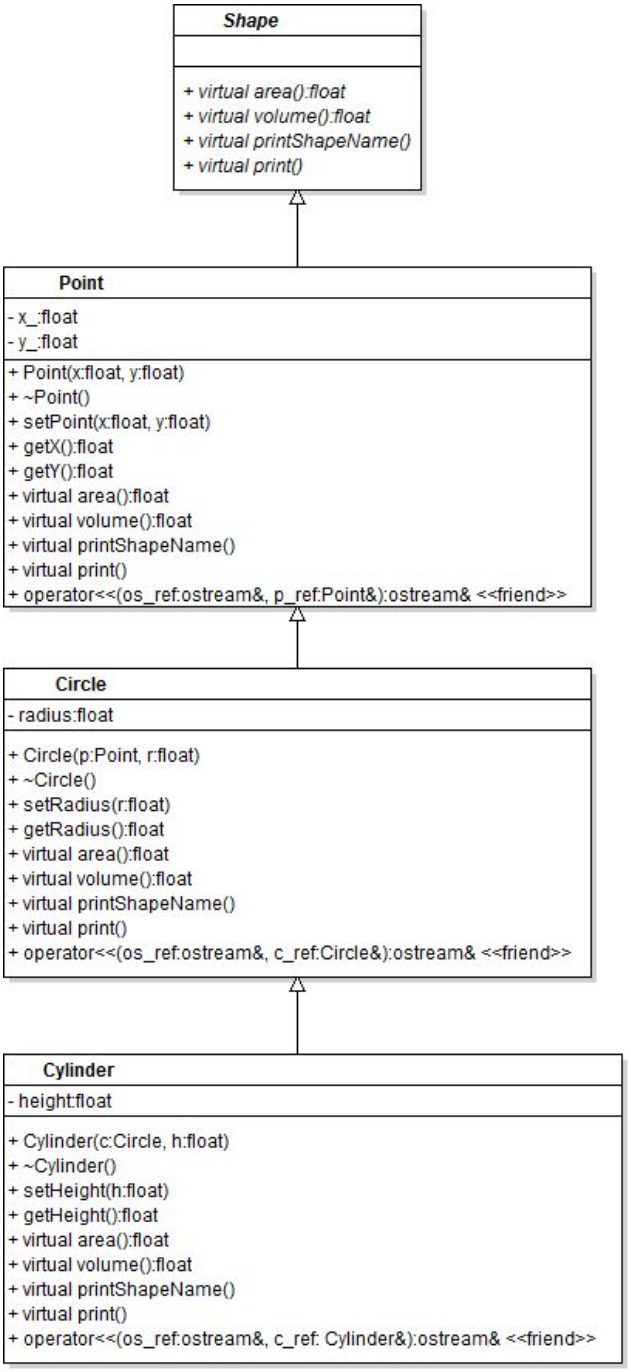
Computing 2 - Labs

Lab 9: Polymorphism among Shapes

You are given the UML diagram shown below.



The header files and the main function are predefined. Their task is the implementation of the associated class functions in the corresponding cpp files. The functions `getX()`, `getY()`, `setPoint(float a, float b)`, `setRadius(float)`, `getRadius()`, `setHeight(float)`, `getHeight()`, `printShapeName()` are already implemented.

Tasks

1. Implement the constructor for the different classes. When you program the constructor for `Cylinder`, remember that `Cylinder` inherits from `Circle` and `Circle` is a point.
2. The purely virtual functions `area(), volume()` are already implemented for `Point`.For the class `Circle`, you must redefine the `area()` function so that it calculates the area of a circle. The function `volume()` is already implemented.
   For the three-dimensional class `Cylinder` you have to redefine both the function `area()` and `volume()` to calculate the surface or volume of a cylinder.
3. Now implement the `print()` function, which prints the attributes (class data elements) of each object.
   For a point this means that the x and y coordinates should be printed.
   For a circle, this means that the x-coordinate, the y-coordinate and the radius are to be output.
   For a cylinder, that the x-coordinate, the y-coordinate, the radius and the height are output.
   For the implementation, use the print functions of the inheriting class to avoid duplication of code
4. In addition, for the classes `Point, Circle and Cylinder` an overloaded stream insertion operator (<<) must be implemented.
   For a point the following output is to be made:
   "`(`"followed by the x value", "followed by the y value"`)`"
   For a circle the following output is to be made:
   "`Center = (`"followed by the x-value", "followed by the y-value"`); Radius =` "followed by the radius-value
   For one cylinder the following output is to be made:
   "`Center = (`“ followed by the x-value, followed by the y-value)”`); Radius =` " followed by the radius-value"`; Height =` " followed by the height-value
   For formatting, it is useful to include the two files iostream, iomanip, and use the setiosflags(ios::showpoint) and setprecision(2) functions.
   Look at the code snip in `main.cpp` line 20-39 to see how the overloaded stream insertion operators are used. Test your function and you should get the output shown below:

```
Shape objects set up using default constructors
Point: (0.00,  0.00)
Circle: Center = (0.00,  0.00); Radius = 0.00
Cylinder: Center = (0.00,  0.00); Radius = 0.00; Height = 0.00


Shape objects set up with user-defined parameters
Point: (7.00,  11.00)
Circle: Center = (7.00,  11.00); Radius = 3.50
Cylinder: Center = (7.00,  11.00); Radius = 3.50; Height = 10.00
```

5. In the `Main` function, insert the missing code for the for loop. In this for loop, you want to use a vector of base class pointers to objects of each concrete class in the hierarchy. The pointer should therefore point to one object each of the classes Point, Circle, and Cylinder. As part of the for loop, the program should use the `printShapeName()` element function to output the name of the object shape to which each base class pointer of the vector points. Then the attributes of each object should be output with the print function. Finally, the area and volume of each object shall be calculated and printed. The desired output is shown on the next page

```
Shape objects set up using default constructors
Point: (0.00,  0.00)
Circle: Center = (0.00,  0.00); Radius = 0.00
Cylinder: Center = (0.00,  0.00); Radius = 0.00; Height = 0.00


Shape objects set up with user-defined parameters
Point: (7.00,  11.00)
Circle: Center = (7.00,  11.00); Radius = 3.50
Cylinder: Center = (7.00,  11.00); Radius = 3.50; Height = 10.00
```

Finally implement a `for` loop in the driver program that uses a vector of `Shape` pointers to objects of each concrete class in the hierarchy, i.e. to `Point`, `Circle` and `Cylinder`. The pointers and the `for` loop are shown below.

```
Shape* arrayOfShapes[3];    // array of base-class pointers
                            // base class pointers are needed for polymorphism

   // aim arrayOfShapes[0] at derived class Point object
   // aim arrayOfShapes[1] at derived class Circle object
   // aim arrayOfShapes[2] at derived class Cylinder object

   // Loop through arrayOfShapes and print the shape name,
   // area, and volume of each object to which the array points
   // using dynamic binding.
   for (int i = 0; i < 3; i++) {


     // your code here

   }
```

As part of the for loop, the program should print the name of the object's shape to which each vector element points using the function `printShapeName`. Then it should print the attributes of each object using the `print` function. Finally, it should print out the Area and Volume of each object. The desired output is shown below:

```
Point:
(7.00,  11.00)
Area = 0.00
Volume =0.00

Circle:
Center = (7.00,  11.00); Radius = 3.50
Area = 38.48
Volume =0.00

Cylinder:
Center = (7.00,  11.00); Radius = 3.50; Height = 10.00
Area = 296.88
Volume =384.84
```