FA17 MP12 - Classes and Inheritance

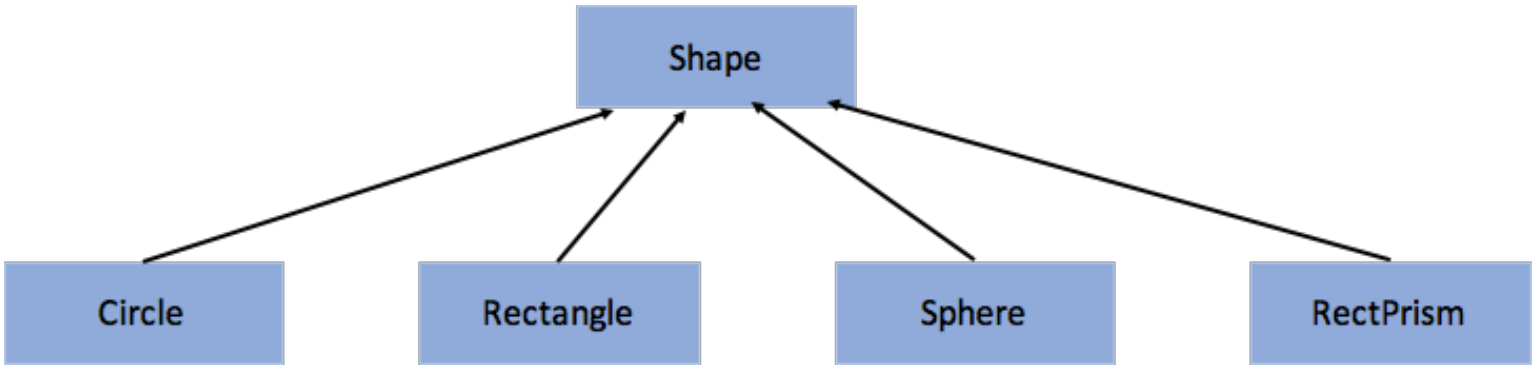
Created by Liu, Iou-Jen, last modified yesterday at 10:06 PM

Due: Thursday, December 7th, by 10:00pm

*** Please work on EWS for this MP to avoid compilation error on verification script.**

Introduction

Your task in MP12 is to implement the *Shape* hierarchy shown here:



Shape class is the base class. Circle, Rectangle, Sphere, RectPrism are derived classes. The base class contains function getName() which return the name (Circle, Rectangle, Sphere, RectPrism) of an object. For each derived classes, you need to implement getArea(), getVolume(), and overloading operator "+" and operator "-". Your program should read the shapes defined in test.txt, and create a vector of *Shape* pointers that points to objects of each input shape. Different constructor should be called according to the input shape. For example, if the input shape is a rectangle, class *Rectangle*'s constructor should be called to initialize the length, width, and name of the rectangle object. After reading the file, given an array of shape pointers, you have to implement MaxArea() and MaxVolume() which return the max area and max volume respectively.

The format of the test cases is as follows:

```
<#objects>
<name0> <var1> <var2> ..
<name1> <var1>…
```

where <#objects> gives the number of objects defined in the test case. Starting from the second line, each line defines a shape object. <name> gives the name of the object. <var1> <var2>…. are the parameter required to initialize an object. For circle and sphere, the only parameter is radius. For rectangle, the parameters are <width> <length>. For rectangle prism, the parameters are <width> <length> <height>.

Below is a sample input-output pair.

Input:

```
4
Circle 2
Rectangle 2 3
Sphere 2
RectPrism 1 2 3
```

Output:

```
max area = 50.2655
max volume = 33.5103
```

The Pieces

In this MP, you are given a set of files:

main.cpp - The source file that contains the main function.

shape.hpp - The header file of the shape hierarchy.

shape.cpp – The source file of the shape hierarchy. **You should write your code here.**

verify.cpp - The source file of the verification program.

check.hpp – The header file of the verification program.

