In-class Programming Project: Polygons
CPSC 298-6 Programming in C++
jbonang@chapman.edu

Introduction

In this project, you'll implement the second of two classes that are derived from an abstract base class, Polygon. The first of these, Rectangle is already defined and implemented. You'll implement a class very similar to Rectangle called Triangle.

The Polygon base class is defined as follows:

```
// ////////////////////////////////////////////////////////////////////////////////////////
// Base Class Polygon - an Abstract Base Class, can't be instantiated
// ////////////////////////////////////////////////////////////////////////////////////////
class Polygon {

public:
    Polygon(double dWidth, double dHeight);
    virtual ~Polygon();
    virtual double area() = 0; // Pure virtual member function - no implementation in base class
    virtual void displayProperties();

protected:
    double m_dWidth;
    double m_dHeight;
    std::string m_strType;
};
```

The area member function is a pure virtual member function; no implementation is provided. This makes Polygon an Abstract Base Class, one that cannot be instantiated (though you can declare pointers to Polygon, as you'll see).

The Assignment

The implementation of Polygon and a class derived from it, Rectangle is provided in source file Polygon_incomplete.cpp.

Follow the TODO comments in the Polygon_incomplete.cpp file.

First, though, build and run the Polygon_incomplete.cpp source file to make sure it's working in your environment.

The output of the Polygon_incomplete.cpp without modifications appears as follows:

```
Polygon Type: Rectangle, Width: 2, Height: 4
Area: 8
Rectangle Destructor called
Polygon Destructor called
```

Now, follow the TODOs in the Polygon_incomplete.cpp source file and implement the Triangle derived class.

The TODOs come in two major groups. The first group involves the implementation of the Triangle class and the second group involved modifications to the main program.

For the first group of TODOs, you'll define the Triangle class and implement its member functions. You can copy the Rectangle implementation for the most part; however, the computation of the area is different.

```
// ////////////////////////////////////////////////////////////////////////////////////////
// Derived Class Triangle ("Concrete" class, can be instantiated)
// ////////////////////////////////////////////////////////////////////////////////////////

/* TODO: Define the Triangle class */


/* TODO: Define the Triangle constructor; make sure to set m_strType to "Triangle" */


/* TODO: Define the Triangle destructor; have it print out "Triangle Destructor called" */


/* TODO: Define the Triangle area member function; Area of a triangle = 0.5 * m_dWdith * m_dHeight */
```

The second group of TODOs occurs in the main program. You'll instantiate the Triangle class and invoke it's methods polymorphically via pointers to Polygon.

```
int main()
{
    Polygon* p_polygonRectangle = new Rectangle(2.0, 4.0);
    /* TODO: Declare a pointer to a Polygon variable (e.g. p_polygonTriangle)
     *       and assign it to a Triangle object allocated using the new operator;
     *       the width of the triangle should be 3.0 and the height 6.0.
     */
    Polygon* p_polygon[2]; // You can create pointers to Polygon, but not a Polygon object

    p_polygon[0] = p_polygonRectangle;
    /* TODO: Assign the triangle object pointer to array element 1 of p_polygon. */

    /* TODO: change "i < 1" to "i < 2" so that the loop iterates over both the Rectangle
     *       and the Triangle.
     */
    for (int i = 0; i < 1; i++)
    {
        // Polymorphic invocation
        p_polygon[i]->displayProperties(); // take advantage of reuse; use base class function
        double dArea = p_polygon[i]->area();
        std::cout << "Area: " << dArea << std::endl;
    }

    delete p_polygonRectangle;
    /* TODO: Delete the allocated Triangle object via the pointer to the Triangle object
     *       that you declared.
     */

    return 0;
```

The output after the Triangle class has been implemented and invoked polymorphically in the main function will appear similar to the following:

```
Polygon Type: Rectangle, Width: 2, Height: 4
Area: 8
Polygon Type: Triangle, Width: 3, Height: 6
Area: 9
Rectangle Destructor called
Polygon Destructor called
Triangle Destructor called
Polygon Destructor called
```