**National University of Computer and Emerging Sciences**



**Lab Manual 11**

**Object Oriented Programming**

|  |  |
| --- | --- |
| Course Instructor | Ms. Syeda Tayyaba Bukhari |
| Lab Instructor (s) | Ms. Fariha Maqbool  Mr. Muhammad Usama |
| Section | BCS-2H |
| Semester | Spring 2023 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

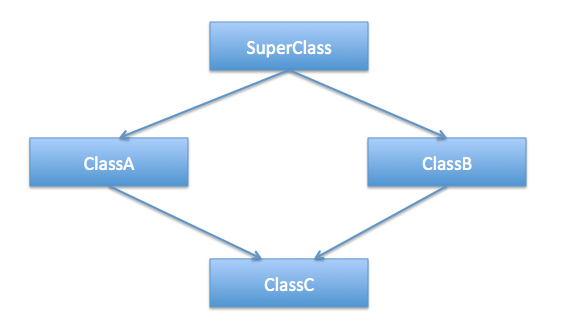
## **Objectives**

After performing this lab, students will practice:

* Learn and identify the need of polymorphism.
* Implement multiple and virtual inheritance.

# **Multiple Inheritance & Polymorphism**

For this exercise, we are going to work on a classical multiple inheritance issue known as ‘diamond problem’. Diamond problem is an ambiguity that arises when two classes A and B inherit from super class or base class, and class C also inherits from both A and B. The class hierarchy structure resembles a diamond as shown below.



1. For simplicity, create a single **.cpp file** with the following classes: **Faculty, Administrator, Teacher, & AdministratorTeacher**.
2. **Faculty** inherits **Administrator** and **Teacher**. While **AdministratorTeacher** has two parents **Administrator** and **Teacher** which represents that an **Administrator** can be a **Teacher** and vice versa

**Task 1:**

1. Add a **print()** method to **Faculty**, **Administrator**, and **Teacher** which displays the class name.
2. In the driver, create a pointer array of 3 **Faculty** objects.
3. Create one object for each of the remaining three classes as well and assign these three object to the **Faculty** object array.
4. Now, in a loop call the print method on the **Faculty** object array and observe the code behavior.
5. You may observe that “Faculty” is displayed on the console 3 times which is wrong.
6. To make corrections, use polymorphism. Make the **print()** method virtual and execute again.
7. This time you will encounter an error. It occurs because the **AdministratorTeacher** object shows ambiguous behavior when calling the **print()** method (It does not know which print method it should call)
8. To resolve this issue, we are going to use **virtual inheritance**. First, make the **print()** method pure virtual in **Faculty.** And add a **print()** method in the **AdministratorTeacher** class as well. Qualify or override this **print()** method by calling the **print()** of either **Teacher** or **Administrator** specifically.
9. Execute the program and report the issue in comment
10. Now, use virtual inheritance i.e. declare **Faculty** inheritance using public virtual keyword for **Teacher** and **Administrator** classes.
11. Execute the program again. This time you will observe that correct class names are displayed on console.

**TASK 2:** Define a pure abstract base class called BasicShape. The BasicShape class should have the following members:

**Private Member Variable:**

area, a double used to hold the shape's area.

**Public Member Functions:**

getArea. This function should return the value in the member variable area.

calcArea. This function should be a pure virtual function.

Next, define a class named Circle. It should be derived from the BasicShape class. It should have the following members:

**Private Member Variables:**

centerX, a long integer used to hold the x coordinate of the circle’s center.

centerY, a long integer used to hold the y coordinate of the circle’s center.

radius, a double used to hold the circle's radius.

**Public Member Functions:**

constructor—accepts values for centerX, centerY, and radius. Should call the overridden calcArea function described below.

getCenterX—returns the value in centerX.

getCenterY—returns the value in centerY.

calcArea—calculates the area of the circle (area = 3.14159 \* radius \* radius) and stores the result in the inherited member area.

Next, define a class named Rectangle. It should be derived from the BasicShape class. It should have the following members:

**Private Member Variables:**

width, a long integer used to hold the width of the rectangle.

length, a long integer used to hold the length of the rectangle.

**Public Member Functions:**

constructor—accepts values for width and length. Should call the overridden calcArea function described below.

getWidth—returns the value in width.

getLength—returns the value in length.

calcArea—calculates the area of the rectangle (area = length \* width) and stores

the result in the inherited member area.

After you have created these classes, create a driver program that defines a Circle object and

a Rectangle object. Demonstrate that each object properly calculates and reports its area

**Note:**

* Follow all the code indentation, naming conventions and code commenting guidelines.
* Make sure your program is executable.