Lab 4: More classes and objects

CPSC 1021

Lab Objective

- Practice writing UML class diagrams
- Practice writing classes in C++
- Practice using private member functions
- Practice using in-class member initialization following modern C++ standards
- Practice writing constructors and using constructors for initializing variables with user provided values when an object is created.

Introduction

In this week's lab, we will continue practicing writing classes, with a focus on writing constructors and following modern C++ guidelines for member initialization. You will first revise your area calculation problem from last lab, this time refining each class to perform member initialization. You will next rewrite your solution for the patient charges problem from Lab 2 to use classes and objects.

Before you start on this lab handout, make sure you and your team members have completed the following readings:

- Chapter 7.6 "Constructors" in the textbook and the corresponding lecture slides and code examples
- Document "Notes on Constructors and Member Initialization" uploaded to Canvas.
- Chapter 7.8 "Private Member Functions"

Collaboration Policy

Work with your assigned team members from Lab 0.

You and your team members can work on the analysis and design part of the project together. You can also collaborate on the main program. But each student should individually work on the classes they have been assigned (see further details below).

You and your partners can work on the assignments together.

Submission Policy

Code solutions to all challenges **can be submitted as teams** this week. **This means only one member from each team SHOULD to submit the code files on Gradescope** (this person must then add all their other team members to the submission). However, **you are required to work on all problems together**.

Areas Calculation: In-class Initializers and Constructors

Start with the code you have submitted for the area calculation problem in Lab 3. Revise each class *declaration* (your .h files, not your .cpp files) as follows:

- use in-class member initialization to initialize each member variable with the value 1.0
- add a default constructor
- add a constructor that initializes each variable with a user provided value

Follow the C++ Core Guidelines as discussed in "Notes on Constructors and Member Initialization" uploaded to Canvas when revising your class declarations.

Submission Instructions

Submit the following 10 files:

1. Circle.h
2. Circle.cpp
3. Rectangle.h
4. Rectangle.cpp
5. Square.h
6. Square.cpp
7. Trapezoid.h
8. Trapezoid.cpp

You do not have to write a client program for this submission. The autograder will compile your submission with a test script that will check if each class uses correct member initialization.

Patient Charges

For this problem, you will develop a solution for the patient charges problem from Lab 2, this time using classes and objects. Write a single class 'Patient' that includes the following member variables:

Explanation	Variable Name	Data Type
The number of days spent in the hospital	days	int
The daily room rate	rate	double
Charges for hospital services (food, lab tests, etc.)	services	double
Medication charges	medication	double
If patient was an inpatient ("I) or outpatient ("O")	patientType	char

Your class needs to include the following functions:

- public setter and getter functions for each member variable, named as follows: set*VariableName* and get*VariableName*
- a private member function `bool validateInput(…)` with a single parameter. This function should be called by each setter function to check that the provided value is greater or equal than zero. The function should return `false` if the provided value is invalid, in which case a setter function should leave the corresponding variable at its default value. Overload this function so that it can be used for both integer values and double values.
- a function `double calcTotalCharges(…)` that calculates a patient's total charges

Your class must use in-class member initialization so that each member variable is initialized to 0 or 0.0 or "I" when an object is created. Your class must also include a constructor that allows initializing each variable with a user provided value when an object is created.

Assignment 1: Write the UML class diagram for the Patient class and submit it via Canvas.

Assignment 2: Write a client program that uses your class. You can start with your solution to lab 2 and refactor your code to use the Patient class. Follow the requirements and sample ran from lab 2 when refactoring your code.

Submit your complete and correct program to Gradescope. Your submission must include 4 files:

- hospitalCharges.cpp (this one includes `main()`)
- Patient.cpp
- Patient.h
- Makefile

Make sure your program compiles and runs on the SoC Linux machines; the autograder automatically assigns 0 points to programs that do not compile.

Code style (proper indentation, consistency, readability, use of comments, etc.) will be considered when grading your submission.