Prüfung 2 Klasse 00-I

Name	

Prüfungsdauer: 90 Minuten

Bitte deutlich schreiben

Lösungen können direkt auf die Aufgabenblätter oder auf die Rückseite der Aufgabenblätter geschrieben werden

PC's sind nicht erlaubt

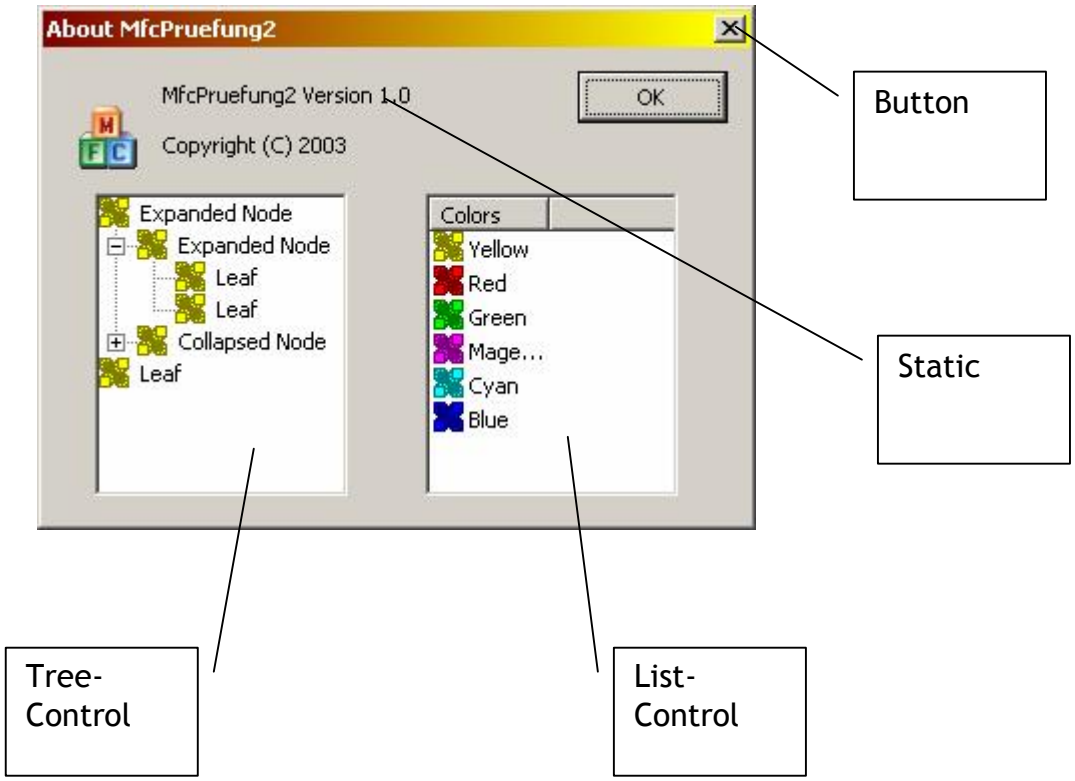
Beliebige Unterlagen und Bücher sind erlaubt

Achte auf Details wie Punkte, Kommas und **Semikolons**

Aufgabe	Punkte	
Allgemeine Fragen	10	
Observer Pattern	10	
Tree Control	10	
List Control	10	
Schlussfragen	10	
Total	50	

1. Allgemeine Fragen

a) Auf folgendem Dialog befinden sich einige Controls. Gib in den Feldern an, wie das jeweilige Control heisst (z.B. List-Control). (4 Punkte)



b) Wie heisst die Klasse, die der Assistent erstellt, um das Rahmenfenster zu abstrahieren? Diese Klasse enthält üblicherweise die CView Objekte und falls vorhanden den Status Bar und den Tool Bar. (1 Punkt)

CFrameView

c) Beim Erstellen eines MFC- Projektes mit Document-View Unterstützung wird eine Klasse erzeugt, die von CDocument abgeleitet ist. Dabei wird immer eine virtuelle Methode überschrieben, die aufgerufen wird wenn ein neues Dokument geladen wird. Wie heisst diese Methode? (1 Punkt)

OnNewDocument

d) Betrachte folgenden Ausschnitt aus der Dokumenten-Klasse. Sie enthält zwei Datenelemente. Schreibe die passende *Serialize* Methode. (4 Punkte)

```
class CMfcPruefung2Doc : public CDocument
{
protected: // create from serialization only
      CMfcPruefung2Doc();
      DECLARE_DYNCREATE(CMfcPruefung2Doc)

// Overrides
      public:
      virtual BOOL OnNewDocument();
      virtual void Serialize(CArchive& ar);

private:
    CString _text;
    int     _zahl;

// REST ist nicht dargestellt

// HIER BITTE METHODE SERIALIZE schreiben (auch Argumente
// nicht vergessen) also was in der .cpp Datei stehen würde

void CMfcPruefung2Doc::Serialize(CArchive& ar)
{
    if(ar.IsStoring())
    {
        ar << _text;
        ar << _zahl;
    }
    else
    {
        ar >> _text;
        ar >> _zahl;
    }
}
```

2. Observer Pattern

a) Folgende Klasse „Daten“ enthält einige Daten, für die es Set- und Get-Methoden gibt. Diese Daten werden nun von verschiedenen anderen Klassen verwendet oder dargestellt. Wende nun das Observer-Pattern an.

