form zur allgemeinen Verwendung zur Verfügung gestellt. In der gleichen Größe wird für die Hintergrundgrafik eine Basisbitmap mit einem Graphics-Objekt erzeugt.

Zur Aktualisierung der Grafikfläche der Form wird in PanelOnPaint die vom Basisprogramm erzeugte Basisbitmap verwendet. Hierzu muss die Anwendung in der Basisbitmap immer die aktuelle Programmgrafik zur Verfügung stellen. Diese kann dann auch zum „Drucken...“ und „Speichern...“ verwendet werden. Die Anwendung zeichnet eine Raumgrafik mit Linien und Farben in die Basisbitmap.

Zum Anwendungs-Projekt gehört die ANWENDUNG1-Klasse und die im Konstruktor erzeugten Basisklassen ANW1_GLOBALS für öffentliche Projektdaten und EVENT_PROCS. EVENT_PROCS besteht aus Methoden, die innerhalb der Botschaften aufgerufen werden. In ihnen bearbeitet die Anwendung das Ereignis und zeichnet die Raumgrafik.

Die Dialogform der Anwendung mit drei Seiten (Tab-Pages), wird in ANW1_DIALOG erstellt. Die hierzu erforderlichen Texte werden in ANW1_DIALOG_TEXTE erzeugt.

# 4.3 Der Ablauf

Zum Programmstart wird im BASISPROJEKT-Konstruktor eine Instanz von BASIS_GLOBALS mit den öffentlichen Datenfeldern des Basisprogramms erzeugt.

Bevor die Form angezeigt wird werden danach auf OnLoad folgende Abläufe durchgeführt:

- Zur Protokollierung der Programmausnahmen wird eine globale StreamWriter-Instanz SWriter für eine Textdatei erzeugt. Der Stream wird erst auf OnClosed wieder geschlossen und steht somit im gesamten Programmablauf für die Ausnahmeprotokollierung zur Verfügung.
- die letzten Einstellungen des Programms werden in Programm_Init zurück geladen.
- eine Panel-Kontrolle mit Ziehleisten, eine Statusbar und das Haupt- und Kontextmenü für die Form werden erzeugt.
- ein Zeitgeber wird gestartet, der bei seinem Ablauf eine Schleife aufruft, in der das Programm bis zu seinem Ende verbleibt. Zum Programmschluss wird auf OnClosed die Schleifenvariable gelöscht, wodurch die Schleife verlassen und das Programm ordnungsgemäß beendet werden kann.

Wenn am Anfang der Programmschleife in einer binären Kontroll- oder Lizenzdatei kein gültiger String vorhanden ist, werden innerhalb des Kontrollablaufes zur Passwortabfrage oder zur Anforderung der Lizenzdatei mehrere Dialogfenster angezeigt.

Der Ablauf zur Anforderung der Lizenzdatei im Einzelnen:

- Nach dem Dialog zur Sprachauswahl wird der Lizenzdialog mit dem Lizenzvertragstext (EULA) in der ausgewählten Sprache angezeigt. Wenn dieser nicht akzeptiert wird, wird der weitere Ablauf abgebrochen und es läuft nur die Testversion des Programms.
- Wird der Lizenzvertrag akzeptiert, kann der Kunde innerhalb der nächsten Dialogform über Radiobutton auswählen, ob die Registrierung für alle Benutzer des Computers oder nur für den gerade angemeldeten Benutzer gültig sein soll. Der ausgewählte Registrierungstext wird in die Zwischenablage kopiert und muss dann per E-Mail an den Verkäufer gesendet werden. Der Registrierungstext besteht aus dem Produktnamen mit Versionsnummer und dem Dateipfad der Anwendung für alle Benutzer oder für den gerade aktuellen Benutzer mit dem Erzeugungsdatum des Ordners.
- Nach Zahlungseingang wird mit dem kleinen Zusatzprogramm aus dem CryptoKeyGenerator-Projekt aus dem Registrierungsschlüssel zusätzlich mit einem internen Projektpasswort der Aktivierungsschlüssel erzeugt. Dieser wird in einer Lizenzdatei an den Kunden zur Freischaltung der Anwendung übermittelt.

Danach werden zur Realisierung des Programmablaufs in Programmschleife ständig die Botschaften an das Programm abgearbeitet und die Methode Basis aufgerufen. In Basis wiederum wird die von der Anwendung zu überschreibende Methode Programm aktiviert. Die Abfrageanzahl der Programmereignisse in jeder Sekunde kann über die Variable EreignisAbfrageMax begrenzt werden (Standardwert 50). Hierdurch steht dem Programm die Computer-Leistung auch kontrolliert zur Verfügung.

Nachdem die Form sichtbar ist und nach jeder Änderung der Formgröße wird auf OnResize ein Grafik-Objekt für die Form und eine Basisbitmap mit dem zugehörigen Grafik-Objekt erzeugt.

Zur Anzeige eines Bildes oder Ausgabe einer Grafik die größer als die Grafikfläche der Form ist, muss die Anwendung die Basisbitmap mit dem zugehörigen Grafik-Objekt in der erforderlichen Größe selbst erzeugen. Des Weiteren muss die öffentliche Instanzvariable Grafik_Anzeige oder Image_Anzeige true gesetzt werden. In OnResize wird dann keine Basisbitmap erzeugt.

Zur Ablaufkontrolle zeichnet das Basisprogramm einfache geometrische Figuren in die Basisbitmap und zum Display. Die Anwendung stellt verschiedene Raumgrafiken zur Auswahl und zeichnet auf dem Raumboden geometrische Figuren.

Durch Auswahl des Menüpunktes „Dialog“ der Anwendung wird eine modeless Dialogform mit zwei Seiten angezeigt: auf der Basis-Seite wird die Aufruffrequenz der Anwendung durch einen auf OnLoad erzeugten Sekunden-Timer ausgegeben und eine Taste zur Information über die Anwendung eingefügt. Die Grafik-Seite ermöglicht die Auswahl der Raumgrafik.

## 4.4 Der Ablauf mit mehreren Threads

Eine erzeugte Programm-Instanz startet im Bedienerthread. Aufgrund der Abarbeitung der Botschaften im Bedienerthread wird dieser im weiteren Ablauf auch Botschaftsthread genannt oder auch als User-Interface-Thread (UI-Thread) bezeichnet. Bei einem Ablauf mit mehreren Threads erzeugt das Programm zusätzlich zu seinem Bedienerthread explizit einen oder mehrere Arbeitsthreads.

Gleichzeitig zu den Botschaften für die Form verrichtet der Arbeitsthread weitere Aufgaben. Aus diesem Grund darf der Arbeitsthread nicht auf Objekte zugreifen, wenn diese gleichzeitig im Botschaftsthread verwendet oder neu erstellt werden. Um dieses zu Vermeiden, stellt der Bedienerthread in ThreadSynchroStart eine Synchronisationsanforderung an den Arbeitsthread und wartet solange bis dieser diese bestätigt. Nach dem Objektzugriff im Bedienerthread wird in dem darauf erforderlichen Aufruf von ThreadSynchroEnde die Synchronisationsanforderung wieder zurückgesetzt.

Eine Synchronisierung des Bedienerthread mit dem Arbeitsthread ist erforderlich:

- In PanelOnResize des Basisprojekts vor dem Erzeugen eines neuen Grafikobjekts und der Basisbitmap.
- Auf PanelOnPaint des Basisprojekts vor der Verwendung der Basisbitmap.
- In OnClosed für den Basis-Dialog vor der Zuweisung der Daten.
- Auf PanelOnResize der Anwendung vor dem Neuzeichnen des Hintergrundes in die Basisbitmap in Groessen.

Komponenten und somit auch Dialogformen müssen im Bedienerthread erzeugt werden. Nur hierdurch können die im Bedienerthread ablaufenden Methoden auf die darin vorhandenen Steuerelemente zugreifen!

Vor der Erzeugung von Komponenten sollte deshalb abgefragt werden, ob die Erzeugermethode im Arbeitsthread ausgeführt wird und somit auf den Bedienerthread umgeleitet werden muss. Zur Umleitung wird die Erzeugermethode durch eine der Invoke-Methoden synchron oder asynchron als Delegat ausgeführt.

Delegaten werden in allen Projekten am Anfang der Methoden zur Erzeugung der Menüs erzeugt und ausgeführt. Des Weiteren im Basisprojekt in Animated_Bitmap zur Animation von geladenen Bitmaps.

Die Bedingte Kompilierungskonstante MULTITHREAD ermöglicht den Ablauf der Programme mit mehreren Threads.

Weitere Anweisungen mit Bedingter Kompilierung innerhalb von #if MULTITHREAD ... #endif:

- Alle using-Anweisungen die den System.Threading-namespace einbinden.
- Die Referenz auf den Arbeitsthread und Basisprojekt-Instanzvariable zur Synchronisierung der Threads.
- Die Startmethode für den erzeugten Arbeitsthread MT_Programmschleife mit den Synchronisationsmethoden ThreadSynchroStart und ThreadSynchroEnde.
- In OnLoad wird im Basisprojekt der Arbeitsthread mit MT_Programmschleife als Startmethode erzeugt.
- Zum Verlassen von Programmschleife für die Einzelversionen nach einem erforderlichen Kontrollablauf.
- Nachdem in OnClosed die Schleifenvariable false gesetzt wurde, wird vor der Freigabe von Programmobjekten das Ende des Arbeitsthreads abgewartet.

## 4.5 Die Projekte für mobile Geräte

Für mobile Geräte unter dem .Net Compact Framework wurden dem DotNet_Projektworkspace zwei Visual C# Smart Device Projekte zugefügt: SmartDevice-Basisprojekt und SmartDevice-Anwendung1.

Die Programme werden über Bedingte Kompilierung und angepasste Kodierung in den auch für das .Net Framework verwendeten C#-Dateien realisiert.

Die Bedingte Kompilierung erfolgt mit der Kompilierungskonstanten „PocketPC“.

Alle Veränderungen an der Kodierung für einen Ablauf auf mobilen Geräten unter dem .Net Compact Framework sind in der Datei Compact-Framework Kodierungsanpassungen.doc zusammengefasst.

## 4.6 Der Aktivierungsschlüssel

Im CryptoKeyGenerator-Projekt wird aus dem Registrierungsschlüssel und einem Projektpasswort eine binäre Lizenzdatei "jk-ware_Lizenzdatei.bin" mit dem Aktivierungsschlüssel erstellt.

Hierzu stellt das Projekt eine Form als FixedToolWindow zur Verfügung. Innerhalb der Form wird in zwei Textboxen der von den Programmen erzeugte Registrierungsschlüssel und das Projektpasswort eingegeben oder aus der Zwischenablage eingeholt. Daraus wird über die Taste „Erzeugen“ eine Lizenzdatei mit dem Aktivierungsschlüssel generiert.

Die Programme überprüfen in der Kontrollablauf-Methode der LIZENZSCHUTZ-Klasse den Aktivierungsschlüssel in der Lizenzdatei. Ist die Lizenzdatei nicht vorhanden oder der Aktivierungsschlüssel nicht korrekt, wird als ein Angebot zum Erwerb einer Lizenzdatei der Kontrollablauf durchgeführt.

## 4.7 Allgemeine Hinweise zur Kodierung

- Für die Übersetzungen des DotNet_Projektworkspace wurde Microsoft Visual Studio 2008 Version 9.0.30729.1 SP mit Microsoft .NET Framework Version 3.5 SP1 und Visual C# 2008 verwendet.
- Allen .Net-Projekten wird als ausreichendes Zielframework .Net Framework 2.0 zugewiesen.
- Zur Erstellung von CLS-kompatiblen Code (CLS = Common Language Specification) wird dem Basisprojekt die Anweisung [assembly: CLSCompliantAttribute(true)] zugefügt. Allen Klassen muss des Weiteren die Anweisung [CLSCompliantAttribute(true)] vorangestellt werden. Zur korrekten Ausführung durch den Compiler müssen die Klassen als public deklariert sein.
- Der C#-Quellcode beinhaltet für die Anzeige im Objektkatalog eine optimierte XML-Dokumentation.
- In InitializeComponent werden durch Neuzuweisungen der Formgröße implizit OnResize-Ereignisse ausgelöst. Um einen Ausnahme-Fehler, z.B. wegen noch nicht erzeugter Objekte zu vermeiden, wird der Aufruf von InitializeComponent unter OnLoad kontrolliert eingefügt.
- Die Klassen mit öffentlichen Datenfeldern für die Anwendung und für das Basisprojekt erben von Klassen mit multilingualen Texten. Hierzu gehören allgemeine Projekttexte, Texte für Fehlermeldungen in Messageboxen und Menütexte. In der MDI-Fensterverwaltung werden nicht für jede gestartete Anwendung öffentliche Texte erzeugt, weil diese als static string-Arrays deklariert sind. Die Texte sind dadurch nur einmal im Speicher vorhanden, werden aber für jede Instanz der öffentlichen Daten neu initialisiert.
- Die Grafikfläche der Form wird als Panel organisiert. Hierdurch werden die Ziehleisten korrekt mit der Statusleiste in die Form integriert und eine geteilte Form ermöglicht.
- Bei der Erweiterung des Hauptmenüs darf der Basismenüpunkt nicht verändert werden. Bei einer Veränderung der Position des „Pause !“-Menüpunktes unterhalb von „Programm“, ist innerhalb der Eigenschaft für diese Instanzvariable die Checkmarke neu festzulegen. Dies gilt ebenso für die Checkmarke der Statusleiste, die in der Eigenschaft für die Instanzvariable StBarAnzeige verändert wird.
- Für einen einfachen Zugriff auf die Ressourcen des Basisprojekts und der Anwendung wurde der Standardnamespace unter allgemeine Eigenschaften des Projekts entfernt.

### 4.7.1 Erweiterungen und Änderungen zur Version 3.0 des Basisprojekts

- Für moderne Geräte (Tablets, Phones) mit multicore-Prozessoren werden Endversionen für das .Net (Compact) Framework mit Multithreadablauf realisiert. Hierzu wurden vom MDI-Projektworkspace von allen relevanten Dateien die Quelltexte mit dem Symbol für Bedingte Compilierung MULTITHREAD in die Dateien des Basisworkspace übernommen. Eine Verlinkung der Dateien im Projektworkspace auf die Dateien im MDI-Projektworkspace kommt nicht in Frage, weil beide verschiedene Projektmappen unter Visual Studio und getrennte Vertriebsprodukte sind.
- Zugriff auf Licht- und Beschleunigungssensoren mit Methoden des Windows API Code Pack in der Datei WAPI_CodePack.cs.

- In Basis() wird in Aufrufkontrolle() AufrufFilterAnz nicht bei einer Spielpause neu berechnet.
- Kontrollmethode() gibt einen Rückgabewert zurück.
- Die Menüs werden in Programmschleife(), nach dem Aufruf von Kontrollmethode() neu erzeugt und nicht mehr in Kontrollmethode(), weil diese auch innerhalb des Programmablaufs aufgerufen wird und es hierbei zu Deadlocks kommen kann.
- Zum Sperren einzelner Menüpunkte werden für das Programm-Menü und das Kontextmenü Ereignisse für das Popup-Ereignis erzeugt. Zum Überschreiben durch die Anwendung werden für diese Ereignisse virtuelle Handler ohne Inhalt zur Verfügung gestellt.
- Der Zugriffsmodifikator für die Methoden Programm, Titeltext und InitializeComponent wird von public zu protected geändert
- Definition allgemeiner öffentlicher Strukturen Point3D, PointF und PointF3D in Basis_Globals.cs für einen Zugriff im ganzen Projekt.
- Neue virtuelle und inhaltslose Botschaftsmethode FormOnMove für das Move-Ereignis der Form zum Überschreiben durch die Anwendungen (das Move-Ereignis der Panels erzeugt keine Botschaften).
- In Programm_Init() werden wegen einer möglichen Ausnahme die Instanzen von BReader und BWriter innerhalb eines finally-blocks geschlossen.

## 4.7.2 Erweiterungen und Änderungen zur Version 2.0

- Für einen schnelleren Programmablauf in der Programmschleife wird die max. Anzahl der Ereignisabfragen auf 50 reduziert (vorher 100).
- Für eine schnellere Bedienbarkeit der Smart Devices (Pocket PC's) wird in der Programmschleife bei einer Programmpause durch Sleep(1) ein Threadwechsel ausgeführt.
- Separate Kontrollmethode für den Lizenzschutz. Der Aufruf erfolgt nach dem Programmstart in Programmschleife und zur zusätzlichen Lizenzkontrolle in den Menüereignissen Programm- und KontextMenueOnClick.
- Zur weitergehenden Anzeige im Objektkatalog und für die kontextsensitive Hilfe wurde der XML-Kommentar für alle Methoden mit <para>-Tags innerhalb der <summary>- und <remarks>-Tags versehen.
- Neben der Erhöhung der globalen Variablen catch_Ausnahmen, werden die innerhalb des Programmablaufes auftretenden Ausnahmen mit Datum und Uhrzeit in Ausnahmeprotokollierung der BASISPROJEKT-Klasse in eine Datei geschrieben. Hierzu wird im Basisprogramm auf OnLoad ein Ausgabestream für die Datei „Titeltext“ + “_Exceptions.txt“ erzeugt, der auf OnClosed wieder geschlossen wird.
- Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden in BPRJMAINCLASS.Main zwei Ausnahmehandler erzeugt, nach deren Aktivierung die Anwendung fortgeführt wird.
- Größenänderungen des Panels werden durch den in ErzeugeFormPanel erzeugten virtuellen Eventhandler PanelOnResize bearbeitet. Hierin werden auch das GDI+ Graphics-Objekt und die Basisbitmap erzeugt. Eine fehlerhafte Panel-Erzeugung in ErzeugeFormPanel wird nicht abgefangen und führt zum Programmabbruch (bisher noch nicht vorgekommen).
- Die Methoden OnPaint und OnMouseDown zur Bearbeitung der Form-Ereignisse Paint und MouseDown sind nicht mehr erforderlich und werden entfernt. Achtung! Für ein Neuzeichnen der Form in PanelOnPaint muss anstelle von Invalidate, FormPanel.Invalidate verwendet werden.
- Neue BASIS_GLOBALS-Instanzvariable Pause_Aussetzen. Wenn true, wird in der Basis-Methode der Pausestatus des Programms, z.B. zur Aktualisierung der Grafikanzeige, vorübergehend ausgesetzt ohne dass dabei die Pause-Menücheckmarken verändert werden.
- Aufgrund des neu eingeführten Eventhandlers PanelOnResize kann auch für Smart Devices die Basis-Methode in der Programmschleife aufgerufen werden und auf den Aufruf in PanelOnPaint verzichtet werden.
- In ErzeugeFormPanel wird für das Scroll-Ereignis des Panels der virtuelle ScrollEventHandler PanelOnScroll erzeugt (nicht für Smart Devices). Für die MouseDown- und MouseMove-Ereignisse werden die EventHandler PanelOnMouseDown und PanelOnMouseMove virtuell zum Überschreiben zur Verfügung gestellt.

- Neue BASIS_GLOBALS-Instanzvariable Pause_Aussetzen. Wenn true, wird in der Basis-Methode der Pausestatus des Programms vorübergehend ausgesetzt, z.B. zur Aktualisierung der Grafikanzeige, ohne dass dabei die Pause-Menücheckmarken verändert werden.
- Die Instanzvariablen Pause und StBarAnzeige der BASIS_GLOBALS-Klasse werden als Eigenschaften definiert, die zusätzlich die Checkmarken des Haupt- und Kontextmenüs mit verändern. Pause wird auch in Programm_Init als Initialisierungsdatum abgespeichert.
- Zum Beenden einer Bildanzeige wird für den Zugriff auf lokale Daten und Methoden die BASISPROJEKT-Methode Pause_Methode eingeführt, in der die Menücheckmarken korrigiert und die Ziehleisten verdeckt werden. Zur Anpassung der Panelgröße wird zusätzlich OnResize() aufgerufen. Der Aufruf erfolgt beim Setzen der Eigenschaft für die globale Instanzvariable Pause.
- In OnResize() wird bei einer Verkleinerung der Form zur Symbolgröße Pause nicht true gesetzt, weil hierdurch über die neue Pause_Methode eine Bildanzeige beendet wird.
- Die Instanzvariable SymbolSize der BASISPROJEKT-Klasse wird zur Instanzvariablen der BASIS_GLOBALS-Klasse.
- Ausgabe der Bitzahl der .Net-Platform in der über-Dialogbox des Menüs und der Basis-Dialogseite. Die Bitanzahl wird von der Länge eines int-Pointers abgeleitet.

