Thema, Ziele: Projekt "CUSEP", Klasse "MyDate", Zusicherungen, TDD

Einführung: das Projekt "CUSEP" – "Se non è vero, è ben trovato."

Herzlich willkommen beim Projekt "CUSEP" – "Confidential Unique Software Engineering Project". CUSEP ist ein hochgeheimes Software-Projekt, welches die aktuellen Unruhen von der Schweiz fernhalten soll.

Obwohl Sie inzwischen vom Schweizerischen Geheimdienst auf ihre Eignung bezüglich Vertrauenswürdigkeit, Verschwiegenheit und Integrität überprüft worden sind (ohne vermutlich etwas davon zu merken), werden Sie in Zukunft jeweils nur stückweise über das Projekt informiert. Dies nach dem Motto, was man nicht kennt, kann man auch nicht verraten.

Da auch andere Personen an diesem Geheimprojekt arbeiten, ist es wichtig, dass Sie sich strikte an alle Anweisungen bezüglich Projektorganisation, Namen von Klassen etc. halten.

Aufgabe 1: Klasse "MyDate" nur mit Stubs

Gegeben ist das folgende .h-File "MyDate.h" der Klasse "MyDate"

```
#ifndef MYDATE_H_
#define MYDATE_H_

class MyDate
{
   private:
      int _day;       // 1..31
      int _month;     // 1..12
      int _year;      // z.B. 2010
      bool _isValid;  // interner Cache fuer Resultat der Gueltigkeitspruefung
   public:
      MyDate();
      // erzeugt neues ungueltiges Datum-Objekt
      MyDate(int day, int month, int year);
      // erzeugt neues initialisiertes Datum-Objekt
      MyDate (const MyDate & x);
      // Copy-Constructor
      MyDate & operator= (const MyDate & x);
      // Zuweisungsoperator
      bool isLeapYear();
      // liefert true falls Schaltjahr
      bool isValid();
      // prueft ob akt. Datum gueltig, d.h. plausibel (nicht 31.2.2009)
      MyDate addDays(int n) const;
      // liefert das Datum, das n Tage nach dem aktuellen Datum liegt.
};
#endif /* MYDATE_H_ */
```

Aufgabe:

Erstellen Sie mit Hilfe von Eclipse ein Projekt "Cusep011" und übernehmen Sie den obigen Teil der Klasse "MyDate".

Erstellen Sie anschliessend auch die Datei ""MyDate.cpp". Implementieren Sie dabei die Memberfunktionen vorläufig nur als sogenannte "Stubs".

Als "Stubs" (Stummel) bezeichnet man Funktionen mit möglichst wenigen Anweisungen, die sich zwar fehlerfrei compilieren lassen, aber in der Regel noch nicht das Richtige tun. Beispiel:

```
bool MyDate::isValid()
{
      return true;
}
```

Aufgabe 2: Klasse "MyDate" mit "main()" zum Testen

Das in der vorherigen Aufgabe erstellte Programm sollte nun fehlerfrei compiliert werden können, es fehlt aber noch das Hauptprogramm, damit es auch ausgeführt werden kann.

Aufgabe:

Ergänzen Sie Ihr Programm durch ein Hauptprogramm "main()". Dieses Hauptprogramm soll zum Austesten Ihrer Klasse "MyDate" dienen.

Erstellen Sie und rufen Sie aus main() zum Testen Funktionen mit assert-Anweisungen der folgenden Art auf (Beispiele):

```
void testLeapYear()
{
    cout << "Schaltjahr-Test:     ";
    MyDate date1 (25, 2, 2010);
    assert ( date1.isValid()  != 0 );    //  != 0  heisst true
    assert ( date1.isLeapYear() == 0 );  //  == 0  heisst false

    MyDate date2 (15, 1, 2100);
    assert ( date2.isValid() );
    assert ( !date2.isLeapYear() );

    cout << "OK" << endl;
}
```

Zur Erinnerung: "to assert" heisst "zusichern". Assert-Anweisungen bezeichnet man deshalb auch als Zusicherungen. Ist der logische Ausdruck in der Klammer der Assert-Anweisung *nicht wahr*, wird das Programm bei seiner Ausführung mit einer entsprechenden Fehlermeldung abgebrochen.

Wenn also alle Assert-Anweisungen ohne Abbruch durchlaufen werden, heisst dass, dass die Klasse "MyDate" alle Tests bestanden hat.

Beachten Sie auch, es werden hier *keine* Daten (via Tastatur) eingelesen, sondern alle Testdaten in der Testfunktion oder in "main()" erzeugt. Es sollen alle Tests *automatisch* ablaufen, weshalb die Überprüfung der Resultate (Vergleich Soll- Ist-Werte) innerhalb der Tests erfolgt.

Achtung: Es ist durchaus sinnvoll die Klasse "MyDate" noch durch einige öffentliche `const`-Methoden zu ergänzen, damit die Klasse umfassend geprüft werden kann. Wie können Sie sonst z.B. die Funktion "addDays()" testen? Diese zusätzlichen Memberfunktionen fürfen aber keinesfalls die Werte der Membervariablen ändern (darum `const`).

Ihr Testprogramm soll möglichst alle Fälle abdecken, so dass Sie bereit sind, eine gute Flasche Wein zu wetten, dass die Klasse "MyDate" keine Fehler mehr enthält, falls sie alle Ihre Tests besteht!

(Stören Sie sich vorläufig nicht daran, dass beim Ausführen des Programms praktisch alle Tests fehlschlagen.)

Aufgabe 3: Klasse "MyDate" implementieren

Implementieren Sie nun die Klasse "MyDate" und testen Sie diese mit Ihrem Testprogramm von Aufgabe 2 aus.

Tipp 1: Für die Implementation können Sie natürlich auch weiter private Memberfunktion (Hilfsfunktionen) wie z.B. "incDay()" zur Klasse hinzufügen.

Tipp 2: Die Memberfunktion "addDays()" darf nicht unterschätzt werden ☺ – siehe Tipp 1.

Die bei Aufgabe 2 gewonnenen Erkenntnisse sollten Ihnen nun dabei helfen, die Aufgabe dieser Klasse besser zu verstehen.

Die Vorgehensweise, das heisst, wenn die Tests vor der Implementierung geschrieben werden, bezeichnet man als "*Test Driven Development*" (TDD).