Project 2 – Classes and Top-down Design

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.

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

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. NOTE: the files DoctorList.h & .cpp and PatientList.h & .cpp have static arrays that have been prepopulated with unique names for the doctors and patients for your convenience. These arrays, since they are static, can be accessed in any source file that includes the .h file.
3. The names of all patients and doctors are unique and can be used to locate the information on a specific patient or doctor.
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 freed.
7. User can print all patient information.
8. User can print 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).
9. 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. List out a set of steps that you will take to implement your solution to the problem. Each step refers to an increment of the program that you will be creating. It is recommended to complete the implementation of a single logical action per step (i.e. a step for listing of doctors/patients, a step for looking up a doctor/patient by name, etc.)
3. Begin implementing your program by using the plan that you created in step 2. For each step, save a snapshot of your program once that step was completed. Each snapshot should be saved to its own directory.
4. Once you have finished implementing your solution, reflect on the process that you followed. Did you wind up with the same classes as you initially identified? Did you need to change any of the functionality or add unexpected details? Did you have to deviate from your plan? Write a description of any details that needed to change as you worked on your solution.

Your responses to tasks 1, 2, and 4 should be submitted as a word document called Answers. Each snapshot should be saved to a directory labelled with a names of the form "Step#" where # refers to the step from your plan. One of these should be your final implementation.

Grading Breakdown

- **[10 pts]** Working menu system to take in user input and call functionality
- **[6 pts]** List all patients
- **[6 pts]** List all doctors
- **[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
- **[15 pts]** Snapshots of previous *working* iterations, including descriptions of changes from previous iteration
- **[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.