

CS-481
C++ Programming

Composition, Inheritance, & Polymorphism

Date: Tue. February 9, 1999

Due: Thrs. February 18, 1999

So far, we have seen how *Composition*, *Inheritance*, and *Polymorphism* can work together to create a rich environment for describing a set of geometric shapes. We've written code for `Points`, `Lines`, and `FancyLines`. Further extensions can be made to these base classes. `Circles`, `Triangles`, `Rectangles`, and other geometric shapes could be added to round out the complement of shapes to our library.

Using the examples from class, extend the "Shapes" library to include these new geometric shapes. Demonstrate the use of *Composition*, *Inheritance*, and *Polymorphism* in your implementation. For each new class, consider providing a *constructor*, *destructor*, *copy constructor*, and *assignment operator*. The base class for all the objects will be an *abstract* class named `Shape`.

Test your graphical objects via implementing a text-based `draw()` function for each type of object. The object doesn't need to physically draw itself, just produce a text message saying that the function has received the draw command and is responding by drawing lines from point to point.

Once the basics are implemented, consider adding a *container* for your shapes, and sequencing through the container using an *iterator*. The container may encapsulate a simple array or a dynamic array (a la the `Stack` class, given earlier).

As always, turn in your *source code* and *output* generated from your test cases to verify the operation of your program.