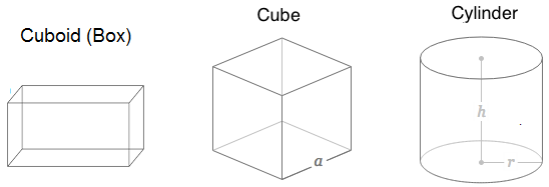
Advanced C++ Programming

Exercise

Suppose we want to calculate total weight of an array of solid objects of different types and densities (could be a box, a cube or a cylinder).



You task is to create a class hierarchy by extending AbstractObject class, implement necessary data members/constructors/operations for each subclass.

Complete source code:
https://campus.cs.le.ac.uk/teaching/resources/CO7105/Surgeries/Surgery4/AbstractObject.zip

The example on Animals used in surgery 4
https://campus.cs.le.ac.uk/teaching/resources/CO7105/Surgeries/Surgery4/Animal.zip

`AbstractObject.h`

```
#include<iostream>
#include<string>

class AbstractObject{
public:
  AbstractObject();
  virtual ~AbstractObject();
  void setDensity(double);
  double getDensity() const;
  virtual double getVolume() const=0 ;
  double getWeight() const;
  friend std::ostream&
    operator<<(std::ostream& os, const AbstractObject& c);
protected:
        std::string objectName_;
        double density_;
};
```

main.cpp

```
#include<iostream>
#include "AbstractObject.h"

constexpr int ITEM_NUMS=3;

int main(){

    Box b(1.0,2.0,3.0,2);
//a=1,b=2,c=3, density=2
    Cube c(1.0,2);
//a=1, density=2
    Cylinder cl(2,1,3.0);
//height=1,radius=1,density=3

AbstractObject **list= new AbstractObject*[ITEM_NUMS];

        list[0]=&b;
        list[1]=&c;
        list[2]=&cl;

    double total_weight=0;

for(int i=0;i<ITEM_NUMS;i++){
cout<<"adding"<<*list[i]<<endl;
total_weight+=list[i]->getWeight();
}

cout<<"total_weight="<<total_weight<< endl;
}
```

Solution

AbstractObject.h

```
#include<iostream>
#include<string>

class AbstractObject{
    public:
         AbstractObject();
         virtual ~AbstractObject();
         void setDensity(double);
         double getDensity() const;
         virtual double getVolume() const=0 ;
         double getWeight() const;

        friend std::ostream&
          operator<<(std::ostream& os, const AbstractObject& c);

    protected:
        std::string objectName_;
        double density_;
};

class Box:public AbstractObject{
    public:
       Box(double x,double y,double z, double density);
        virtual ~Box();
        double getVolume() const;
    protected:
        double x_;
        double y_;
        double z_;
};

class Cube:public Box{
    public:
```

```
        Cube(double w,double density);
         virtual~Cube();
};

class Cylinder:public AbstractObject{
    public:
         Cylinder(double height,double radius,double density);
         virtual ~Cylinder();
         double getVolume() const;
    protected:
         double height_;
         double radius_;
};
```

AbstractObject.cpp

```
#include <iostream>
#include "AbstractObject.h"

constexpr double PI=3.14159265;

std::ostream& operator<<(std::ostream& os, const AbstractObject& c){
    os<<c.objectName_<<" weight="<<c.getWeight();
    return os;
}

AbstractObject::AbstractObject()
 :objectName_{"abstract object"},density_{0}{
    std::cout<<"abstract object constructor"<<std::endl;
}
AbstractObject::~AbstractObject(){
    std::cout<<"abstract object destructor"<<std::endl;
}

void AbstractObject::setDensity(double d){
    density_=d;
}

double AbstractObject::getDensity() const{
    return density_;
}

double AbstractObject::getWeight() const{
    return getVolume()*getDensity();
```

```
}

Box::Box(double x,double y, double z, double density)
 :x_{x},y_{y},z_{z}{
      setDensity(density);
      objectName_="Box";
      std::cout<<"Box constructor"<<std::endl;
}

Box::~Box(){
     std::cout<<"Box destructor"<<std::endl;
}

double Box::getVolume() const{
    return x_*y_*z_;
}

Cube::Cube(double w,double density):Box(w,w,w,density){
    objectName_="Cube";
     std::cout<<"Cube constructor"<<std::endl;
}

Cube::~Cube(){
     std::cout<<"Cube destructor"<<std::endl;
}

Cylinder::Cylinder(double height,double radius,double density)
:height_{height},radius_{radius}{
    setDensity(density);
      objectName_="Cylinder";
     std::cout<<"Cylinder constructor"<<std::endl;
}
Cylinder::~Cylinder(){
    std::cout<<"Cylinder destructor"<<std::endl;
}
double Cylinder::getVolume() const{
    return PI*radius_*radius_*height_;
}
```