Object Oriented Programming Concepts Based Assignment: A "Student Lecturer Appointment" Application

OCTOBER 28, 2019

HIMANSHU CHAUDHARY
ROLL NO 197920

Object Oriented Programming Concepts Based

Assignment: A "Student Appointment" Application

Purpose

The purpose of this assignment is to help us learn or review

(1) the fundamentals of the C++ programming language.

(2) the inheritance , composition and various other OOPs concepts.

(3) how to create user space menu-based applications.

Background

It is necessary for the students to regularly meet their lecturers to ask questions about their work. Naturally, the lecturers are busy people and can not be found in their rooms most of the time. Hence, it is necessary to create an application that can help students and the lecturers to meet. For this purpose, I have coded a program that can help students take appointments from the lecturers.

The Task

I have to code a lecturer appointment system for students. In this system, student will have roll number, name, surname, department, start date, email and phone no. properties. Lecturers will have id number, name, surname, academic department, email, phone properties. The appointment will have student, lecturer, date, start and finish times. The relevant information about the students, lecturers and appointments should be read from the relevant files during the program start. The new information input to the program should be save to the files and kept in these files. The student number and employment numbers are unique and can be used to distinguish individuals. The user interface should allow the user to add, list, delete and update students, lecturers and appointments respectively. It should be possible to search for appointments and make sure that the appointments do not overlap for the same users. Furthermore, each users' appointments

for a given interval should be printed on the screen. In this work I have taken advantage of composition, inheritance and various other properties of OOPS.

OOPS Concepts that are used for implementation of this system:

- Inheritance
- Constructor overloading (Default Constructor)
- Constructor overloading (Parameterized Constructor)
- Static Member
- Static Member function
- Composition
- Templates
- Friend Function
- Operator Overloading
- Function Overloading
- Function Overriding
- If else ladder
- File Handling
- Virtual Function
- Friend Class
- Command Line Arguements

Inheritance:

Child classes Student and Lecturer are inheriting parent class People.

Constructor overloading (Default Constructor):

Default Constructor is used to give default values to the properties of classes i.e. Student, People and Appointment classes.

Constructor overloading (Parameterized Constructor):

Parameterized Constructor is used to give userdefined values to the properties of classes i.e. Student, People and Appointment classes.

Static Member:

Static keyword is used to define global array for holding no. of object in three different files those are Student, People and Appointment classes.

Static Member function:

Static Member is used to give default values to static count array i.e. zero.

Composition:

Composition is used to define has a relationship here I have used to define has a relationship between Appointment and Student Lecturer classes. Appointment is having Student and Lecturer.

- **Templates:**

  Here I have used templates in

  template<class T>

  void countFile(char *filename, T filenum)

  here I am using template for passing current file number for which I have to count number of objects in that file.

- **Friend Function:**

  For Student class I have defined Display function i.e. to display all the students currently enrolled as a Friend of student Class.

- **Operator Overloading:**

  Here I have overloaded extraction operator (<<) to display all the lecturers currently present in our system.

- **Function Overloading:**

  Here I have overloaded getNumber(int _number) function for integer and String data type.

- **Function Overriding:**

  Here I have used Function overriding

  void setNumber(string _number)

  Here Student and Lecturer classes are provided their different implementation for this.

- **If else Ladder:**

  I have used If else Ladder for implementing Menu for the System.

- **File Handling:**

  I have used File Handling to provide persistent storage to my system. Here I have created three files to store the information for Student Lecturer and Appointment.

- **Virtual Function:**

  Here I have used Virtual function in People class for

  virtual void setPhone(string _phone)=0;

  Student and Lecturer classes are provided their different implementation for this.

- **Friend Class:**

  Here I have defined Appointment as a friend class of Student and Lecturer class.

