Project 4 – Strings, Vectors, and File I/O with Classes

Overview

In software development projects, it is important to be able to identify objects and classes in a project description. Equally important is the ability to follow a systematic approach to creating a solution to the problem. In this project, you are asked to utilize the design skills discussed in class to develop a solution to a programming problem. This solution should include the use of std::string, std::vector, and file I/O using the fstream library.

Learning Objectives

The focus of this assignment is on the following learning objectives:

- Be able to identify class and objects from a problem description
- Be able to identify class functionality
- Be able to identify an incremental approach to developing the program solution
- Be able to implement the program solution based on the identified approach
- Understand how the program solution can evolve over time
- Understand how to incorporate currently existing libraries to simplify solutions and reduce effort
- Understand the use of std::vector to maintain dynamic structures

Prerequisites

To complete this project, you need to make sure that you have the following:

- C++ and the g++ compiler
- A C++ IDE or text editor (multiple editors exist for developers)
- An understanding of the material presented in class.
- An understanding of the material covered in ZyBooks.

Problem Description

You are to implement a program for scheduling a doctor's daily schedule. For this project, you will need to utilize the concepts of classes and top-down design that has been discussed in class. The list of requirements and constraints for the system are as follows:

1. The system must be able to manage multiple patients and doctors.
2. For each doctor, the system records the doctor's name, age, and a specialty. For each patient, the system records the patient's name, age, and gender. **NOTE:** the files doctors.txt and patients.txt have the list of doctors and patients (and their respective details) that will involved in the scheduling. These files should be read in and used to create Doctors and Patients to be used by your system. The lists of Doctors and Patients should be stored in a dynamic data structure (std::vector).
3. The names of all patients and doctors, which should be stored as std::string, are unique and can be used to locate the information on a specific patient or doctor. This should be accomplished by using methods available in the std::string library for string comparisons.

4. Each doctor has a maximum of eight appointments in a day, each an hour long, with the workday starting at 08:00AM. It is the responsibility of the user to schedule appointments with a doctor.
5. The user can add an appointment by indicating the doctor, the time slot, and the patient. This action fails if the selected time slot is already filled.
6. The user can remove an appointment by indicating the doctor and the time slot that should be cancelled.
7. User can request all patient information to be displayed on screen.
8. User can request all doctor information, including a list of their timeslots and their status (if they are free or the name of the patient that they will see during that appointment), to be displayed on screen.
9. User can request all patient information to be written to a file with a name that is provided by the user.
10. User can request all doctor information, including a list of their timeslots and their status (if they are free or the name of the patient that they will see during that appointment), to be written to a file with a name that is provided by the user.
11. User can enter name of doctor and a time slot to see if that appointment time is available.

Tasks

For this project, you must complete the following tasks in the order that they are described:

1. From the problem description, create a list of all classes that you can identify. For each class, list the associated member variables and identify an initial set of member functions.
2. Provide a detailed account of what modifications were necessary to update this program to satisfy the new requirements.
3. The Doctor and Patient classes should implement a method called "toString()" that uses std::stringstream to build a std::string to return for display (or writing to a file).

Your responses to tasks 1 and 2 should be submitted as a word document called Answers.

Grading Breakdown

- **[10 pts]** Working menu system to take in user input and call functionality
- **[4 pts]** List all patients
- **[4 pts]** List all doctors
- **[5 pts]** Write patient information to file
- **[5 pts]** Write doctor information to file
- **[5 pts]** Write patient information to file
- **[5 pts]** Doctor and Patient classes have a toString() method that returns std::string
- **[8 pts]** Look up patient by name (print info)
- **[10 pts]** Look up doctor by name (print info + schedule)
- **[8 pts]** Adding appointments
- **[8 pts]** Removing appointments
- **[6 pts]** Check for appointments
- **[10 pts]** Sufficient and clear class divisions

- **[8 pts]** Clear, easy-to-read code
- **[5 pts]** Reflection on process

Submission

Points will be deducted for not following these instructions. Before submitting this project in eLearning make sure that you follow the following steps:

1. Make sure that your name appears at the top of each file. This should be done as a comment for any source code files.
2. Copy each snapshot directory and your Responses document into a directory called "Project 2". Do not include any additional files. Zip up this directory into a .zip (points will be deducted for using any other compressed format).

Turn your zipped up project into eLearning.