## 4.7.3 Erweiterungen und Änderungen zur Version 1.1

- Neue Projekte für Smart Devices „SmartDevice-Basisprojekt" und "SmartDevice-Anwendung1" als leeres Projekt zum Erstellen von .Net Compact-Framework Anwendungen für die Windows CE Zielplattform hinzugefügt.
- Angezeigte Ziehleisten werden in BASISPROJEKT::OnKeyDown mit Pfeil- und Sondertasten der Tastatur bewegt. Unter dem .Net Compact-Framework werden die Ziehleisten nicht automatisch angezeigt. Deshalb erfolgt für Smart Devices die Bildnavigation in BASISPROJEKT::SmartDevice_KeyDown mit den Navigationstasten. Die Bildposition wird dann in PanelOnPaint realisiert.
- Die Druckerausgabe in BASISPROJEKT::BasisBitmapDrucken wird über Anzeigen der PageSetup- und darauf folgender PrintPreview-Dialogform durchgeführt, wenn die Ausgabe mit dem Standard Drucker-Dialog von der System.Windows.Forms.PrintDialog-Klasse nicht realisiert werden kann oder vom Anwender abgebrochen wird.
- Aufgrund von Problemen mit der Mausposition als Datenelement von HelpEventArgs zur Anzeige der erweiteren Hilfetexte in BASIS_DIALOG::BASIS_DIALOG_HelpRequested und ANW1_DIALOG:: Anw1_DIALOG_HelpRequested, wird die aktuelle Mausposition von der Control-Klasse verwendet.
- Der Lizenzablauf erhält einen neuen Konstruktor, dem als zweiten Parameter der Ordnerpfad der Anwendung übergeben wird. Dieser wird in BASISPROJEKT::Programm_Init auch für Smart Devices erzeugt und als Datenfeld der BASISPROJEKT-Klasse abgespeichert.
- In LIZENZSCHUTZ::Kontrollablauf wird für die Passwortabfrage in der binären Kontrolldatei der Kontrolltext aus dem Programmpfad und der Erstellungszeit des Programmordners gebildet.
- Die multilingualen Lizenztexte („EULA") für den Lizenzdialog in der LIZENZSCHUTZ-Klasse in Basis_Procs.cs auf den 05.06.2009 korrigiert. Hierbei wurden nur drei kleine grammatikalische Korrekturen an dem deutschen und englischen Text vorgenommen.
- Der Erläuterungstext für die Registrierungsschlüssel-Dialogform wurde verändert.

## 4.7.4 Für neue Versionen vorbehalten

- Die Main-Methode der CryptoKeyGenerator-Klasse für einen externen Server-Aufruf mit Argumenten für den Registrierungsschlüssel und das Passwort versehen. Hierzu den params-Modifizierer einsetzen (s. 7. Programmierrezepte für Visual C# .Net, S. 436f und 8. C# IT-Tutorial, S. 264f).
- Die mit dem Windows-Installer erstellten Installationsdateien für Windows unter .Net zusätzlich in einer einfach komprimierten Datei (z.B. im Zip-Format) für andere Plattformen bereitstellen, da der Windows-Installer ebenso wie InstallShield Express 11.0 keine Dateien in Intermediate Language erzeugt.
- Drag und Drop realisieren: beginnen mit einem Lernprogramm oder mit Petzold, Kapitel 24 oder der DataFormats-Klasse.

## 4.8 Literaturquellen


1. Moses, Nowak: C# Programmieren unter .Net, Franzis' 2002,
   ISBN 3-7723-7224-4: empfehlenswert zur Einführung in die Programmiersprache C#.

2. Petzold: Programming Windows with C#, Microsoft 2002, ISBN 0-7356-1370-2: empfehlenswert zur Einführung in die Form-Programmierung unter .Net.

3. Schwichtenberg, Eller: Programmierung mit der .Net-Klassenbibliothek, Addison-Wesley 2002, ISBN 3-8273-1905-6: zusätzlich zur Beschreibung der .Net-Klassenbibliothek von Microsoft wird als eBook ein kostengünstiger Überblick geliefert.

4. Maslo, Freiberger: .Net-Framework Developer's Guide, Markt&Technik 2002, ISBN 3-8272-6142-2: zusätzlich zur Beschreibung der .Net-Klassenbibliothek von Microsoft wird als eBook ein kostengünstiger Überblick geliefert.

5. Sells: Windows Forms Programming in C#, Addison-Wesley 2004, ISBN 0-321-11620-8: beschreibt tiefer gehend die Probleme der .Net-Programmierung für MDI- und Multithread-Umgebungen.

6. Bayer: Das C#-Codebook, Addison-Wesley 2003, ISBN 3-8273-2050-X, mit nützlichen C#-Codes unter .Net auch als kostengünstiges eBook.

7. Jones: Programmierrezepte für Visual C# .Net, Microsoft Press Deutschland, ISBN 3-86063-091-1, mit nützlichen Programmierrezepten zu Visual C# .Net.

8. Schildt: C# IT-Tutorial, mitp-Verlag, ISBN 3-8266-0834-8, zur Auffrischung der C#-Grundlagen

9. Kühnel: Das umfassende Handbuch zu Visual C# 2008 und Visual C# 2010, Galileo Computing als <openbook>

10. Wigley, Moth, Foot: Mobile Development Handbook, Microsoft Press 2007, ISBN 0-7356-2358-9, als Einführung für die Programmierung von mobilen Geräten unter dem .Net Compact Framework.

# 5 Die Projektentwicklung

## 5.1 Die Projektmappe

Die Projektmappe für den .Net Projektworkspace enthält sechs Projekte unter Microsoft Visual Studio 2008 Version 9.0.30729.1 SP. Die hieraus erstellte Microsoft Visual Studio Solution-Datei DotNet_Projektworkspace.sln wird auch von nachfolgenden Entwicklungsumgebungen akzeptiert und darauf umgewandelt. Allerdings sind die SmartDevice-Projekte nur unter VS2008 verfügbar.

Zur Kompilierung der Projekte wurde laut Anzeige des Info-Dialogs unter dem Hilfe-Menüpunkt Microsoft Visual C# 2008 mit .NET Framework Version 3.5 SP1verwendet. Unter Anwendung der Projekteigenschaften wird für die .Net-Projekte als Zielframework .NET Framework Version 2.0 eingesetzt.

Die sechs Projekte sind in alphabetischer Reihenfolge Anwendung1, Basisprojekt, CryptoKeyGenerator, LicenceProtectorSample, SmartDevice-Anwendung1 und SmartDevice-Basisprojekt.

## 5.2 Programmierung in C#

Die Programme sollen die Anforderungen für alle vorhandenen und zukünftigen .Net-Plattformen erfüllen und einen Ablauf unter der Common Language Runtime ermöglichen.

Hierzu ist allen Klassendeklarationen das [CLSCompliantAttribute(true)] vorangestellt. Es ist zu beachten, dass der Zugriff auf die Klassen public ist, da sonst der Compiler die Überprüfung nicht durchführen kann.

Der Quellcode wird mit #region und #endregion in Regionen unterteilt. Hierdurch wird ein schneller und gezielter Zugriff auf die einzelnen Anweisungen des Programms möglich.

Die Beschreibungen für alle Klassen, Methoden und öffentlichen Variablen sind im Quellcode über den Deklarationen als XML-Kommentar aufgeführt. Zur weitergehenden Anzeige im Objektkatalog und für die kontextsensitive Hilfe wurde der XML-Kommentar für alle Methoden mit <para>-Tags innerhalb der <summary>- und <remarks>-Tags versehen.

Neben der Erhöhung der globalen Variablen catch_Ausnahmen, werden die innerhalb des Programmablaufes auftretenden Ausnahmen mit Datum und Uhrzeit in Ausnahmeprotokollierung der BASISPROJEKT-Klasse in eine Datei geschrieben. Hierzu wird im Basisprogramm auf OnLoad ein Ausgabestream für die Datei „Titeltext“ + “_Exceptions.txt“ erzeugt, der auf OnClosed wieder geschlossen wird.

Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden in BPRJMAINCLASS.Main zwei Ausnahmehandler erzeugt, nach deren Aktivierung die Anwendung fortgeführt wird.

## 5.3 Die Ressourcen

Alle Ressourcen werden zur Absicherung in ein gleichnamiges Verzeichnis innerhalb des Projektworkspace abgelegt.

Zur Einbindung in das Projekt müssen die *.resx-Dateien geöffnet und die erforderlichen Ressourcen darin eingefügt werden. Hierdurch werden die Dateien nochmals in ein von Visual C# erzeugtes Projekt-Verzeichnis „Resources“ abgelegt.

Diese Ressourcen werden in die Assemblies der Projekte eingebunden. Es handelt sich vorerst nur um Symbole (Iconen) und Bitmaps (Images).

Der Zugriff auf die Ressourcen erfolgt über die Anweisungen

```
System.ComponentModel.ComponentResourceManager resources = new
                System.ComponentModel.ComponentResourceManager(typeof(jkBPrj.BASISPROJEKT));
Icon = resources.GetObject("Basis1") as Icon;
Image = resources.GetObject("logoPictureBox.Image") as Image;
```

Zur Ermittlung der Type-Klasse der einzuholenden Ressource kann folgende Anweisungsfolge verwendet werden:

```
System.Reflection.Assembly assem = System.Reflection.Assembly.GetExecutingAssembly();
foreach (string resourceName in assem.GetManifestResourceNames() )
   MessageBox.Show (resourceName, Text);
```

Hierdurch werden die Namen aller in der Assembly vorhandenen Ressourcen in einer Messagebox ausgegeben.

Weitere Zugriffmöglichkeiten bieten folgende Anweisungen

```
System.Reflection.Assembly assem = System.Reflection.Assembly.GetExecutingAssembly();
System.Resources.ResourceManager resman = new
        System.Resources.ResourceManager("jkBPrj.BPRJMAINCLASS", assem);
string ResText = (string)resman.GetObject("TestString1");
MessageBox.Show (ResText, Text);
Icon BasisIcon = resman.GetObject("Basis1") as Icon;
```

Oder mit einer Stream-Anweisung

```
System.IO.Stream stream = assem.GetManifestResourceStream("Anw1.ico");
if (stream != null)
{
        BPrj.Icon = new Icon(stream);
        stream.Close();
}
```

Für einen erfolgreichen Zugriff als Stream, sind die Ressourcen-Dateien über die Buildvorgang-Eigenschaft dem Projekt explizit als „eingebettete Ressource“ anzugeben.

Achtung: Wenn die „eingebetteten Ressourcen“ zusätzlich in einer .resx-Datei aufgeführt sind, werden diese Dateien zweimal in die .exe-Datei der Anwendung eingebunden, wodurch sich die Dateigröße u.U. merklich erhöht.

Hinweis 2 zum Icon der Anwendung!

Das Anwendungsicon nimmt eine Sonderstellung unter den Ressourcen ein. Es muss in den Projekteigenschaften der Anwendung unter Symbol und Manifest explizit angegeben werden. Hierdurch wird vom Compiler im Projektordner nochmals eine Kopie des Icons abgelegt, die nicht entfernt werden darf.

## 5.4 Der Entwurf der Programm-Formen

Die Programm-Formen werden manuell ohne Designer-Unterstützung entworfen.

Dies betrifft insbesondere den Entwurf der multilingualen Menüs und deren Erweiterungen durch die abgeleiteten Anwendungen.

Weitere Probleme in Visual Studio Express:

- die Designer-Ansicht für die Programm-Formen steht nicht immer zur Verfügung.
- Im DEBUG-Ablauf entsteht beim Schließen der Designer-Ansicht für die Anwendung1 durch die Grafikausgabe eine sich wiederholende catch-Ausnahme. Diese kann verhindert werden, wenn die Ausgabe der Messagebox in BASISPROJEKT.Programm entfernt wird.
- Der Designer übernimmt die Ressourcenverwaltung, wodurch selbst verwaltete Ressourcen (Icon) entfernt werden.

### 5.4.1 InitializeComponent

Als exemplarisches Beispiel wird hier InitializeComponent des Basisprojekts aufgeführt.

Weil die Form des Basisprogramms mit den Menüs manuell codiert wird, darf der Inhalt der Methode manuell mit dem Code-Editor verändert werden. Auf die Designerunterstützung wird wegen der aufgeführten Probleme verzichtet.

Der Aufruf erfolgt wegen eines implizit ausgeführten OnResize-Ereignis nicht im Konstruktor.

Kodierung 5-1: BASISPROJEKT.InitializeComponent

```
///**************************************************************************
/// Version 3.0 vom 16.11.09
///**************************************************************************
protected virtual void InitializeComponent()
{
        // BASISPROJEKT
    this.SuspendLayout();
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Inherit;
    this.BackColor = System.Drawing.Color.Bisque;
    this.Name = "BASISPROJEKT";
    this.Text = "Basisprojekt";
    this.ResumeLayout(false);
}
```

## 5.5 Der Entwurf der Dialog-Formen

Alle Dialogformen werden mit Designer-Unterstützung entworfen.

Die Kodierung wird automatisch durch den Designer in InitializeComponent erstellt und deshalb in diesem Handbuch nicht aufgeführt.

# 6 Das Basisprojekt

Das Basisprojekt stellt eine Form mit vielfältigen Eigenschaften zur Vererbung bereit. Diese Eigenschaften sind

- ein Hauptmenü
- ein Kontextmenü
- eine Statusleiste
- ein Panel mit automatischen Ziehleisten
- Drucken der Programmgrafik
- Speichern der Programmgrafik
- Laden einer Grafikdatei
- Lizenzschutz

## 6.1 Die Klassen

Das Basisprojekt besteht aus den Klassen BPRJMAINCLASS, BASISPROJEKT, BASIS_GLOBALS, BASIS_TEXTE, BASIS_MENUE_TEXTE, den Klassen für die Basisdialogform BASIS_DIALOG, BASIS_DIALOG_TEXTE, Dialogformklassen zur Informationsausgabe ABOUTBOX, UEBERDIALOG und INFO_DIALOG und den LIZENZSCHUTZ-Klassen mit den Dialogformklassen SPRACHE_DIALOG, LIZENZ_DIALOG, PASSWORT_DIALOG, REGISTRIERUNGS_DIALOG und AKTIVIERUNGS_DIALOG.

In der BASISPROJEKT-Klasse stehen über die Bedingte Kompilierungskonstante WAPICP noch Methoden für den Zugriff auf Sensoren über das Windows API Code Pack zur Verfügung.

Alle Klassen sind in den Namensraum jkBPrj eingeschlossen.

## 6.2 Die BPRJMAINCLASS-Klasse

BPRJMAINCLASS enthält neben der statischen Methode Main, die durch den Compiler als Startobjekt für das Basisprogramm aufgerufen wird, noch zwei Methoden als Handler von globalen Ausnahmen.

Für einen kontrollierten Ablauf wird in der statischen Main-Methode in einem try-catch-Block ein BASISPROJEKT-Objekt mit einer separaten Anweisung erzeugt und die Standardmeldungsschleife für das Basisprogramm in einer zweiten Anweisung mit Application.Run gestartet, wodurch auch die Form sichtbar und bedienbar wird. Nach einer hierin abgefangenen und protokollierten Ausnahme wird das Basisprogramm beendet.

Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden deshalb zwei Ausnahmehandler erzeugt, nach deren Aktivierung das Basisprogramm fortgeführt werden kann.

Kodierung 6-1: Die BPRJMAINCLASS-Klasse

## 6.3 Die BASISPROJEKT-Klasse

Eine Instanz dieser Klasse wird mit new jkBPrj.BASISPROJEKT von Form ableitend erzeugt und stellt hierdurch mit den eigenen Klassen eine grundlegenden Basis für eine Anwendung zur Vererbung bereit.

Die Basisprojektklasse wird in WAPI_Codepack.cs um Methoden zur Erfassung von Sensordaten mit dem Windows API Code Pack erweitert (s. Kap. 14).

Für einen unterschiedlichen Lizenzschutz-Ablauf sind drei verschiedene Konstruktoren vorhanden.

Diese Klassendefinition muss an erster Stelle des Quellcodes stehen, sonst ist keine Designer-Unterstützung der Form möglich.

Hinweis 3 zur Designer-Ansicht der Programme!

Im DEBUG-Ablauf entsteht beim Schließen der Designer-Ansicht für die Anwendungungen durch die Grafikausgabe eine sich wiederholende catch-Ausnahme. Um diese zu verhindern, wird eine Ausnahmemeldung in BASISPROJEKT.Programm nicht in einer Messagebox sondern mit Debug.WriteLine() zum Ausgabefenster ausgegeben.

Kodierung 6-2: Die BASISPROJEKT-Klasse

```
#region BASISPROJEKT-Klasse
///*****************************************************************************
/// Version 3.0 vom 10.05.14
///*****************************************************************************
[CLSCompliantAttribute(true)]
public partial class BASISPROJEKT : Form
{

    #region Datenfelder und Konstruktoren #endregion

    #region Programmschleifen, Titeltext und InitializeComponent #endregion

    #region Basisprogramme, Pause_Methode, Ausnahmeprotokollierung  und Programm-Initialis. #endregion

    #region StatusBar, FormPanel, Haupt- und Kontext-Menü erzeugen #endregion

    #region Bearbeitung der Form-Ereignisse #endregion

    #region Bearbeitung der Menü-Ereignisse #endregion

    #region Speichern, Laden und Drucken der BasisBitmap #endregion

    #region Sensoren über das Windows API Code Pack einbinden #endregion

}
#endregion
```

## 6.3.1 Die Instanzvariablen

Auf die implizit privaten Instanzvariablen kann nur innerhalb der BASISPROJEKT-Klasse zugegriffen werden. Einzig public ist die Referenz auf die in den Konstruktoren erzeugte Instanz von BASIS_GLOBALS für einen Zugriff auf die öffentlichen Datenfelder des Basisprojekts aus den abgeleiteten Klassen. Des Weiteren ist das interne Datenfeld der Aufrufkontrolle Sekunden und DirectoryPath für einen informativen Lesezugriff public.

Kodierung 6-3: die Instanzvariablen der BASISPROJEKT-Klasse

## 6.3.2 Die Konstruktoren

Wird BASISPROJEKT ohne Parameter erzeugt, wird kein Lizenzschutz durchgeführt, weil das öffentliche Datenfeld Lizenz_Ablauf false initialisiert ist.

Der Lizenzschutz von jk-ware wird durchgeführt, wenn der erste Parameter ein string ist. Dieser string wird als Passwort oder als Projektschlüssel verwendet.

Ist der zweite Parameter true, wird ein Lizenzschutz mit Registrierungsschlüssel durchgeführt. Mit einem Zusatzprogramm aus dem CryptoKeyGenerator-Projekt wird hieraus zusammen mit dem Passwort eine

Lizenzdatei erstellt. Ist der zweite Parameter false, wird das Programm nur durch eine Passwortabfrage geschützt.

Ist der erste Parameter bool, wird ein erweiterter Lizenzschutz mit dem Licence Protector der Mirage GmbH durchgeführt.

Der Konstruktor wird direkt in Main oder als abgeleitete Basisklasse bei der Erzeugung einer Anwendung aktiviert.

Alle Konstruktoren erstellen eine Instanz von der BASIS_GLOBALS-Klasse für die öffentlich zugänglichen Datenfelder des Basisprojekts.

Kodierung 6-4: der BASISPROJEKT-Konstruktor

Hinweis 4 zu den Ein- und Ausgabedaten der Kodierungen!

Wenn die Ein- und Ausgabedaten keiner Klasse zugeordnet sind, handelt es sich um Instanzvariable (Datenfelder) der aktuellen Klasse.

