Hochschule Harz	FB Automatisierung und Informatik
3. Labor: Dialog	Grafische Nutzerschnittstellen mit .net Thema: Erstellen eines dialogbasierten Programms

Versuchsziele

Konzepte der .net-Programmierung am Beispiel eines Dialogfenster anwenden können.

Kenntnisse in der Implementierung von GUI-Elementen und Befehlen zum Ein- und Auslesen von textbasierten und binären Dateien.

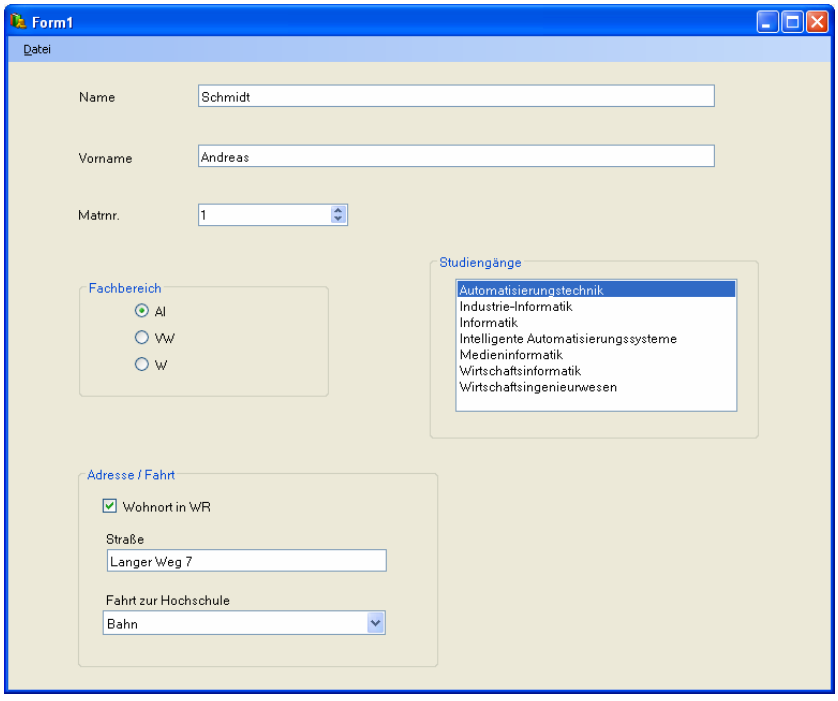
Grundlagen

Die .net-Architektur vereinfacht die GUI-Programmieren durch ein komplexes Framework. Es ist vollständig objektorientiert entwickelt wurden. Die GUI-Elemente können à la Java direkt adressiert werden.

Überblick

Erstellen Sie ein Dialogprogramm in der MFC zum Verwalten von „Studentendaten“. Im ersten Teil werden die Dialogelemente in das Fenster eingetragen und die Steuerung untereinander programmiert. Der zweite Teil liest und schreibt die Daten des Fensters in zwei Dateien. Dazu werden im Anhang die Funktionen vorgestellt.

Folgendes Fenster soll entwickelt werden:



Projekterstellung

Aufgaben

1. Teilaufgaben

Folgende Aufgaben / Funktionalitäten realisiert werden:

- Erstellen des Dialogfensters (Details siehe unten)
- Namen der GUI-Elemente

  EName
  EVorname
  numericUpDownMatrnr
  rbAI
  rbVW
  rbW
  lstudiengaenge
  chkWR
  EStrasse
  cmbFahrt Ändern des DropDownStyle

  <u>Hinweis:</u>
  Es dürfen keine negativen Matrikelnummern vergeben werden.
  Attribut:
  - Minimum
  - Maximum

- Inhalte der Liste und ComboBox:

  <u>Inhalte der Studiengänge:</u>
  AI: Automatisierungstechnik
  Industrie-Informatik
  Informatik
  Intelligente Automatisierungssysteme
  Medieninformatik
  Wirtschaftsinformatik
  Wirtschaftsingenieurwesen

  VW: Öffentliche Verwaltung
  Verwaltungsökonomie
  Europäisches Verwaltungsmanagement
  Verwaltungsmanagement / eGovernment

  W: Betriebswirtschaftslehre
  BWL / Dienstleistungsmanagement
  Wirtschaftspsychologie
  Tourismusmanagement

  <u>Inhalte der Fahrtmöglichkeiten:</u>
  Bahn
  Bus
  Fahrrad
  Motorrad
  PKW

Zu Fuß

Hinweise:
Die Inhalte der Liste müssen direkt als Quellcode eingeben werden (siehe Skript).

