**COSC 2436 Programming Fundamentals III - Data Structures**  
**Bob Comer, Professor of Computer Studies**

**Program 2 - Rectangle Class with Interface**

Before you start this assignment, be sure to read through Chapter 1 and C++ Interlude 1 of the textbook. In addition, please read Appendix I C++ Documentation Systems on pages 809 and 810.

This reading assignment introduces several topics that may be new to you, including:

* using a class template to separate the functionality of a class implementation from the type of data used in the class
* inheriting data fields (data members) and methods from a base class
* using an abstract class as a public interface for an abstract data type (ADT)

**Assignment objectives**

In this assignment, you are going to implement a relatively simple Rectangle class that is split into a header file and an implementation file. You will also create an abstract base class (interface) that defines the public interface for your class.

Note that you will **not** use a class template for this assignment. But you will use an abstract class with inheritance.

**Assignment Details**

Create a class called *Rectangle* to represent a rectangle. You will also write a small driver program to test your class. **DO NOT CREATE A CLASS TEMPLATE.** For this assignment, we will just create a class.

In Blackboard, I included example code for a Circle class taken from your COSC 1337 textbook that includes an interface file. Before you start this assignment, you may want to put a copy of the Circle files in a folder. Then create a project, add these files to the project, and be sure that you can compile and run the example code.

**Rectangle Class Members**

Your class should have private data fields (member variables or data members) of type *double* to represent the *length* and *width* of the rectangle.

Your class should have the following public methods (member functions):

* A default (no argument) constructor that sets initializes the length and width of the rectangle to 1.0..
* A method called *set* that allows the client to update the rectangle's length and width. This method will have two parameters that allow the client to provide new values for the length and width. This method has a prerequisite - the new values provided for the length and width must both be larger than 0.0. If they are both larger than zero, the Rectangle object will be updated and the method will return true. Otherwise, the object will not be updated and the method will return false.
* A constant method called *getLength* that returns the length of the rectangle.
* A constant method called *getWidth* that returns the width of the rectangle.
* A constant method called *getArea* that returns the area of the rectangle. The area of a rectangle is the length times the width.

Your method prototypes should be:

Rectangle();

bool set( double newLength, double newWidth );

double getLength() const;

double getWidth() const;

double getArea() const;

If you have not used inheritance and/or abstract classes before, you may want to implement and test your class before you continue to the next step. You must keep your test code separate from your class code, and you must put class definition in a class header file and the implementations for the class methods in a class implementation file.

**Rectangle Interface File**

Create an interface file for your Rectangle class similar to the interface files in Chapter 1 and C++ Interlude 1 in your textbook. Since you are creating a class and not a class template, your implementation file may more closely resemble the example Circle class in Blackboard.

It is important to note:

* Constructor methods are not included in the abstract class.
* You must include a virtual destructor for the abstract class with an empty implementation (see the examples).

Include documentation in your interface file similar to the documentation (comments) in the examples in the textbook and in the example Circle class. You must use the comment formatting described in Appendix I of the textbook.

**Debugging and Testing Tips**

1. If you see linker error messages like:
2. Undefined reference to vtable ...
3. undefined reference to 'interface-destructor-method'

you have a problem with the destructor in the interface file. Check again that this destructor matches the destructors in the example programs. Also, you may have to delete the object (.o) files in your project folder before you try to compile and link your changes.

1. When you are testing a method that contains an if statement, be sure that you try calling the method with data that will make the condition of the if statement true, and also with data that will make the condition false.

**Requirements:**

1. Your program must be split into 4 files: the driver program, the class interface, the class header file (class definition), and class implementation file.
2. The driver program for each programming project should begin with "header comments" containing:  
   - Your name  
   - Programming assignment number  
   - A short description of the program  
   - Course  
   - The date  
   For example:  
   // Programmer:      your name  
   // Project number: 2  
   // Project Desc:     Quadratic Expression class  
   // Course:              COSC 2436 PF III Data Structures  
   // Date:                  date of submission
3. Use a standard indentation convention in your code (see examples in textbook).
4. Use meaningful variable names.
5. Do not use global variables.
6. Your class methods should not contain any input or output operations. All input and output should be done in your driver.

**Programming Assignment Deliverables**

When you have completed and tested your code, submit the following files in Blackboard:

1. your driver (test program)
2. RectangleInterface.h
3. Rectangle.h
4. Rectangle.cpp

**Important:** Only submit files that contain C++ code. Do not send me executable files, project or make files, object code files, etc.

You may want to create a zipped folder containing the 4 files and submit the zipped file.

[Return to C++ Home Page](http://www.austincc.edu/comer/cosc2436/index.htm)

**Copyright: © 2017 by the Austin Community College   
Department of Computer Studies. All rights reserved.   
Comments to:** [**Bob Comer**](mailto:comer@austincc.edu) **Last updated: September 16, 2017**