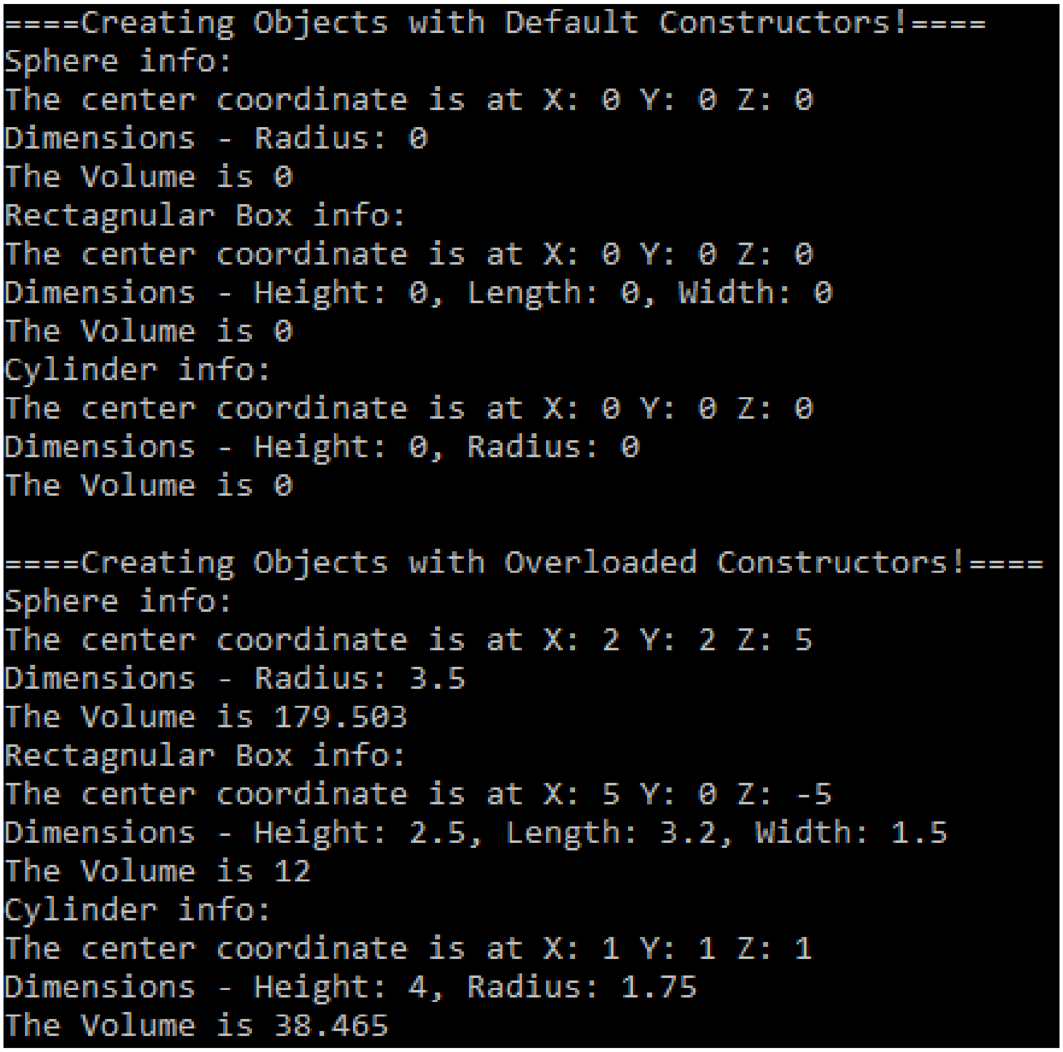
```cpp
/*
Daniel Avila April 22nd, 2020 Section 19
Lab 11: Inheritance
Description: To use multiple classes derived from a base class to be able
Description: to access its functions as a template
*/
#include <iostream>
#include "Cylinder.h"//To be able to create Cylindrical Object from
#include "RectangularBox.h"//To be able to create a Rectangular Box Object
#include "Sphere.h"//To be able to a create Spherical Object
using namespace std;

int main()
{
    cout << "====Creating Objects with Default Constructors!====" << endl;
    cout << "Sphere info:" << endl;
    shared_ptr<Sphere> sphere = make_shared<Sphere>();//Pointer to create a default sphere object-no parameters
    cout << "The center coordinate is at X: " << sphere->getX() << " Y: " << sphere->getY() << " Z: " << sphere->getZ() << endl;//Inherits the functions from the base class to call the default coordinates-values of 0
    cout << "Dimensions - Radius: " << sphere->getRadius() << endl;//Gets the default radius from the sphere class-value of 0
    cout << "The Volume is " << sphere->calcVolume() << endl;//Calculates the unique volume because of the specific formula for a sphere-0 radius so everything becomes 0

    cout << "Rectagnular Box info:" << endl;
    shared_ptr<RectangularBox> rectbox = make_shared<RectangularBox>();//Pointer to create a default rectangular box object-no parameters
    cout << "The center coordinate is at X: " << rectbox->getX() << " Y: " << rectbox->getY() << " Z: " << rectbox->getZ() << endl;//Gets the coordinates from the base class which are set to 0 by default
    cout << "Dimensions - Height: " << rectbox->getHeight() << ", Length: " << rectbox->getLength() << ", Width: " << rectbox->getWidth() << endl;//The h,l,w comes from its class where it is set to 0
    cout << "The Volume is " << rectbox->calcVolume() << endl;//Object of rectangular box points to member function of its class because the volume formula is specific to it-results in 0

    cout << "Cylinder info:" << endl;
    shared_ptr<Cylinder> cylinder = make_shared<Cylinder>();//Cylinder object created with no parameters so it uses default constructor
    cout << "The center coordinate is at X: " << cylinder->getX() << " Y: " << cylinder->getY() << " Z: " << cylinder->getZ() << endl;//Gets the coordinates from the base class which are set to 0
    cout << "Dimensions - Height: " << cylinder->getHeight() << ", Radius: " << cylinder->getRadius() << endl;//Height and Radius are specific to cylinder class so it points to its own member function to get 0 for both
    cout << "The Volume is " << cylinder->calcVolume() << endl << endl;//Cylinder volume formula is specfic to it so it points to its own member function in the class


    cout << "====Creating Objects with Overloaded Constructors!====" << endl;
    cout << "Sphere info:" << endl;
    sphere = make_shared<Sphere>(2, 2, 5, 3.5);//Using same object name but with overloaded parameters to use the overloaded constructors
    cout << "The center coordinate is at X: " << sphere->getX() << " Y: " << sphere->getY() << " Z: " << sphere->getZ() << endl;//Set to 2, 2, 5 respectively for the coordinates with inherits from the base class' member function
    cout << "Dimensions - Radius: " << sphere->getRadius() << endl;//Overloaded parameter passes it to the radius so it overwrites default and displays 3.5
    cout << "The Volume is " << sphere->calcVolume() << endl;//Calculates with 3.5 instead of 0 to get an actual result of 179.503

    cout << "Rectagnular Box info:" << endl;
    rectbox = make_shared<RectangularBox>(5, 0, -5, 2.5, 3.2, 1.5);//Using same object name but with overloaded parameters to use the overloaded constructors
    cout << "The center coordinate is at X: " << rectbox->getX() << " Y: " << rectbox->getY() << " Z: " << rectbox->getZ() << endl;//Using inheritance, the parameters use the overloaded constructors in the base class as 5, 0, -5