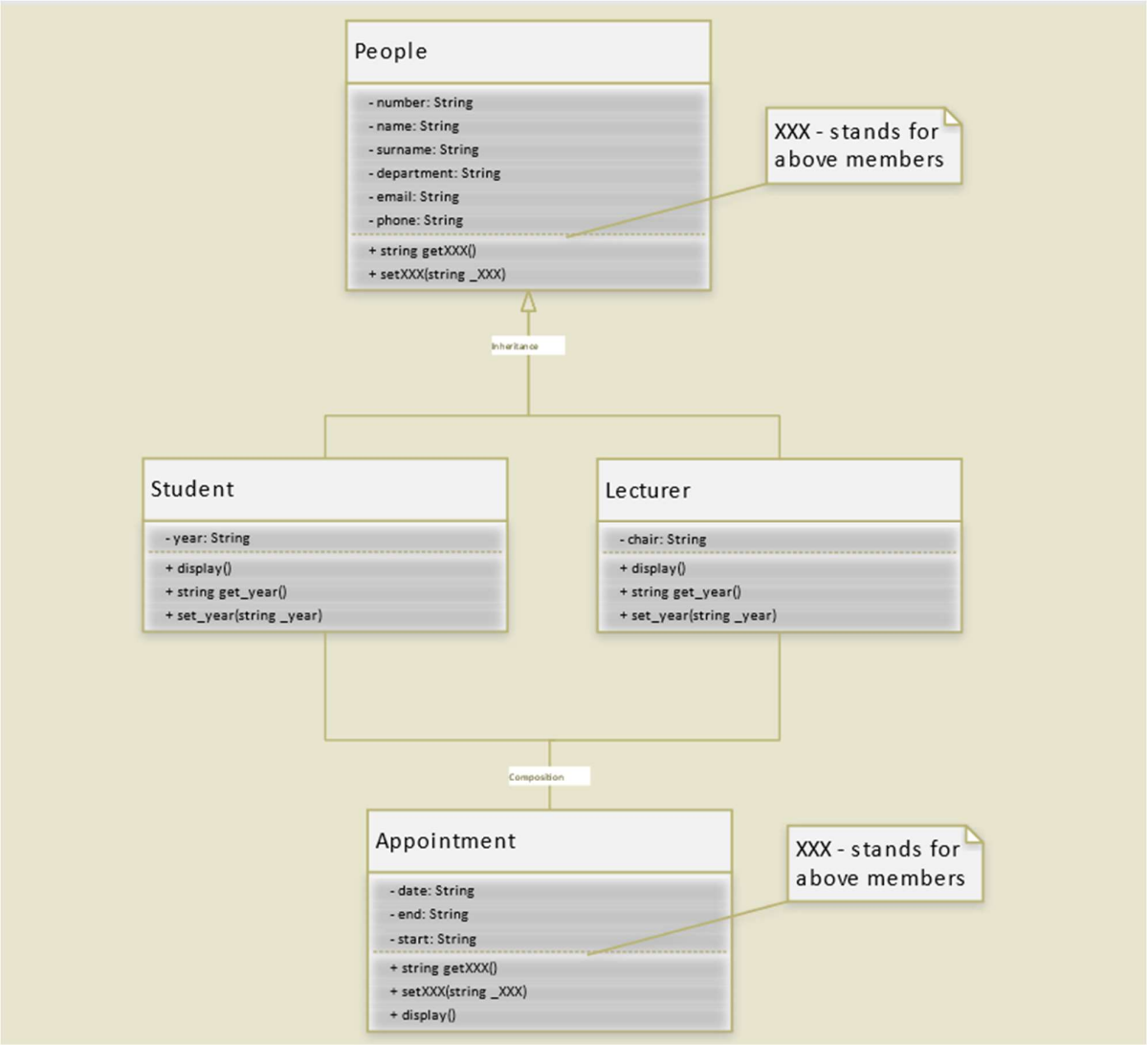
- **Command Line Arguments:**

Here I have used command line arguments to take file names of Student Lecturer and Appointment.

