SYDE 121 – Digital Computation
Lab Number 10

Exercise 1: Creating and Using the Shapes Class

Learning Objectives: Practice creating a class. Practice with inheritance. Practice redesigning and reimplementing functional code into object-oriented code.

Read This First

For this lab you will be reusing some of the code (possibly re-writing, if needed) from Lab #5, and an old lab for this course. In Lab #5, you were asked to create a program that output a solid diamond. In previous years, SYDE 121 students were asked to output a hollow square (see the posted solution code, called square.cpp). In practice, there are many different shapes that could be output, given that same type of input from the user (e.g., a numeric "size"), for example, a square, rectangle, triangle, diamond, pyramid, etc.. The user could also indicate whether the shape should be hollow, filled or, striped. Since much of the same type of code will be used to do the outputting for each type of shape, it makes sense to create a "shape" class that can do much of the basic work needed, and create several derived classes (i.e. inherited classes) for each of the specific shapes (i.e. a Triangle class inherited from the base Shape class.

What to Do

Create a Dev-Cpp project called Shapes. Set up your project using an interface file (shape.h), an implementation file (shape.cpp), and the main file (using_shapes.cpp).

The shape.h and shape.cpp will be used to declare and define the base class Shape and your derived classes (described below). Review Lab #7 for proper construction and use of the separate interface and implementation files, including how to properly "wrap" header files in compiler directives. The following enum should be included in the shape.h interface file to be available both to your main and to all of your classes and member functions.

```
enum Shape_Type {SQUARE, DIAMOND, PYRAMID, TRIANGLE};
```

Download the *main.cpp* file from the Lab 10 LEARN dropbox. This file will serve as the basis for your *using_shapes.cpp* file. You will need to add your own documentation, and any additional functionality as required to meet the lab criteria.

Your Shape class should contain the following member data variables:

- `shape_type` of type `Shape_Type`
- `shape_size` of type `int`
- `isFilled` of type `bool`

The Shape class should also contain the following member functions:

- a default constructor that initializes the `shape_type` member variable to square (i.e. an object of type Shape will be automatically defined as a square).
- an overloaded constructor that has one `Shape_Type` parameter and initializes the `shape_type` member variable to the passed in argument.
- A `defineSize` member function that prompts the user to enter the size of the shape (as Shape is by default a square, it should ask for a positive even number between 0 and 20). The member variable `shape_size` should be assigned the entered value.
- A `defineFill` member function that prompts the user to enter a fill type for printing the shape (0 = hollow, 1 = solid fill). Depending on the entered value, the member variable `isFilled` should be set to true or false (true means filled, false means solid).
- A `getSize` accessor member function that returns the value of the `shape_size` member variable.
- A `setSize` mutator member function that sets the value of `shape_size` member variable to the passed in integer argument.
- A `printFilledShape` member function that has one `ofstream` parameter `out_stream`, and prints the shape to the `out_stream` using solid fill (as Shape is by default a square, it should print out a square of `shape_size` in solid fill).
- A `printHollowShape` member function that has one `ofstream` parameter `out_stream`, and prints the shape to the `out_stream`

using hollow fill (as Shape is by default a square, it should print out a square of `shape_size` with hollow fill).

- A `getIsFilled` member function that returns the current `bool` value of the member variable `isFilled`.

You must decide which variables and which functions should be defined as private or public, based on good abstract data type programming practices.

Once the Shape class is defined, the four specific shape classes, Square, Diamond, Pyramid, Triangle, should be defined. Each class should be defined as a child class of the parent (or base) class Shape. Each of these derived classes should redefine the `defineSize`, `printFilledShape`, and `printHollowShape` to work more specifically for their given shape. For instance, since Shape's `defineSize` member function currently asks the user to enter an even number between 0 and 20 this would not be appropriate for the Diamond shape. Thus, Diamond's `defineSize` member function should ask the user for an odd number between 1 and 19. Similarly, each shape will need a different set of commands to output the shape properly; thus, two new print member functions are needed for each specific shape class.

Given these requirements, the Square class declaration, for example, would look as follows:

```
class Square : public Shape {

public:

   Square( void );  // default constructor

   void defineSize();

   void printFilledShape( ofstream& out_stream );

   void printHollowShape( ofstream& out_stream );

};
```

Table 1 illustrates how each of the different shape type, Square, Diamond, Pyramid, and Triangle, should be displayed (i.e. formatted) in the output file.

Create a "*README_shapes.txt"* file, and create detailed instructions on how to run your *using_shapes.cpp* program, including what input is expected, and what output will be produced (i.e. file name and contents of the file).

Table 1. Each of the four different shapes, Square, Diamond, Pyramid, and Triangle, should be displayed as shown below in the output file, for the two different fill types.

Filled Square of size 6:	Hollow Square of size 6:

`******`
`******`
`******`
`******`
`******`
`******`
	`******`
	`*    *`
	`*    *`
	`*    *`
	`*    *`
	`******`

Filled Diamond of size 5:	Hollow Diamond of size 5:

`  *`
`***`
`*****`
`***`
`  *`
	`  *`
	`* *`
	`*   *`
	`* *`
	`  *`

Filled Pyramid of size 7:	Hollow Pyramid of size 7:

`   *`
`  ***`
`*****`
`*******`
	`   *`
	`  * *`
	`*   *`
	`*******`

Filled Triangle of size 8:	Hollow Triangle of size 8:

`       *`
`      **`
`     ***`
`    ****`
`   *****`
`  ******`
` *******`
`********`
	`       *`
	`      **`
	`     * *`
	`    *  *`
	`   *   *`
	`  *    *`
	` *     *`
	`********`

What to submit

Submit your three source files *using_shapes.cpp*, *shape.cpp*, *shape.h* and *README_shapes.txt* files to the Lab Assignment 10 LEARN dropbox.

Due Date

All materials for this lab are due **Friday, November 28 by 1:30pm.** Remember to give yourself plenty of time before the due date to complete the lab.