## 6.4 Die öffentlichen Datenfelder des Basisprojekts

Datenfelder, die für andere Klassen öffentlich zugänglich sein müssen, werden in einer eigenen Klasse BASIS_GLOBALS aufgeführt. Auf diese Datenfelder kann hierdurch über eine separate Referenz zugegriffen werden, die auch austauschbar ist. Die BASIS_GLOBALS-Klasse wird in den Konstruktoren des Basisprojekts erzeugt.

Im Konstruktor werden die öffentlichen Datenfelder des Basisprojekts initialisiert und ein Generator für Pseudozufallszahlen erzeugt.

Von BASIS_GLOBALS werden automatisch Objekte von BASIS_TEXTE und BASIS_MENUE_TEXTE als Basisklassen abgeleitet. Diese enthalten statische zwei- oder dreidimensionale Arrays in denen die mehrsprachigen Texte für das Basisprojekt und das Menü eingetragen werden.

Die Menütexte enthalten zusätzlich einen kleinen Erläuterungstext zur Menüauswahl, der für die Ausgabe zur Statusleiste verwendet wird.

Kodierung 6-5: die BASIS_GLOBALS-Klasse

### 6.4.1 Mehrsprachigkeit

Die mehrsprachige Bedienung der Programme wird über statische zweidimensionale string-Arrays realisiert, die als öffentliche Datenfelder zugänglich sind. Statische Arrays haben den Vorteil, dass sie auch bei mehreren erzeugten Instanzen nur einmal im Speicher vorhanden sind.

Beispiele:

Eine Messagebox mit einer mehrsprachigen Nachricht wird mit MessageBox.Show (BASIS_TEXTE.MessageBoxTexte[User_Language-1,1], Text) angezeigt. Hierbei ist MessageBox.Show eine Methode der System.Windows.Forms-Klasse. Text übergibt den Titeltext der Form. BASIS_TEXTE.MessageBoxtexte[,] wählt ein Element eines zweidimensionalen statischen Arrays aus. User_Language-1 ist die ausgewählte Textsprache und erst die darauf folgende eins gibt den Text für die Messagebox an.

Eine mehrsprachige Textausgabe zum Panel1 der Statusleiste erfolgt in dieser Art und Weise mit der Anweisung StBarPanel1.Text = BASIS_TEXTE.BasisTexte [BGlobals.User_Language-1,2]. Mit BGlobals als

Instanz der öffentlichen Datenfelder, die außerhalb der BASISPROJEKT-Klasse für den Zugriff auf die Texte erforderlich ist.

### 6.4.2 Die mehrsprachigen Basistexte

Die BASIS_TEXTE-Klasse enthält allgemein zugängliche multilinguale Texte für das Basisprojekt und erbt BASIS_MENUE_TEXTE.

Im Konstruktor werden die allgemeinen multilingualen Texte für das Basisprojekt in den Textarrays bereitgestellt.

Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

Kodierung 6-6: Die BASIS_TEXTE-Klasse

### 6.4.3 Die multilingualen Texte des Basismenüs

Die BASIS_MENUE_TEXTE-Klasse enthält allgemein zugängliche multilinguale Texte für das Basismenü mit einem kleinen Erläuterungstext zur Menüauswahl. Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

Kodierung 6-7: Die BASIS_MENUE_TEXTE-Klasse

## 6.5 Das Panel

Für eine korrekte Gruppierung der Ziehleisten und der Statusleiste innerhalb der Form muss eine Panel-Kontrolle erzeugt werden. Hierdurch kann die Grafikfläche mit Ziehleisten ohne die Statusleiste verschoben werden.

Eine unkorrekte Erzeugung der Panel-Instanz führt zum Abbruch der Anwendung (bisher noch nicht vorgekommen).

Hinweis 5 zu den Ziehleisten!

Weil die Statusleiste zur Grafikfläche der Form gehört, wird diese mit verschoben, wenn die Form mit Ziehleisten erzeugt wird. Hierbei handelt es sich noch um eine grundlegende Eigenart der Windows-Programmierung mit leider aufwendigen Folgen für den Programmierer.

Zur Behandlung der Panel-Ereignisse Resize, Paint, MouseDown, MouseMove und Scroll werden die Methoden PanelOnResize, PanelOnPaint, PanelOnMouseDown, PanelOnMouseMove und PanelOnScroll als Handler eingetragen (PanelOnScroll nicht für Smart Devices).

Nicht für Smart Devices wird für das Move-Ereignis der Form die Ereignismethode FormOnMove als Handler eingetragen, weil das Panel selbst kein Move-Ereignis erzeugt.

Für Smart Devices wird ein Pixel-Button erzeugt und in die untere rechte Ecke des Panels vorerst unsichtbar platziert. Zur Anzeige der Ziehleisten muss dieser Button auf PanelOnPaint neu platziert und sichtbar gemacht werden.

Wegen einer Bildverkleinerung durch die Ziehleisten erfolgt die Bildverschiebung aber mit den Navigationstasten der SmartDevice.

Der Aufruf erfolgt im Form-Ereignis OnLoad.

Kodierung 6-8: ErzeugeFormPanel

## 6.6 Die Statusleiste

Zur Anzeige von Programminformationen wird eine Statusleiste mit einem Panel erzeugt (nicht für PocketPC).

Dem Panel wird ein erster Text und übergeordnet ein Tooltipptext zugewiesen.

Im Compact-Framework kann kein Panel erzeugt werden. Deshalb wird nur der Text an die Statusleiste selbst ausgegeben.

Die Statusleiste wird nur angezeigt, wenn das öffentliche Datenfeld StBarAnzeige true ist.

Der Aufruf erfolgt im Form-Ereignis OnLoad.

Kodierung 6-9: ErzeugeStatusbar

## 6.7 Der Titeltext

Schreibt den mehrsprachigen Text der Titelleiste der Form des Basisprogramms oder einer überschreibenden Anwendung.

Der Aufruf erfolgt im Basisprojekt auf OnLoad und in ErzeugeFormMenues.

Kodierung 6-10: Titeltext

## 6.8 Die Grafikobjekte

Das Basisprojekt stellt zwei Grafikobjekte für die Ausgabe zur Grafikfläche (Client Area) des Panels und zur Ausgabe in eine Basisbitmap in der Größe der Grafikfläche zur Verfügung.

Die Anwendung kann über diese Grafikobjekte entweder direkt in die Client Area zeichnen oder alles zuvor in eine Basisbitmap ablegen, um diese dann insgesamt zur Client Area zu übertragen.

Die Grafikobjekte werden bei einer neuen Panelgröße in der in ErzeugeFormPanel für das Resize-Ereignis des Panels eingetragenen Methode PanelOnResize neu erzeugt.

Ist die zu zeichnende Grafik größer als die Grafikfläche der Form, muss die Basisbitmap mit dem davon abgeleiteten Grafikobjekt von der Anwendung selbst erzeugt werden. Dazu muss die öffentliche Variable Grafik_Anzeige true sein, wodurch in PanelOnResize keine Basisbitmap erzeugt wird.

## 6.9 Bearbeitung der Form-Ereignisse für das Basisprojekt

### 6.9.1 OnLoad

In OnLoad werden das Haupt- und Kontextmenü, eine Statusbar, eine Panel-Kontrolle über der Form und die Instanz zum Schreiben der Fehlermeldungen in eine Datei erzeugt.

Des Weiteren werden die Startdaten für das Programm aus der Initialisierungsdatei eingelesen und das Symbol für das Basisprogramm aus den Ressourcen geladen.

Nur für MULTITHREAD wird ein Arbeitsthread mit MT_Programmschleife als Startmethode und ein Zeitgeber zur Aktivierung von Programmschleife gestartet.

Anm.: In Programmscheife wird für den Ablauf mit MULTITHREAD nur ein erforderlicher Kontrollablauf durchgeführt.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal vor dem Load-Ereignis aufgerufen, bevor die Form angezeigt wird.

Kodierung 6-11: OnLoad für das Basisprojekt

## 6.9.2 OnResize

OnResize überschreibt die geerbte Form-Methode und durchläuft, wenn die Form nicht zum Symbol verkleinert oder wieder hergestellt wird, zur Aktivierung der eingetragenen Resize-Handler die Basismethode.

OnResize wird bei einer Größenänderung der Form vor dem Resize-Ereignis aufgerufen.

Kodierung 6-12: OnResize für das Basisprojekt

## 6.9.3 PanelOnResize

PanelOnResize erzeugt zur Ausgabe direkt zum Display zur allgemeinen Verwendung ein GDI+ Grafik-Objekt, das in der neuen Größe des Panels erstellt wird.

In dieser Größe wird auch eine Basisbitmap mit einem davon abgeleiteten GDI+ Grafik-Objekt erstellt. Die Basisbitmap wird aber nicht erstellt, wenn sie extern zur Verfügung gestellt wird (Grafik_Anzeige true) oder eine geladene Bilddatei angezeigt wird (Image_Anzeige true). In beiden Fällen wird PanelOnResize vorzeitig verlassen.

Nicht für Smart Devices (PocketPC): für die erzeugten Grafik-Objekte, wird die Grafikqualität und der Pixeloffset entsprechend der Eingabedaten gesetzt.

Für MULTITHREAD wird vor der Behandlung der Grafik-Objekte der Bedienerthread mit dem Arbeitsthread synchronisiert.

PanelOnResize wird bei einer Größenänderung des Panels als eingetragener Handler für das Resize-Ereignis aufgerufen.

Kodierung 6-13: PanelOnResize für das Basisprojekt

## 6.9.4 PanelOnPaint

PanelOnPaint gibt die Basisbitmap zur Client Area des Panels aus. Hierbei wird die AutoScroll-Position der Ziehleisten berücksichtigt, wenn die Bitmap eine andere Größe als das Panel hat (Grafik_Anzeige oder Image_Anzeige true).

Für MULTITHREAD wird vor dem Zugriff auf das Grafik-Objekt der Bedienerthread mit dem Arbeitsthread synchronisiert.

PanelOnPaint ist ein eingetragener Handler für das Paint-Ereignis des Panels und wird aufgerufen, wenn die visuelle Darstellung des Panels erneuert werden muss.

Für Smart Devices werden aufgrund einer fehlenden AutoScrollMinSize-Anweisung die Bildlaufleisten bei einem größeren Bild auch dann nicht angezeigt, wenn nur ein Bildausschnitt ausgegeben wird. Aus diesem Grund wird ein Bildausschnitt in der Größe der Client Area des Panels ausgegeben. Mit den Navigationstasten der Smart Device kann dann der Bildausschnitt ausgewählt werden.

Alternativ kann die gesamte Bitmap auf die Größe der Client Area des Panels vergrößert oder verkleinert angezeigt werden, wodurch aber ein verzerrtes Abbild entsteht.

Die Bildlaufleisten werden bei einem größeren Bild angezeigt, wenn ein Pixel-Button zum Panel an der rechten unteren Bildseite ausgegeben wird. Wegen einer Bildverkleinerung durch die Ziehleisten, wird dieser Ablauf aber nicht verwendet.

Kodierung 6-14: PanelOnPaint für das Basisprojekt

## 6.9.5 PanelOnMouseDown

Nach einer auf dem Panel gedrückten rechten Maustaste wird die öffentliche Eigenschaft Pause true, wodurch auch die Pause-Checkmarken beider Menüs gesetzt werden.

Der Aufruf von PanelOnMouseDown erfolgt als eingetragener Handler für das MouseDown-Ereignis des Panels bei einer auf dem Panel gedrückten Maustaste.

Kodierung 6-15: PanelOnMouseDown für das Basisprojekt

## 6.9.6 PanelOnMouseMove

PanelOnMouseMove ist der in ErzeugeFormPanel eingetragene Handler für das MouseMove-Ereignis des Panels und wird virtuell und inhaltslos zum Überschreiben durch das Programm definiert.

Kodierung 6-16: PanelOnMouseMove für das Basisprojekt

## 6.9.7 FormOnMove (nicht für PocketPC)

FormOnMove ist der in ErzeugeFormPanel eingetragene Handler für das Move-Ereignis der Form und wird virtuell und inhaltslos zum Überschreiben durch das Programm definiert.

Kodierung 6-17: FormOnMove für das Basisprojekt

## 6.9.8 PanelOnScroll (nicht für PocketPC)

PanelOnScroll ist nicht für Smart Devices eine virtuelle Methode zur Verwaltung der Ziehleisten, die von der Anwendung überschrieben werden kann.

Aufruf als Handler für das PanelOnScroll-Ereignis.

Kodierung 6-18: PanelOnScroll für das Basisprojekt

## 6.9.9 OnKeyDown (nicht für PocketPC)

Nach einer gedrückten Pause-Taste wird die öffentliche Eigenschaft Pause true, wodurch auch die Pause-Checkmarken beider Menüs gesetzt werden.

Auf betätigte Pfeiltasten, auch der Zehnertastatur, wird bei einer angezeigten Ziehleiste, die Ziehleiste sonst der Mauszeiger um 20 Pixel weiter bewegt (mit gedrückter Strg-Taste um 5 Pixel).

Nach Betätigung der Pos 1-, Ende-, Bild-Auf- und Bild-Ab-Tasten werden die Ziehleisten entsprechend positioniert.

Bei der Bewegung der Ziehleisten wird die Ober- und Untergrenze des bildlauffähigen Bereiches beachtet.

OnKeyDown überschreibt die geerbte Form-Methode und wird bei jedem Tastendruck aufgerufen, wenn die Taste nicht als Shortcut-Taste für ein Menüelement verwendet wird.

Kodierung 6-19: OnKeyDown für das Basisprojekt

## 6.9.10 SmartDevice_KeyDown (nur für PocketPC)

Behandelt die Ereignisse von den Navigationstasten der Smart Device.

Der Aufruf erfolgt als ein in OnLoad eingetragener Handler für ein KeyDown-Ereignis der Smart Device.

Kodierung 6-20: SmartDevice_KeyDown

## 6.9.11 MobileDevice_Hibernate (nur für PocketPC)

Behandelt das Hibernate-Ereignis der Smart Device (Hibernate = fig. eine Pause einlegen, faulenzen).

Der Aufruf erfolgt als ein in OnLoad eingetragener Handler für das Hibernate-Ereignis der Smart Device.

```
///*************************************************************************
/// Version 2.0 vom 03.05.11
/// Parameter "kea": Daten des KeyDown-Ereignisses
/// Eingabedaten:
/// -
/// Ausgabedaten:
/// BASIS_GLOBALS-Datenfelder:
/// Pause true
///*************************************************************************
private void MobileDevice_Hibernate(object sender, EventArgs ea)
{
    BGlobals.Pause = true;
}
```

## 6.9.12 OnClosed

OnClosed schreibt in Programm_Init wichtige Programmdaten für den nächsten Programmstart in die Datei unter Init_Dateiname.

Nachdem die Schleifenvariable von der Programmschleife false gesetzt wurde, wird für MULTITHREAD auf die Beendigung des Arbeitsthreads gewartet, in dem die Programmschleife läuft.

Wenn während des Programmablaufes innerhalb der try-Blöcke catch-Ausnahmen abgefangen wurden, wird die Anzahl in einer Messagebox angezeigt und auf die Ausnahmeprotokolldatei verwiesen.

Erst danach werden noch vorhandene öffentliche Objekte explizit freigegeben, da diese noch für die Anzeige der Messagebox benötigt werden.

OnClosed überschreibt die von der Form geerbte Methode und wird einmal aufgerufen, nach dem die Form geschlossen wurde.

Kodierung 6-21: OnClosed für das Basisprojekt

## 6.9.13 Dispose

Die von Form verwendeten Ressourcen (mit Ausnahme des Speichers) werden freigegeben.

Dispose überschreibt die geerbte Form-Methode und ruft diese selbst als Basismethode auf. Der Aufruf erfolgt nach OnClosed, nachdem die Form geschlossen wurde.

Dispose wurde als Standardmethode unverändert übernommen.

Kodierung 6-22: Dispose für das Basisprojekt

# 6.10 Der Basisprogrammablauf

## 6.10.1 Die Initialisierung des Basisprogramms

In Programm_Init werden relevante öffentliche Datenfelder des Basisprojekts in eine für das Schreiben neu erzeugten Datei unter Filename geschrieben oder aus einer bestehenden Datei eingelesen.

Nach dem Lesen der Datenfelder ist in dem öffentlichen Datenfeld FileStreamPos die aktuelle Streamposition abgelegt.

Hinweis 6: zur Erweiterung der Init-Daten in neuen Versionen des Basisprogramms!

Für einen fehlerfreien Zugriff auf neu zugefügte Datenfelder in den alten Init-Dateien, wird das Datenende auf die konstante BASISDATENENDE vorsorglich nach oben verschoben. Hierbei ist zu beachten das eine string-Erweiterung nur einen Positionswert aus den alten Init-Dateien beansprucht und der eingelesene „string“ somit fehlerhaft ist. Integerwerte erhalten auch erstmalig einen falschen Wert, werden aber mit einer korrekten Positionsanzahl eingelesen.

Der Aufruf zum Lesen der Datenfelder erfolgt nach dem Programmstart in OnLoad. Der Aufruf zum Schreiben erfolgt zum Programmende auf OnClosed.

Programm_Init ist eine virtuelle Methode, die von der Anwendung zur Abspeicherung eigener Datenfelder überschrieben werden kann.

Kodierung 6-23: Programm_Init für das Basisprogramm

## 6.10.2 Die Programmschleife

Die Programmschleife wird als Ereignishandler eines in OnLoad erzeugten Timers durchlaufen. Der Ablauf erfolgt bis zum Verlassen der Schleife mit ProgLoop false.

Zu Beginn wird der aufrufende Timer gestoppt und die Kontrollmethode für die Lizenzkontrolle ausgeführt.

Wenn der Kontrollablauf korrekt abläuft, wird die öffentliche Variable TESTVERSION false und das Programm läuft in der Endversion.

Zur Anpassung der Menüs an die Bedienersprache werden diese nach dem Kontrollablauf erzeugt. Hierbei wird auch der Text der Titelleiste aktualisiert.

Für MULTITHREAD erfolgt nach der Lizenzkontrolle der Rücksprung aus der Methode, ohne in der Programmschleife zu verbleiben.

Nur für die Einzelthread-Versionen wird darauf in einer Programmschleife ständig Basis aufgerufen und die Programm-Ereignisse abgearbeitet. Zur weitergehenden Optimierung kann die Anzahl dieser Ereignisabfragen begrenzt werden (Standardwert 50).

Kodierung 6-24: die Programmschleife mit Lizenzkontrolle

## 6.10.3 Die Kontrollmethode

In der Kontrollmethode wird die Lizenzkontrolle durchgeführt.

Wenn der Kontrollablauf korrekt abläuft, wird die öffentliche Variable TESTVERSION false und das Programm läuft in der Endversion.

Ist bei einem Kontrollablauf der Aktivierungsschlüssel nicht korrekt, wird der Anwender über eine Messagebox darüber informiert und TESTVERSION true.

Ist bei einem Passwort-Kontrollablauf das Passwort nicht korrekt wird der Programmablauf beendet.

Die Aufrufe erfolgen in Programmschleife und zur zusätzlichen Lizenzkontrolle in Programm- und KontextMenueOnClick nach einem Pausewechsel.

Kodierung 6-25: die Kontrollmethode für die Lizenzkontrolle

## 6.10.4 Die Programmschleife für MULTITHREAD

MT_Programmschleife ist die Startmethode für einen auf OnLoad erzeugten Arbeitsthread.

Wenn der Bedienerthread keine Synchronisationsanforderung stellt, wird zur Abarbeitung des Anwendungsprogramms in einer Schleife solange Basis aufgerufen bis ProgLoop false wird.

Bei einer Programmpause oder nach einer Synchronisationsanforderung wird der Thread sofort gewechselt.

Zu Beginn wird in einer Schleifenabfrage auf die Ausführung der ersten OnResize-Botschaft mit der Erzeugung der GDI+ Graphics-Objekte gewartet.

Der Arbeitsthread wird beendet, wenn auf OnClosed ProgLoop false wird und MT_Programmschleife verlassen wurde. Vorher wird für einen korrekten Programmabschluss noch einmal die Basis-Methode aufgerufen.

Eine Lizenzkontrolle darf in dieser Methode nicht durchgeführt werden, da in einem Arbeitsthread bei einem Aufruf von Dialogforms das Programm ohne Fehlermeldung abstürzt.

