FEJLETT PROGRAMOZÁSI TECHNIKÁK (C++)
10.2. GYAKORLAT

Cél:

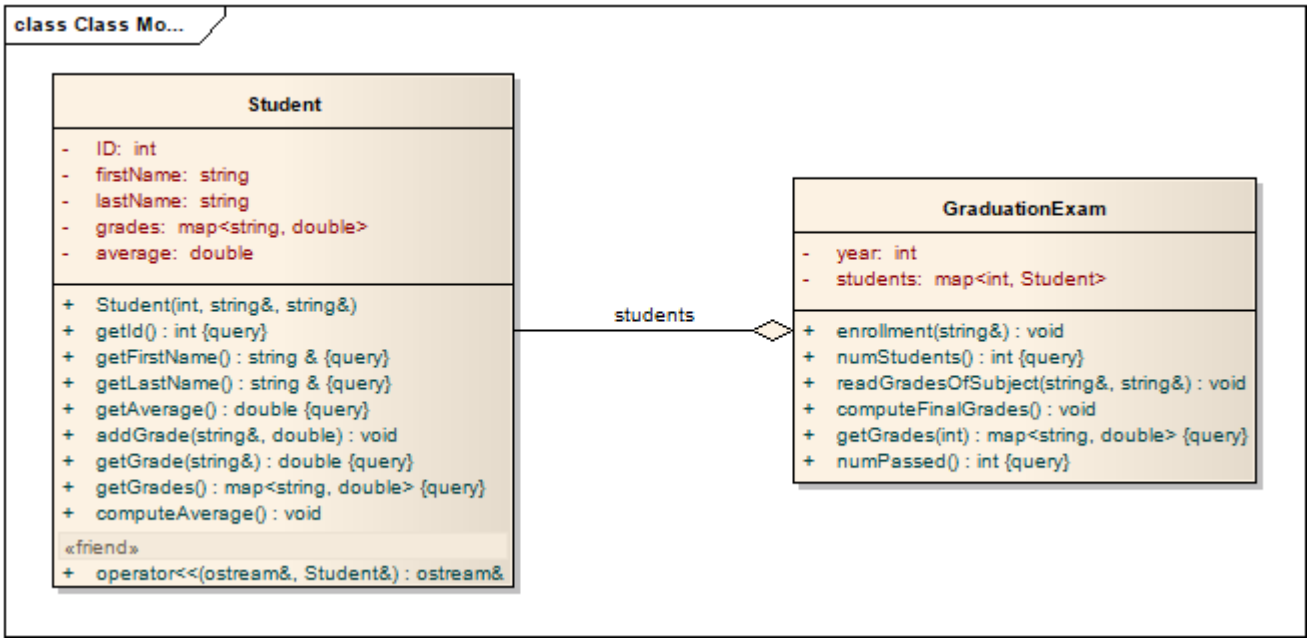
- A **map** asszociatív tároló gyakorlása
- Interfészek (absztrakt osztályok tiszta virtuális függvényekkel)
- OO tervezés

Érettségi

Egy szöveges állomány (`names.txt`) érettségiző diákok azonosítóit, illetve neveit tartalmazza. Minden diáknak egy vezetékneve és egy keresztneve van. Az állomány minden sora egyetlen diák adatait tartalmazza a következő sorrendben: `azonosító vezetéknév keresztnév`.

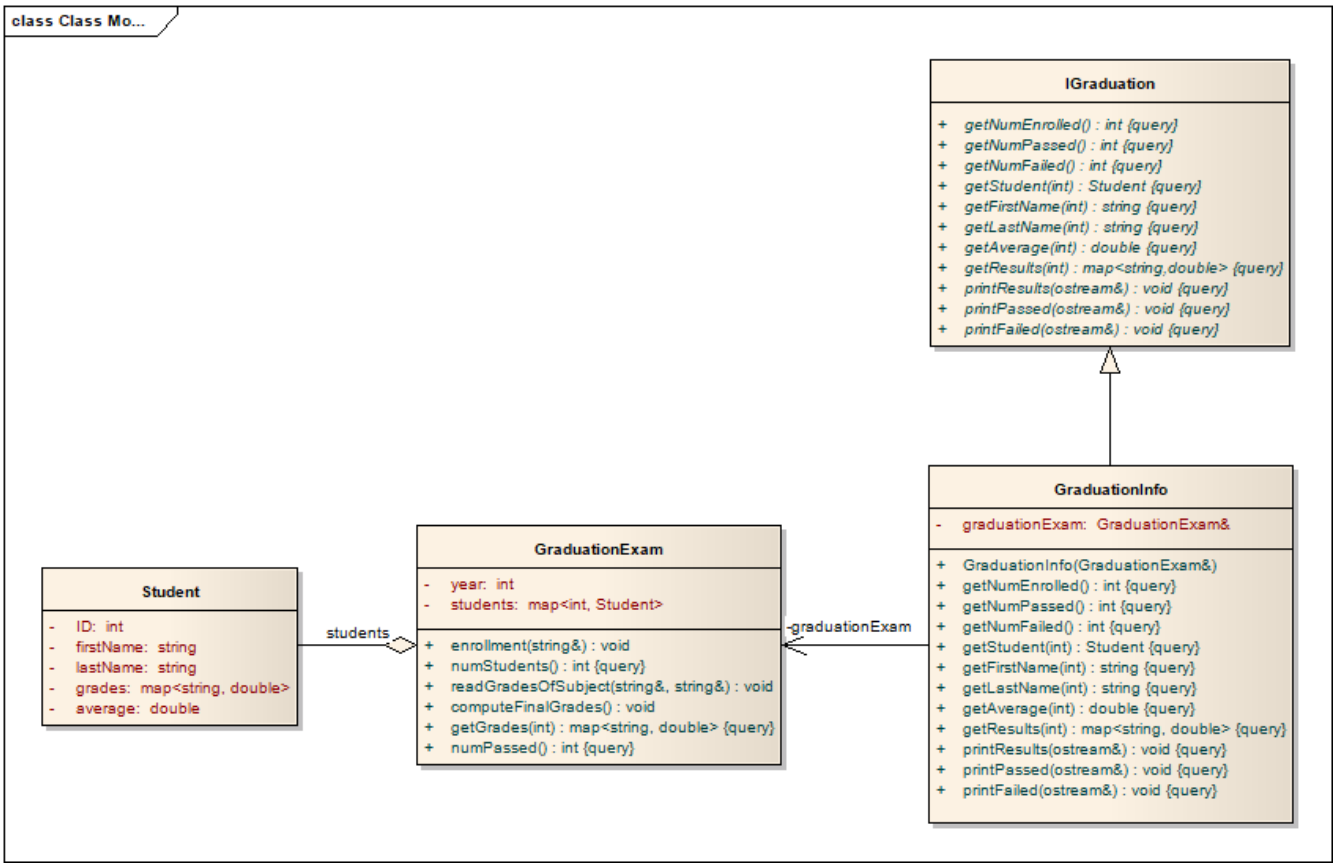
Az érettségi dolgozatokat külön bizottságok javítják, ezért a jegyeket minden érettségi tárgy esetén külön állományokban helyezik el. Összesen három érettségi tantárgyból adjuk meg a jegyeket (`hungarian.txt`, `romanian.txt` és `math.txt`), de készítse el úgy a programot, hogy az **érettségi tárgyak száma ne legyen korlátos**. A jegyeket tartalmazó állományok minden sora egy azonosítót és egy jegyet tartalmaz fehér karakterekkel elválasztva. A bemeneti állományok elérhetők Moodle-ben.