    cout << "Dimensions - Height: " << rectbox->getHeight() << ", Length: " << rectbox->getLength() << ", Width: " << rectbox->getWidth() << endl;//Uses the 2.5, 3.2, 1.5 from arguments to set to its member function's use
    cout << "The Volume is " << rectbox->calcVolume() << endl;//Calculates the volume with its specific fomula using the argument values to get 12

    cout << "Cylinder info:" << endl;
    cylinder = make_shared<Cylinder>(1, 1, 1, 4, 1.75);//Using same object name but with overloaded parameters to use the overloaded constructor
    cout << "The center coordinate is at X: " << cylinder->getX() << " Y: " << cylinder->getY() << " Z: " << cylinder->getZ() << endl;//Uses 1, 1, 1 to set the coordinates to these values from the base class
    cout << "Dimensions - Height: " << cylinder->getHeight() << ", Radius: " << cylinder->getRadius() << endl;//Uses 4, 1.75 to set the height and radius to it in order to calculate the volume in the class' member function
    cout << "The Volume is " << cylinder->calcVolume() << endl;//Uses the 4 and 1.75 to calculate the volume spcific to a cylinder and returns 38.465 for the volume

    system("pause>nul");
    return 0;
}
```

```cpp
#ifndef GEOMETRICALSHAPE_H
#define GEOMETRICALSHAPE_H
#include <iostream>
using namespace std;

class GeometricalShape
{
private:
    int xPosition;//x coordinate
    int yPosition;//y coordinate
    int zPosition;//z coordinate
public:
    GeometricalShape();//default
    GeometricalShape(int x, int y, int z);//overloaded
    int getX();//returns x coordinate
    int getY();//returns y coordinate
    int getZ();//returns z coordinate
    virtual void Abstract() = 0;//Test abstraction
};
GeometricalShape::GeometricalShape()//default constructor that sets it equal to 0
{
    xPosition = 0;
    yPosition = 0;
    zPosition = 0;
}
GeometricalShape::GeometricalShape(int x, int y, int z)//overloaded constructor
{
    xPosition = x;//first parameter is set equal to the variable xPosition
    yPosition = y;//second parameter is set equal to the variable yPosition
    zPosition = z;//third parameter is set equal to the variable zPosition
}
int GeometricalShape::getX()
{
    return xPosition;//gets x value
}
int GeometricalShape::getY()
{
    return yPosition;//gets y value
}
int GeometricalShape::getZ()
{
    return zPosition;//gets z value
}
#endif // !GEOMETRICALSHAPE_H
```

```
#ifndef CYLINDER_H
#define CYLINDER_H
#include "GeometricalShape.h"//base class
#include <cmath>

//VF= 3.14 * r^2 * H

class Cylinder : public GeometricalShape//To inherit the base class
{
private:
    double height;//height belongs to cylinder's volume
    double radius;//radius belongs to cylinder's volume
public:
    Cylinder();//default
    Cylinder(int x, int y, int z, double h, double r);//overloaded
    double calcVolume();//calculates volume
    double getHeight();//returns height
    double getRadius();//returns radius
    virtual void Abstract() {}//tests for abstraction
};
Cylinder::Cylinder()//default constructor
{
    height = 0.0;//default height value
    radius = 0.0;//default radius value
}
Cylinder::Cylinder(int x, int y, int z, double h, double r) : GeometricalShape(x, y, z)//x,y,z are used with the base class' function to set the coordinates
{//last 2 parameters use Cylinder's member functions; overloaded function
    height = h;//fourth parameter is set equal to height
    radius = r;//fifth parameter is set equal to radius
}
double Cylinder::calcVolume()//Function calculates the volume using the height and radius values
{
    double volume;
    volume = (3.14 * pow(radius, 2) * height);//volume formula to calculate
    return volume;//returns the volume value
}
double Cylinder::getHeight()
{
    return height;//gets height value