Kodierung 6-26: die Programmschleife für MULTITHREAD

## 6.10.5 Die Basis-Methode

Basis durchläuft Aufrufkontrolle und nicht bei einer Programmpause, einer auszusetzenden Pause oder einer angezeigten Bilddatei die Anwendung in Programm.

Der Aufruf erfolgt ständig in Programmschleife.

Struktogramm

Kodierung 6-27: die Basis-Methode

### 6.10.5.1 Die Aufrufkontrolle

Die Anzahl der Aufrufe von Aufrufkontrolle werden in jeder Sekunde neu berechnet und als gefilterter Wert ohne Schwankungen, wenn diese unter 2% bleiben, in das öffentliche Datenfeld AufrufFilterAnz eingetragen.

Aufrufkontrolle wird ständig in Basis aufgerufen.

Kodierung 6-28: die Aufrufkontrolle

### 6.10.5.2 Das virtuelle Basisprogramm

Programm von dieser BASISPROJEKT-Klasse zeichnet wahlfrei geometrische Figuren in die BasisBitmap und direkt zum Display.

Programm ist eine virtuelle Methode, die von der Anwendung als Schnittstelle zum Basisprojekt überschrieben werden muss.

Programm wird ständig in Basis aufgerufen, wenn keine Pause ist.

Kodierung 6-29: das virtuelle Programm des Basisprojekts

## 6.10.6 Die Synchronisation der Threads

ThreadSynchroStart startet die Synchronisation des Bedienerthreads (UI-Thread) mit dem Arbeitsthread. Der Aufruf von ThreadSynchroStart darf nur im Bedienerthread erfolgen.

Kodierung 6-30: ThreadSynchroStart

ThreadSynchroEnde beendet die Synchronisation des Bedienerthreads (UI-Thread) mit dem Arbeitsthread. ThreadSynchroEnde muss nach jedem Aufruf von ThreadSynchroStart aufgerufen werden.

Kodierung 6-31: ThreadSynchroEnde

## 6.10.7 Den Arbeitsthread pausieren

Arbeitsthread_pausieren pausiert den Arbeitsthread.

In der Pausezeit wird zur Vermeidung eines Deadlock eine Synchronisation des Bedienerthreads mit dem Arbeitsthread ermöglicht.

Ein Aufruf muss im Arbeitsthread erfolgen.

Kodierung 6-32: Arbeitsthread_pausieren

## 6.10.8 Methodenaufruf über einen Delegaten

Komponenten und somit auch Dialogforms müssen im Bedienerthread erzeugt werden. Nur hierdurch können die im Botschaftsthread ablaufenden Methoden auf die darin vorhandenen Steuerelemente zugreifen!

Vor der Erzeugung von Komponenten sollte deshalb abgefragt werden, ob die Erzeugermethode im Arbeitsthread ausgeführt wird und somit auf den Bedienerthread umgeleitet werden muss. Zur Umleitung wird die Erzeugermethode durch eine der Invoke-Methoden synchron oder asynchron als Delegat ausgeführt.

Delegaten werden in allen Projekten am Anfang der Methoden zur Erzeugung der Menüs erzeugt und ausgeführt. Des Weiteren im Basisprojekt in Animated_Bitmap zur Animation von geladenen Bitmaps, in der Pause_Methode und in Titeltext mit string-Parameter.

Kodierung 6-33: Methodenaufruf über einen Delegaten

## 6.10.9 Die Pause_Methode

Die Pause_Methode bearbeitet einen neuen Pausestatus, beendet eine Bildanzeige und ermöglicht einen Zugriff auf lokale Daten und Methoden.

Zum Beenden einer Bildanzeige werden die Ziehleisten verdeckt und zur Anpassung der Panelgröße PanelOnResize aufgerufen.

Der Aufruf erfolgt beim Setzen der Eigenschaft der globalen Instanzvariablen Pause.

Kodierung 6-34: Die Pause_Methode für das Basisprogramm

## 6.10.10 Die Ausnahmeprotokollierung

Ausnahmeprotokollierung erhöht das globale Datenelement catch_Ausnahmen um eins und schreibt für die ersten 25 Ausnahmen den Ausnahmetext mit Datum und Zeit in die Datei der StreamWriter-Instanz SWriter "Titeltext" + "_Exceptions.txt".

Kodierung 6-35: Ausnahmeprotokollierung für das Basisprogramm

## 6.11 Die multilingualen Basismenüs

Das Hauptmenü für die Form des Basisprogramms wird multilingual in ErzeugeFormMenues zusammengestellt. Zum Abschluss der Methode wird das Kontextmenü durch Aufruf von ErzeugeKontextMenue entworfen.

Die Anwendung kann durch Überschreiben der virtuellen Methoden mit vorangehendem Aufruf der Basismethode die Elemente beider Menüs verändern und erweitern.

Die einzelnen Menüelemente werden über die Einträge in den dreidimensionalen Menütext-Arrays zusammengestellt. Hierbei bestimmt die Größe der zweiten Array-Dimension die Anzahl der Menüelemente. Die erste Dimension wählt die Textsprache und die dritte Dimension den Text selbst aus.

Die Menüs werden nach dem Programmstart auf OnLoad erstmalig zur Verfügung gestellt. Im weiteren Programmablauf müssen die Menüs nach einem Sprachwechsel in SpracheMenueOnClick und nachdem der Basis-Dialog geschlossen wurde in DialogResultOK neu erzeugt werden.

Nach einem Wechsel der Hilfetextsprache in HilfeTextSpracheMenueOnClick und einer gewechselten Statusbaranzeige in BasisMenueOnClick korrigiert ein neuer Entwurf nur die Checkmarke. Weitergehend kann hierdurch aber das gesamte Menüelement durch die Anwendung verschoben werden.

Zur Aktualisierung der Titelleiste der Form wird Titeltext aufgerufen, was nach einem Wechsel der Bedienersprache erforderlich ist. Danach wird der Panel- und Tooltiptext der Statusbar multilingual korrigiert.

Für alle Menüelemente werden Handler für die Select-Ereignisse und, wenn erforderlich, für die Click-Ereignisse eingetragen.

Für Smart Devices werden keine Eventhandler für Select-Ereignisse erzeugt. Des Weiteren stehen keine RadioChecks, keine Tastatur-Shortcuts und für die Statusbar kein Panel- und Tooltiptext zur Verfügung. Der Paneltext wird direkt zur Statusleiste ausgegeben.

Hinweis 7 zur Menübeschreibung!

Im weiteren Ablauf werden die Begriffe Menüelement und Menüpunkt gleichbedeutend verwendet. Ein Menüelement kann selbst wieder aus mehreren Menüelementen bestehen und ein Menüpunkt kann selbst wieder mehrere Menüpunkte enthalten. Der Menüindex gibt den Menüpunkt aus mehreren zusammengehörenden Menüelementen von null beginnend an.

### 6.11.1 Der Basisentwurf des Hauptmenüs

Das Hauptmenü wird nur der Form zugewiesen, wenn das öffentliche Datenfeld BasisHauptMenueAnzeige true ist.

Dem Menüelement Pause wird die Taste F3 zur schnellen Aktivierung zugewiesen. Mit F1 wird die Einführung des Hilfetextes angezeigt. Weitere Shortcuts werden den Menüpunkten zur Anzeige der Dialogforms zugewiesen (keine Shortcuts für PocketPC).

Nur für MULTITHREAD gilt, wenn ErzeugeFormMenues nicht in ihrem Erzeuger-Thread (UI-Thread) aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

Kodierung 6-36: ErzeugeFormMenues für das Basisprogramm

### 6.11.2 Der Basisentwurf des Kontextmenüs

ErzeugeKontextMenue entwirft das multilinguale Kontextmenü für die Form des Basisprogramms (für SmartDevices unter dem Compact-Framework ohne Eventhandler für das Select-Ereignis und RadioCheck-Button).

Die Anwendung kann durch Überschreiben der virtuellen Methode mit vorangehendem Aufruf der Basismethode das Kontextmenü verändern und erweitern.

Der Aufruf erfolgt in ErzeugeFormMenues nach dem Entwurf des Basis-Hauptmenüs.

Kodierung 6-37: ErzeugeKontextMenue für das Basisprogramm

# 6.12 Bearbeitung der Basismenü-Ereignisse

## 6.12.1 Das Abschlussereignis für das Basismenü

Nicht für Smart Devices unter dem .Net Compact Framework gibt OnMenuComplete den Starttext des Basisprogramms wieder zum ersten Panel der StatusBar aus.

OnMenuComplete überschreibt die von der Form geerbte Methode und wird aufgerufen, nachdem die Menübehandlung abgeschlossen wurde.

Kodierung 6-38: OnMenuComplete für das Basisprogramm

## 6.12.2 Hauptmenüelement-Ereignisse

Nicht für Smart Devices unter dem .Net Compact Framework gibt HauptMenueOnSelect den Hilfetext für den ausgewählten Hauptmenüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Hauptmenüelement ausgewählt wurde.

Kodierung 6-39: HauptMenueOnSelect für das Basisprogramm

## 6.12.3 Programmmenüelement-Ereignisse

ProgrammMenueOnClick bearbeitet die Clicks der Programm-Menüelemente, die von dem Basisprojekt erzeugt wurden. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Programm-Menüelement angeklickt wurde. Für PocketPC ist der public-Zugriffsmodifizierer wegen direkter Methodenaufrufe erforderlich.

Nach Auswahl des Pause ! - Menüpunktes wird der Programmstatus gewechselt und in der Eigenschaft für Pause in beiden Menüs die Pause-Checkmarke korrigiert. Ist Image_Anzeige true, wird eine laufende Bildanimation beendet (nicht für PocketPC), die Ziehleisten ausgeblendet, Image_Anzeige false und PanelOnResize zur Erzeugung neuer Grafikobjekte aufgerufen.

Bei einem Pausewechsel wird für die Endversion in der Kontrollmethode nochmals eine Lizenzkontrolle durchgeführt.

Kodierung 6-40: ProgrammMenueOnClick für das Basisprogramm

ProgrammMenueOnPopup ist der in ErzeugeFormMenues eingetragene Handler zum Aktivieren oder Deaktivieren von Programm-Menüpunkten der inhaltslos zum Überschreiben für die Anwendung definiert wird.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Popup-Ereignisse eingetragener Handler, bevor der Programm-Menüpunkt ausgeklappt wird.

Kodierung 6-41: ProgrammMenueOnPopup für das Basisprogramm

Nicht für Smart Devices unter dem .Net Compact Framework gibt ProgrammMenueOnSelect den Hilfetext für den ausgewählten Programm-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Programm-Menüelement ausgewählt wurde.

Kodierung 6-42: ProgrammMenueOnSelect für das Basisprogramm

## 6.12.4 Kontextmenüelement-Ereignisse

KontextMenueOnClick bearbeitet die Clicks der Kontext-Menüelemente, die von dem Basisprojekt erzeugt wurden. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Kontext-Menüelement angeklickt wurde.

Nach Auswahl des Pause ! - Menüpunktes wird der Pausestatus gewechselt und in der Eigenschaft für Pause in beiden Menüs die Pause-Checkmarke korrigiert. Ist Image_Anzeige true, wird eine laufende Bildanimation beendet (nicht für PocketPC), die Ziehleisten ausgeblendet, Image_Anzeige false und PanelOnResize zur Erzeugung neuer Grafikobjekte aufgerufen.

Bei einem Pausewechsel wird für die Endversion in der Kontrollmethode nochmals eine Lizenzkontrolle durchgeführt.

Nach Auswahl von "Basis" wird die Dialogform des Basisprojekts modal erzeugt, angezeigt und wieder entfernt.

Nach Auswahl von "Drucken" wird in BasisBitmapDrucken die Basisbitmap ausgedruckt (nicht für Smart Devices). Nach Auswahl von "Speichern" wird in BasisBitmapSpeichern die Basisbitmap in eine Grafikdatei abgespeichert.

Kodierung 6-43: KontextMenueOnClick für das Basisprogramm

KontextMenueOnPopup ist der in ErzeugeFormMenues eingetragene Handler zum Aktivieren oder Deaktivieren von Kontext-Menüpunkten der inhaltslos zum Überschreiben für die Anwendung definiert wird.

Der Aufruf erfolgt als ein in ErzeugeFormMenue für Popup-Ereignisse eingetragener Handler, bevor das Kontextmenü angezeigt wird.

Kodierung 6-44: KontextMenueOnPopup für das Basisprogramm

Nicht für Smart Devices unter dem .Net Compact Framework gibt KontextMenueOnSelect den Hilfetext für den ausgewählten Kontext-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Kontext-Menüelement ausgewählt wurde.

Kodierung 6-45: KontextMenueOnSelect für das Basisprogramm

## 6.12.5 Basismenüelement-Ereignisse

BasisMenueOnClick bearbeitet die Clicks der Basis-Menüelemente, die von dem Basisprojekt erzeugt wurden. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Basis-Menüelement angeklickt wurde.

Nach Auswahl von "Basis" wird die Dialogform des Basisprojekts modal erzeugt, angezeigt und wieder entfernt.

Nach Auswahl von "Drucken" wird in BasisBitmapDrucken die Basisbitmap ausgedruckt (nicht für Smart Devices). Nach Auswahl von "Speichern" wird in BasisBitmapSpeichern die Basisbitmap in eine Grafikdatei abgespeichert. Nach Auswahl von "Laden" wird die Basisbitmap aus einer Grafikdatei erzeugt.

"Statusleiste" zeigt oder verdeckt die Statusleiste mit gewechselter Eigenschaftsvariablen StBarAnzeige und hierdurch korrigierter Checkmarke.

Kodierung 6-46: BasisMenueOnClick für das Basisprogramm

Nicht für Smart Devices unter dem .Net Compact Framework gibt BasisMenueOnSelect den Hilfetext für den ausgewählten Basis-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Element des Basis-Menüs ausgewählt wurde.

Kodierung 6-47: BasisMenueOnSelect für das Basisprogramm

## 6.12.6 Hilfemenüelement-Ereignisse

HilfeMenueOnClick bearbeitet die Clicks der Hilfe-Menüelemente, die von dem Basisprojekt erzeugt wurden. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Hilfe-Menüelement angeklickt wurde. Für PocketPC ist der public-Zugriffsmodifizierer wegen direkter Methodenaufrufe erforderlich.

Nach Auswahl von "Einführung" und "Menübefehle" wird aus den multilingualen HTML-Hilfetextdateien ein Inhaltsverzeichnis mit dem Einführungstext oder der Beschreibung der Menübefehle aufgerufen.

Nach Auswahl von "über..." wird eine Dialogform mit multilingualer Information über das Basisprogramm angezeigt.

Kodierung 6-48: HilfeMenueOnClick für das Basisprogramm

Nicht für Smart Devices unter dem .Net Compact Framework gibt HilfeMenueOnSelect den Hilfetext für den ausgewählten Hilfe-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Hilfe-Menüelement ausgewählt wurde.

Kodierung 6-49: HilfeMenueOnSelect für das Basisprogramm

## 6.12.7 Sprachemenüelement-Ereignisse

SpracheMenueOnClick bearbeitet die Clicks der Sprache-Menüelemente, die von dem Basisprojekt erzeugt wurden. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Sprache-Menüelement angeklickt wurde.

Nach Anwahl eines Sprache-Menüpunktes werden die öffentlichen Datenfelder User_Language und Help_Language verändert und das Menü durch Aufruf von ErzeugeFormMenues in der neuen Sprache erzeugt. Der Paneltext der StatusBar und der Formtitel werden hierbei in der neuen Sprache aktualisiert.

Kodierung 6-50: SpracheMenueOnClick für das Basisprogramm

Nicht für Smart Devices unter dem .Net Compact Framework gibt SpracheMenueOnSelect den Hilfetext für den ausgewählten Sprache-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Sprache-Menüelement ausgewählt wurde.

Kodierung 6-51: SpracheMenueOnSelect für das Basisprogramm

### 6.12.8 Hilfetextsprachemenüelement-Ereignisse

HilfeTextSpracheMenueOnClick bearbeitet die Clicks der Hilfetextsprache-Menüelemente, die von dem Basisprojekt erzeugt wurden. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Hilfetextsprache-Menüelement angeklickt wurde.

Nach Anwahl eines Hilfetextsprache-Menüpunktes wird das öffentliche Datum Help_Language verändert und das Menü durch Aufruf von ErzeugeFormMenues neu erzeugt. Hierdurch kann das ganze Menüelement durch die Anwendung verschoben werden.

Kodierung 6-52: HilfeTextSpracheMenueOnClick für das Basisprogramm

Nicht für Smart Devices unter dem .Net Compact Framework gibt HilfeTextSpracheMenueOnSelect den Hilfetext für den ausgewählten Hilfetextsprache-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Hilfetextsprache-Menüelement ausgewählt wurde.

Kodierung 6-53: HilfeTextSpracheMenueOnSelect

## 6.13 Speichern, Laden und Drucken der Basisbitmap

### 6.13.1 Abspeichern der Basisbitmap in eine Datei

BasisBitmapSpeichern speichert die Basisbitmap in eine Datei nach Auswahl des Dateinamens über die Standard-Dialogform "Speichern unter".

Der Aufruf erfolgt in BasisMenueOnClick und KontextMenueOnClick nach der Menüauswahl "Speichern...".

Die Dateierweiterungen .wmf, .emf, .exif, .mbmp und .ico werden nicht als Dateityp in der Speichern-Dialogform aufgeführt, weil bei einer Auswahl dieser Dateitypen die Datei immer im Standard .png-Format abgespeichert wird.

Kodierung 6-54: BasisBitmapSpeichern

### 6.13.2 Laden einer Grafikdatei in die Basisbitmap

BasisBitmapLaden lädt die Basisbitmap aus einer Datei nach Auswahl des Dateinamens über die Standard-Dialogform "Öffnen".

Der Aufruf erfolgt in BasisMenueOnClick nach der Menüauswahl "Laden...".

Von dem ausgewählten Bild wird eine Bitmap mit dem zugehörigen Grafik-Objekt erzeugt. Wenn vorhanden, werden beide Objekte vorher gelöscht. Nicht auf Smart Devices wird auch die Wiedergabe animierter Bitmaps unterstützt.

Bei einer Programmausnahme wird die Ausnahmeprotokollierung aufgerufen und eine zusätzliche Messagebox angezeigt. Danach werden durch Aufruf von PanelOnResize normale Grafikobjekte erzeugt.

Kodierung 6-55: BasisBitmapLaden

#### 6.13.2.1 Wiedergabe einer animierten Bitmap (nicht für PocketPC)

Animated_Bitmap gibt den nächsten Rahmen einer geladenen animierten Basisbitmap zur Form aus. Aufruf als Event-Handler für die animierte Bitmap aus BasisBitmapLaden.

Die Ausführung wird über einen neu erzeugten Delegaten auf den Bediener-Thread umgeleitet, wenn InvokeRequired true zurückgibt.

Kodierung 6-56: Animated_Bitmap

## 6.13.3 Ausgabe der Basisbitmap zum Drucker (nicht für Smart Devices)

BasisBitmapDrucken startet eine Druckerausgabe nach Auswahl eines Druckers über die Standard-Dialogform "Drucken" oder, wenn nicht erfolgreich, über die Seitenvorgabe- und Druckervorschau-Dialoge. BasisBitmapDrucken wird erst nach der gesamten Datenübertragung zum Drucker einschließlich einer nachfolgenden Druckervorschau verlassen.

Eine Vorschau der Druckerausgabe wird nach Anzeige der Standard-Dialogform "Drucken" durch den Druckertreiber selbst durchgeführt. Die Standard-Dialogform "Drucken" wird unter den 64-Bit Windows-Versionen nur als extended-Dialog angezeigt.

Aufruf in BasisMenueOnClick und KontextMenueOnClick nach der Menüauswahl "Drucken...".

Kodierung 6-57: BasisBitmapDrucken

### 6.13.3.1 Ausgabe der Druckerseiten

OnPrintPage gibt die einzelnen Seiten der Basisbitmap zum ausgewählten Drucker aus. Der Aufruf erfolgt als ein in BasisBitmapDrucken eingetragener Handler für das PrintPage-Ereignis des Drucker-Dokuments.

