COMSC-200 Lab 12
Using Polymorphism

GOOD PROGRAMMING PRACTICES *show / hide*

ABOUT THIS ASSIGNMENT
In this lab session, we continue with the **GeometryHomework** solution from labs 1-4, so that it uses inheritance. You will also practice programming with classes by doing exercises from our textbook.

As you complete this assignment, post the required files to the COMSC server. You may post them all at once, or one-at-a-time as you complete them, as you prefer. The individual labs are graded in order, starting with "a", which must be fully correct before "b" is graded, and so on. This assignment is worth 50-points, to be awarded after all labs are successfully completed. Use the "Submit your SP2015 work" link on the class website to post your file for this lab, using the **lab12** folder provided.

Lab 12 Using Polymorphism [`GeometryHomework6.cpp`]
Make structural changes to the GeometryHomework program from lab 11, applying polymorphism in order to greatly simplify the coding instructions in the `main` function. These include:

1. Include an abstract class, **Shape**, and use it as the polymorphic parent of each of the three 2-dimensional shapes: circle, square, and rectangle. The 3-dimensional shapes should continue to derive from the 2-dimensional shapes, so that there are three generations of inheritance in use.

2. Declare three *generic* dimension variables in the **Shape** class, as `const` data members. Use these to store up to three dimensions for any object. *Remove the private data members from the six derived classes, which were originally used to store dimensions.* Yes, Square objects will have 3 inherited dimension values, but only one will get used, and the other two ignored.

3. The six shape classes should now include only public members, so there is no longer a need for a private section. So for these 6 classes, change the keyword `class` to `struct`, and lose the `public:` designation inside of it.

4. Declare this function in **Shape**: `virtual void output(ostream&) const = 0;`. This should match the output functions that already exist in the derived classes. Include an empty virtual destructor, too.

4a. [OPTIONAL] Add an overloaded stream insertion operator for the abstract base class, which calls the **output** function. Use the stream insertion operator for outputting in **main**. Since these call a public getter, there is no need for them to access private members directly, so no friend relationships are needed for them. Since **output** is polymorphic, the correct derived-class' version of **output** will be invoked!

5. Use a vector of `const Shape*`'s, initially empty. Replace the detailed code structures for outputting objects and deleting memory in **main** with simple traversing `for` loops. You can use either index-based or iterator-based loops per your choice. In either case, the `switch` structure should no longer appear. Also, remove the shape ID array, and all references to shape ID codes -- *these are no longer necessary.*

Vector iterators: You can use either `operator[]` and the `size()` function, or you can use "iterators". Declare it something like this: `vector<const Shape*>::iterator it;`. Remember that `it` returns *pointer to a pointer* -- you have to dereference it to get a `const Shape*` like this: `*it`, which dynmically

casts to the proper derived class pointer. For dynamic binding to work, do something like this: `(*it)->output();`.

Do *not* use more than one source file -- that means no **.h** files, and just a single **.cpp**. *Post* ***GeometryHomework6.cpp*** *to the COMSC server for credit.* Save this file, because modifications will be made to this program in lab 13. The GeometryHomework series is the subject of seven (7) lab assignments.

How to pause a console program: *show / hide*

GRADING POLICY *show/hide*