Design

A typical flow chart of program that might look like this:

Flow Chart Design using UML tool Microsoft Visio:



Functionality

A typical execution of program from the shell might look like this:

$>./a.out a.txt b.txt c.txt

Sample file formats should be as follows:

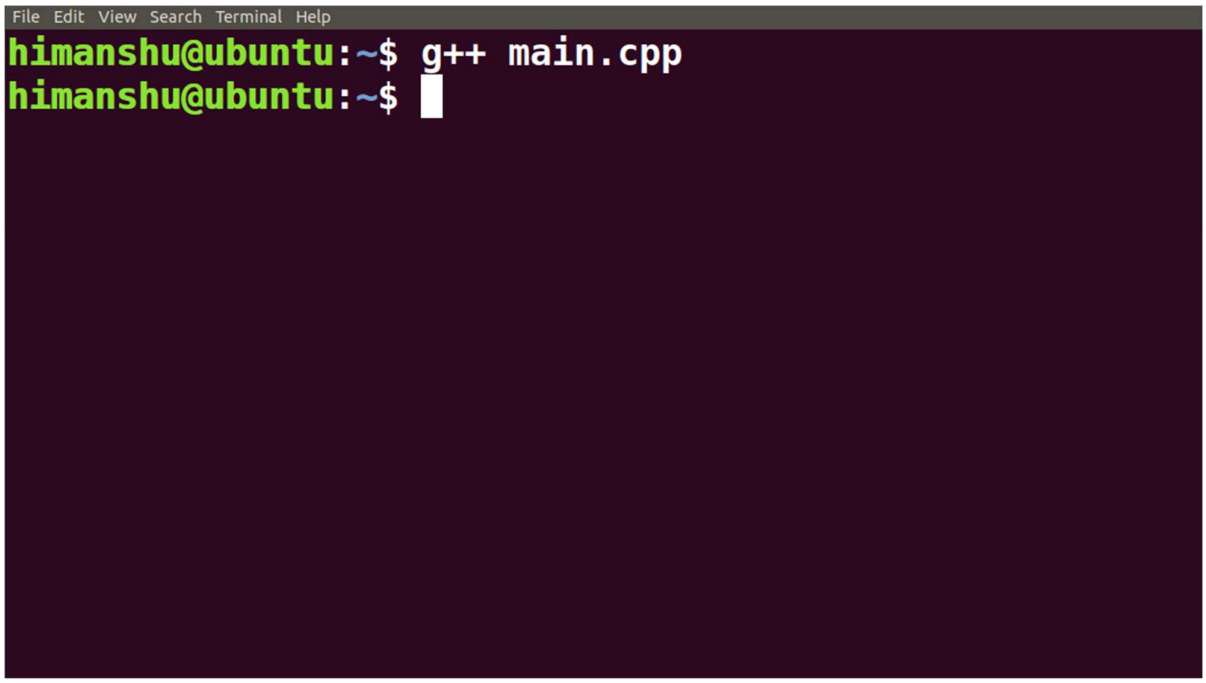
a.txt
123456 Himanshu Chaudhary CS 2019 himanshu@gmail.com +917982713217

b.txt
1234 Shekhar Joshi CS CSK@gmail.com +919999999999 Prof.Dr.