Kodierung 6-58: OnPrintPage

# 7 Die Dialogform des Basisprogramms

Der Basisdialog besteht aus einer dreiseitigen Dialogform mit Registerkarten zur Seitenauswahl und wird in der Klasse BASIS_DIALOG durch Vererbung von der Form-Klasse erstellt. Durch ihn können grundlegende Einstellungen für das aktive Programm vorgenommen werden.

Auf der Basisseite kann das Hauptmenü und die Statusleiste ein- und ausgeblendet werden, die Sichbarkeit der Form eingestellt und ein transparenter Hintergrund für die Grafikfläche der Form ausgewählt werden.

Auf der Grafikseite kann die Qualität der Grafikausgabe und der Pixelausgleich bei der Darstellung ausgewählt werden.

Innerhalb der Spracheseite kann die Sprachauswahl für die Bediener- und die Hilfetextsprache vorgenommen werden.

Weil die Dialogformen mit dem Designer entworfen werden und für das .Net Compakt-Framework viele Änderungen und Einschränkungen erforderlich sind, wird für die SmartDevice-Projekte eine neue Quellcodedatei verwendet (s.a. Kodierungsanpassungen für das .Net Compact-Framework).

Im weiteren Ablauf wird hier nur die Dialogform für das .Net-Framework behandelt.

Kodierung 7-1: Die BASIS_DIALOG-Klasse

```
///*************************************************************************
/// Version 3.0 vom 22.09.13
///*************************************************************************
[CLSCompliantAttribute(true)]      // markiert die Klasse als CLS-kompatibel
public class BASIS_DIALOG : Form
{

    #region Datenfelder und Konstruktor #endregion

    #region Bearbeitung der Standard-Ereignisse für den Dialog #endregion

    #region InitializeComponent und Ereignis-Handler für die Kontrollen #endregion

}
```

## 7.1 Die Instanzvariablen der Basisdialogform

Die Dialogkontrollen wurden von dem Designer bei der Erstellung der Dialogformen zugefügt und dürfen nicht manuell geändert werden.

Kodierung 7-2: Die Instanzvariablen der BASIS_DIALOG-Klasse

## 7.2 Der Konstruktor der Basisdialogform

In InitializeComponent werden die mit dem Designer erstellten Basis-Dialogseiten aufgebaut und eine Instanz von BASIS_DIALOG_TEXTE erstellt.

Der Aufruf erfolgt als Konstruktor für die BASIS_DIALOG-Klasse nach Menüpunktauswahl "Basis..." in BasisMenueOnClick und KontextMenueOnClick.

Kodierung 7-3: Der Konstruktor der Basisdialogform

# 7.3 Bearbeitung der Ereignisse für die Basisdialogform

## 7.3.1 OnLoad für die Basisdialogform

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten. Des Weiteren werden erweiterte Hilfetexte und Tooltipps eingeführt.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

Kodierung 7-4: OnLoad für die Basisdialogform

## 7.3.2 BASIS_DIALOG_HelpRequested

BASIS_DIALOG_HelpRequested zeigt eine Messagebox mit dem ausführlichen Hilfetext der ausgewählten Kontrolle an.

Der Aufruf erfolgt als ein im Designer eingetragener Handler für das HelpRequested-Ereignis der Basis-Dialogform nach Anwahl einer Kontrolle über den ?-Button oder die F1-Taste.

Das Ereignis wird auch ausgelöst, wenn der Kontrolle über die HelpProvider-Klasse ein Popup-Hilfetext zugewiesen wurde und dieser durch SetShowHelp für die Kontrolle deaktiviert ist.

Kodierung 7-5: BASIS_DIALOG_HelpRequested

## 7.3.3 OnClosed für die Basisdialogform

OnClosed ruft zur Realisierung der Dialog-Einstellungen DialogResultOK auf.

Für MULTITHREAD wird vor der Veränderung der globalen Daten in DialogResultatOK der Bedienerthread mit dem Arbeitsthread synchronisiert.

OnClosed überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, nachdem die Dialogform geschlossen wurde.

Kodierung 7-6: OnClosed für die Basisdialogform

DialogResultOK realisiert alle Dialog-Einstellungen auf Schließen des Dialoges mit dem OK-Button und erzeugt in ErzeugeFormMenues neue Menüs.

Der Aufruf erfolgt auf OnClosed bevor die modale Dialog-Form geschlossen wird.

Kodierung 7-7: DialogResultOK für die Basisdialogform

# 7.4 Mehrsprachige Basisdialogtexte

Die Dialogtexte werden mehrsprachig in der Klasse BASIS_DIALOG_TEXTE zur Verfügung gestellt. Sie bestehen aus dem Titeltext, den Texten für die Registerkarten und Messageboxtexten, die alle ohne Tooltips auskommen.

Die weiteren Texte für die Kontrollen aller Dialogseiten enthalten Tooltips und, wenn erforderlich, zusätzlich erweiterte Hilfetexte.

Eine Instanz von BASIS_DIALOG_TEXTE wird im Konstruktor der BASIS_DIALOG-Klasse erzeugt.

Hinweis 8 zu den Tooltipps!

Wenn ein erweiterter Hilfetext zur Verfügung steht, wird dem Tooltip ein Hinweis in der Form (-> ?-Hilfe) zugefügt.

## 7.5 Die Basisdialogform mit dem Designer erzeugen

InitializeComponent erzeugt die Basis-Dialogseiten mit den Kontrollen. Der Aufruf erfolgt im BASIS_DIALOG-Konstruktor.

Die Kodierung wird automatisch mit dem Designer in deutscher Sprache erzeugt und darf nicht manuell geändert werden.

BASIS_DIALOG_HelpRequested ist als Handler für das HelpRequested-Ereignis der Dialogform eingetragen und Info_Button_Click ist der Handler für das Click-Ereignis des Info-Button.

```
///*****************************************************************************
///  Version 1.0 vom 01.08.06
///*****************************************************************************
private void InitializeComponent()
{...}
```

## 7.6 Ereignisse für die Kontrollen der Basisdialogform

### 7.6.1 InfoButton_Click für die Basisdialogform

InfoButton_Click zeigt die Ueber-Dialogform mit Informationen über das Basisprojekt an.

Der Aufruf erfolgt als ein eingetragener Handler für das Click-Ereignis des "über"...-Button von der Basis-Dialogseite.

Kodierung 7-8: InfoButton_Click für die Basisdialogform

### 7.6.2 HilfeButton_Click für die Basisdialogform

HilfeButton_Click ruft die HTML-Hilfe multilingual für die aktuelle Dialogseite auf. Der Aufruf erfolgt als eingetragener Handler für das Click-Ereignis des Hilfe-Button der Basis-Dialogform.

Hinweis 9 zu den Hilfetexten für die Basisdialogseiten!

Die Hilfe für die Basisdialogseiten wird über Tooltips und erweiterte Hilfetexte realisiert. Zur Zeit ist kein Hilfe-Button in die Dialogform eingebunden und es sind auch keine Hilfetexte für die Dialogseiten in den HTML-Hilfedateien vorhanden. Für Smart Devices unter dem .Net Compact Framework stehen keine Tooltips und erweiterte Hilfetexte zur Verfügung. Hier wird die HTML-Hilfe über einen Hilfe-Button mit Hilfetexten für die Dialogseiten realisiert. Diese Dialogseiten werden aber getrennt von den Dialogseiten für das .Net Framework erstellt.

Kodierung 7-9: HilfeButton_Click für die Basisdialogform

# 8 Standard Dialogformen des Basisprojekts

Zur Anzeige von Programminformationen stellt der Basisworkspace die folgenden, mit dem Designer erstellten und zu verwaltenden Dialogformen zur Verfügung.

## 8.1 Das Infofeldformular

Diese Dialogform wurde durch Auswahl von Windows Forms / Infofeld zum Basisworkspace als neues Element hinzugefügt. Dadurch werden auch die Quellcodedateien AboutBox.cs, AboutBox.Designer.cs und AboutBox.resx automatisch zum Projekt hinzugefügt.

In dem Infofeldformular werden die in der AssemblyInfo-Datei eingetragenen Projekt-Informationen aufbereitet angezeigt. Dies sind im Einzelnen von der Datei AssemblyInfo.cs die Attribute AssemblyTitle, AssemblyProduct, AssemblyVersion, AssemblyCopyright, AssemblyCompany und AssemblyDescription.

Diese Informationen können aber noch nach der Formular-Erzeugung vor der Anzeige der Dialogform überschrieben werden.

Das Infofeldformular wird von der Anwendung 1 und dem Formen-Verwaltungsprogramm nach Menüauswahl „Hilfe / über“ zur Anzeige von Programminformationen aufgerufen.

## 8.2 Die Über-Dialogform

Als Alternative zu dem Infofeldformular steht eine Dialogform mit einer Ausgabetextbox und den Button „OK“ und „WWW“ zur Verfügung. Die Anzeige wird in den Quellcodedateien Ueber.cs, Ueber.designer.cs und Ueber.resx realisiert.

In das Textfeld werden der BasisText1 mit Informationen über die Version des Basisprojekts und Copyrights ausgegeben. Des Weiteren der vollständige Assemblyname des Programms, die verwendete Common Language Runtime CLR, die Betriebssystemversion und die Bitanzahl der .Net Platform, die über die Länge eines int-Pointers ermittelt wird.

Über den „WWW“-Button wird die Entwicklerseite von jk-ware aufgerufen. Hierzu wird auch eine erforderliche Einwahl in das Internet durchgeführt.

Die Über-Dialogform wird von dem Basisprogramm nach Menüauswahl „Hilfe / über“ zur Anzeige von Informationen über das Basisprojekt verwendet.

## 8.3 Ein Informationsdialog

Als Alternative zu der Messagebox des .Net-Frameworks wird eine einfache Dialogform mit einer Ausgabetextbox und einem OK-Button zur Verfügung gestellt. Die Anzeige wird in den Dateien Info.cs, Info.Designer.cs und Info.resx realisiert.

Der Informationsdialog wird innerhalb des Kontrollablaufes für den Lizenzschutz bei einem fehlenden Aktivierungsschlüssel zur Informationsanzeige verwendet.

# 9 Methoden für den Lizenzschutz

In der LIZENZSCHUTZ-Klasse wird nach einer Passworteingabe in einer binären Kontrolldatei ein Programmpfad verglichen oder in einer zugesandten Lizenzschutzdatei ein kryptographischer Schlüssel kontrolliert.

Der Lizenzschutz wird innerhalb der Quellcodedatei Basis_Procs.cs realisiert.

Kodierung 9-1: LIZENZSCHUTZ-Klasse

```
#region Methoden und Dialogformen für den Lizenzschutz
///*************************************************************************
/// Version 1.1 vom 10.10.09
///*************************************************************************
[CLSCompliantAttribute(true)]      // markiert die Klasse als CLS-kompatibel
public class LIZENZSCHUTZ
{

    #region Datenfelder und Konstruktor #endregion

    #region Methoden für den Kontrollablauf #endregion

    #region Dialogformen für die Lizenzkontrolle #endregion

} //Ende von class LIZENZSCHUTZ
#endregion
```

## 9.1 Instanzvariable und Konstruktor

Kodierung 9-2: Die Instanzvariablen der LIZENZSCHUTZ-Klasse

Kodierung 9-3: Der LIZENZSCHUTZ-Konstruktor

## 9.2 Der Kontrollablauf

Der Kontrollablauf kontrolliert bei einer Passwortabfrage in der Kontrolldatei einen abgelegten Programmpfad mit der Erstellzeit des Ordners oder bei einer Lizenzabfrage in der Lizenzdatei einen Aktivierungsschlüssel.

Ist das Programmpasswort nicht korrekt, werden Dialogformen zur Sprachauswahl, zur Akzeptanz eines Lizenzvertrages („EULA“) und zur Eingabe des Passwortes ausgegeben.

Bei einer Lizenzabfrage wird bei einer fehlenden Lizenzdatei oder einem unkorrekten Aktivierungsschlüssel innerhalb der Lizenzdatei nach dem Dialog zur Akzeptanz des Lizenzvertrages ein Dialog mit dem Registrierungstext angezeigt.

Kodierung 9-4: Der Kontrollablauf

## 9.3 Die Lizenzdateikontrolle

In Lizenzdateikontrolle wird der Aktivierungsschlüssel in der Lizenzdatei unter Filename überprüft.

Wenn die Lizenzdatei nicht vorhanden ist oder der Aktivierungsschlüssel unkorrekt ist, werden nach dem Dialog zur Sprachauswahl noch Dialoge zur Akzeptanz oder Ablehnung eines Lizenzvertrags („EULA“) und zum Kopieren des Registrierungsschlüssels angezeigt.

Der Aufruf erfolgt in Kontrollablauf.

Kodierung 9-5: Lizenzdateikontrolle

## 9.4 Regenerieren des Aktivierungsschlüssels

Dekrypt_Aktivierungsschluessel regeneriert den Aktivierungsschlüssel aus den Daten der Lizenzdatei in Dateiname.

Der Aufruf erfolgt in Lizenzdateikontrolle.

Kodierung 9-6: Dekrypt_Aktivierungsschluessel

# 10 Dialogformen für den Lizenzschutz

Für einen Zugriff auf die gemeinsamen Instanzvariablen für den Lizenzschutz, sind alle Klassen für die Dialogformen innerhalb der LIZENZSCHUTZ-Klasse aufgeführt.

Die Dialogformen für den Lizenzschutz wurden alle manuell ohne Designer-Unterstützung entworfen.

Die Dialogform zur Eingabe eines Aktivierungsschlüssels wird innerhalb des Lizenzschutzablaufes nicht eingesetzt, da der Aktivierungsschlüssel in einer Lizenzdatei enthalten ist, die an den Kunden übermittelt wird.

Nachfolgend werden die Quelltexte hauptsächlich sich selbst erläuternd aufgelistet.

Zur Einsparung von Seiten wird die Initialisierung der String-Arrays für die Dialogtexte nur in den Sprachen Deutsch und Englisch aufgeführt.

## 10.1 Die Dialogform zur Sprachauswahl

Die SPRACHE_DIALOG-Klasse erzeugt eine Dialogform zur Sprachauswahl.

Kodierung 10-1: Die SPRACHE_DIALOG-Klasse

```
///*************************************************************************
/// Version 1.1 vom 04.10.09
///*************************************************************************
[CLSCompliantAttribute(true)]      // markiert die Klasse als CLS-kompatibel
public class SPRACHE_DIALOG : Form
{

    #region Datenfelder, Konstruktor und InitializeComponent #endregion
    #region Bearbeitung der Ereignisse für die Sprache-Dialogform #endregion

}
```

### 10.1.1 Die Instanzvariablen

Kodierung 10-2: Die Instanzvariablen

### 10.1.2 Der Konstruktor

Der Konstruktor für die SPRACHE_DIALOG-Klasse initialisiert ein zweidimensionales Array mit multilingualen Texten für die Kontrollen und erstellt die Sprache-Dialogform in InitializeComponent.

Kodierung 10-3: Der Konstruktor

### 10.1.3 InitializeComponent

In InitializeComponent wird die Sprache-Dialogform mit den Kontrollen aufgebaut. Die Codierung wurde manuell in deutscher Sprache durchgeführt.

Der Aufruf erfolgt im Konstruktor.

Kodierung 10-4: InitializeComponent

### 10.1.4 Bearbeitung der Ereignisse für die Sprache-Dialogform

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

Die Form wird wegen Problemen mit FormStartPosition.CenterParent in InitializeComponent hier manuell zentriert (nicht für PocketPC).

Kodierung 10-5: OnLoad für die Sprachedialogform

In OnClosed wird die ausgewählte Sprache an das Datenfeld Language zugewiesen.

OnClosed überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, nachdem die Dialogform geschlossen wurde.

Kodierung 10-6: OnClosed für die Sprachedialogform

## 10.2 Die Dialogform zur Anzeige des Lizenzvertrages (EULA)

Die LIZENZ_DIALOG-Klasse erzeugt eine Dialogform zur Anzeige und Akzeptanz des Lizenzvertrages (EULA).

Kodierung 10-7: Die LIZENZ_DIALOG-Klasse

```
///*****************************************************************************
/// Version 1.1 vom 04.10.09
///*****************************************************************************
[CLSCompliantAttribute(true)]      // markiert die Klasse als CLS-kompatibel
public class LIZENZ_DIALOG : Form
{

    #region Datenfelder, Konstruktor mit den Lizenztexten und InitializeComponent #endregion
    #region Bearbeitung der Ereignisse für die Dialogform und die Kontrollen #endregion

}
```

### 10.2.1 Die Instanzvariablen

Kodierung 10-8: Die Instanzvariablen

### 10.2.2 Der Konstruktor

Der Konstruktor für die LIZENZ_DIALOG-Klasse initialisiert ein zweidimensionales Array mit multilingualen Texten für die Kontrollen und erstellt die Lizenz-Dialogform in InitializeComponent.

Kodierung 10-9: Der Konstruktor

### 10.2.3 InitializeComponent

In InitializeComponent wird die Lizenz-Dialogform mit den Kontrollen aufgebaut. Die Codierung wurde manuell in deutscher Sprache durchgeführt.

Der Aufruf erfolgt im Konstruktor.

Kodierung 10-10: InitializeComponent

### 10.2.4 Bearbeitung der Ereignisse für die Lizenzdialogform und die Kontrollen

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

Die Form wird wegen Problemen mit FormStartPosition.CenterParent in InitializeComponent hier manuell zentriert (nicht für PocketPC).

Kodierung 10-11: OnLoad für die Lizenzdialogform

PanelOnPaint gibt den sprachabhängigen Lizenztext zum Panel aus.

Für Smart-Devices wird zur Anzeige der Bildlaufleisten ein Button links-unten zum Panel ausgegeben.

PanelOnPaint ist der eingetragene Handler für das Paint-Ereignis des Panels aus InitializeComponent.

Kodierung 10-12: PanelOnPaint für die Lizenzdialogform

## 10.3 Die Dialogform zur Passworteingabe

Die PASSWORT_DIALOG-Klasse erzeugt eine Dialogform zur Passworteingabe.

Kodierung 10-13: Die PASSWORT_DIALOG-Klasse

```
///*****************************************************************************
/// Version 1.1 vom 04.10.09
///*****************************************************************************
[CLSCompliantAttribute(true)]     // markiert die Klasse als CLS-kompatibel
public class PASSWORT_DIALOG : Form
{

    #region Datenfelder, Konstruktor und InitializeComponent #endregion
    #region Bearbeitung der Ereignisse für die Dialogform und die Kontrollen #endregion

}
```

### 10.3.1 Die Instanzvariablen

Kodierung 10-14: Die Instanzvariablen

### 10.3.2 Der Konstruktor

Der Konstruktor für die PASSWORT_DIALOG-Klasse initialisiert ein zweidimensionales Array mit multilingualen Texten für die Kontrollen und erstellt die Passwort-Dialogform in InitializeComponent.

Kodierung 10-15: Der Konstruktor

### 10.3.3 InitializeComponent

In InitializeComponent wird die Passwort-Dialogform mit den Kontrollen aufgebaut. Die Codierung wurde manuell in deutscher Sprache durchgeführt.

Der Aufruf erfolgt im Konstruktor.

Kodierung 10-16: InitializeComponent

### 10.3.4 Bearbeitung der Ereignisse für die Passwortdialogform und die Kontrollen

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten.

Lautet das zu kontrollierende Passwort "Testversion", wird "Testversion" in die Passwort-Textbox eingetragen.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

Die Form wird wegen Problemen mit FormStartPosition.CenterParent in InitializeComponent hier manuell zentriert (nicht für PocketPC).

Kodierung 10-17: OnLoad für die Passwortdialogform

PasswortDialogPanelOnPaint gibt einen sprachabhängigen Erläuterungstext zum Panel aus.

PasswortDialogPanelOnPaint ist der eingetragene Handler für das Paint-Ereignis des Panels aus InitializeComponent.

Kodierung 10-18: PasswortDialogPanelOnPaint für die Passwortdialogform

OnClosed kontrolliert das eingegebene Passwort mit dem Kontrollpasswort unter Berücksichtigung des Start- und Ende-Index. Bei einem korrekt eingegebenen Passwort ist DialogResult OK.

