

Aufgabe O1.3

Ich verberge die in der Aufgabenstellung genannten Funktionen durch eine private inheritance:

```
class CCashOffice : private CRoom
{ ... }
```

Stellvertretend für die anderen Methoden ist *GetNumberOfCashOffice* implementiert:

```
// retrieve the private value of CRoom::m_nNumberOfCashOffice
Ordinal CCashOffice::GetNumberOfCashOffice() const
{
    return CRoom::GetNumberOfRoom();
}
```

Ich habe mich dabei bemüht, die Methoden der Basisklasse *CRoom* zu benutzen, um nicht Code doppelt zu schreiben, was die Fehleranfälligkeit erhöhen könnte und bei Programmänderungen zu Inkonsistenzen führt.

Insgesamt halten sich die Erweiterungen gegenüber O1.2 sehr in Grenzen, die Änderungen/Erweiterungen sind **blau** markiert.

CCashOffice.h:

```
////////////////////////////////////
// Softwarebauelemente I, Aufgabe 01.3
//
// author:          Stephan Brumme
// last changes:    January 15, 2001

#ifndef 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 : private CRoom
{
private:
    Ordinal m_nNumberOfCounter;

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

    // destructor (not necessary)
    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);

    // access m_nNumberOfRoom of CRoom
    Ordinal GetNumberOfCashOffice() const;
    void SetNumberOfCashOffice(const Ordinal nNumberOfCashOffice);
}
```

```

    // deliver m_nArea of CRoom
    Ordinal GetAreaOfCashOffice() const;

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

#endif // !defined(AFX_CASHOFFICE_H__26E07000_EA3B_11D4_9BB7_AFE07846A21__INCLUDED_)

```

CCashOffice.cpp:

```

////////////////////////////////////
// Softwarebauelemente I, Aufgabe 01.3
//
// author:          Stephan Brumme
// last changes:    January 15, 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;
}

// retrieve the private value of CRoom::m_nNumberOfCashOffice
Ordinal CCashOffice::GetNumberOfCashOffice() const
{
    return CRoom::GetNumberOfRoom();
}

// change private CRoom::m_nNumberOfCashOffice
void CCashOffice::SetNumberOfCashOffice(const Ordinal nNumberOfCashOffice)
{
    CRoom::SetNumberOfRoom(nNumberOfCashOffice);
}

// deliver CRoom::m_nArea
Ordinal CCashOffice::GetAreaOfCashOffice() const
{
    return CRoom::GetArea();
}

// 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;
}

```

01_3.cpp:

```
////////////////////////////////////
// Softwarebauelemente I, Aufgabe 01.3
//
// author:          Stephan Brumme
// last changes:    January 15, 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;

// next line will throw a compilation error due to private inheritance
// cout<<myCashOffice.GetArea()<<endl;

// but the new method works well
cout<<"cashoffice 1 covers "<<myCashOffice.GetAreaOfCashOffice()<<" acres"<<endl;

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