Készítsen objektumorientált programot, amely beolvassa az adatokat és kiszámítja az érettségi átlagokat. Csak azon diákoknak számítunk átlagot, akik minden tantárgyból megírták az ötöst. Az érettségi akkor sikeres, ha minden tantárgyból megírta a diák legalább az ötöst és az érettségi tárgyak átlaga legalább hatos. A számítások elvégzése után írassa ki a sikeresen érettségizett diákok számát.



FEJLETT PROGRAMOZÁSI TECHNIKÁK (C++)
10.2. GYAKORLAT

Egészítse ki az elkészített programot egy szolgáltatással, amely segítségével adatokat tudunk lekérdezni az érettségiről. Ennek érdekében készítsen egy interfész típusú osztályt (**IGraduation**, *absztrakt osztály tiszta virtuális függvényekkel)*, majd implementálja az interfészt (*publikus származtatás az absztrakt osztályból*) egy konkrét osztályban (**GraduationInfo**).



```
#ifndef GRADUATIONEXAM_H
#define GRADUATIONEXAM_H


#include "Student.h"
#include <map>

using namespace std;

class GraduationInfo;
class GraduationExam {
    int year;
    map<int, Student> students;
    friend class GraduationInfo;
public:
    // A names.txt fájlban lévő hallgatókat beiratkoztatni az érettségire
    void enrollment(const string& filename);

    int numStudents() const { return students.size();}


    // Valamely tantárgy jegyeit tartalmazó fájl beolvasása és a jegyek
    // elhelyezése a megfelelő diákhoz
    void readGradesOfSubject(const string& subject, const string& filename);

    // Kiszámítja az érettségi átlagát minden diáknak
    void computeFinalGrades();

    // Visszatéríti adott azonosítójú diák jegyeit tantárgyakkal együtt
    map<string, double> getGrades( int studentID) const;

    // Sikeresen érettségizők száma
    int numPassed() const;

};


#endif //GRADUATIONEXAM_H
```

```
class IGraduation {
public:
   virtual int getNumEnrolled() const = 0;//tiszta virtuális függvény
   // ...

}
```

Ha helyesen dolgozott, akkor az alábbi programnak kell működnie:

```
#include <iostream>
#include "Student.h"
#include "GraduationExam.h"
#include "GraduationInfo.h"
#include "IGraduation.h"

string subjects[] {"Romanian", "Hungarian", "Math"};

int main() {
   GraduationExam exam;
   exam.enrollment("names.txt");
   cout<<exam.numStudents()<<endl;
   exam.readGradesOfSubject(subjects[0], "romanian.txt");
   exam.readGradesOfSubject(subjects[1], "hungarian.txt");
   exam.readGradesOfSubject(subjects[2], "math.txt");
   exam.computeFinalGrades();
   cout<<exam.numPassed()<<endl;
   // Service
   IGraduation * graduation = new GraduationInfo( exam );
   cout<<"Graduation Information"<<endl;
   cout<<"Number of enrolled students: "<<graduation->getNumEnrolled()<<endl;
   cout<<"Number of passed students: "<<graduation->getNumPassed()<<endl;
   cout<<"Number of failed students: "<<graduation->getNumFailed()<<endl;
   int ID;
   for( ;; ){
       cout<<"Enter an ID <0 for termination>: ";
       cin>>ID;
       if( ID == 0 ){
           break;
       }
       cout<<"Information:"<<endl;
       try{
           const Student& student = graduation->getStudent( ID );
           cout<<student<<endl;
       }catch( domain_error& e){
           cout<<e.what()<<endl;
       }
   }
   delete graduation;
   return 0;
}
```