OnClosed überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, nachdem die Dialogform geschlossen wurde.

Kodierung 10-19: OnClosed für die Passwortdialogform

## 10.4 Die Dialogform zum Kopieren des Registrierungsschlüssels

Die REGISTRIERUNGS_DIALOG-Klasse erzeugt eine Dialogform zum Kopieren des ausgewählten Registrierungsschlüssels in die Zwischenablage.

Kodierung 10-20: Die REGISTRIERUNGS_DIALOG-Klasse

```
///*************************************************************************
/// Version 1.1 vom 12.10.09
///*************************************************************************
[CLSCompliantAttribute(true)]      // markiert die Klasse als CLS-kompatibel
public class REGISTRIERUNGS_DIALOG : Form
{

    #region Datenfelder, Konstruktor und InitializeComponent #endregion
    #region Bearbeitung der Ereignisse für die Dialogform und die Kontrollen #endregion

}
```

### 10.4.1 Die Instanzvariablen

Kodierung 10-21: Die Instanzvariablen

### 10.4.2 Der Konstruktor

Der Konstruktor für die REGISTRIERUNGS_DIALOG-Klasse initialisiert ein zweidimensionales Array mit multilingualen Texten für die Form und erstellt die Registrierungs-Dialogform in InitializeComponent.

Kodierung 10-22: Der Konstruktor

### 10.4.3 InitializeComponent

In InitializeComponent wird die Registrierungs-Dialogform mit den Kontrollen aufgebaut. Die Codierung wurde manuell in deutscher Sprache durchgeführt.

Der Aufruf erfolgt im Konstruktor.

Kodierung 10-23: InitializeComponent

### 10.4.4 Bearbeitung der Ereignisse für die Registrierungsdialogform und die Kontrollen

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

Die Form wird wegen Problemen mit FormStartPosition.CenterParent in InitializeComponent hier manuell zentriert (nicht für PocketPC).

Kodierung 10-24: OnLoad für die Registrierungsdialogform

RegistrierungsDialogPanelOnPaint gibt einen sprachabhängigen Erläuterungstext zum Panel aus. Hierzu wird auch die Position der Bildlaufleisten abgefragt.

RegistrierungsDialogPanelOnPaint ist der eingetragene Handler für das Paint-Ereignis des Panels aus InitializeComponent.

Kodierung 10-25: RegistrierungsDialogPanelOnPaint für die Registrierungsdialogform

Copy_Button_Click kopiert den Registrierungsschlüssel von dem Label in die Zwischenablage.

Der Aufruf erfolgt als eingetragener Handler für das Click-Ereignis des Copy-Button (Accept-Button) aus InitializeComponent.

Kodierung 10-26: Copy_Button_Click für die Registrierungsdialogform

In RB_RadioButton_Click wird nach Auswahl einer neuen Registrierungsart der Labeltext zum zugehörigen Registrierungsschlüssel gesetzt.

Der Aufruf erfolgt als eingetragener Handler für das Click-Ereignis der Radio-Buttons aus InitializeComponent.

Kodierung 10-27: RB_RadioButton_Click für die Registrierungsdialogform

## 10.5 Die Dialogform zur Eingabe des Aktivierungsschlüssels

Die AKTIVIERUNGS_DIALOG-Klasse erzeugt eine Dialogform zur Eingabe des Aktivierungsschlüssels.

Diese Dialogform wird innerhalb des Lizenzschutzablaufes nicht eingesetzt, weil der Aktivierungsschlüssel in einer Lizenzdatei enthalten ist, die an den Kunden übermittelt wird.

# 11 Der Mirage-Lizenzschutz

Die Methode ProtectMe von der Mirage Computer Systems GmbH wurde unverändert innerhalb der LIZENZSCHUTZ-Klasse des Basisprojekts zur Verfügung gestellt und von der Anwendung 1 und dem Formen-Verwaltungsprogramm in den Programmablauf eingebunden.

Der Mirage-Lizenzschutz wird nur durchgeführt, wenn das Symbol MirageLP für bedingte Kompilierung den Eigenschaften für das Erstellen der Projekte zugefügt ist.

Der Aufruf der Kontrollmethode, in der ProtectMe aufgerufen wird, erfolgt durch die Anwendung 1 in der Programmschleife des Basisprojekts.

Kodierung 11-1: Mirage Lizenzschutz-Methode ProtectMe

# 12 Auf Sensoren zugreifen

Zur Erfassung von Sensordaten wird das Windows API Code Pack eingesetzt.

Hierzu müssen dem Visual Studio-Projekt zwei Verweise auf die benötigten dll-Dateien des Windows API Code Pack hinzugefügt werden: Microsoft.WindowsAPI.CodePack und Microsoft.WindowsAPICodePack.Sensors.

Des Weiteren ist in den Projekteigenschaften unter „Erstellen“ das Symbol für Bedingte Compilierung „WAPICP“ einzutragen.

Die Methoden für den Zugriff auf Sensoren sind in der Quellcodedatei WAPI_CodePack.cs enthalten, mit der die BASISPROJEKT-Klasse um das Windows API Code Pack erweitert wird.

Hinweis 10 zum Windows API Code Pack

Die im Windows API Code Pack 1.1 integrierten Klassen entsprechen nicht der Common Language Spezification (CLS)!

## 12.1 Die Sensordaten

Die Felder für die Sensoren werden mit der Instanz der BASISPROJEKT-Klasse erzeugt.

Kodierung 12-1: Felder für die Sensoren

## 12.2 Den Beschleunigungssensor einrichten

BS_Sensor_Init ist die Einrichtungsmethode für den Beschleunigungssensor.

Das Abfrageintervall des Beschleunigungssensors wird auf das kleinstmögliche gesetzt.

Zur Weitergabe neuer Sensordaten wird eine Ereignismethode eingerichtet.

Wenn der Beschleunigungssensor korrekt eingerichtet ist, gibt die Methode true zurück.

Kodierung 12-2: BSSensor_Init

## 12.3 Die Ereignismethode für den Beschleunigungssensor

BSSensor_Handler ist die Ereignismethode für den Beschleunigungssensor, wenn neue Daten vorliegen.

Wenn der Beschleunigungssensor gültig ist, werden die aktuellen Daten dem Beschleunigungsvektor übergeben.

Der Ereignishandler wird in BSSensor_Init erzeugt.

Für MULTITHREAD wird vor der Veränderung von öffentlichen Instanzvariablen der Bedienerthread mit dem Arbeitsthread synchronisiert.

Kodierung 12-3: Ereignismethode BSSensor_Handler

## 12.4 Den Beschleunigungssensor freigeben

Freigabemethode für den Beschleunigungssensor.

Der Eventhandler wird entfernt und der Beschleunigungssensor zu null gesetzt.

Kodierung 12-4: BSSensor_Exit

## 12.5 Den Umgebungslichtsensor einrichten

BS_Sensor_Init ist die Einrichtungsmethode für den Umgebungslichtsensor.

Das Abfrageintervall des Umgebungslichtsensors wird auf das kleinstmögliche gesetzt.

Zur Weitergabe neuer Sensordaten wird eine Ereignismethode eingerichtet.

Wenn der Umgebungslichtsensor korrekt eingerichtet ist, gibt die Methode true zurück.

Kodierung 12-5: Lichtsensor_Init

## 12.6 Die Ereignismethode für den Umgebungslichtsensor

Lichtsensor_Handler ist die Ereignismethode für den Umgebungslichtsensor, wenn neue Daten vorliegen.

Wenn der Umgebungslichtsensor gültig ist, wird die aktuelle Lichtstaerke in ein Feld abgespeichert.

Der Ereignishandler wird in Lichtsensor_Init erzeugt.

Für MULTITHREAD wird vor der Veränderung von öffentlichen Instanzvariablen der Bedienerthread mit dem Arbeitsthread synchronisiert.

Kodierung 12-6: Ereignismethode Lichtsensor_Handler

## 12.7 Den Umgebungslichtsensor freigeben

Freigabemethode für den Umgebungslichtsensor.

Der Eventhandler wird entfernt und der Umgebungslichtsensor zu null gesetzt.

Kodierung 12-7: Lichtsensor_Exit

# 13 Eine Beispielanwendung

Das Projekt Anwendung1 realisiert ein Beispielprogramm, das von der BASISPROJEKT-Klasse abgeleitet wird.

Das Programm der Anwendung 1 gibt wahlfrei einfache geometrische Figuren auf den Raumboden der Hintergrundgrafik aus. Hierzu wird die virtuelle Methode der BASISPROJEKT-Klasse überschrieben.

Zusätzlich zu den geerbten Fähigkeiten des Basisprogramms, wird nach jeder Größenänderung der Form auf PanelOnResize eine Raumgrafik in die Hintergrundbitmap gezeichnet, die auf PanelOnPaint in das Grafikobjekt für die Ausgabe zur Client Area kopiert wird.

Des Weiteren erweitert die Anwendung das Basismenü und stellt eine Dialogform mit zwei Seiten zur Verfügung. Die Raumgrafik kann über das Menü und auch über die Dialogform gewechselt werden. Zur Auswahl stehen drei verschiedene Muster, durchsichtig und einfarbig.

## 13.1 Die Klassen der Anwendung 1

Das Projekt Anwendung1 besteht aus den Klassen ANW1MAINCLASS, ANWENDUNG1, ANW1_GLOBALS, ANW1_TEXTE, ANW1_MENUE_TEXTE, EVENT_PROCS für die erweiterte Ereignisbearbeitung und den Klassen für die Dialogform ANW1_DIALOG und ANW1_DIALOG_TEXTE. Alle Klassen sind in dem Namensraum Anw1 aufgeführt.

## 13.2 Die ANW1MAINCLASS-Klasse

ANW1MAINCLASS enthält neben der statischen Methode Main, die durch den Compiler als Startobjekt für die Anwendung aufgerufen wird, noch zwei Methoden als Handler von globalen Ausnahmen.

Für einen kontrollierten Ablauf wird in der statischen Main-Methode in einem try-catch-Block ein ANWENDUNG1-Objekt mit einer separaten Anweisung erzeugt und die Standardmeldungsschleife für die Anwendung in einer zweiten Anweisung mit Application.Run gestartet, wodurch auch die Form sichtbar und bedienbar wird. Nach einer hierin abgefangenen und protokollierten Ausnahme wird die Anwendung beendet.

Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden deshalb zwei Ausnahmehandler erzeugt, nach deren Aktivierung die Anwendung fortgeführt werden kann.

Kodierung 13-1: Die ANW1MAINCLASS-Klasse

## 13.3 ANWENDUNG1-Klasse

Die Klasse ANWENDUNG1 ist von der BASISPROJEKT-Klasse abgeleitet und realisiert eine kleine Beispielanwendung. Sie enthält neben dem Konstruktor auch override-Methoden, die virtuelle Methoden des Basisprojekts überschreiben.

Diese Klassendefinition muss an erster Stelle des Quellcodes stehen, sonst ist keine Designer-Unterstützung der Form möglich.

Kodierung 13-2: Die ANWENDUNG1-Klasse

```
#region ANWENDUNG1-Klasse
///*****************************************************************************
/// Version 3.0 vom 03.06.14
///*****************************************************************************
[CLSCompliantAttribute(true)]
public class ANWENDUNG1 : jkBPrj.BASISPROJEKT
{
```

```
    #region Datenfelder, Konstruktor, Programm, Programm_Init, Titeltext und InitializeComponent
    #endregion

    #region Erweiterung der Basismenüs #endregion

    #region Handler für die Form-Ereignisse #endregion

    #region Bearbeitung der Menü-Ereignisse #endregion

    #region Handler für die Ereignisse vom Lichtsensor #endregion

}
#endregion
```

## 13.3.1 Die Instanzvariablen der Anwendung 1

Auf die implizit privaten Instanzvariablen kann nur innerhalb der ANWENDUNG1-Klasse zugegriffen werden. Einzig public ist die Instanz ANW1_DIALOG für die Dialogform der Anwendung1 zum Löschen der Referenz nach dem Schließen der Dialogform.

Kodierung 13-3: Die Instanzvariablen der ANWENDUNG1-Klasse

## 13.3.2 Der Konstruktor der ANWENDUNG1-Klasse

Im ANWENDUNG1-Konstruktor werden Instanzen von den Basisklassen ANW1_GLOBALS und EVENT_PROCS erzeugt. Der Aufruf erfolgt in ANW1MAINCLASS.Main.

Dem Konstruktor der Basis wird für einen Kontrollablauf mit Passwort- oder Aktivierungsschlüsselabfrage ein Programmpasswort übergeben. Bei einer Passwortabfrage wird "Testversion" als gültiges Passwort eingetragen. Für die Abfrage eines Aktivierungsschlüssels muss der dem Passwort folgende Parameter true sein.

Der Lizenzschutz des mirage Licence Protector wird eingebunden, wenn der erste Parameter bool ist. Der zweite string-Parameter ist hierzu der Project Secure Key (PSK). Zur Kompilierung der hierfür erforderlichen Methoden und Variablen muss die Bedingte Kompilierungskonstante MirageLP definiert sein.

Kodierung 13-4: Der Konstruktor der ANWENDUNG1-Klasse

## 13.3.3 Der Titeltext für die Anwendung 1

Titeltext schreibt für die Anwendung 1 den mehrsprachigen Text in die Titelleiste der Form.

Titeltext überschreibt die geerbte Methode des Basisprojekts. Der Aufruf erfolgt durch das Basisprogramm.

Kodierung 13-5: Titeltext für die Anwendung 1

## 13.3.4 InitializeComponent für die Anwendung 1

InizializeComponent überschreibt die geerbte Methode des Basisprojekts und wird durch das Basisprogramm aufgerufen.

Weil die Form der Anwendung 1 mit den Menüs manuell codiert wird, darf der Inhalt dieser Methode mit dem Code-Editor verändert werden. Auf die Designerunterstützung wird wegen weitergehender Probleme verzichtet.

Kodierung 13-6: InitializeComponent für die Anwendung 1

# 13.4 Der Programmablauf der Anwendung 1

## 13.4.1 Der Programmstart mit Lizenzkontrolle

Durch das ausgewählte Startobjekt innerhalb der Projekt-Eigenschaften für die Anwendung 1, wird nach dem Programmstart die ANW1MAINCLASS des Namensraums Anw1 ausgeführt. Des Weiteren werden die virtuellen und überschreibbaren Methoden der BASISPROJEKT-Klasse durch gleichnamige, überschreibende Methoden der ANWENDUNG1-Klasse ersetzt, weil ANWENDUNG1 von BASISPROJEKT abgeleitet ist.

Nach dem Programmstart wird OnLoad von der Anwendung ausgeführt und darin OnLoad des Basisprojekts aufgerufen. In OnLoad für das Basisprojekt wird nur für MULTITHREAD ein Arbeitsthread mit MT_Programmschleife als Startmethode und ein Zeitgeber zur Aktivierung von Programmschleife gestartet.

In Programmscheife wird für den Ablauf mit MULTITHREAD nur ein erforderlicher Lizenzschutz-Kontrollablauf durchgeführt. Der eigentliche Programmablauf wird durch den Arbeitsthread in MT_Programmschleife ausgeführt, worin die Basis-Methode ständig aktiviert wird. In der Basis-Methode wird die von der Anwendung überschriebene Methode Programm virtuell aufgerufen.

## 13.4.2 Die Initialisierung der Anwendung 1

Programm_Init liest oder schreibt relevante öffentliche Variable des Basisprogramms und von dieser Anwendung 1 in eine Initialisierungsdatei.

Hierzu wird die virtuelle Methode des Basisprojekts überschrieben und die Basismethode selbst aufgerufen. Der Aufruf erfolgt innerhalb des Basisprogramms.

Kodierung 13-7: Programm_Init der Anwendung 1

## 13.4.3 Die Ausführung der Anwendung 1

Die Anwendung 1 gibt wahlfrei einfache geometrische Figuren auf den Raumboden der Hintergrundgrafik aus. Hierzu wird die virtuelle Methode von BASISPROJEKT überschrieben.

Der Aufruf der virtuellen Basisklassen-Methode erfolgt in Basis von BASISPROJEKT nur, wenn keine Programmpause.

Kodierung 13-8: Programm von der Anwendung 1

# 13.5 Erweiterungen und Änderungen zur Version 3.0

- Für moderne Geräte  (Tablets, Phones) mit multicore-Prozessoren werden Endversionen für das .Net (Compact) Framework mit Multithreadablauf realisiert. Hierzu wurden vom MDI-Projektworkspace von allen relevanten Dateien die Quelltexte mit dem Symbol für Bedingte Compilierung MULTITHREAD in die Dateien der Anwendung 1 übernommen.
- Die Raumgrafik wird mit dem Sensorwert für das Umgebungslicht gezeichnet (wenn vorhanden). Hierzu wird die Grafik-Dialogseite um eine Editierkontrolle für das Umgebungslicht mit Sensortaste erweitert.
- Zur Realisierung der Raumgrafik mit oder ohne Farbwechsel wird die Methode Groessen() mit dem Parameter Farbwechsel aufgerufen.
- Für die Methoden Programm, Titeltext und InitializeComponent wird der Zugriffsmodifikator wie im Basisprojekt von public zu protected geändert.
- In Programm_Init() werden wegen einer möglichen Ausnahme die Instanzen von BReader und BWriter innerhalb eines finally-blocks geschlossen.

## 13.6 Erweiterungen und Änderungen zur Version 2.0

- Zur weitergehenden Anzeige im Objektkatalog und für die kontextsensitive Hilfe den XML-Kommentar für alle Methoden mit <para>-Tags innerhalb der <summary>- und <remarks>-Tags versehen.
- Neben der Erhöhung der globalen Variablen catch_Ausnahmen, werden die innerhalb des Programmablaufes auftretenden Ausnahmen mit Datum und Uhrzeit durch Aufruf von Ausnahmeprotokollierung() der BASIS_PROCS-Klasse in die Datei „Anwendung 1_Exceptions.txt“ geschrieben.
- Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden in ANW1MAINCLASS. Main zwei Ausnahmehandler erzeugt, nach deren Aktivierung die Anwendung fortgeführt wird.
- Aufgrund von Änderungen im Basisprojekt, wird die OnResize-Methode der Form zu dem in ErzeugeFormPanel() eingetragenen Handler PanelOnResize() für das Resize-Ereignis des Panels geändert. PanelOnResize() wird über das Basisprojekt in OnResize() nur aktiviert, wenn die Form nicht zum Symbol verkleinert wird oder vom Symbol wieder hergestellt wurde.
- Aufruf von PanelOnResize() in RaumgrafikmenueOnClick().
- In EPPOCS.Groessen() wird nicht mehr der FormWindowState abgefragt und durch FormPanel.Invalidate() zur Anzeige des Hintergrundes ein PanelOnPaint-Ereignis angestoßen.
- Zur Anpassung an eine neue Sprachauswahl wird in ErzeugeFormMenues() eine geöffnete Anwendungsdialogform durch Aufruf ihres Load-Handlers neu aufgebaut.
- Der AboutBox werden vor der Anzeige eine neue Imagedatei und zwei LinkLabels zugefügt. Dies gilt nur für die Anzeige über den Menüpunkt „Hilfe/über...“. Nach der Anzeige durch den „über“-Button der Basis-Dialogseite wird derzeit nur der Beschreibungstext neu zugefügt.

## 13.7 Die öffentlichen Datenfelder der Anwendung 1

Datenfelder, die für andere Klassen öffentlich zugänglich sein müssen, werden in einer eigenen Klasse ANW1_GLOBALS aufgeführt. Auf diese Datenfelder kann hierdurch über eine separate Referenz zugegriffen werden. ANW1_GLOBALS wird einmal bei der Erzeugung der Instanz im Konstruktor der ANWENDUNG1-Klasse aufgerufen.

Der Konstruktor von ANW1_GLOBALS initialisiert die öffentlichen Datenfelder der Anwendung 1.

Von ANW1_GLOBALS werden automatisch Objekte von ANW1_TEXTE und ANW1_MENUE_TEXTE als Basisklassen abgeleitet. Diese enthalten statische zwei- oder dreidimensionale Arrays in denen die mehrsprachigen Texte für das Basisprojekt und das Menü eingetragen werden.

