Abend 19, Übung Notendarstellung

Ziel

Wir sind in der Lage eine Klasse Note zur Abstraktion einer Note zu schreiben. Mit der Klasse Notenliste verwalten wir die Noten eines Fachs, können diese ausgeben und mit Hilfe des TsuZeichnen-Programms ist sogar eine einfache Grafik möglich.

Übung Notenliste

Wichtig ist zunächst, dass ihr Euch die neueste Version des TsuZeichnen.exe-Programms und der Hilfsklassen besorgt. Vergesst nicht das TsuZeichnen.exe in das Verzeichnis zu kopieren, in dem sich Euer Projekt befindet und es von dort zu registrieren (mit der Option "-regserver" das Programm starten, oder das .bat-File in das Verzeichnis kopieren und laufen lassen!).

Klasse Note

Die Klasse Note ist sehr simpel aufgebaut, sie kapselt im Grunde genommen nur einen double-Wert, der dem Notenwert entspricht. Ein Noten-Objekt ist auch in der Lage den Benutzer mit der Methode "einlesen" selber nach dem Notenwert zu fragen (über die Konsole : cin). Die Note kann sich auch selber darstellen, entweder mit der Methode "ausgeben" ohne Parameter oder mit der anderen Version dieser Methode, die als Parameter ein ostream-Objekt übernimmt (dadurch ist auch die Ausgabe in eine Datei möglich, dazu unten mehr!)
Die Methode holeGerundeteNote haben wir gemeinsam in der Stunde angesehen:

```
double Note::holeGerundeteNote(enum Note::Rundung rundung) const
{
    double rundungsWert = rundung;  // durch geschickte
                                    // Wahl des enums
                                    // können wir etwas
                                    // vereinfachen
    double korrektur = 1.0 / rundungsWert / 2.0;

    int hilfswert = (int)((m_notenWert + korrektur) * rundungsWert);

    double gerundeterWert = (double)hilfswert / rundungsWert;

    return gerundeterWert;
}
```

In der Note haben wir den Enum Rundung definiert. Wenn wir ein Variable des Typs Rundung verwenden wollen, ist es nötig den Sichtbarkeitsoperator :: zu verwenden (Note::Rundung). Da wir die einzelnen Werte im enum

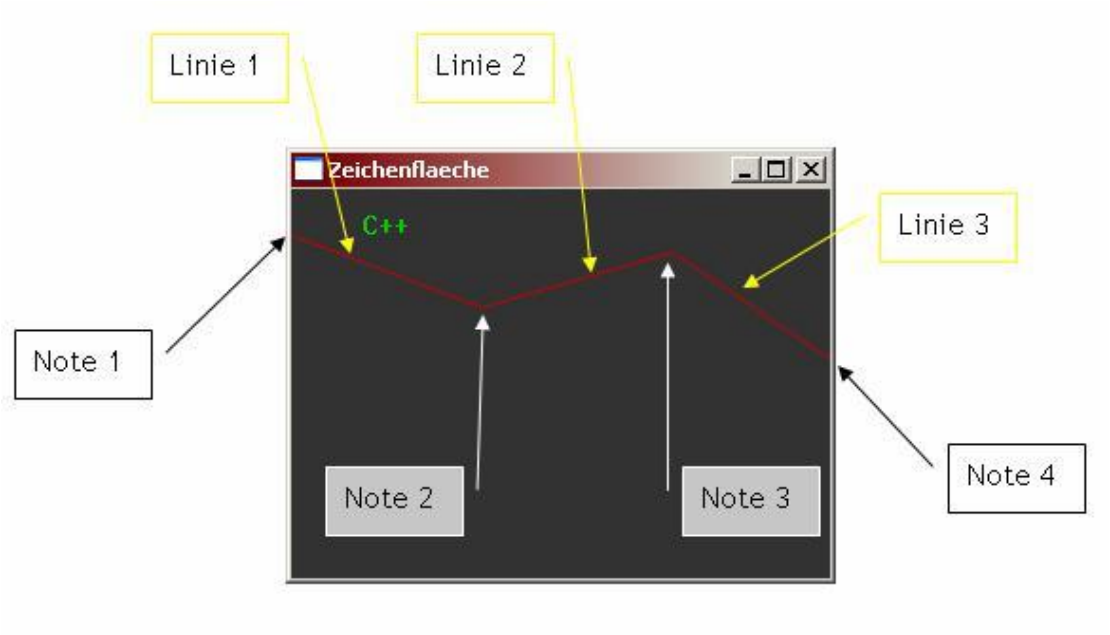
geschickt gewählt haben, können wir direkt einen double rundungsWert definieren, der so einen dieser Werte haben kann : [1.0, 2.0, 4.0, 10.0]
Für die Rundung brauchen wir einen Korrekturwert, der zur Note addiert wird, bevor sie multipliziert wird. Durch das multiplizieren wird ist der relevante Teil der Note vor dem Dezimalpunkt. Durch umwandeln in einen int verlieren wir den Nachkommateil (der Wert wird ungenauer). Jetzt können wir die Multiplikation rückgängig machen, indem wir wieder durch den rundungsWert teilen.
Spiele diesen Algorithmus (Rechenvorschrift) von Hand mit verschiedenen Werten Schritt für Schirtt durch falls Du nicht sicher bist, was hier überhaupt passiert.
Die anderen Methoden der Klasse entsprechen genau meinem Geschmack, denn sie sind einfach und klein und sollten leicht nachvollziehbar sein.