Fügen Sie drei Methoden in die Klasse, um die Liste zusetzen:
setStudiengaenge_AI()
setStudiengaenge_VW()
setStudiengaenge_W()

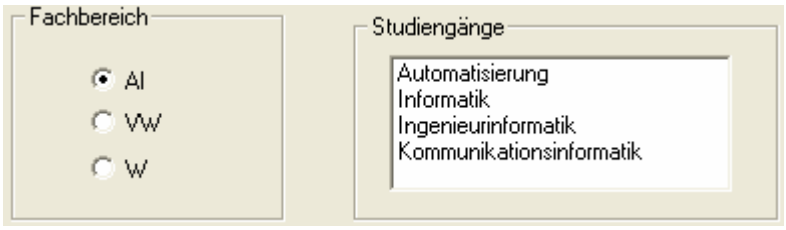
Die Methode „setStudiengaenge_AI()“ wird in der Methode onInitDialog aufgerufen.

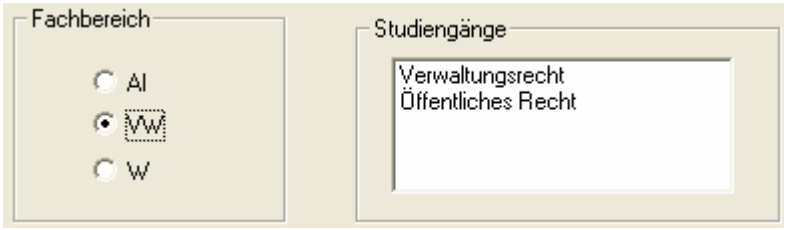
- Setzen des Radiobuttons „Fachbereichs AI“ beim Start des Programms
- Erstellen der Click-Methode für die Fachbereiche (Doppelklick auf die richtigen RadioButtons)

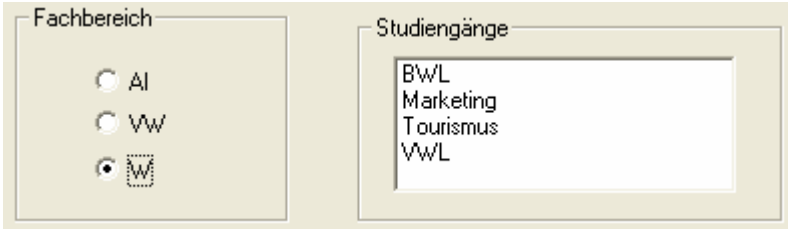
  Hinweis:

  Die Events müssen pro Radiobutton definiert werden!

  Dialogbeispiele:







- Der Adressteil soll nur verfügbar sein, wenn der Student in Wernigerode wohnt. Dazu müssen zum einen im Formular beide GUI-Elemente deaktiviert werden. Zum anderen müssen im Quelltext die Elemente aus- bzw. eingeschaltet werden (onClick-Event der Checkbox). Das Aus- bzw. Anschalten von Elementen verwendet die Methode enabled. Die Musterlösung ist auf meiner Homepage verfügbar.

2. Teilaufgabe, Einbau der Menüs

Einbau folgender Menüeinträge mit Funktionalität:

Menü Datei

- Hauptmenü Datei Name: MainFile
- Öffnen Name: MnOpen
- Speichern unter Name: MnSaveAs
- Schließen Name: MnClose

In den einzelnen Open/Save-Event-Methoden sollen die Dateien ausgewählt werden können. Dabei müssen aber zwei Dateierweiterungen möglich sein (txt und bin). Je nach Erweiterung muss dann einer der folgenden Methoden aufgerufen werden:

- load1 Laden Ascii-Datei
- save1 Speichern Ascii-Datei
- load2 Laden Binär-Datei
- save2 Speichern Binär-Datei

3. Teilaufgabe /Speichern und Laden in einer ASCII-Datei (Stream)

Die Daten sollen mittels der Klasse StreamReader bzw. StreamWriter geladen und gespeichert werden:

- Verwendet werden die Stream-Methoden
- Das Format wird von Ihnen vollständig selbständig definiert

4. Teilaufgabe /Speichern und Laden in einer Binär-Datei (Stream)

Die Daten sollen geladen und gespeichert werden:

