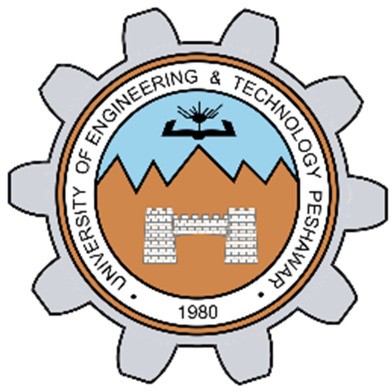
**Friend Function and Friend Class**

## LAB # 07



**Fall 2020**

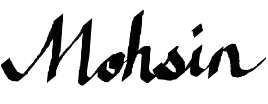
**CSE208L Object Oriented Programming Lab**

Submitted by: **Syed Mohsin Shah**

Registration No. : **19PWCSE1749**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”



Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

## **Engr. Sumayyea Salahuddin**

February 2, 2021

Department of Computer Systems Engineering University of Engineering and Technology, Peshawar

# **Objectives of the Lab:**

* Understand the difference between static and dynamic binding.
* 2 Demonstrate polymorphism with the help of virtual function and base class pointer.
* 3 Develop abstract class using pure virtual function.
* 4 Complete all the tasks available in CISCO Chapter 6.

Activity

# **Title:**

# Polymorphism

# **Problem Analysis:**

Define an abstract base class **shape** that includes protected data members for area and volume of a shape, public methods for computing area and volume of a shape (make these functions **virtual**), and a display function to display the information about an object. Make this class abstract by making display function pure virtual.

Derive a concrete class **point** from the **shape** class. This **point** class contains two protected data members that hold the position of **point**. Provide no-argument and 2-argument constructors. Override the appropriate functions of base class.

Derive a class **Circle** publicly from the **point** class. This class has a protected data member of radius. Provide a no-argument constructor to initialize the fields to some fixed values. Provide a 3-argument constructor to initialize the data members of **Circle** class to the values sent from outside. Override the methods of base class as required.

Derive another class **Cylinder** from the **Circle** class. Provide a protected data member for height of cylinder. Provide a no-argument constructor for initializing the data members to default values. Provide a 4-argument constructor to initialize x- and y-coordinates, radius, and height of cylinder. Override the methods of base class.

Write a driver program to check the polymorphic behavior of this class.

**Algorithm:**

UML diagram for the above problem is given below:

|  |
| --- |
| **Shape** |
| * Area – float * Volume - float |
| * virtual float area() * virtual float volume() * virtual void display() = 0 |

* + First make class Shape:
    - Declare Area and Volume as protected integer data members.
    - Define virtual function area() that Prints Empty.
    - Define virtual function volume() that Prints Empty.
    - Define pure virtual function display().

|  |
| --- |
| **Point** |
| * x – int * y – int |
| * Point() * Point(int a, int b) * display() |

* + Next make class Point:
    - Declare x and y as protected integer data members.
    - Define parameter-less constructor to set x = 0, y = 0.
    - Define 2 parameter constructor to set x = a, y = b.
    - Define function display() to print x and y.
  + Next make class Circle:

|  |
| --- |
| **Circle** |
| * radius - float |
| * Circle() * Circle(int a, int b, int r) * area() * display() |

* + - Declare radius as protected integer data member.
    - Define parameter-less constructor to set radius = 0.
    - Define 3 parameter constructor to set x = a, y = b, radius = r.
    - Define area() that returns Area = 2 \* PI \* radius \* radius
    - Define function display() to print radius and area().
  + Next make class Cylinder:

|  |
| --- |
| **Cylinder** |
| * height - float |
| * Cylinder() * Cylinder(int a, int b, int r, int h) * area() * volume() * display() |

* + - Declare height as protected integer data member.
    - Define parameter-less constructor to set height = 0.
    - Define 4 parameter constructor to set x = a, y = b, radius = r, height = h.
    - Define area() that returns Area = 2 \* PI \* radius \* height + 2 \* PI \* radius \* radius.
    - Define volume() that returns Volume = PI \* radius \* radius \* height.
    - Define function display() to print height, area() and volume().
  + In main function, create object pointer to s and objects p, c, cy of each class input values into it.
  + Refer to each derived class and call display() pointed from s.