1. Definiere eine Klasse für die Observer, nenne sie z.B. „DatenObserver“ mit Methoden, die aufgerufen werden, wenn sich die Daten ändern. Es ist Dir überlassen, ob es eine oder mehrere Methoden sind. (3 Punkte)

2. Ergänze bei der Deklaration (Header-Datei) der Klasse „Daten“ die Methoden, die es braucht damit sich Observer (DatenObserver) Ein- oder Austragen können und die Kollektion (Container) dafür. Wähle eine geeignete Klasse aus der STL und vergiss nicht das #include. (2 Punkte)

3. Für die Klasse „Daten“ musst du die beiden Methoden für das Ein- und Austragen implementieren. (2 Punkte)

4. In den beiden Set-Methoden musst du noch den Code ergänzen um die „DatenObserver“ über Änderungen zu benachrichtigen. (3 Punkte)

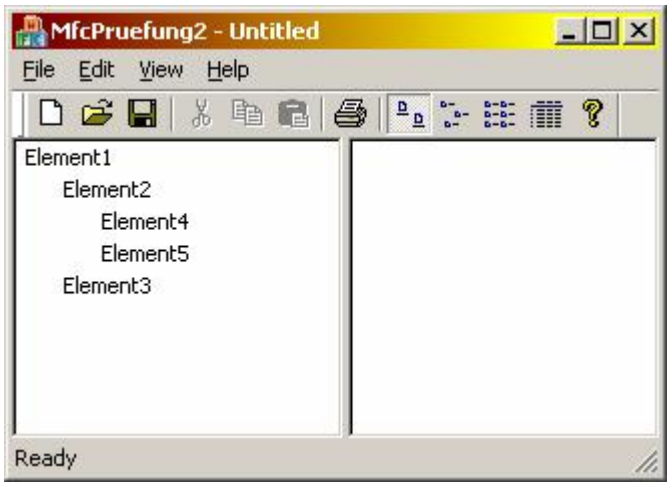
Total 10 Punkte

```
#ifndef DATEN_H
#define DATEN_H

#include <string>
// 2.
// Hier #include für den STL-Container ergänzen
#include <set>


// 1.
// Definiere hier die Klasse DatenObserver

class DatenObserver
{
    public:
        virtual void dataChanged() = 0;
};









// 2.
// Definiere hier einen Datentypen für den Container

typedef std::set<DatenObserver*> DatenObservers;


class Daten
{
public:
    Daten();

    void setText(const std::string& text);
    const std::string& getText() const;
    void setZahl(int zahl);
    int getZahl() const;

// 2.
// Hier die Methode für das Ein- und Austragen der Observer
    void addObserver(DatenObserver* observer);
    void removeObserver(DatenObserver* observer);



private:
    std::string _text;
    int         _zahl;
// 2.
// Hier ein Datenelement für die ObserverKollektion definieren

    DatenObservers _observers;


};
#endif
```

```
#include "Daten.h"

// 3.
// Hier Methoden zum Ein- und Austragen definieren


void Daten::addObserver(DatenObserver* observer)
{
    _observers.insert(observer);
}

void Daten::removeObserver(DatenObserver* observer)
{
    _observers.erase(observer);
}




void Daten::setText(const std::string& text)
{
    _text = text;
// 4.
// Hier alle DatenObserver benachrichtigen

    DatenObservers::iterator it = _observers.begin();
    DatenObservers::iterator end = _observers.end();

    for( ; it != end; ++it)
    {
        (*it)->dataChanged();
    }






}

void Daten::setZahl(int zahl)
{
    _zahl = zahl;
// 4.
// Hier alle DatenObserver benachrichtigen

    DatenObservers::iterator it = _observers.begin();
    DatenObservers::iterator end = _observers.end();

    for( ; it != end; ++it)
    {
        (*it)->dataChanged();
    }



}
```

3. TreeControl

a) Wir haben eine Klasse „CLeftView“, die von CTreeView abgeleitet ist. Ergänze die Methode OnInitialUpdate so, dass sich folgende Struktur ergibt.



