Stephan Brumme, SST, 3.FS, Matrikelnr. 70 25 44
mail@stephan-brumme.com

Aufgabe O1.2

Diese Aufgabe ist lediglich eine Erweiterung von O1.1, so dass ich *CRoom* unverändert übernehmen konnte, ebenso brauchten fast alle Methoden nur sehr wenig verändert werden.

Grundsätzlich versuche ich, so viel wie möglich Arbeit durch *CRoom* erledigen zu lassen, um auch für *CCashOffice* eine Stabilität gewährleisten zu können, falls sich die interne Struktur von *CRoom* bei gleicher Schnittstelle verändern sollte.
Ebenso habe ich in *CRoom* bereits polymorphe Methoden benutzt. In der folgenden Tabelle sind sie grau hinterlegt, Pfeile weisen darauf hin, dass die Methode unverändert vererbt wird:

CRoom		CCashOffice
CRoom		CCashOffice
~CRoom		~CCashOffice
=		=
Copy		Copy
==		==
EqualValue		EqualValue
GetNumberOfRoom	⇒	
SetNumberOfRoom	⇒	
		GetNumberOfCounter
		SetNumberOfCounter
GetArea	⇒	
Show		Show

Die Destruktoren sind jeweils virtuell deklariert, um sicherzustellen, dass sie jeweils korrekt aufgerufen werden. Im Beispiel sind zwar Destruktoren überhaupt nicht notwendig, ich halte es aber für stilistisch besser, jede Klasse mit einem Destruktor zu versehen.

Stellvertretend für die anderen Methoden gebe ich hier die Deklaration und die Implementierung des Vergleichsoperators an:
Deklaration:

```
virtual Boolean operator==(const CCashOffice& cashOffice) const;
```

Implementation:

```
// compare the CashOffice with another one
Boolean CCashOffice::operator==(const CCashOffice& cashOffice) const
{
	// use the CRoom compare function
	return (CRoom::operator==(cashOffice) &&
		m_nNumberOfCounter == cashOffice.m_nNumberOfCounter);
}
```

CCashOffice.h:

```
////////////////////////////////////////////////////////
// Softwarebauelemente I, Aufgabe O1.2
//
// author:            Stephan Brumme
// last changes:      January 14, 2001

#if !defined(AFX_CASHOFFICE_H__26E07000_EA3B_11D4_9BB7_AFEE07846A21__INCLUDED_)
#define AFX_CASHOFFICE_H__26E07000_EA3B_11D4_9BB7_AFEE07846A21__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


#include "Room.h"


class CCashOffice : public CRoom
{
```

```
private:
	Ordinal m_nNumberOfCounter;

public:
	// constructor (former Init !)
	CCashOffice(Ordinal nNumberOfRoom, Ordinal nArea, Ordinal nNumberOfCounter) :
		CRoom(nNumberOfRoom, nArea), m_nNumberOfCounter(nNumberOfCounter) {}

	// destructor (not neccessary)
	virtual ~CCashOffice() {}

	// compare two rooms
	virtual Boolean operator==(const CCashOffice& cashOffice) const;
	virtual Boolean EqualValue(const CCashOffice& cashOffice) const;

	// copy one room to another one
	virtual CCashOffice& operator=(const CCashOffice& cashOffice);
	virtual Boolean      Copy     (const CCashOffice& cashOffice);

	// access m_nNumberOfCounter
	Ordinal GetNumberOfCounter() const;
	void    SetNumberOfCounter(const Ordinal nNumberOfCounter);

	// display the attributes
	// only for internal purposes !
	virtual void Show() const;
};

#endif // !defined(AFX_CASHOFFICE_H__26E07000_EA3B_11D4_9BB7_AFEE07846A21__INCLUDED_)
```

CCashOffice.cpp:

```
////////////////////////////////////////////////////////
// Softwarebauelemente I, Aufgabe O1.2
//
// author:            Stephan Brumme
// last changes:      January 14, 2001

#include "CashOffice.h"
#include <iostream>


// compare the CashOffice with another one
Boolean CCashOffice::operator==(const CCashOffice& cashOffice) const
{
	// use the CRoom compare function
	return (CRoom::operator==(cashOffice) &&
		m_nNumberOfCounter == cashOffice.m_nNumberOfCounter);
}


// compare the CashOffice with another one
// routes call down to "==" operator
Boolean CCashOffice::EqualValue(const CCashOffice& cashOffice) const
{
	return (operator==(cashOffice));
}


// copy one CashOffice to another one
// prevents user from copying an object to itself
CCashOffice& CCashOffice::operator=(const CCashOffice& cashOffice)
{
	// cannot copy an object to itself
	if (this != &cashOffice)
	{
		// copy all variables
		// first copy the attributes of CRoom
		CRoom::operator=(cashOffice);
		// and now our new ones of CCashOffice
		m_nNumberOfCounter = cashOffice.m_nNumberOfCounter;
	}
```

```
	return *this;
}


// copy one CashOffice to another one
// prevents user from copying an object to itself
// calls "=" operator internally
Boolean CCashOffice::Copy(const CCashOffice &cashOffice)
{
	// cannot copy an object to itself
	if (this == &cashOffice)
		return false;

	// use "=" operator
	operator=(cashOffice);

	return true;
}


// retrieve the private value of m_nNumberOfCounter
Ordinal CCashOffice::GetNumberOfCounter() const
{
	return m_nNumberOfCounter;
}


// change private m_nNumberOfCounter
void CCashOffice::SetNumberOfCounter(const Ordinal nNumberOfCounter)
{
	m_nNumberOfCounter = nNumberOfCounter;
}


// display the attributes
// only for internal purposes !
void CCashOffice::Show() const
{
	// open namespace
	using namespace std;

	// display the crap using default output

	// the CRoom member variables
	CRoom::Show();
	// and the CCashOffice ones
	cout << "The number of counter is "<<GetNumberOfCounter()<<"."<<endl;
}
```

O1_2.cpp:

```
////////////////////////////////////////////////////////
// Softwarebauelemente I, Aufgabe O1.2
//
// author:            Stephan Brumme
// last changes:      January 14, 2001


#include "CashOffice.h"
#include <iostream>


using namespace std;


void main()
{
	CCashOffice myCashOffice(2,3,7);
	CCashOffice myCashOffice2(4,5,42);

	myCashOffice.Show();
	myCashOffice2.Show();
```

```
        // I wrote some lines twice to verify the overloaded operators

        cout<<"Compare cashoffices ... "<<(myCashOffice==myCashOffice2)<<endl;
//      cout<<"Compare cashoffices ... "<<myCashOffice.EqualValue(myCashOffice2)<<endl;

        cout<<"Now copy myCashOffice to myCashOffice2 ..."<<endl;
        myCashOffice2 = myCashOffice;
//      myCashOffice2.Copy(myCashOffice);

        cout<<"Compare again ... "<<(myCashOffice==myCashOffice2)<<endl;
//      cout<<"Compare again ... "<<myCashOffice.EqualValue(myCashOffice2)<<endl;

        cout<<"Verify the Copy-To-Itself protection
..."<<myCashOffice.Copy(myCashOffice)<<endl;
}
```