# **Flowchart**

# **C++**

## **Source Code:**

#include<iostream>

#include<math.h>

using namespace std;

class Shape //Abstact Class

{

protected:

float Area;

float Volume;

public:

virtual float area()

{

cout<<"\nEmpty Shape";

}

virtual float volume()

{

cout<<"\nEmpty Shape";

}

virtual void display() = 0;

};

class Point: public Shape

{

protected:

int x;

int y;

public:

Point():x(0),y(0){}

Point(int a, int b):x(a),y(b){}

//Overridden

void display()

{

cout << "Point's Information:" << endl;

cout << "\nX = " << x;

cout << "\nY = " << y << endl;

}

};

class Circle: public Point

{

protected:

float radius;

public:

Circle():radius(0){}

Circle(int a, int b, float r)

{

x = a;

y = b;

radius = r;

}

//Overridden

float area()

{

Area = M\_PI \* radius \* radius;

return Area;

}

void display()

{

cout << "\nCircle's Information: " << endl;

cout << "\nX = " << x;

cout << "\nY = " << y;

cout << "\nRadius = " << radius;

cout << "\nArea = " << area() << endl;

}

};

class Cylinder: public Circle

{

protected:

float height;

public:

Cylinder():height(0){}

Cylinder(int a, int b, float r, float h)

{

x = a;

y = b;

radius = r;

height = h;

}

//Overridden

float area()

{

Area = (2 \* M\_PI \* radius \* radius) + (2 \* M\_PI \* radius \* height);

return Area;

}

float volume()

{

Volume = M\_PI \* radius \* radius \* height;

return Volume;

}

void display()

{

cout << "\nCylinder's Information: " << endl;

cout << "\nX = " << x;

cout << "\nY = " << y;

cout << "\nRadius = " << radius;

cout << "\nHeight = " << height;

cout << "\nArea = " << area();

cout << "\nVolume = " << volume() << endl;

}

};

int main()

{

Shape \*s;

Point p(1, 2);

Circle c(3, 4, 5);

Cylinder cy(6, 7, 8, 9);

s = &p;

s->display();

s = &c;

s->display();

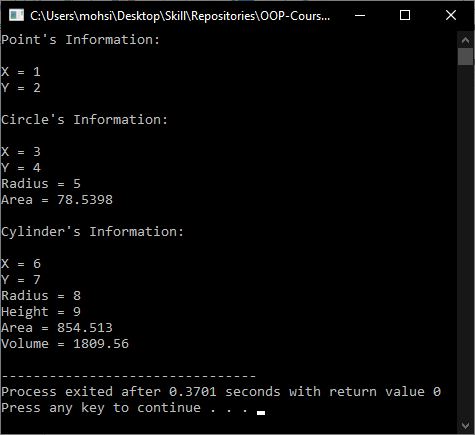
s = &cy;

s->display();

return 0;

}

## **Output:**



# **Java**

## **Source Code:**

### **Shape.java:**

**package** lab8;

**abstract** **class** Shape

{

**protected** **float** Area;

**protected** **float** Volume;

**public** **float** area()

{

System.***out***.println("Empty Shape");

**return** 0;

}

**public** **float** volume()

{

System.***out***.println("Empty Shape");

**return** 0;

}

**abstract** **void** display();

}

### **Point.java:**

**package** lab8;

**public** **class** Point **extends** Shape

{

**protected** **int** x;

**protected** **int** y;

**public** Point()

{

**this**.x = 0;

**this**.y = 0;

}

**public** Point(**int** a, **int** b)

{

**this**.x = a;

**this**.y = b;

}

//Overridden

**public** **void** display()

{

System.***out***.println("Point's Information: ");

System.***out***.println("X = " + **this**.x);

System.***out***.println("Y = " + **this**.y);

}

}

### **Circle.java:**

**package** lab8;

**public** **class** Circle **extends** Point

{

**protected** **float** radius;

**public** Circle()

{

**this**.x = 0;

**this**.y = 0;

**this**.radius = 0;

}

**public** Circle(**int** a, **int** b, **float** r)

{

**this**.x = a;

**this**.y = b;

**this**.radius = r;

}

//Overridden

**public** **float** area()

{

Area = (**float**)(Math.***PI*** \* Math.*pow*(**this**.radius, 2));

**return** Area;

}

**public** **void** display()

{

System.***out***.println("\nCircle's Information: ");

System.***out***.println("X = " + x);

System.***out***.println("Y = " + y);

System.***out***.println("Radius = " + **this**.radius);

System.***out***.println("Area = " + area());