Details

Shape hierarchy

The header file of the shape hierarchy is given. Your task is to implement the member functions of the class.

```
//shape.hpp
class Shape{
public:
  Shape(string name, int id);
  string getName();
  virtual double getArea() = 0;
  virtual double getVolume() = 0;
private:
  string name_;
};
```

Read Input File

You have to implement CreateShapes() to read the input file and initialize corresponding objects. CreateShapes() return a vector of pointers (vector<Shape*>) point to the objects initialized according to the input data.

```
vector<Shape*> CreateShapes(char* file_name);
```

For example, to initialize a Circle object with radius 2, you may use the following codes:

```
Shape* shape_ptr = new Circle(2);
```

To read the input file, you can use *ifstream.* You can find more information about *ifstream* on the website(http://www.cplusplus.com/reference/fstream/ifstream/). An example of using *ifstream* to read input file is as follows:

```
//test.txt
circle 2
```

```
//in main()
String name
int r
ifstream ifs ("test.txt", std::ifstream::in);
ifs >> name >> r;
ifs.close();
```

By executing the code piece above, "circle" is stored in *name* and 2 is stored in *r*.

MaxVolume() and MaxArea()

Given a vector of object pointers, you have to implement MaxArea() and MaxVolume(). MaxArea() and MAxVolume() compute each objects area and volume, and return the maximum area and maximum volume respectively.

```
double MaxArea(vector<Shape*> shapes);
double MaxVolume(vector<Shape*> shapes);
```

Operator Overloading

For the purposes of this MP, we have defined the addition and subtraction of 2 Rectangles/Circles/Spheres/RectPrisms to be as follows:

Rectangles
Given **R3 = R1 + R2**: it follows that
length R3 = length R1 + length R2
width R3 = width R1 + width R2
Given **R3 = R1 - R2**: it follows that
length R3 = max(0, length R1 - length R2)
width R3 = max(0, width R1 - width R2)
Circles
Given **C3 = C1 + C2**: it follows that
radius C3 = radius C1 + radius C2
Given **C3 = C1 - C2**: it follows that
radius C3 = max(0, radius C1 - radius C2)
Rectangular Prisms
Given **RP3 = RP1 + RP2**: it follows that
length RP3 = length RP1 + length RP2
width RP3 = width RP1 + width RP2
height RP3 = height RP1 + height RP2

Given **RP3 = RP1 - RP2**: it follows that
length RP3 = max(0, length RP1 - length RP2)
width RP3 = max(0, width RP1 - width RP2)
height RP3 = max(0, height RP1 - height RP2)
Sphere
Given **S3 = S1 + S2**: it follows that
radius S3 = radius S1 + radius S2
Given **S3 = S1 - S2**: it follows that
radius S3 = max(0, radius S1 - radius S2)

Computing Area and Volume

The fomulas for computing areas and volumes are listed as below for your reference.

Rectangle R

area of R = length * width

volume of R = 0

Circle C

area of C = radius^2 * PI

volume of C = 0

Rectangular Prism RP

area of RP = 2 * (l ength * width + length * height + width * height)

volume of RP = length * width * height

Sphere S

area of S = 4 * PI * radius^2

Building and Testing

To compile your program, type the following command:

```
make
```

To execute your program, type :

```
./mp12 test1.txt
```

After you finish your implementation, you could verify your program by using a provided test program. To Verify your program, use the following command

```
make verify
./verify_mp12 test1.txt
```

The program will invoke a hidden checker to test your program and display the verification results.

```
------------------- Begin Verifying MP12 ---------------------
getName() 16/16
Rectangle: 16/16
Circle: 16/16
Sphere: 16/16
RectPrism: 16/16
MaxArea(): 10/10
MaxVolume(): 10/10
Your total Score for MP12: 100/100
--------------------- End Verifying MP12 ----------------------
```

Grading Rubric

- *Functionality (100%)*
  - getName() (16%)
  - Class Rectangle (16%)
    - 4% for each function
  - class Circle (16%)
    - 4% for each function
  - class Sphere (16%)
    - 4% for each function
  - class RectPrism (16%)
    - 4% for each function
  - MaxArea() (10%)
  - MaxVolume() (10%)
- **If your code does not compile, you will get 0.**

No labels