Notenliste

Die Klasse Notenliste ist sehr ähnlich aufgebaut wie die Klasse CDListe oder Menu aus der letzten Übung CDListe. Auch hier haben wir ein Array für eine begrenzte Anzahl Noten-Objekte.
Ein Objekt der Klasse Notenliste hat auch immer den Namen des Faches, für welches diese Liste gilt (Datenelement: m_Fach von der Klasse string). Es gibt Methoden zum hinzufügen einer Note, zum Asugeben der ganzen Liste und zur Berechnung des Durchschnitts. Eine zweite Version der Methode ausgeben hat als Parameter ein Objekt der Klasse ostream. Diese Methode können wir verwenden um die Liste in eine Datei zu schreiben.
Zur grafischen Ausgabe fügen wir ein Datenelement der Klasse ZeichenFlaeche hinzu. Das heisst für jede Notenliste (jedes Fach) erscheint ein Fenster als Zeichenfläche. Da die Klasse ZeichenFlaeche nur einen Konstruktor mit Parametern hat, müssen wir dieses Objekt in der Initialisierungsliste des Konstruktors der Klasse Notenliste initialisieren.

```
Notenliste::Notenliste(const std::string& Fach)
           :m_Fach(Fach), m_AnzahlNoten(0),
            m_flaeche(Breite, Hoehe)
{
    m_flaeche.setzeFarbe(RGB(50,50,50));
    Text derText(50,20);
    derText.setzeText(m_Fach);
    derText.setzeFarbe(RGB(10,200,10));
    m_flaeche.textHinzufuegen(derText);
}
```

Hier wird auch gleich ein Text in die Zeichenfläche geschrieben, der als Titel dient. Ich verwende hier einfach den Namen des Faches.
Die Noten werden grafisch als Linien dargestellt. Diese Linien verwalten wir, wie die einzelnen Noten in einem Array, wobei es eine Linie weniger hat als Noten.



In der Methode "hinzufuegen", in der eine Note in die Liste eingetragen wird, wird, sobald zwei Noten vorhanden sind die private Methode "neueLinie" aufgerufen.
Private Methoden wie "neueLinie" und auch "berechneY" dienen hier dazu den Code übersichtlicher und modularer zu gestalten.
Die Methode neueLinie ist recht aufwendig und braucht einen klaren Kopf ;-)
Am besten man hat hier einen Zeichnung der Grafik vor sich und versucht die einzelnen Schritte und Elemente auf Papier herzuleiten oder nachzuvollziehen.

```
void Notenliste::neueLinie()
{
    int AnzahlLinien = m_AnzahlNoten-1;

    int xBreite = Breite / AnzahlLinien;

    for(int i = 0; i < AnzahlLinien; ++i)
    {
        int x1 = i * xBreite;
        int x2 = x1 + xBreite;
        Note note1 = m_Noten[i];
        Note note2 = m_Noten[i+1];
        int y1 = berechneY(note1);
        int y2 = berechneY(note2);

        m_linien[i].setzeLinie(x1, y1, x2, y2);
    }
    Linie& neueLinie = m_linien[AnzahlLinien-1];
    neueLinie.setzeFarbe(RGB(200,0,0));
    m_flaeche.linieHinzufuegen(neueLinie);
}
```

