CS 1131

جامعة عفت

EFFAT UNIVERSITY

CS 1131

Advanced Programming - Fall 2021

Author: **Flowra Almufadda (S19105762) , Mehreen Junaid (S20106424)**

Haifa Albalochi (S19105548) , Mageda Alasabi (S19105541)

Instructor: **Akila Sarirete**

Date of Submission: December 19 2021

Contents

1 Introduction

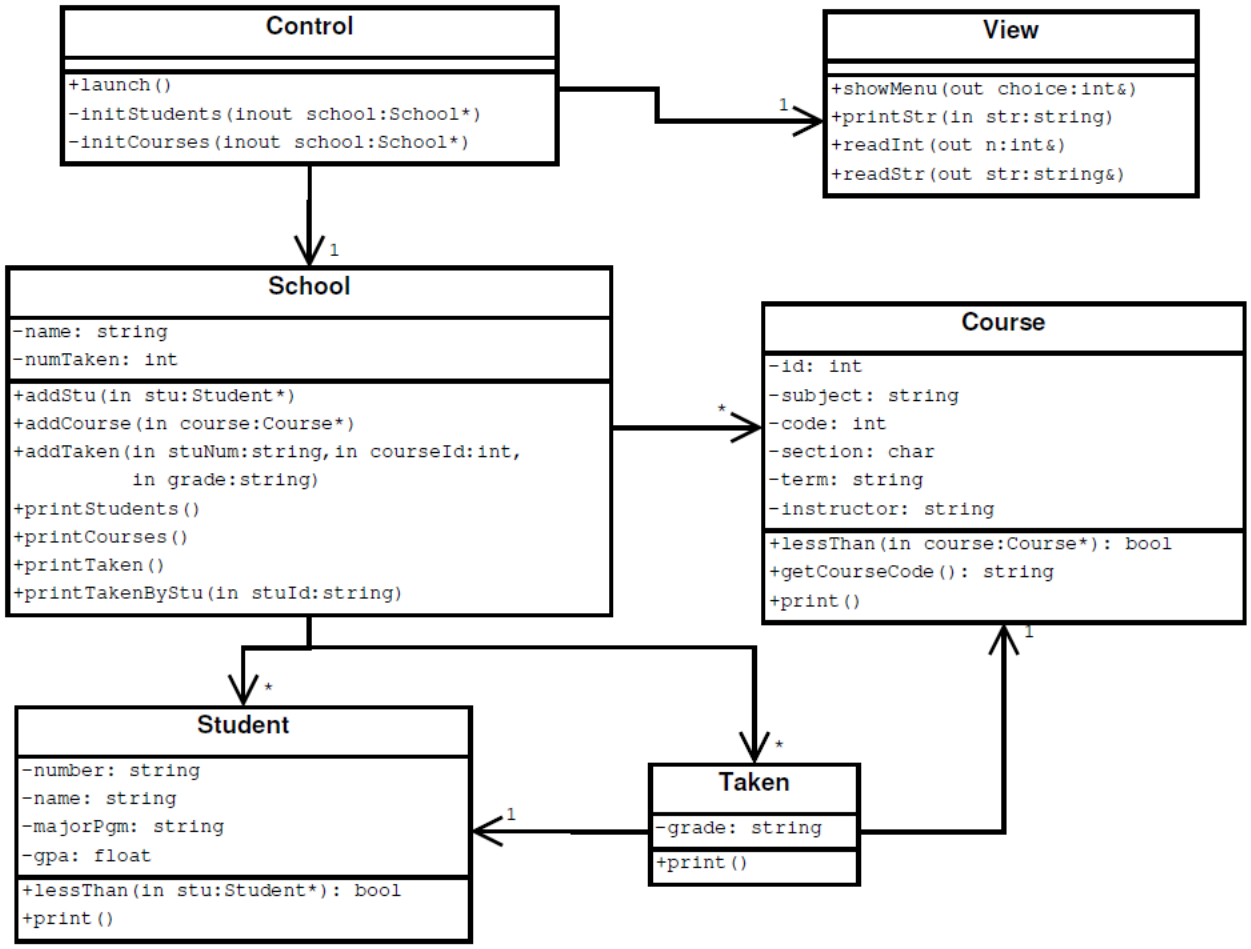
1.1 Goal and Learning outcomes

For this project, you will write a C++ program to manage the data for a school with students and courses. You will implement your program using objects from the different classes, based on a UML class diagram provided for you.