Die Menütexte enthalten zusätzlich einen kleinen Erläuterungstext zur Menüauswahl, der für die Ausgabe zur Statusleiste verwendet wird.

Kodierung 13-9: die ANW1_GLOBALS-Klasse

### 13.7.1 Die mehrsprachigen allgemeinen Texte der Anwendung 1

Die ANW1_TEXTE-Klasse enthält mehrdimensionale, öffentlich zugängliche Textarrays für multilinguale Projekt- und Messagebox-Texte der Anwendung 1 und erbt die Menütexte von ANW1_MENUE_TEXTE. Eine Instanz wird mit der ANW1_GLOBALS-Klasse erzeugt.

Im Konstruktor werden die multilingualen Texte für die Anwendung 1 in den Textarrays bereitgestellt.

Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

Kodierung 13-10: Die ANW1_TEXTE-Klasse

### 13.7.2 Die mehrsprachigen Menütexte der Anwendung 1

Die ANW1_MENUE_TEXTE-Klasse enthält mehrdimensionale, öffentlich zugängliche Textarrays für die multilingualen Menütexte der Anwendung 1 mit Erläuterungstexten. Eine Instanz wird mit der ANW1_TEXTE-Klasse erzeugt.

Im Konstruktor werden die multilingualen Menütexte für die Anwendung 1 in den Textarrays bereitgestellt.

Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

Kodierung 13-11: Die ANW1_MENUE_TEXTE-Klasse

## 13.8 Erweiterung der Basismenüs für die Anwendung 1

In ErzeugeFormMenues wird das multilinguale Haupt- und Kontextmenü des Basisprojekts für die Anwendung 1 erweitert. Hierzu wird die gleichnamige virtuelle Basismethode überschrieben und selbst aufgerufen. Danach werden die Menüpunkte verändert und erweitert.

Eine geöffnete Anwendungsdialogform wird zur Anpassung an eine neue Sprachauswahl durch Aufruf ihres Load-Handlers neu aufgebaut.

Die Aufrufe erfolgen innerhalb des Basisprojekts in OnLoad, SpracheMenueOnClick und HilfeTextSpracheMenueOnClick.

Nur für MULTITHREAD gilt: wenn diese Methode nicht von ihrem Erzeuger-Thread aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

Kodierung 13-12: ErzeugeFormMenues für die Anwendung 1

## 13.9 Erweiterung des Kontextmenüs für die Anwendung 1

ErzeugeKontextMenue erweitert das Basis-Kontextmenü für die Anwendung 1. Hierzu wird die gleichnamige virtuelle Basismethode überschrieben und selbst aufgerufen.

Der Aufruf erfolgt innerhalb des Basisprojekts in ErzeugeFormMenues.

Kodierung 13-13: ErzeugeKontextMenue für die Anwendung 1

## 13.10 Form-Ereignisse für die Anwendung 1

### 13.10.1 OnLoad

OnLoad Initialisiert die Form und die Anwendung1. Hierzu wird die überschriebene Basismethode aufgerufen.

OnLoad wird nur einmal als Handler für das Load-Ereignis aufgerufen, bevor die Form angezeigt wird.

In der Basismethode werden die Menüs und die StatusBar- und Panel-Kontrolle erzeugt und zur Initialisierung die virtuellen Methoden Programm_Init, InitializeComponent und Titeltext aufgerufen.

Kodierung 13-14: OnLoad für die Anwendung 1

### 13.10.2 PanelOnResize

PanelOnResize ruft die überschriebene Basismethode und für die erweiterte Ereignisbehandlung Groessen von der EVENT_PROCS-Klasse auf.

Der Aufruf erfolgt als Handler für das Resize-Ereignis des Panels und explizit auf OnLoad und nach einer Raumgrafikauswahl in RaumgrafikMenueOnClick.

Die Basismethode erzeugt zur allgemeinen Verwendung ein GDI+ Grafik-Objekt abgeleitet von dem Panel in der neuen Größe der Client Area des Panels. In dieser Größe wird auch eine Basisbitmap mit einem zugehörigen GDI+ Grafik-Objekt erzeugt.

Kodierung 13-15: PanelOnResize für die Anwendung 1

### 13.10.3 OnClosed

OnClosed gibt von der Anwendung erzeugte Objekte frei (wenn vorhanden) und ruft danach OnClosed von der Basis auf.

Der Aufruf erfolgt als Handler für das Closed-Ereignis.

Kodierung 13-16: OnClosed für die Anwendung 1

### 13.10.4 Dispose

Dispose gibt die von Form verwendeten Ressourcen (mit Ausnahme des Speichers) frei und ruft danach die überschriebene Basismethode der Form auf. Dispose wurde als Standardmethode unverändert übernommen.

Der Aufruf erfolgt nachdem OnClosed für die Anwendung 1 durchlaufen wurde.

Kodierung 13-17: Dispose für die Anwendung 1

## 13.11 Erweiterte Ereignisbearbeitung in der EVENT_PROCS-Klasse

Die EVENT_PROCS-Klasse enthält Methoden für die Anwendung 1 zur erweiterten Ereignisbearbeitung und zum Zeichnen der Raumgrafik.

Kodierung 13-18: Die EVENT_PROCS-Klasse

```
#region EVENT_PROCS-Klasse
///*************************************************************************
/// Version 3.0 vom 05.06.14
///*************************************************************************
[CLSCompliantAttribute(true)]    // markiert die Klasse als CLS-kompatibel
public class EVENT_PROCS
{

    #region Datenfelder und Konstruktor #endregion

    #region Erweiterte Ereignisbearbeitung #endregion

    #region Zeichnen der Raumgrafik in die Basisbitmap #endregion

}
#endregion
```

### 13.11.1 Die Instanzvariablen

Auf die implizit privaten Instanzvariablen kann nur innerhalb der EVENT_PROCS-Klasse zugegriffen werden.

Kodierung 13-19: Die Datenfelder der EVENT_PROCS-Klasse

### 13.11.2 Der Konstruktor

Als Parameter werden dem Konstruktor der EVENT_PROCS-Klasse die Referenzen auf die Instanzen der öffentlichen Datenfelder der Anwendung 1 und des Basisprojekts übergeben. Hierdurch wird in EVENT_PROCS der Zugriff auf das Basisprojekt und die Form ermöglicht.

Diese Instanz wird im Konstruktor der ANWENDUNG1-Klasse erzeugt.

Kodierung 13-20: Der Konstruktor der EVENT_PROCS-Klasse

### 13.11.3 Groessen auf PanelOnResize für die Anwendung 1

Groessen berechnet wichtige Programmvariable und ruft Hintergrund auf, wenn keine Imagedatei angezeigt wird.

Für MULTITHREAD wird der Ablauf zuvor mit dem Arbeitsthread synchronisiert.

Groessen wird nach jeder Ausführung von PanelOnResize aufgerufen und weitergehend mehrfach zur Aktualisierung der Raumgrafik.

In PanelOnResize wird bei einer Programmausnahme auch eine Ausnahmeprotokollierung durchgeführt.

Kodierung 13-21: Groessen auf PanelOnResize für die Anwendung 1

## 13.12 Zeichnen der Raumgrafik in die Basisbitmap

Hintergrund zeichnet die ausgewählte Raumgrafik in die vom Basisprojekt zur Verfügung gestellte Basisbitmap oder gibt eine neue Farbe für den Hintergrund der Form an. Der Aufruf erfolgt in Groessen.

Kodierung 13-22: Hintergrund

### 13.12.1 Raumfarben

Raumfarben zeichnet die Raumwände mit linear abnehmenden Farbwerten und den Boden mit der Hintergrundfarbe. Hierbei wird für alle Farbwerte durch diffuse Reflexion der Umgebungslichtwert berücksichtigt.

Der Aufruf erfolgt in Hintergrund.

Kodierung 13-23: Raumfarben

### 13.12.2 Hatchmuster (nicht für Smart Devices)

Hatchmuster zeichnet die Raumwände mit Hatch-Mustern. Der Aufruf erfolgt in Hintergrund.

Kodierung 13-24: Hatchmuster

### 13.12.3 LinearGradient Farbmuster (nicht für Smart Devices)

LinearGradient_Farbmuster zeichnet die Raumwände mit dem LinearGradientBrush mit zwei Wandfarben und vier wechselnden Raumfarben.

Hierbei wird für alle Farbwerte durch diffuse Reflexion der Umgebungslichtwert berücksichtigt.

Der Aufruf erfolgt in Hintergrund.

Kodierung 13-25: LinearGradient_Farbmuster

### 13.12.4 PathGradientmuster (nicht für Smart Devices)

PathGradientmuster zeichnet die Raumwände und den Raumboden mit PathGradient-Mustern. Der Aufruf erfolgt in Hintergrund.

Kodierung 13-26: PathGradientmuster

### 13.12.5 Diffuse_Reflexion

Ausgehend von einem Farbwert in der Nähe berechnet Diffuse_Reflexion einen diffusen Farbwert in der Entfernung.

Hierbei wird auch der Umgebungslichtwert aus der Editierkontrolle der Grafikdialogseite oder von einem aktiven Sensor berücksichtigt.

Kodierung 13-27: Diffuse_Reflexion

### 13.12.6 HgrLinien (nur für Smart Devices)

HgrLinien zeichnet die Raumwände mit Linien. Der Aufruf erfolgt in Hintergrund.

Kodierung 13-28: HgrLinien

#### 13.12.6.1 Linienschar (nur für Smart Devices)

Linienschar zeichnet Linien in eine Raumwand. Der Aufruf erfolgt in HgrLinien.

Kodierung 13-29: Linienschar

## 13.13 Ereignisse von den Menüs der Anwendung 1

### 13.13.1 Das Abschließende Menüereignis

Nicht für Smart Devices unter dem .Net Compact Framework gibt OnMenuComplete den Starttext der Anwendung 1 wieder zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als überschreibender Handler der geerbten Form-Methode, nachdem die Menübehandlung abgeschlossen wurde.

Kodierung 13-30: OnMenuComplete für die Anwendung 1

### 13.13.2 Ereignisse für das Programmmenüelement

ProgrammMenueOnClick bearbeitet die Clicks der von der Anwendung 1 erzeugten Elemente des Programm-Menüs.

Bei Auswahl von "Dialog" wird die modeless Dialogform der Anwendung nur dann erzeugt und angezeigt, wenn nicht schon eine erzeugt wurde. Auf Menüanwahl "Beenden !" wird die Anwendung beendet.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Programm-Menüpunkt angeklickt wurde. Die gleichnamige Methode des Basisprojekts wird überschrieben. Diese wird selbst nur für den Pause-Menüindex aufgerufen. Für PocketPC ist der public-Zugriffsmodifizierer wegen direkter Methodenaufrufe erforderlich.

Nur für MULTITHREAD wird vorab geprüft, ob der Aufruf von dem Erzeuger-Thread erfolgt. Ist dies nicht der Fall, wird ProgrammMenueOnClick erneut durch ihre Delegat-Methode synchron im Erzeuger-Thread ausgeführt.

Kodierung 13-31: ProgrammMenueOnClick für die Anwendung 1

Nicht für Smart Devices unter dem .Net Compact Framework gibt ProgrammMenueOnSelect den Hilfetext für den ausgewählten Programm-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Programm-Menüpunkt ausgewählt wurde. Die gleichnamige Methode des Basisprojekts wird überschrieben und nur für den Pause-Menüindex aufgerufen.

Kodierung 13-32: ProgrammMenueOnSelect für die Anwendung 1

## 13.13.3 Ereignisse für das Raumgrafikmenüelement

RaumgrafikMenueOnClick bearbeitet die Clicks der Raumgrafik-Menüelemente. Die Raumgrafik wird durch Aufruf von Groessen mit Farbwechsel in die neue BasisBitmap gezeichnet.

Der Aufruf erfolgt als ein in ErzeugeFormMenues und ErzeugeKontextMenue für Click-Ereignisse eingetragener Handler, wenn ein Raumgrafik-Menüelement angeklickt wurde. Wenn die Raumgrafik über die Dialogform geändert wurde erfolgt der Aufruf über PerformClick (für PocketPC durch einen direkten Aufruf der Methode).

Kodierung 13-33: RaumgrafikMenueOnClick für die Anwendung 1

Nicht für Smart Devices unter dem .Net Compact Framework gibt RaumgrafikMenueOnSelect den Hilfetext für den ausgewählten Raumgrafik-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Raumgrafik-Menüelement ausgewählt wurde.

Kodierung 13-34: RaumgrafikMenueOnSelect für die Anwendung 1

## 13.13.4 Ereignisse für das Kontextmenü

KontextMenueOnClick bearbeitet die Clicks der von der Anwendung 1 erzeugten Elemente des Kontextmenüs.

Bei Auswahl von "Dialog" wird die modeless Dialogform der Anwendung nur dann erzeugt und angezeigt, wenn nicht schon eine erzeugt wurde.

Der Aufruf erfolgt als ein in ErzeugeKontextMenue für Click-Ereignisse eingetragener Handler, wenn ein Kontextmenüpunkt angeklickt wurde. Die gleichnamige Methode des Basisprojekts wird überschrieben und selbst aufgerufen, wenn der Menüindex kleiner als sechs ist.

Kodierung 13-35: KontextMenueOnClick für die Anwendung 1

Nicht für Smart Devices unter dem .Net Compact Framework gibt KontextMenueOnSelect den Hilfetext für den ausgewählten Kontextmenüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeKontextMenue für Select-Ereignisse eingetragener Handler, wenn ein Kontextmenüpunkt ausgewählt wurde. Die gleichnamige Methode des Basisprojekts wird überschrieben und nur aufgerufen, wenn der Menüindex kleiner als sechs ist.

Kodierung 13-36: KontextMenueOnSelect für die Anwendung 1

## 13.13.5 Ereignisse für das Hilfemenüelement

HilfeMenueOnClick bearbeitet die Clicks der für die Anwendung 1 relevanten Elemente des Hilfe-Menüs.

Nach Auswahl von "Einführung" und "Menübefehle" wird aus den multilingualen HTML-Hilfetextdateien "Anw1_3_0_?Language?.chm" ein Inhaltsverzeichnis mit dem Einführungstext oder der Beschreibung der Menübefehle aufgerufen.

Nach Auswahl von "über..." wird eine About-Dialogbox mit multilingualer Information über die Anwendung 1, einer Bitmap und zwei Webseiten-Links angezeigt.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Hilfemenüpunkt angeklickt wurde. Der gleichnamige Handler des Basisprojekts wird überschrieben. Für PocketPC ist der public-Zugriffsmodifizierer wegen direkter Methodenaufrufe erforderlich.

Kodierung 13-37: HilfeMenueOnClick für die Anwendung 1

### 13.13.5.1 Das Click-Ereignis für die LinkLabels (nicht für PocketPC)

WebLinkLabel_LinkClicked bearbeitet die Clicks auf die LinkLabels innerhalb des Beschreibungstextes der AboutBox.

Der Aufruf erfolgt als ein in HilfeMenueOnClick für Click-Ereignisse von den LinkLabels eingetragener Handler.

Kodierung 13-38: WebLinkLabel_LinkClicked

# 13.14 Die Ereignismethode für den Umgebungslichtsensor

Lichtsensor_Handler ist die Ereignismethode für den Umgebungslichtsensor, wenn ein neuer Umgebungslichtwert vorliegt.

Die Raumgrafik für den neuen Sensorwert wird durch einen Aufruf von EVENT_PROCS.Groessen realisiert.

Für MULTITHREAD wird Lichtsensor_Handler erneut durch einen Delegaten im Erzeuger-Thread aufgerufen, wenn dies erforderlich ist.

Lichtsensor_Handler überschreibt die virtuelle Methode des Basisprojekts und ruft diese zu Beginn direkt auf.

Kodierung 13-39: Ereignismethode Lichtsensor_Handler

# 14 Die Dialogform der Anwendung 1

Die Dialogform der Anwendung 1, die in der ANW1_DIALOG-Klasse durch Vererbung von der Form-Klasse erzeugt wird, besteht aus drei Seiten mit TabControls zur Seitenauswahl.

Auf der Programmseite kann der Ablauf mit dem Pause / Start-Taster unterbrochen und wieder gestartet werden. In einem Textfeld wird die Anzahl der Aufrufe der Anwendung 1 in einer Sekunde ausgeben. Mit Betätigung der „über...“-Taste wird eine weitere Dialogform mit Informationen zur Programmversion angezeigt.

Von der Grafikseite kann über fünf Optionsfelder die Raumgrafik ausgewählt werden.

Die Grafik 2-Seite enthält vorerst keine Kontrollen.

Hinweis 11 zur Dialogform der Anwendung 1!

Zur Zeiteinsparung wurde die Klasse ANW1_DIALOG in der Quellcodedatei Anw1_Dialog.cs insgesamt vom MDI-Projektworkspace übernommen. Die Kodierungen mit Bedingter Kompilierung für MDI_PRJ wurden entfernt. Auch die Dokumentation wurde im ganzen Kapitel übernommen und daraufhin angepasst.

Kodierung 14-1: die ANW1_DIALOG-Klasse

```
///**************************************************************************
/// Version 3.0 vom 01.06.14
///**************************************************************************
[CLSCompliantAttribute(true)]     // markiert die Klasse als CLS-kompatibel
public class ANW1_DIALOG : Form
{

    #region Datenfelder und Konstruktor #endregion

    #region InitializeComponent für den Visual Studio Designer #endregion

    #region Ereignisse zur Verwaltung der Dialogform #endregion

    #region Ereignisse für die Kontrollen der Programm Dialogseite und den Timer #endregion

    #region Ereignisse für die Kontrollen der Grafik-Dialogseite #endregion

    #region Ereignisse für die Kontrollen der Grafik 2-Dialogseite #endregion

}
```

## 14.1 Die Instanzvariablen

Die Dialogkontrollen wurden von dem Designer bei der Erstellung der Dialogformen zugefügt und dürfen nicht manuell geändert werden.

Kodierung 14-2: die Instanzvariablen der Anwendung 1-Dialogform

## 14.2 Der Konstruktor

Im Konstruktor für die ANW1_DIALOG-Klasse werden die als Parameter übergebenen Referenzen als Instanzvariable abgespeichert. Danach wird eine Instanz von ANW1_DIALOG_TEXTE erstellt und in InitializeComponent die mit dem Designer erstellte Dialogform für die Anwendung 1 aufgebaut.

Der Aufruf erfolgt nach Menüpunktauswahl "Dialog..." in  ProgrammMenueOnClick und KontextMenueOnClick.

Kodierung 14-3: der Konstruktor für die Dialogform der Anwendung 1

# 14.3 Die Dialogform mit dem Visual Studio-Designer erzeugen

InitializeComponent erzeugt die Dialogseiten der Anwendung 1 mit den Kontrollen. Der Aufruf erfolgt im ANW1_DIALOG-Konstruktor.

Die Kodierung wird automatisch mit dem in Visual Studio integrierten Designer in deutscher Sprache erzeugt und darf nicht manuell geändert werden.

ANW1_DIALOG_HelpRequested ist als Handler für das HelpRequested-Ereignis der Dialogform eingetragen. OKButton_Click, AbbrechenButton_Click, HilfeButton_Click, PauseButton_Click, Info_Button_Click, RGRadioButton_Click sind die eingetragenen Handler für die Click-Ereignisse der Button-Kontrollen.

```
///**************************************************************************
/// Version 2.0 vom 08.08.10
/// Ein- und Ausgabedaten:
/// alle Kontrollen der Dialogform der Anwendung 1
///**************************************************************************
private void InitializeComponent()
{...}
```

# 14.4 Ereignisse zur Verwaltung der Dialogform

## 14.4.1 LoadHandler

LoadHandler initialisiert die Kontrollen mit Daten und multilingualen Texten und führt ToolTips und erweiterte Hilfetexte ein.

LoadHandler ist der im Konstruktor eingetragene Handler für das Load-Ereignis.

Die Aufrufe erfolgen bevor die Dialogform angezeigt wird und explizit zur Korrektur der Kontrollen nach einer Menüerzeugung in ErzeugeFormMenues.

Zur vollständigen Initialisierung der Anwendung 1-Dialogform werden die Init-Methoden der einzelnen Dialogseiten aufgerufen.