Die Anzahl Linien ist, wie ihr unschwer erkennen könnt die Anzahl Noten minus eine. Die Noten haben untereinander einen festen Abstand in x-Richtung, den ich in dieser Methode xBreite nenne.
Danach kommt eine Schleife in der ich für jede Linie, die dargestellt wird, die jeweiligen Koordinaten berechne. Die X-Koordinaten sind einfach nachzuvollziehen, zur Berechnung der beiden Y-Werte brauchen wir die beiden Noten, die durch die Linie verbunden werden. Die Berechnung der Y-Koordinate ist ein wenig aufwendiger und ist darum in eine eigene Methode gewandert : "berechneY"
Die Methode "berechneY" berechnet also eine Y-Koordinate aus der Note und verwendet hierfür die Bildschirmhöhe und den ungerundeten Notenwert.

Die main-Funktion

Die main-Funktion ist in dieser Lösung sehr einfach gehalten und dient eigentlich nur einem grundsätzlichen Funktionstest der Klasse Notenliste. Schön wäre hier auch ein Menu, das es dem Benutzer ermöglicht neue Notenlisten zu erzeugen, oder eine allgemein bessere Nutzerführung. Übrigens wäre auch eine aufwendigere grafische Ausgabe denkbar, mit Hilfslinien, einer Linie für den Durchschnitt und so weiter. Eurer Kreativität sind kaum Grenzen gesetzt...
Wichtig
In der main-Funktion ist es wichtig ein Objekt der Klasse ComSystem zu erzeugen, denn diese initialisiert im Konstruktor das Com-Subsystem des Betriebssystems und räumt im Destruktor wieder auf.

```
#include "Notenliste.h"
#include <string>
#include <iostream>
#include <conio.h>
#include <fstream>
#include "ZeichenFlaeche.h"

using namespace std;

int main()
{
    ComSystem dasComSystem;    ...
```

Ausgabe in eine Datei

Schön wäre es die Noten auch direkt in eine Datei schreiben zu können. In C++ ist es nun so, dass eine Datei für die Ausgabe ganz ähnlich verwendet werden kann wie die Konsole, also das cout-Objekt.
Hier ein kleines Beispiel:

```
#include <fstream>
using namespace std;
int main()
{
    ofstream dateiOut("Test.txt");

    dateiOut << "Hallo Welt" << endl;
    dateiOut << "C++ ist cool" << endl;
    return 0;
}
```

Wenn Du dieses Programm kompilierst und laufen lässt, sollte sich danach eine Datei "Test.txt" im Projekt-Verzeichnis befinden. Diese Datei kannst Du mit dem Notepad Editor oder einem anderen Texteditor betrachten.
Das Objekt cout für die Konsolenausgaben ist ein Objekt der Klasse ostream. Ein ofstream ist ein spezieller ostream, der zur Ausgabe eine Datei (file) verwendet, sonst aber genauso ist wie ein ostream. Diese Spezialisierung werden wir bald unter dem Stichwort "Vererbung" kennen lernen.
Die Methoden zur Ausgabe in der Klasse Notenliste und Note sind jetzt in einer Version mit einer ostream-Referenz vorhanden. Wir können hier jetzt entweder das altbekannte cout-Objekt oder aber ein Objekt der Klasse ofstream also eine Datei mitgeben.

```
int main()
{
    ComSystem dasComSystem;

    cout << "Fach ? ";
    string Fach;
    cin >> Fach;

    Notenliste eineNotenliste(Fach);

    bool weiter = true;
    while(weiter)
    {
        Note eineNote;
        eineNote.einlesen();
        eineNotenliste.hinzufuegen(eineNote);

        cout << "weiter [j/n] ? ";
        string eingabe;
        cin >> eingabe;
        if(eingabe != "j")
        {
            weiter = false;
        }
    }
    // Hier in der Konsole ausgeben
    eineNotenliste.ausgeben(cout);
    // Datei erzeugen
    ofstream dateiOut("Note.txt");
    // und zur Ausgabe an
    // Notenliste übergeben
    eineNotenliste.ausgeben(dateiOut);

    getch();
    return 0;
}
```