- Verwendet werden die Stream-Methoden
- Erstellen Sie eine Methode für den Schalter „Speichern Stream Binär“. Überprüfen Sie am Anfang der Methode, ob alle Elemente Werte haben.
- In dieser Aufgabe brauchen Sie nur den **Nachnamen**, die **Matrikelnummer** und den Wert für den **Wohnort** speichern.
- Speichern Sie als erstes die Daten mittels der Stream-Methode write.
- Erstellen Sie eine Methode für den Schalter „Laden Stream Binär“.
- Laden Sie die Daten, Methode (read), und stellen Sie den ursprünglichen Zustand wieder her.
- Das Format wird von Ihnen vollständig selbständig definiert. Beachten Sie dabei, dass Sie die String wieder einlesen müssen.

BEISPIEL-CODE

Einfügen der Elemente einer ListBox

```
liste.Items.Clear();
liste.Items.add("eintrag1");
liste.Items.add("eintrag2");
liste.SelectedIndex = 3;
```

zusätzliche Methoden:

Anzahl: m_liste_ctrl.GetCount();

Löschen: m_liste_ctrl.DeleteString(i);

Gesamtlöschen: m_liste_ctrl.ResetContent();

Aktuellen Index: GetCurSel

Index setzen: SetCurSel(i)

Index setzen: SetSel(i);

Methoden von string

Trim() Löschen der Leerzeichen am Anfng und am Ende

OpenDialog

```
OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter = "java files (*.java)|*.java|All files (*.*)|*.*";
openFileDialog1.FilterIndex = 1;// zählt von 1 !
openFileDialog1.DefaultExt = ".java";
openFileDialog1.InitialDirectory = "c:\\daten";
openFileDialog1.Multiselect = false;
openFileDialog1.RestoreDirectory = true;
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
        string sFilename = openFileDialog1.FileName.ToString();
}
```

SaveDialog

```
SaveFileDialog saveFileDialog1 = new SaveFileDialog();
saveFileDialog1.Filter  = "java files (*.java)|*.java|All files (*.*)|*.*";
saveFileDialog1.FilterIndex = 1;// zählt von 1 !
```

```
saveFileDialog1.DefaultExt = ".txt";
saveFileDialog1.InitialDirectory = "c:\\daten";
saveFileDialog1.RestoreDirectory = true;
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
  string sFilename = saveFileDialog1.FileName.ToString();
}
```

Stream-Methoden

Klasse FileStream

Die Klasse FileStream dient als Basisklasse für alle weiteren Klassen (ähnlich FileInputStream).
Wichtig:
Sie ist die Basisklasse zum Lesen UND zum Schreiben !

Konstruktor:
FileStream aFile = new FileStream(sFilename, FileMode.Create);

Konstanten:

- FileMode.Create
- FileMode.CreateNew
- FileMode.Open
- FileMode.Append
- FileMode.OpenOrCreate
- FileMode.Truncate

Klasse StreamReader

Die Klasse StreamReader dient zum Einlesen von ASCII-Dateien.

Konstruktor:
StreamReader inFile = new StreamReader (aFile);

Öffentliche Methoden

ReadLine	Lesen einer Zeile
ReadToEnd	Lesen eines Strings biz zum Ende der Datei
Read	Lesen eines einzelnen Zeichens
Close	Schließen der Datei

Klasse StreamWrite

Die Klasse StreamWrite dient zum Schreiben von ASCII-Dateien.

Konstruktor:
StreamWriter inFile = new StreamWriter (aFile);

Öffentliche Methoden: Alle Werte werden ins Ascii-Format transformiert

WriteLine(String)	Lesen einer Zeile
WriteLine(Char)	Schreiben eines einzelnen Zeichens
WriteLine(Boolean)	Schreiben einesBool#schen Wertes
WriteLine(Decimal)	Schreiben eines Decimals
WriteLine(Double)	Schreiben einesDouble-Wertes
WriteLine(Int32)	Schreiben eines Integer-Wertes
WriteLine(Int64)	Schreiben eines Integer-Wertes
WriteLine(Object)	Schreiben eines Objected
WriteLine(Single)	Schreiben eines Single-Wertes
WriteLine(UInt32)	Schreiben eines Integer-Wertes
WriteLine((UInt64)	Schreiben eines Integer-Wertes
Close	Schließen der Datei

Methoden zum Konvertieren (String nach Integer bzw. Double)

Convert.ToInt32("1234");
Convert.ToDouble("1234");
Convert.ToInt16("1234");

Klasse BinaryWriter

Die Klasse BinaryWriter dient zum Schreiben von Binär-Dateien.