Nur für MULTITHREAD gilt, wenn diese Methode nicht von ihrem Erzeuger-Thread aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

Kodierung 14-4: LoadHandler für die Anwendung 1-Dialogform

## 14.4.2 Anw1_DIALOG_HelpRequested

Anw1_DIALOG_HelpRequested zeigt eine Message-Box mit dem ausführlichen Hilfetext der ausgewählten Kontrolle an. Der Aufruf erfolgt als ein im Designer eingetragener Handler für das HelpRequested-Ereignis der Basis-Dialogform nach Anwahl einer Kontrolle über den ?-Button oder die F1-Taste.

Das Ereignis wird auch ausgelöst, wenn der Kontrolle über die HelpProvider-Klasse ein Popup-Hilfetext zugewiesen wurde und dieser durch SetShowHelp für die Kontrolle deaktiviert ist.

Kodierung 14-5: Anw1_DIALOG_HelpRequested für die Anwendung 1-Dialogform

## 14.4.3 ANW1_DIALOG_FormClosing

Für die Einzelanwendung wird nur die für die ANWENDUNG 1 eingetragene Referenz auf das ANW1_DIALOG-Objekt zu null gesetzt.

ANW1_DIALOG_FormClosing wird als ein im Konstruktor eingetragener Handler für das FormClosing-Ereignis aufgerufen bevor die Dialogform geschlossen wird.

Kodierung 14-6: ANW1_DIALOG_FormClosing für die Dialogform der Anwendung 1

### 14.4.4 ANW1_DIALOG_FormClosed

Gibt von dieser Klasse erzeugte Objekte und das ANW1_DIALOG-Objekt selbst frei.

ANW1_DIALOG_FormClosed wird als ein im Konstruktor eingetragener Handler
für das FormClosed-Ereignis aufgerufen bevor die Dialogform geschlossen wird.

Kodierung 14-7: ANW1_DIALOG_FormClosed für die Dialogform der Anwendung 1

### 14.4.5 OKButton_Click

OKButton_Click speichert den letzten Dialogseitenindex, setzt die Initialisierungsvariable false und schließt den Dialog mit Close.

Der Aufruf erfolgt als Handler für das Click-Ereignis des OK-Button und als Accept-Ereignis für den Dialog.

Kodierung 14-8: OKButton_Click für die Anwendung 1-Dialogform

### 14.4.6 AbbrechenButton_Click

In AbbrechenButton_Click werden die Einstellungen der Dialogform beim ersten Aufruf wieder hergestellt.

Für MULTITHREAD wird vor der Veränderung der globalen Variablen der Bedienerthread mit dem Arbeitsthread synchronisiert.

Der Aufruf erfolgt als Handler für das Click-Ereignis des Abbrechen-Button und als Cancel-Ereignis für den Dialog.

Kodierung 14-9: AbbrechenButton_Click für die Anwendung 1-Dialogform

### 14.4.7 HilfeButton_Click

Ruft für die aktuelle Dialogseite die HTML-Hilfe auf.

Aufruf als Handler für das Click-Ereignis des Hilfe-Button.

Kodierung 14-10: HilfeButton_Click für die Dialogform der Anwendung1

## 14.5 Mehrsprachige Texte für die Dialogform

Die Dialogtexte werden mehrsprachig in der Klasse ANW1_DIALOG_TEXTE zur Verfügung gestellt. Sie bestehen aus dem Titeltext, den Texten für die TabControls und Messageboxtexten, die alle ohne Tooltips auskommen.

Die weiteren Texte für die Kontrollen aller Dialogseiten enthalten Tooltips und, wenn erforderlich, zusätzlich erweiterte Hilfetexte. Wenn ein erweiterter Hilfetext zur Verfügung steht, wird dem Tooltip ein Hinweis in der Form (-> ?-Hilfe) oder (-> Hilfetext) zugefügt.

Eine Instanz von ANW1_DIALOG_TEXTE wird im Konstruktor der ANW1_DIALOG-Klasse erzeugt.

Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

Kodierung 14-11: Die ANW1_DIALOG_TEXTE-Klasse

# 14.6 Ereignisse für die Kontrollen der Programm-Dialogseite

## 14.6.1 Programm_Init

Programm_Init initialisiert die Kontrollen der Programm-Dialogseite mit Daten (Kontrollen_Init true) und multilingualen Texten und ToolTips (Text_Init true).

Mit Anw1Dialog_Init false, werden die Rückladewerte für den Abbruch der Dialogseite als globale Daten der Anwendung abgespeichert.

Des Weiteren wird ein Sekunden-Timer für den Eintrag der Aufrufe/s in den AufrufeLabel erzeugt.

Die Aufrufe erfolgen in LoadHandler und explizit im Programmablauf zur Aktualisierung der Kontrollen der Programm-Dialogseite.

Nur für MULTITHREAD gilt, wenn diese Methode nicht von ihrem Erzeuger-Thread aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

Kodierung 14-12: Programm_Init für die Programm-Dialogseite

## 14.6.2 Die Timer-Methode AufrufeSekTimer

Timer-Methode des Anw1DlgSekTimer für den Eintrag der Aufrufe/s in ein Label der Programm-Dialogseite. Der Timer wird mit einem Intervall von einer Sekunde in Programm_Init erzeugt.

Kodierung 14-13: Timer-Methode AufrufeSekTimer

## 14.6.3 PauseButton_Click

Unterbricht oder Startet die Anwendung 1.

Der Aufruf erfolgt als Ereignis-Handler des "Pause / Start"-Button auf der Basis-Dialogseite.

Kodierung 14-14: PauseButton_Click für die Anwendung 1-Dialogform

## 14.6.4 InfoButton_Click

InfoButton_Click zeigt eine About-Dialogbox mit multilingualen Informationen über die Anwendung 1 an.

Der Aufruf erfolgt als Ereignis-Handler für den "über"...-Button der Basis-Dialogseite.

Kodierung 14-15: InfoButton_Click für die Anwendung 1-Dialogform

# 14.7 Ereignisse für die Kontrollen der Grafik-Dialogseite

## 14.7.1 Grafik_Init

Grafik_Init initialisiert die Kontrollen der Grafik-Dialogseite mit Daten (Kontrollen_Init true) und multilingualen Texten und ToolTips (Text_Init true).

Mit Anw1Dialog_Init false, werden die Rückladewerte für den Abbruch der Dialogseite als globale Daten der Anwendung abgespeichert.

Die Aufrufe erfolgen in LoadHandler und explizit im Programmablauf zur Aktualisierung der Kontrollen der Grafik-Dialogseite.

Nur für MULTITHREAD gilt, wenn diese Methode nicht von ihrem Erzeuger-Thread aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

Kodierung 14-16: Grafik_Init für die Grafik-Dialogseite

## 14.7.2 RgRadioButton_Click

RgRadioButton_Click bearbeitet eine neue Auswahl der Raumgrafik.

Der Aufruf erfolgt als Ereignis-Handler der Optionsfelder auf der Grafik-Dialogseite.

Kodierung 14-17: RgRadioButton_Click für die Anwendung 1-Dialogform

## 14.7.3 GrafikDialogseite_CheckBox_Click

GrafikDialogseite_CheckBox_Click bearbeitet einen Click auf das Auswahlfeld der Grafik-Dialogseite für den Umgebungslichtsensor.

Für MULTITHREAD wird vor der Veränderung der globalen Variablen der Bedienerthread mit dem Arbeitsthread synchronisiert.

Der Aufruf erfolgt durch den in InitializeComponent eingetragenen Handler für die Click-Ereignisse der Checkboxen.

Kodierung 14-18: GrafikDialogseite_CheckBox_Click

## 14.7.4 Umgebungslicht_Textbox_LostFocus

Umgebungslicht_TextBox_LostFocus verarbeitet die Eingabe in die Textkontrolle für das Umgebungslicht.

Ein korrekt umgewandelter Eingabetext wird zum Umgebungslichtwert. Hierzu wird auch der Wert der ScrollBar angepasst.

Die Raumgrafik für den neuen Umgebungslichtwert wird durch einen Aufruf von EVENT_PROCS.Groessen realisiert.

Der Aufruf erfolgt durch den in InitializeComponent eingetragenen Handler für das LostFocus-Ereignis, wenn der Eingabefocus die Textkontrolle verlässt.

Kodierung 14-19: Umgebungslicht_TextBox_LostFocus für die Grafik-Dialogseite

## 14.7.5 Umgebungslicht_ScrollBar_ValueChanged

Umgebungslicht_ScrollBar_ValueChanged verarbeitet die Bedienung der Drehfelder für die Textkontrolle des Umgebungslichts.

Ein korrekter Bildlauffeldwert wird zum Umgebungslichtwert. Hierzu wird auch der Text in der Textkontrolle angepasst.

Die Raumgrafik für den neuen Umgebungslichtwert wird durch einen Aufruf von EVENT_PROCS.Groessen realisiert.

Der Aufruf erfolgt durch den in InitializeComponent eingetragenen Handler für das ValueChanged-Ereignis, wenn der Benutzer das Bildlauffeld verschiebt.

Kodierung 14-20: Umgebungslicht_ScrollBar_Valuechanged für die Grafik-Dialogseite

# 15 Die Projekte für mobile Geräte unter dem .Net Compact Framework

Für mobile Geräte unter dem .Net Compact Framework wurden dem Projektworkspace zwei Visual C# Smart Device Projekte zugefügt: SmartDevice_Basisprojekt und SmartDevice_Anwendung1.

Hinweis 12 zu den Projekten für mobile Geräte!

Die Projekte stehen in der Professional-Version nur unter Visual Studio 2008 aber nicht unter Visual Studio 2010 zur Verfügung. Es ist auch keine Datei für eine erweiterte Installation bekannt, die eine Kompilierung von Smart-Projekten für das .Net Compact Framework unter Visual Studio 2010 ermöglicht.

Dafür können unter Visual Studio 2010 für Smart Devices Projekte über das XNA Game Studio und Silverlight angelegt werden.

Die Programme werden über Bedingte Kompilierung und angepasste Kodierung in den auch für das .Net Framework verwendeten C#-Dateien realisiert.

Die Bedingte Kompilierung erfolgt mit der Kompilierungskonstanten „PocketPC“.

Für moderne Geräte (Tablets, Phones) mit multicore-Prozessoren werden ab der Version 3.0 die Endversionen für das .Net Compact Framework auch mit Multithreadablauf realisiert.

Aufgrund des in der Version 2.0 neu eingeführten Eventhandlers PanelOnResize kann auch für Smart Devices die Basis-Methode in der Programmschleife aufgerufen werden und auf den Aufruf in PanelOnPaint verzichtet werden. Für eine bessere Bedienbarkeit der Device wird bei einer Programmpause mit Sleep(1) ein Threadwechsel veranlasst.

Weitere Veränderungen an der Kodierung für einen Ablauf auf mobilen Geräten unter dem .Net Compact Framework sind in der Datei Compact-Framework Kodierungsanpassungen.doc zusammengefasst.

## 15.1 Das Basisprojekt für das .Net Compact Framework

Das SmartDevice-Basisprojekt erhält die gleiche Ordnerstruktur wie das Basisprojekt. Dem Projekt werden die Dateien des Basisprojekts über Hinzufügen \ vorhandenes Element \ Hinzufügen als Link zugefügt.

Weil die Dialogformen mit dem Designer entworfen werden und viele Einschränkungen erforderlich sind, wird für SmartDevice-Projekte eine neue Quellcodedatei verwendet. Für das Basisprojekt wurde hierfür die Sourcedatei der Dialogform von dem .Net-Framework übernommen und umbenannt zu Basis_Dialog_CF.cs in das SmartDevice-Basisprojekt eingefügt.

Unter dem .Net Compact Framework sind für das Basisprojekt insbesondere folgende Einschränkungen bemerkenswert:

- keine Druckerausgabe wegen fehlender System.Drawing.Printing-Klasse
- für die Statusleiste kann kein Panel erzeugt werden
- keine animierte Bitmaps wegen fehlender System.Drawing.ImageAnimator-Klasse
- kein teilweise sichtbares Programmfenster und kein transparenter Hintergrund für die Form
- keine Grafikqualität und kein Pixeloffset-Modus für das Graphics-Object
- keine automatische Anzeige des Kontextmenüs:
  das Kontextmenü wird nach betätigter Enter-Taste in SmartDevice_KeyDown explizit angezeigt.
- kein Aufruf der Windows-Hilfe zur Anzeige der HTML-Hilfedateien:
  die HTML-Hilfedateien werden durch einen direkten Aufruf des Internet-Explorers mit dem anzuzeigenden Dateinamen angezeigt.

- keine automatischen Bildlaufleisten bei größeren Bildern:
  mit den Navigationstasten der Smart Device wird in SmartDevice_KeyDown ein Bildausschnitt ausgewählt, der in PanelOnPaint in der Größe der Client Area des Panels ausgegeben wird.
- kein Tooltip und keine erweiterte Hilfe innerhalb der Dialogformen:
  die Hilfe für die Dialogformen wird durch einen Hilfebutton realisiert über den die HTML-Hilfedateien durch einen direkten Aufruf des Internet-Explorers angezeigt werden.

## 15.2 Die Beispielanwendung für das .Net Compact Framework

Das Projekt der Anwendung1 für SmartDevices erhält die gleiche Ordnerstruktur wie das Projekt für die Anwendung1. Dem Projekt werden alle Verweise von dem SmartDevice-Basisprojekt kopiert und von der Anwendung1 über Hinzufügen \ vorhandenes Element \ Hinzufügen als Link zugefügt.

Für die Dialogformen der Anwendung1 wird eine neue Quellcodedatei verwendet. Hierzu wird die Designerdatei für die Dialogform von dem Anwendung1-Projekt für das .Net-Framework in das SmartDevice-Projekt hinzugefügt und umbenannt zu Anw1_Dialog_CF.cs.

Für die Beispielanwendung kommen unter dem .Net Compact Framework zu den Einschränkungen für das Basisprojekt noch folgende hinzu:

- Keine erweiterten GDI+ Brushes und kein HatchBrush:
  in der Anwendung1 kann der Hintergrund nur mehrfarbig mit SolidBrushes gezeichnet werden. Anstelle der Raummuster wird die Liniengrafik aus WIN32 übernommen. Ein TextureBrush unter Verwendung einer Bitmap ist auch möglich, wird aber nicht verwendet.
- Kein durchsichtiger Hintergrund wegen fehlender Form.TransparencyKey-Eigenschaft.

# 16 Das CryptoKeyGenerator-Projekt für den Aktivierungsschlüssel der Anwendungen

Das CryptoKeyGenerator-Projekt erstellt eine Form als „FixedToolWindow“ mit einer unveränderlichen Größe.

Das Programm erzeugt den für die Anwendungen erforderlichen kryptographischen Aktivierungsschlüssel in einer Binärdatei.

## 16.1 Die CryptoKeyGenerator-Klasse

Der kryptographische Schlüsselgenerator erzeugt aus einem Registrierungsschlüssel mit einem Passwort in der Datei "jk-ware_Lizenzdatei.bin" einen Aktivierungsschlüssel. Zur Kontrolle wird der regenerierte Aktivierungsschlüssel als Text in ein Label der Form ausgegeben.

Kodierung 16-1: die CryptoKeyGenerator-Klasse

```
#region CryptoKeyGenerator-Klasse
///*************************************************************************
/// Version 1.1 vom 11.10.09
///*************************************************************************
public class CryptoKeyGenerator : Form
{

    #region Instanzvariable, Main() und Konstruktor #endregion

    #region InitializeComponent() und Ereignis-Handler für die Kontrollen #endregion

    #region Enkrypten und Dekrypten des Aktivierungsschlüssels #endregion

}
#endregion
```

### 16.1.1 Die Instanzvariablen

Auf alle implizit privaten Instanzvariablen kann nur innerhalb der CryptoKeyGenerator-Klasse zugegriffen werden.

Kodierung 16-2: Die Instanzvariablen

### 16.1.2 Die Startmethode Main()

Main als Haupteinstiegspunkt für die CryptoKeyGenerator-Klasse.

Kodierung 16-3: Die Startmethode Main()

### 16.1.3 Der Konstruktor

Der Konstruktor ruft die für die Designerunterstützung der Form erforderliche Methode InitializeComponent auf.

Kodierung 16-4: Der Konstruktor

### 16.1.4 Die Designermethode InitializeComponent

InitializeComponent ist eine erforderliche Methode für die Designerunterstützung der Form.

Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden. Der Aufruf erfolgt im Konstruktor der CryptoKeyGenerator-Klasse.

Kodierung 16-5: InitializeComponent für die CryptoKeyGenerator-Klasse

## 16.2 Ereignis-Handler für die Formkontrollen des Aktivierungsschlüsselgenerators

### 16.2.1 OnClick_RegkeyEinfuegenButton

OnClick_RegkeyEinfuegenButton kopiert den Text aus der Zwischenablage in die Textbox für den Registierungsschlüssel.

Der Aufruf erfolgt als Handler für das Click-Ereignis des Einfügen-Buttons für den Registierungsschlüssel.

Kodierung 16-6: OnClick_RegkeyEinfuegenButton des Aktivierungsschlüsselgenerators

### 16.2.2 OnClick_AnwPwEinfuegenButton

OnClick_AnwPwEinfuegenButton kopiert den Text aus der Zwischenablage in die Textbox für das Anwendungspasswort.

Der Aufruf erfolgt als Handler für das Click-Ereignis des Einfügen-Buttons für das Anwendungspasswort.

Kodierung 16-7: OnClick_AnwPwEinfuegenButton des Aktivierungsschlüsselgenerators

### 16.2.3 OnClick_ErzeugenButton

In OnClick_ErzeugenButton wird Erzeuge_Aktivierungsschluessel aufgerufen, worin der Aktivierungsschlüssel erzeugt und in einer Datei abgespeichert wird.

Danach wird der in Dekrypt_Aktivierungsschluessel regenerierte Registrierungsschlüssel zum Label der Form ausgegeben.

OnClick_ErzeugenButton wird als Handler für das Click-Ereignis des Erzeugen-Button aufgerufen.

Kodierung 16-8: OnClick_ErzeugenButton des Aktivierungsschlüsselgenerators

## 16.3 Enkrypten und Dekrypten des Aktivierungsschlüssels

### 16.3.1 Erzeuge_Aktivierungsschluessel

Erzeuge_Aktivierungsschluessel erzeugt den Aktivierungsschlüssel und schreibt ihn in die Datei unter Dateiname.

Der Aufruf erfolgt in OnClick_ErzeugenButton.

Kodierung 16-9: Erzeuge_Aktivierungsschluessel

## 16.3.2 Dekrypt_Aktivierungsschluessel

Regeneriert den Aktivierungsschlüssel aus den Dateidaten unter Dateiname. Aufruf zur Kontrolle des Aktivierungsschlüssel in OnClick_ErzeugenButton().

Kodierung 16-10: Dekrypt_Aktivierungsschluessel

# 17 Der erweiterte Lizenzschutz im LicenseProtectorSample-Projekt

Zur Absicherung der Anwendungen mit dem Lizenzschutz der Mirage GmbH, wird das von dem Unternehmen zur Verfügung gestellte Beispielprojekt übernommen.

Zum korrekten Ablauftest muss allerdings eine neue Lizenzdatei angefordert werden oder besser noch eine neue Version des Licence Protector von www.mirage-systems.de heruntergeladen und installiert werden.

Für einen Start des Beispielprogramms mit einer alten Lizenzdatei wurde innerhalb der Anweisungsfolge in Main() true zu false verändert:

```
// Run the application if the licence check passed (failed)
if(ProtectMe()==false)  //true, mit einer neuen Lizenzdatei csharp.lic
{
    Application.Run(new FormMain());
}
else
{
    Application.Exit();
}
```

Die Integration des License Protector in die Projekte wird im Kapitel 11 „Der Mirage-Lizenzschutz“ erläutert.

# 18 Übersichtsdiagramm, Kommentarwebseiten und Struktogramme in Dateien

- Übersichtsdiagramm

BASISPROJEKT:

- Struktogramme
- Strukturblöcke

ANWENDUNG1:

- Struktogramme

# 19 Kodierungen

# 20 Hinweise