Midterm Programming Project: Inheritance
Topics: Classes, Inheritance, Polymorphism
CPSC-298-6 Programming in C++
jbonang@chapman.edu

Assignment

In this project, you'll implement two C++ classes that represent simple shapes: a rectangle class and a circle class. Both of these classes will be derived from a base class called Shape. All classes will implement functions to scale the shape object, compute its area, compute its perimeter, and display its properties.

Each shape will have three properties: width, height and type. The type is a string and is either "Shape", "Rectangle", or "Circle." The width and height are double precision floating point values.

Each class must also have a constructor and a destructor method. The constructor will be an overloaded constructor that accepts width and height arguments which are assigned to the corresponding member variables.

The base class, Shape, is an abstract base class; not all of its functions will have implementations and it cannot itself be instantiated.

The shape class should be similar to the following:

```
class Shape
{

public:
    Shape(double dHeight, double dWidth);
    virtual ~Shape();
    virtual void scale(double dScaleFactor) = 0;
    virtual double area() = 0;
    virtual double perimeter() = 0;
    virtual void displayProperties();

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

The member functions scale, area and perimeter are pure virtual functions without implementations, as denoted by the = 0 after their prototypes. (This syntax for pure

virtual functions is confusing; C++ should have introduced a new keyword for this instead.)

The Shape class's constructor, destructor and displayProperties member functions must have implementations.

The displayProperties member function uses std::cout to display the shape type string, m_strType, the width and the height:

```
Shape Type: Circle, height: 2, width: 2
```

The member variables of the Shape class must be declared protected so that they are inherited by the derived classes Rectangle and Circle.

You'll derive the Rectangle class and the Circle class from Shape:

```
class Rectangle : public Shape
{
public:
    Rectangle(double dHeight, double dWidth);
    virtual ~Rectangle();
    void scale(double dScaleFactor);
    double area();
    double perimeter();
};
```

```
class Circle : public Shape
{
public:
    Circle(double dHeight, double dWidth);
    virtual ~Circle();
    void scale(double dScaleFactor);
    double area();
    double perimeter();
};
```

The scale(), area() and perimeter() member functions must be implemented for both of these classes, along with the constructor and destructor. Both classes may use the displayProperties member function implemented in the base class.

The perimeter of a rectangle is the sum of its sides (and you have the width and height).

The perimeter of circle, its circumference, is (π * diameter). The width of a circle is equal to its height which equals its diameter.

The area of a rectangle is just (width * height).

The area of a circle is given by 1/4 * π * (Diameter^2). (The diameter is squared.)

To scale a shape, you multiply the shapes width and height by a scale factor.

After you've defined all the functions of your base and derived classes, implement a main program which instantiates those classes using the new operator with the class constructors:

```
int main()
{
    Rectangle* p_shapeRectangle = new Rectangle(2.0, 3.0);
    Circle* p_shapeCircle = new Circle(2.0, 2.0);

    Shape* p_shapes[2];

    p_shapes[0] = p_shapeCircle;
    p_shapes[1] = p_shapeRectangle;
```

Also declare an array of pointers to Shape with two array elements. Assign the first (zero'th) element to the pointer to the Circle shape and the second element, index [1], to the Rectangle shape.

Next, iterate over the elements of the p_shapes array of pointers to shapes using a for loop.

Within the loop, do the following:

- display the properties of the shape (using displayProperties)
- compute the area of the shape and print it out using std::cout
- compute the perimeter of the shape and print it out using std::cout
- Scale the shape by a factor of two
- display the properties of the shape (using displayProperties) (again)
- compute the area of the shape and print it out using std::cout (again)
- compute the perimeter of the shape and print it out using std::cout (again)

The first iteration will display data for the Circle, the second for the Rectangle.

Remember that when you invoke a public member function within the for loop, you will need to use the -> operator:

```
double dArea = p_shapes[i]->area();
```

When you test your program, use the values shown above when instantiating the Circle and Rectangle classes. The Circle should have a width (and thus a height) value of 2.0 (it's diameter). The Rectangle should have a height of 2.0 and a width of 3.0. The output of your program should appear similar to the following screen capture.

```
Shape Type: Circle, height: 2, width: 2
Area: 3.1416, Perimeter: 6.28319
Shape Type: Circle, height: 4, width: 4
Area: 12.5664, Perimeter: 12.5664
Shape Type: Rectangle, height: 2, width: 3
Area: 6, Perimeter: 10
Shape Type: Rectangle, height: 4, width: 6
Area: 24, Perimeter: 20
```

Deallocate the Shape objects at the end of the main program.

Submit your Midterm Programming Project as you would any assignment.