Konstruktor:
BinaryWriter outFile = new BinaryWriter(aFile);

Öffentliche Methoden

Name	**Description**
Write(Boolean)	Writes a one-byte Boolean value to the current stream, with 0 representing false and 1 representing true.
Write(Byte)	Writes an unsigned byte to the current stream and advances the stream position by one byte.
Write(array<Byte>[]()[])	Writes a byte array to the underlying stream.
Write(Char)	Writes a Unicode character to the current stream and advances the current position of the stream in accordance with the Encoding used and the specific characters being written to the stream.
Write(array<Char>[]()[])	Writes a character array to the current stream and advances the current position of the stream in accordance with the Encoding used and the specific characters being written to the stream.
Write(Decimal)	Writes a decimal value to the current stream and advances the stream position by sixteen bytes.
Write(Double)	Writes an eight-byte floating-point value to the current stream and advances the stream position by eight bytes.
Write(Int16)	Writes a two-byte signed integer to the current stream and advances the stream position by two bytes.
Write(Int32)	Writes a four-byte signed integer to the current stream and advances

	the stream position by four bytes.
Write(Int64)	Writes an eight-byte signed integer to the current stream and advances the stream position by eight bytes.
Write(SByte)	Writes a signed byte to the current stream and advances the stream position by one byte.
Write(Single)	Writes a four-byte floating-point value to the current stream and advances the stream position by four bytes.
Write(String)	Writes a length-prefixed string to this stream in the current encoding of the BinaryWriter, and advances the current position of the stream in accordance with the encoding used and the specific characters being written to the stream.
Write(UInt16)	Writes a two-byte unsigned integer to the current stream and advances the stream position by two bytes.
Write(UInt32)	Writes a four-byte unsigned integer to the current stream and advances the stream position by four bytes.
Write(UInt64)	Writes an eight-byte unsigned integer to the current stream and advances the stream position by eight bytes.
Write(array<Byte>[]()[], Int32, Int32)	Writes a region of a byte array to the current stream.
Write(array<Char>[]()[], Int32, Int32)	Writes a section of a character array to the current stream, and advances the current position of the stream in accordance with the Encoding used and perhaps the specific characters being written to the stream.
Close	Closes the current BinaryWriter and the underlying stream.

Klasse BinaryReader

Die Klasse BinaryReader dient zum Einlesen von Binär-Dateien.

Konstruktor:
BinaryReader inFile = new BinaryReader (aFile);

Öffentliche Methoden

Name	**Description**
ReadBoolean	Reads a Boolean value from the current stream and advances the current position of the stream by one byte.
ReadByte	Reads the next byte from the current stream and advances the current position of the stream by one byte.
ReadBytes	Reads count bytes from the current stream into a byte array and advances the current position by count bytes.
ReadChar	Reads the next character from the current stream and advances the current position of the stream in accordance with the Encoding used and the specific character being read from the stream.
ReadChars	Reads count characters from the current stream, returns the data in a character array, and advances the current position in accordance with the Encoding used and the specific character being read from the stream.
ReadDecimal	Reads a decimal value from the current stream and advances the current position of the stream by sixteen bytes.
ReadDouble	Reads an 8-byte floating point value from the current stream and advances the current position of the stream by eight bytes.
ReadInt16	Reads a 2-byte signed integer from the current stream and advances the current position of the stream by two bytes.

ReadInt32	Reads a 4-byte signed integer from the current stream and advances the current position of the stream by four bytes.
ReadInt64	Reads an 8-byte signed integer from the current stream and advances the current position of the stream by eight bytes.
ReadSByte	Reads a signed byte from this stream and advances the current position of the stream by one byte.
ReadSingle	Reads a 4-byte floating point value from the current stream and advances the current position of the stream by four bytes.
ReadString	Reads a string from the current stream. The string is prefixed with the length, encoded as an integer seven bits at a time.
ReadUInt16	Reads a 2-byte unsigned integer from the current stream using little-endian encoding and advances the position of the stream by two bytes.
ReadUInt32	Reads a 4-byte unsigned integer from the current stream and advances the position of the stream by four bytes.
ReadUInt64	Reads an 8-byte unsigned integer from the current stream and advances the position of the stream by eight bytes.
Close	Closes the current reader and the underlying stream.