AGH – UNIVERSITY OF SCIENCE AND TECHNOLOGY

Project documentation for

Student Grade Book Application

Object-oriented programming languages

Michał Bogucki
Mirosław Kołodziej

lecturer: Rafał Frączek

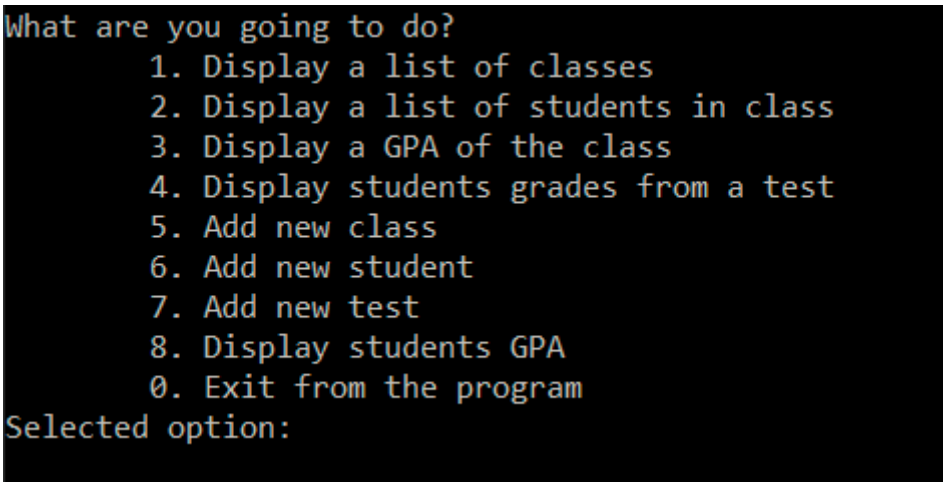
15.01.2022

1. Project description

Project keeps track of students (with a student class that has their name, average, and scores) in a class and their grades. User can assign scores on tests and assignments to the students and display average grades for students and for the class.

2. User's manual

After opening a program, the user should see a simple menu with list of available options.



User should choose one of the displayed options by writing a number of selected option. Next menus are similar to the first menu.

3. Compilation

Project can be built on all systems, standard c++ like gcc should be enough.

4. Source files

The project consists of the following source files:

- *clearscreen.h*, *clearscreen.cpp* – declaration and implementation of the `ClearScreen` function,
- *DataHandler.h*, *DataHandler.cpp* – declaration and implementation of the `DataHandler` class,
- *GradeBook.h*, *GradeBook.cpp* – declaration and implementation of the `GradeBook` class,
- *MarkOps.h* – declaration and implementation of the `MarkOps` struct,
- *GradeBook.h*, *GradeBook.cpp* – declaration and implementation of the `GradeBook` class,
- *OOPProject.cpp* – implementation of the `main` function,
- *Student.h*, *Student.cpp* – declaration and implementation of the `Student` class,
- *StudentTest.h*, *StudentTest.cpp* – declaration and implementation of the `StudentTest` class.

5. Dependencies

None.

6. Class description

In the project the following classes and structures were created:

- `DataHandler` – a structure that loads and saves data from .txt file.
  - `static vector<GradeBook> loadData()` – returns vector of the grade books,
  - `static void saveData(vector<GradeBook> gradeBooks)` – saves data to .txt file,
- `GradeBook` – a class that represents grade book for the school class.
  - `int getGradeBookIdOffset()` – getter for `gradeBookIdOffset`,
  - `int getID()` – getter for `id`,
  - `string getName()` – getter for `schoolClassName`,
  - `void addStudent(Student &student)` – adds `student` to `studentVector`,
  - `vector<Student> getStudents()` – getter for `studentVector`,
  - `void setStudentTests(vector<StudentTest> studentTestsVector)` – setter for `studentVector`,
  - `void printTestWithScores(int i)` – prints information about tests with scores,
  - `float getStudentMean(Student& student)` – returns student GPA,
  - `void printStudentsMean()` – prints students GPA,
  - `float getClassMean()` – returns class GPA,
  - `void addTest(StudentTest& test)` – adds `test` to `studentTestVector`,
  - `void addScoresForTest(StudentTest& test)` – adds scores of students for `test`,
  - `void printStudents()` – prints information about students,
  - `void printTests()` – prints information about tests,
  - `int getStudentTestVectorSize()` – returns size of `studentTestVector`,
  - `StudentTest getTest(int i)` getter for tests from `studentTestVector`,
- `MarkOps` – a struct that helps with assigning marks for students.
  - `static string getMark(int score, int maxScore)` – returns mark,
- `Student` – a class that represents students.
  - `int getID()` – getter for `id`,
  - `void setID()` – setter for `id`,

- `string getName()` – getter for `name`,
- `string getLastName()` – getter for `lastName`,
- `void setTestResultsMap(map<int, int> testResultsMap)` – setter for `testResultsMap`,
- `void setTestResultsMap(map<int, int> testResultsMap)` – setter for `testResultsMap`,
- `void setTestMarksMap(map<int, int> testMarksMap)` – setter for `testMarksMap`,
- `map<int, int> getScoresMap()` – getter for `testResultsMap`,
- `map<int, string> getMarksMap()` – getter for `testMarksMap`,
- `float getMeanMark()` – calculates GPA for a student,
- `void addScore(int testId, int testScore,int maxScore)` – adds score of student for test,
- `void print()` – prints information about student,

- `StudentTest` – a class that represents a test.
  - `int getID()` – getter for `id`,
  - `string getName()` – getter for `name`,
  - `string getSubject()` – getter for `subject`,
  - `string getDate()` – getter for `date`,
  - `int getMaxScore()` – getter for `maxScore`,
  - `void print()` – prints information about test.

7. Resources

- data.txt - file containing saved files that serve as storage for data - used for loading/saving data. The file structure in a manned as described in DataHandler.cpp code:

```
/*
Scheme:
    tokens are values that allow to get around *txt save file
    splitted by , delimiter.

    gradebook.size\n
    gradebook.id,gradebook.name,students.size,tests.size\n
    students,\n
    [student0.id,student0.name,student0.lastName,student0.scoresMap,student0.marksMap,
    .
    .
    .
    studentN.id,studentN.name,studentN.lastName,studentN.scoresMap,studentN.marksMap,],\n
    studentTests,\n
    [studentTest0.id,studentTest0.name,studentTest0.subject,studentTest0.maxScore,studentTest0.date,
    .
    .
    .
    studentTestN.id,studentTestN.name,studentTestN.subject,studentTestN.maxScore,studentTestN.date,],\n
    gradebook<id>,\n
    for scores and marks map:

    [,k1,v1, ... ,kN,vN],

*/
```

8. Future development

For future development the best possibilities are:

- Graphic UI - implement application generated user interface not to use the console and make operating in application more intuitive and easier.

- DataBase storage - implement database handler structures and database like MySQL, because of relations in structures. The second idea is to store id references to each of the structures and implement a non relational database or other persistence layer to store objects used.

- Implement more functional methods and objects that will make it easier to operate in program

- Implement More generic structures to wrap underlying types that implement most generic methods.

9. Other

None.