}

}

### **Cylinder.java:**

**package** lab8;

**public** **class** Cylinder **extends** Circle

{

**protected** **float** height;

**public** Cylinder()

{

**this**.x = 0;

**this**.y = 0;

**this**.radius = 0;

**this**.height = 0;

}

**public** Cylinder(**int** a, **int** b, **float** r, **float** h)

{

**this**.x = a;

**this**.y = b;

**this**.radius = r;

**this**.height = h;

}

//Overridden

**public** **float** area()

{

Area = (**float**) ((2 \* Math.***PI*** \* Math.*pow*(radius, 2)) + (2 \* Math.***PI*** \* radius \* **this**.height));

**return** Area;

}

**public** **float** volume()

{

Volume = (**float**) (Math.***PI*** \* Math.*pow*(radius, 2) \* **this**.height);

**return** Volume;

}

**public** **void** display()

{

System.***out***.println("\nCylinder's Information: ");

System.***out***.println("X = " + x);

System.***out***.println("Y = " + y);

System.***out***.println("Radius = " + radius);

System.***out***.println("Height = " + height);

System.***out***.println("Area = " + area());

System.***out***.println("Volume = " + volume());

}

}

### **Main.java:**

**package** lab8;

**class** Main

{

**public** **static** **void** main(String[] args)

{

Shape p = **new** Point(1, 2);

Shape c = **new** Circle(3, 4, 5);

Shape cy = **new** Cylinder(6, 7, 8, 9);

p.display();

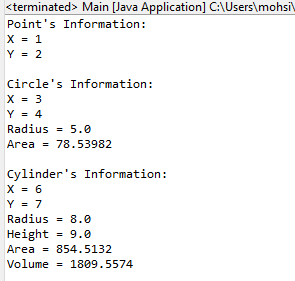
c.display();

cy.display();

}

}

## **Output:**



# **Python**

## **Source Code:**

from abc import ABC, abstractmethod  
import math  
  
  
class Shape(ABC):  
 Area = 0  
 Volume = 0  
  
 def area(self):  
 print(**"Empty Shape"**)  
  
 def volume(self):  
 print(**"Empty Shape"**)  
  
 def display(self):  
 pass  
  
  
class Point(Shape):  
 x = 0  
 y = 0  
  
 def \_\_init\_\_(self):  
 pass  
  
 def \_\_init\_\_(self, a, b):  
 self.x = a  
 self.y = b  
  
 def display(self):  
 print(**"Point's Information: "**)  
 print(**"X = "**, self.x)  
 print(**"Y = "**, self.y)  
  
  
class Circle(Point):  
 radius = 0  
  
 def \_\_init\_\_(self):  
 pass  
  
 def \_\_init\_\_(self, a, b, r):  
 self.x = a  
 self.y = b  
 self.radius = r  
  
 def area(self):  
 self.Area = math.pi \* self.radius \*\* 2  
 return self.Area  
  
 def display(self):  
 print(**"**\n**Circle's Information: "**)  
 print(**"X = "**, self.x)  
 print(**"Y = "**, self.y)  
 print(**"Radius = "**, self.radius)  
 print(**"Area = "**, self.area())  
  
  
class Cylinder(Circle):  
 height = 0  
  
 def \_\_init\_\_(self):  
 pass  
  
 def \_\_init\_\_(self, a, b, r, h):  
 self.x = a  
 self.y = b  
 self.radius = r  
 self.height = h  
  
 def area(self):  
 self.Area = (2 \* math.pi \* self.radius \*\* 2) + (2 \* math.pi \* self.radius \* self.height)  
 return self.Area  
  
 def volume(self):  
 self.Volume = (math.pi \* (self.radius \*\* 2) \* self.height)  
 return self.Volume  
  
 def display(self):  
 print(**"**\n**Cylinder's Information: "**)  
 print(**"X = "**, self.x)  
 print(**"Y = "**, self.y)  
 print(**"Radius = "**, self.radius)  
 print(**"Height = "**, self.height)  
 print(**"Area = "**, self.area())  
 print(**"Volume = "**, self.volume())  
  
  
p = Point(1, 2)  
c = Circle(3, 4, 5)  
cy = Cylinder(6, 7, 8, 9)  
  
p.display()  
c.display()  
cy.display()

## **Output: Conclusion:**

These programs helps us in building the fundamental concepts of Polymorphism in C++. It teaches us the prerequisite fundamentals for higher level programming. We learn the various concepts about the Virtual Functions and Pure Virtual Functions in OOP with the help of these programs.