COM S 213 – Fall 2002

ASSIGNMENT #8: Interfaces

DATE GIVEN: 11/2/02 (10/31/02)

DATE DUE: 11/7/02

PURPOSE:
To apply what we’ve learned about Interfaces and Polymorphism

ASSIGNMENT:
You will need to define a new interface for Shapes. As a matter of fact, call the interface Shapes. It will define the following API calls:

- `double computerArea()`
- `void draw()`
- `void setOrigin(int x, int y)`

Now, make sure you declare those api calls in a manner consistent with interfaces as described in class.

Once you have the interface defined, you should derive three classes from Shape:

- Square
- Circle
- Rectangle

Each should implement the interface and should have the following routines specific to that class:

**SQUARE**
- `Square(x, y, w)` 
  *constructor which takes x, y, width*
- `SetWidth(w)` 
  *member function which allows width to be set*

**CIRCLE**
- `Circle(x, y, r)` 
  *constructor which takes x, y and radius*
- `SetRadius()` 
  *member function which allows the radius to be set*
Note: You may implement the derived classes totally in their respective header files to avoid a plethora of files needing to be turned in.

For implementing the `draw()` function in each shape, simply print out the origin and dimensions to `cout` (for extra credit, try using text characters to make a graphical representation of the shape that is proportional to the dimensions of the object)

Now, to test your implementation you should run the following code in your main() functions:

```cpp
Shape *shapeArray[3];

ShapeArray[0] = new Square(3,4,5);
ShapeArray[1] = new Circle(4,4,8);
ShapeArray[2] = new Rectangle(6,7,10,20);

for (int j=0; j<2; j++)
{
    shapes[j]->draw();
    shapes[j]->setOrigin(j*2,j*3);
    shapes[j]->draw();
    cout << "Area of shape " << j+1 << " is " <<
         shapes[j]->computeArea() << endl;
    cout << endl << endl;
}
```

**SUGGESTIONS**
This one is fairly straightforward, email me with questions.