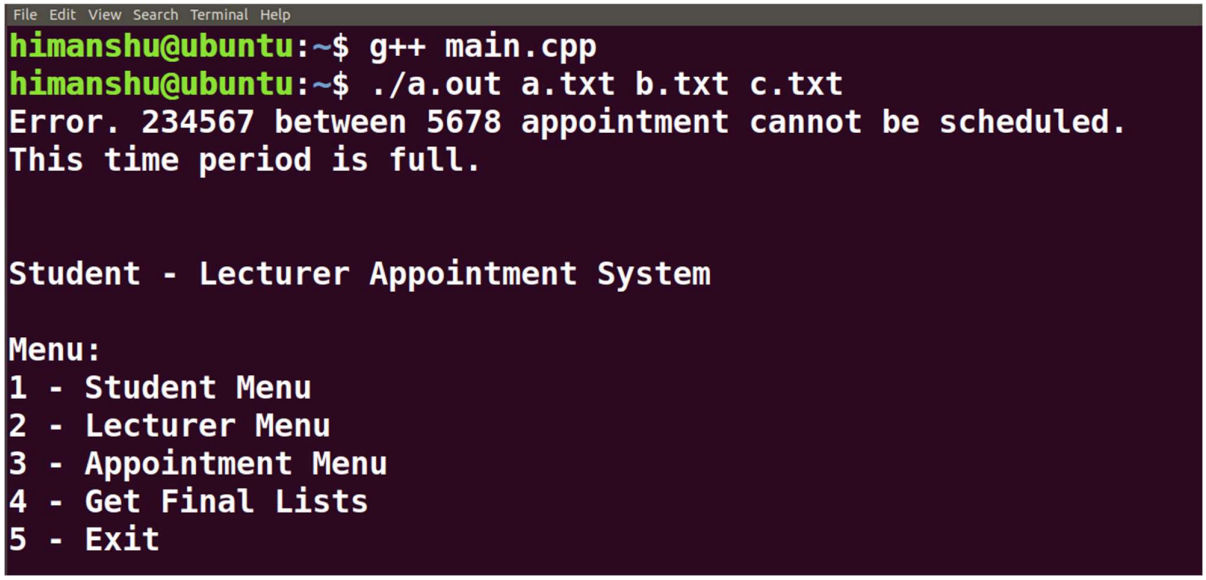
c.txt
123456 1234 12.12.2018 16:00 16:30

Execution:

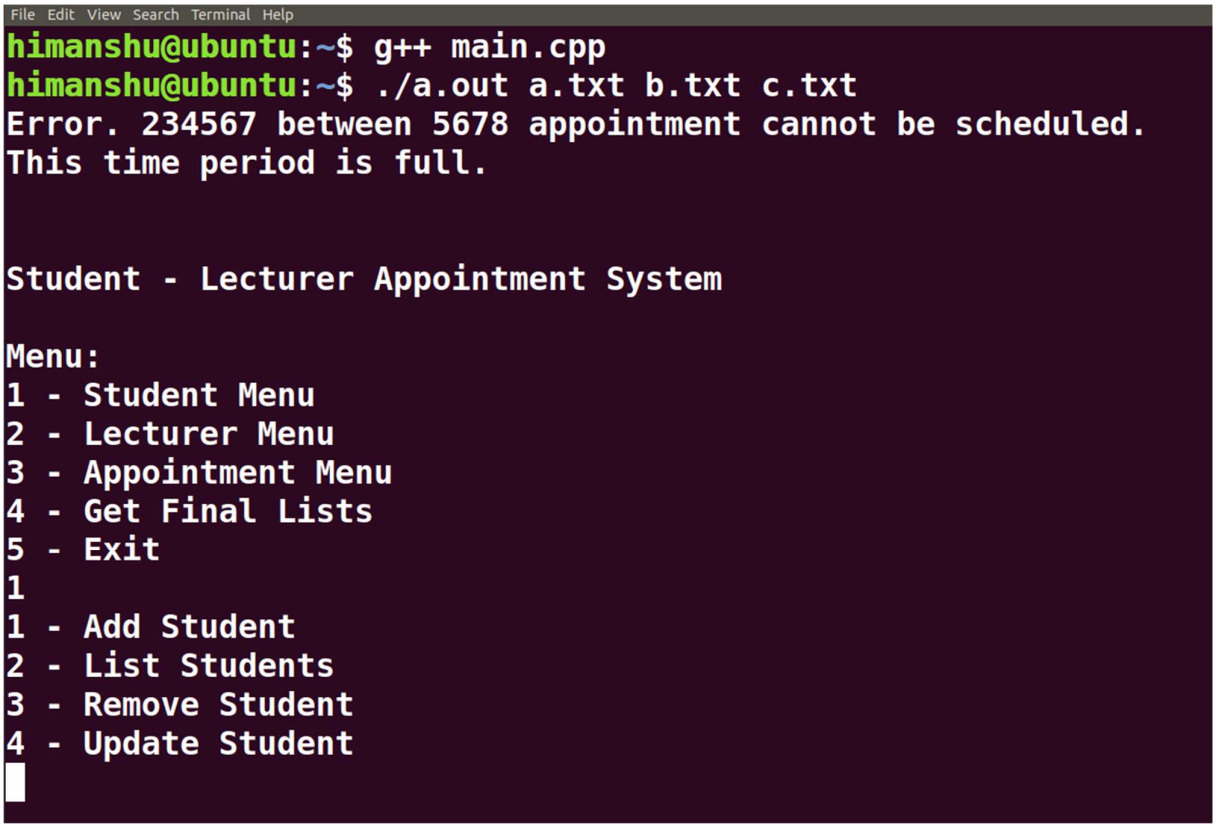
- **Compilation:**



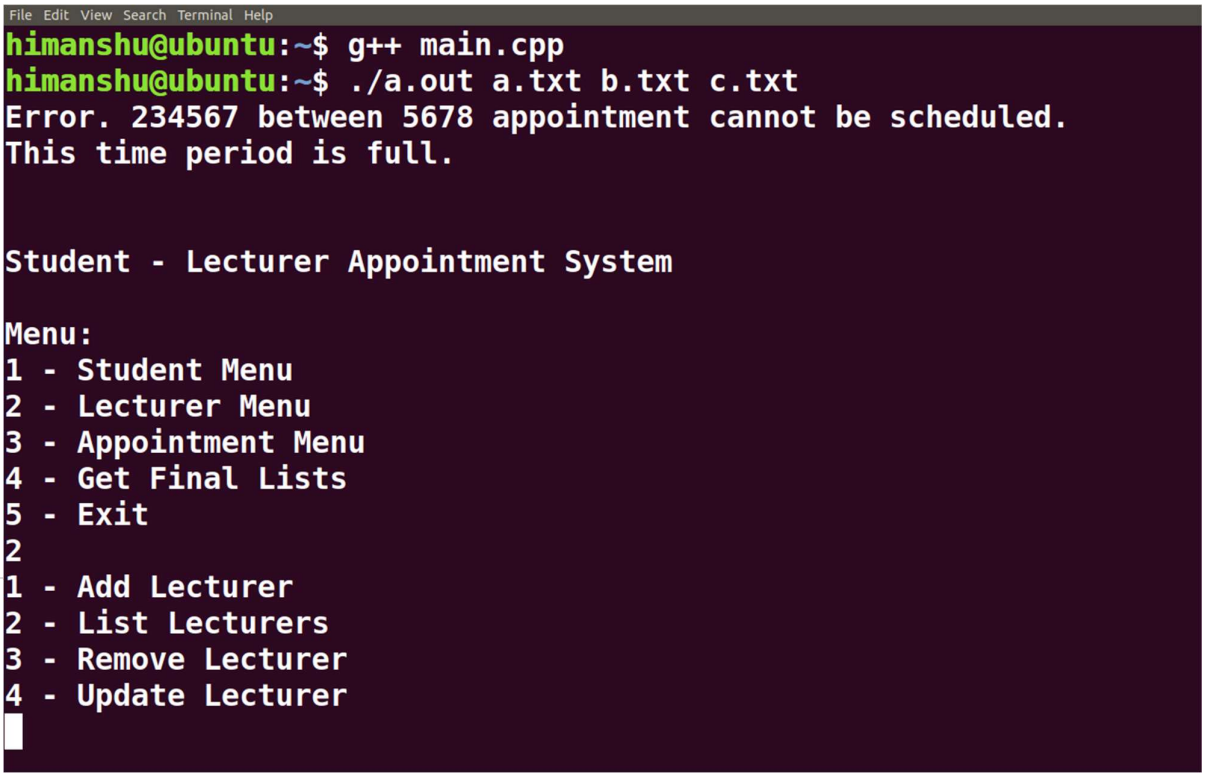
- **Running object file with three files as Command line Arguments**



- **Student Menu**



- **Lecturer Menu**



- **Appointment Menu**