Als erstes musst du auf das Tree Control zugreifen. (1 Punkt)
Danach ist es nötig alle Elemente aus dem Tree Control zu löschen. (1 Punkt)
Füge nun die Elemente ein und achte darauf welches jeweils das Parent Item von einem anderen ist. (5 Punkte)
Total 7 Zeilen Code. Total 7 Punkte.

```
void CLeftView::OnInitialUpdate()
{
	CTreeView::OnInitialUpdate();

	CTreeCtrl& treeCtrl = GetTreeCtrl();

	treeCtrl.DeleteAllItems();

	HTREEITEM el1 = treeCtrl.InsertItem(“Element1”);
	HTREEITEM el2 = treeCtrl.InsertItem(“Element2”, el1);
	treeCtrl.InsertItem(“Element3”, el1);
	treeCtrl.InsertItem(“Element4”, el2);
	treeCtrl.InsertItem(“Element5”, el2);

}
```

b) Wie heisst die Methode, mit der man herausfinden kann welches Element in einem Tree Control an einer bestimmten Stelle (Koordinaten) liegt? (1 Punkt)

HitTest

c) Welcher Stil muss bei einem Tree Control mit SetWindowLong gesetzt werden, damit es die Plus und Minus Icons gibt? Wie heisst diese Konstante? (1 Punkt)

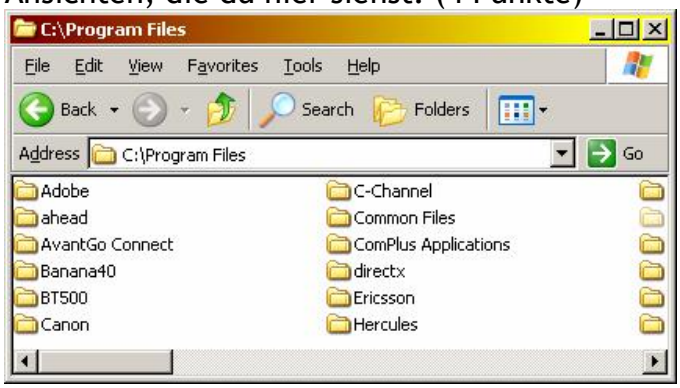
TVS_HASBUTTONS

d) Wie heisst das oberste (oder unterste, je nach Betrachtungsweise) Element in einem Tree? (1 Punkt)

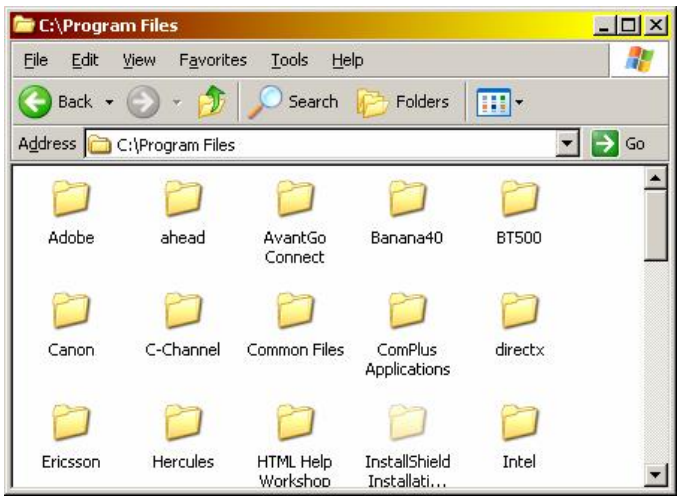
Root

4. List Control

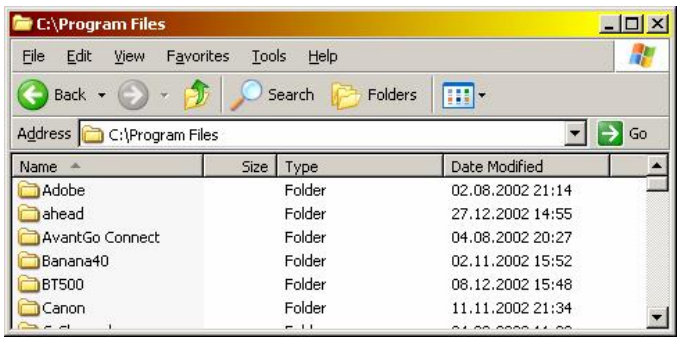
a) Das List Control bietet verschiedene Ansichten. Benenne die vier Ansichten, die du hier siehst. (4 Punkte)