- practice implementing a design that is given as a UML class diagram
- implement a program separated into control, view, entity, and collection objects
- work with statically allocated and dynamically allocated arrays

2 Problem Statement and Design

2.1 UML class diagram



3 Implementation

The implementation process was done and dealt with 8 classes. All 7 has both header and implementation files, while-as main() had none, aka just a .cpp file.

3.1 Student class

The purpose of this class is to manage the list of students by acquiring the information of a student and adding it into the list, as well as display the list if requested by the user.
The class contains both a header and implantation files. Where the variables (stuNumber, stuName, stuMajor, stuGPA) were created. The functions required are:

- Constructors (Default + overloaded)
- Getter functions + getData() / Getting the data from the user.
- Setter functions + setData() / Setting the data from the student.
- lessThan() / Comparing two students according to data.
- print() / Displays information of a selected student.

3.2 Course class

The purpose of this class is to manage the list of courses by acquiring the information of a course and adding it into the list, as well as display the list if requested by the user.
The class contains both a header and implantation files. Where the variables (courseID, courseCode, courseSection, courseSubject, courseTerm, courseInstructor) were created. The functions required are:

- Constructors (Default + overloaded)
- Getter functions + getData() / Getting the data from the user.
- Setter functions + setData() / Setting the data of the course.
- lessThan() / Comparing two courses according to data.
- print() / DIsplays information of selected course.

3.3 Taken class

The purpose of this class is to manage the courses and students, by classifying the courses on whether a specific student has taken the course or not.
The class contains both a header and implantation files. Where the variables (grade) and pointers (stuTakenCrs, crsTakenStu) were created. The functions required are:

- Constructors (Default + overloaded)
- Getter functions + getData() / Getting the data from the user.
- Setter functions + setData() / Setting the data of the course.
- lessThan() / Comparing two courses according to data.
- print() / Displays information of selected course.

3.4 DynArray class

The purpose of this class is to dynamically manage the array of students listed that can be found in the student.h file.
The class contains both a header and implantation files. Where the pointer (arr2[MAX-ARR]) was created. The functions required are:

- Constructors (Default + destructor)
- add() / Add a student to the system or array.
- find() / Find the student from the array.
- lessThan() / Compare two students according to data.
- print() / Displays information of selected student.

3.5 StatArray class

The purpose of this class is to statically manage the array of courses listed that can be found in the course.h file.
The class contains both a header and implantation files. Where the pointer (arr2[MAX-ARR]) was created. The functions required are:

- Constructors (Default + destructor)
- add() / Add a course to the system or array.
- find() / Find the course from the array.
- lessThan() / Compare two courses according to data.
- print() / Displays information of selected course.

3.6 School class

The purpose of this class is to connect all the files and classes in this program into one file and manages everything. It is considered to be the heart of the program, the main of it.
The class contains both a header and implantation files. Where the variables (name, numTaken) were created. The functions required are:

- Constructors (Default + destructor)
- Adder functions / Add information into the system.
- Print functions / Print information from the system for the user.

3.7 Control class

The purpose of this class is control the program, it is connected to the classes school (which is the main class) and view (the class that focuses on the I/O mechanics).
The class contains both a header and implantation files. Where the variables (view) was created. The functions required are:

- Constructors (Default)
- launch() / Launch the system.
- Initializers / Initializes the students and courses in the system.

3.8 Main program

4 Conclusion

4.1 Functional Dependencies

According to the given UML class diagram included in the introduction, the relations were shown between each class listed.