}
double Cylinder::getRadius()
{
    return radius;//gets radius value
}
#endif // !Cylinder
```

```
#ifndef SPHERE_H
#define SPHERE_H
#include "GeometricalShape.h"//base class
#include <cmath>

//VF= 4.0/3.0 * 3.14 * r^3

class Sphere : public GeometricalShape//To inherit the base class
{
private:
    double radius;//radius belongs to a sphere's volume
public:
    Sphere();//default constructor
    Sphere(int x, int y, int z, double r);//overloaded constructor
    double getRadius();//returns radius
    double calcVolume();//calculates the volume
    virtual void Abstract() {}//tests abstraction
};
Sphere::Sphere()//default constructor
{
    radius = 0.0;//default radius set to 0
}
Sphere::Sphere(int x, int y, int z, double r) : GeometricalShape(x, y, z)//Values x, y, z are sent to base class to use its overloaded constructor
{
    radius = r;//fourth parameter is set equal to radius
}
double Sphere::getRadius()
{
    return radius;//gets radius value
}
double Sphere::calcVolume()//Function calculates the volume using the radius value
{
    double volume;
    volume = (4.0 * 3.14 * pow(radius, 3)) / 3.0;//volume formula to calculate
    return volume;//returns the volume value
}
#endif // !SPHERE_H
```

```cpp
#ifndef RECTANGULARBOX_H
#define RECTANGULARBOX_H
#include "GeometricalShape.h"//base class

//VF= H * L * W

class RectangularBox : public GeometricalShape//To inherit the base class
{
private:
    double height;//height belongs to reactangular box's volume
    double length;//length belongs to reactangular box's volume
    double width;//width belongs to reactangular box's volume
public:
    RectangularBox();//default constructor
    RectangularBox(int x, int y, int z, double h, double l, double w);//overloaded constructor
    double calcVolume();//calculates volume
    double getHeight();//returns height
    double getLength();//returns length
    double getWidth();//returns width
    virtual void Abstract() {}//tests for abstraction
};
RectangularBox::RectangularBox()//default constructor
{
    height = 0.0;//default height value
    length = 0.0;//default length value
    width = 0.0;//default width values
}
RectangularBox::RectangularBox(int x, int y, int z, double h, double l, double w) : GeometricalShape(x, y, z)//first 3 parameters use the base class function
{//last 3 parameters use the rectangularbox class functions
    height = h;//fourth parameter is set equal to height
    length = l;//fifth parameter is set equal to length
    width = w;//sixth parameter is set equal to width
}
double RectangularBox::calcVolume()//Function calculates the volume using the height, length, and width values
{
    double volume;
    volume = (height * length * width);//volume formula to calculate
    return volume;//returns the volume value
}
double RectangularBox::getHeight()
{
    return height;//gets height value
}
double RectangularBox::getLength()
{
    return length;//gets length value
}
double RectangularBox::getWidth()
{
    return width;//gets width value
}
#endif // !RECTANGULARBOX_H
```



```
====Creating Objects with Default Constructors!====
Sphere info:
The center coordinate is at X: 0 Y: 0 Z: 0
Dimensions - Radius: 0
The Volume is 0
Rectagnular Box info:
The center coordinate is at X: 0 Y: 0 Z: 0
Dimensions - Height: 0, Length: 0, Width: 0
The Volume is 0
Cylinder info:
The center coordinate is at X: 0 Y: 0 Z: 0
Dimensions - Height: 0, Radius: 0
The Volume is 0

====Creating Objects with Overloaded Constructors!====
Sphere info:
The center coordinate is at X: 2 Y: 2 Z: 5
Dimensions - Radius: 3.5
The Volume is 179.503
Rectagnular Box info:
The center coordinate is at X: 5 Y: 0 Z: -5
Dimensions - Height: 2.5, Length: 3.2, Width: 1.5
The Volume is 12
Cylinder info:
The center coordinate is at X: 1 Y: 1 Z: 1
Dimensions - Height: 4, Radius: 1.75
The Volume is 38.465
```