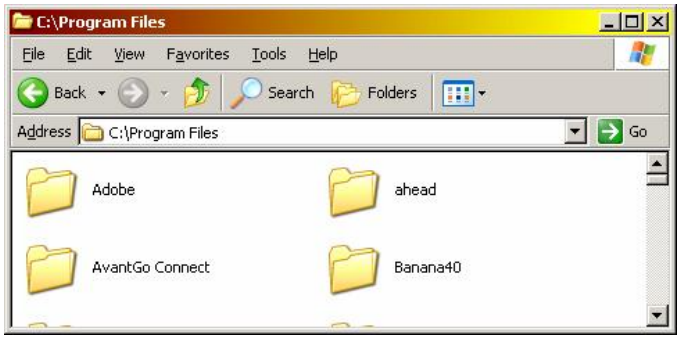
Liste



Kleine Symbole



Details



Grosse Symbole

b) Füge in folgender Methode CMfcPruefung2View::OnInitialUpdate() eine Kolonne mit der Überschrift „Name“ in das List Control ein. CMfcPruefung2View ist von CListView abgeleitet.
Als erstes gilt es auf das List Control zuzugreifen. (1 Punkt)
Danach kannst du die Kolonne (linksbündig) einfügen. (2 Punkte)
Du musst nicht vorher alle Kolonnen löschen!
(Total 3 Punkte)

```
void CMfcPruefung2View::OnInitialUpdate()
{
	CListView::OnInitialUpdate();

	CListrCtrl& listCtrl = GetListCtrl();

	listCtrl.InsertColumn(0, „Name“, LVCFMT_LEFT);


}
```

c) Wie heisst die Methode von CListCtrl um ein Element einzufügen? Versuche zu raten, falls du die Information nicht findest. (1 Punkt)

InsertItem

d) Wie heisst die Methode von CListCtrl um alle Elemente zu löschen? Auch das kannst du erraten. (1 Punkt)

DeleteItem

e) Die Klasse CListView ist von CView abgeleitet. Von welcher Klasse ist CListCtrl abgeleitet? Vorsicht es gibt keine Klasse CCtrl!
Hinweis: Auch CView ist davon abgeleitet. (1 Punkt)

CWnd

5. Schlussfragen

a) Welches häufig verwendete Windows-Programm hat als Benutzer-Interface eine geteilte Ansicht mit einem Tree Control auf der linken Seite und einem List Control auf der rechten, wie unser Notenverwaltungsprogramm? (1 Punkt)

Explorer

b) Wie nennt man es, wenn eine Klasse von mehreren anderen Klassen ableitet? (1 Punkt)

- O Vielfachvererbung
- O Multiableitung
- O **Mehrfachvererbung ß**
- O Zwillingsvererbung

c) Mach in der folgenden Klasse Fred die Methode doIt const falls es geht, indem du einfach den Code ergänzt. (1 Punkt)

```
class Fred
{
public:
    Fred() : _value(0)
    {
    }

    int doIt(int count) const
    {
        int result = _value * count;
        return result;
    }

private:
    int _value;
};
const ist ok, denn Fred ändert sich nicht
```

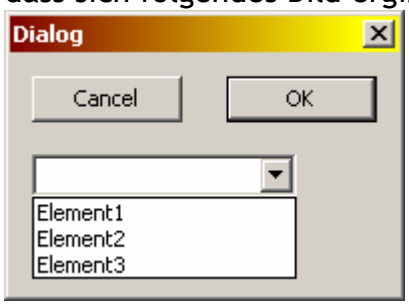
d) Wie heisst die Methode von CWnd, mit der man absolute Bildschirmkoordinaten in Client-Koordinaten umrechnen kann? (1 Punkt)

ScreenToClient

e) Schreibe für folgende Klasse CMouse die Methode getMousePosition. Hinweis: Die Klasse CPoint leitet von der POINT Struktur ab, kann also verwendet werden, als wäre eine solche Struktur. (ca. 3-5 Zeilen) (2 Punkte)

```
class CMouse
{
public:
    CPoint getMousePosition() const;
};


// Hier die Methode definieren

CPoint CMouse::getMousePosition() const
{
    CPoint result;
    GetCursorPos(&result);
    return result;
}
```

f) Gegeben sei ein CComboBox Objekt „combobox“. Fülle sie so auf, dass sich folgendes Bild ergibt (Reihenfolge der Einträge beachten):



Rufe einfach Methoden auf das Objekt combobox auf um Elemente einzutragen, du musst nichts includieren oder sonst was schreiben. (3 Zeilen) (2 Punkte)

```
combobox.InsertString(-1, „Element1“);
combobox.InsertString(-1, „Element2“);
combobox.InsertString(-1, „Element3“);
```

g) Welche Klasse der MFC abstrahiert einen string? (1 Punkt)

CString

h) Die Schlussfrage, aber umformuliert:
Wie viele Fehler soll ein Programm von Dir enthalten? (1 Punkt)
KEINEN