- Control is functionally dependent on View and School.
- School is functionally dependent on Course, Taken, and Student.
- Taken is functionally dependent on Student and Course.

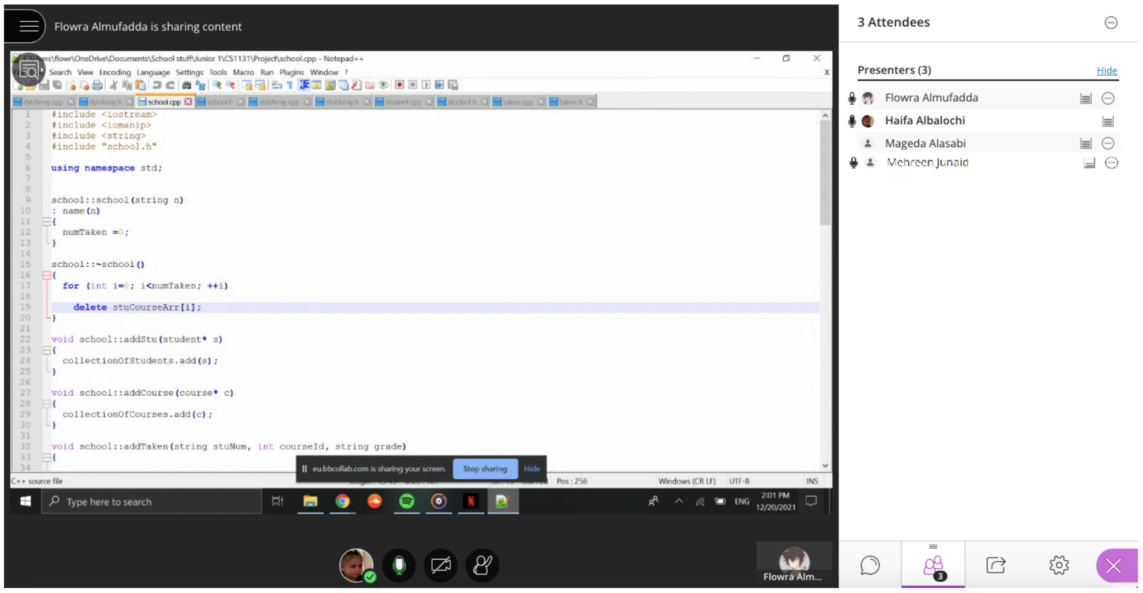
4.2 Launch Instructions

The way the program is used is it starts with the main() file, and launches the school system. It offers a list of functions that the user can selected and the switch case will lead the user to the next class related to the selected function. Such as that if the user selects adding students then it will lead to the student section where the user can use the functions offered to the corresponding class.

4.3 Team Management and Timeline

The process of this project was done together. Where the group sets up an online meeting room using BlackBoard Collaborate and worked on the code using one of the member's computer and have her share her screen to the whole meeting room and work on it from her side. If the group faces a difficulty, it is left to be done after the meeting, where each student works on it individually and then contact each other if any solutions were found. The files were shared using Google Drive.

The timeline of this process began the moment the project was assigned to the group. And was worked on since the it started till the submission day. The work was done together most of the time, unless faced with a difficult problem, then that was spent individually.



```cpp
#include <iostream>
#include <iomanip>
#include <string>
#include "school.h"

using namespace std;


school::school(string n)
: name(n)
{
  numTaken =0;
}

school::~school()
{
  for (int i=0; i<numTaken; ++i)

    delete stuCourseArr[i];
}

void school::addStu(student* s)
{
  collectionOfStudents.add(s);
}

void school::addCourse(course* c)
{
  collectionOfCourses.add(c);
}

void school::addTaken(string stuNum, int courseId, string grade)
{
```

This screenshot displays the process of how the work was done. Where one of the students shares her screen of her PC for everyone to see, and the work is discussed using the voice call function installed in BlackBoard Collaborate. Most of the time, the screen that was shared the most was Flowra's screen.