Computer Science 312
Principles of Programming Languages
Fall 2022
Assignment 6

Due: 11:59 p.m., Tuesday, 12/6

Overview

For this assignment, you will write a program in C++ that works with shapes and exhibits the four main features of objected-oriented programming: encapsulation, inheritance, polymorphism, and dynamic method binding.

Description

The base class for the three basic 3D shapes (sphere, box, and cone) is named **`Shape`**. Its data and methods are shown below and should be followed *exactly.* All data members in classes should be assigned to the most restrictive access mode (**`public`**, **`protected`**, or **`private`**) possible.

data: **`type (string), loc (Point), color (string),`** and **`next (Shape *)`**
- **`color`** must be accessible in derived classes, **`next`** must be accessible by any function

`Shape (string type, string color, Point loc);`
- constructor that sets the parameters given to the class variables

`void print_color (void);`
- must be declared as **`virtual`** so that derived classes may override it; by default, it prints a label, **`Color:`** and then the string stored in **`color`** (use **`cout`**)

`double compute_volume (void);`
- must be declared as pure **`virtual`** – derived classes are therefore forced to provide an implementation for it

`void print_type (void);`
`void print_loc (void);`
- declared as regular class methods that print out the shape type and loc(ation), respectively (use **`cout`** and see output for format)

The **`Sphere`**, **`Box`**, and **`Cone`** classes inherit from **`Shape`** and add their own data and methods. The code for the **`Cone`** class is provided on the webpage – please use it as a template for creating the **`Sphere`** and **`Box`** classes. Each class **must** provide an implementation for **`compute_volume`**, and **may** provide an implementation for **`print_color`** if it prints out the object's color(s) differently than the default in the **`Shape`** class.

`Sphere` data: **`center (Point), radius (double)`**

`Sphere (string type, string color, Point center, double radius);`
- its location is its **`center`**

Box data: **length (double), width (double), height (double), tbcolor (string)**
- **tbcolor** is the top and bottom color; the side color should be stored in **color** (from **Shape**)

```
Box (string type, string color, string tbcolor, double length,
       double width, double height, Point loc);
```
- its location is **loc**

Volume equations:
- Sphere: $\frac{4}{3}\pi r^3$
- Box: $length * width * height$

The **Point** class is off to itself and is used to store 3D points (**x**, **y**, and **z**). It has the following members:

Point data: **x (double), y (double), and z (double)**

```
Point () {}
Point (double x, double y, double z) { set (x, y, z); }
```
- two constructors that are implemented in the header file as shown

```
void set (double x, double y, double z)
```
- make this method **inline** and provide its very simple implementation in the header file

```
double length ();
```
- computes and returns the length of a point vector: **sqrt (x^2 + y^2 + z^2)**
- used in **Cone::compute_volume**
- **#include <cmath>** for **sqrt**

```
Point operator- (Point& p);
```
- overloaded operator to compute ***this – p**
- reference parameter for efficiency
- used in **Cone::compute_volume**

```
void print (void);
```
- prints the **Point** as shown in the output (use **cout**)

The **main** function should appear as follows:

```
int main ()

(
   Shape *list;

   read_objs (&list);

   print_objs (list);

   // add loop here to return any allocated space to the system

   return (0);
}
```

The functions shown above appear in main.cpp also:

`void read_objs (Shape **list)`

- most of the code is linked from the webpage
- you must fill in the reads from **`std`** input using **`cin`**
- you must call **`new`** to create nodes to link in the **`list`**
- you must complete the code to initialize the **`list`** and link the node to the beginning of the **`list`**

`void print_objs (Shape *list)`

- declare a local variable of type **`Shape*`** to traverse the linked **`list`**
- for each element in the list, use the methods in **`Shape`** and its subclasses to print the type, color, and loc (note that the appropriate method will be called in **`Shape`** or one of its subclasses due to dynamic binding)
- use **`cout`** to print the final line, **`Volume:`** and the volume computed by **`compute_volume`**

Compiling and Executing the Code

To compile all of the C++ files in your directory, you can type:

```
g++ *.cpp –o shape
```

and then run the code using

```
./shape < objs.txt
```

The **`objs.txt`** file is provided on the webpage. You can check the output of your program by comparing it to the provided output on the webpage.

Submitting the Code

Submit the following files through Blackboard:

```
box.h
box.cpp
sphere.h
sphere.cpp
point.h
point.cpp
shape.h
shape.cpp
main.cpp
```