**National University of Computer and Emerging Sciences**



**Lab Manual 09**

**Object Oriented Programming**

|  |  |
| --- | --- |
| Course Instructor | Syeda Tayyaba Bukhari |
| Lab Instructor (s) | Fariha Maqbool  Muhammad Usama |
| Section | BDS-2B |
| Semester | Spring 2022 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

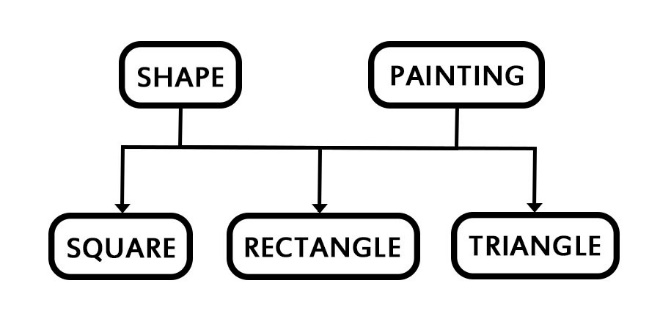
## Objectives

After performing this lab, students shall be able to:

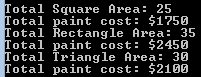
* Inheritance. (Accessibility of public, private, and protected members during public, private, and protected inheritance, Multiple Inheritance)
* Virtual functions

# **Exercise 1: Multiple Inheritance**

In this exercise, we are working with the following classes to implement multiple inheritance: **Shape, Painting, Square, Rectangle,** and **Triangle.**

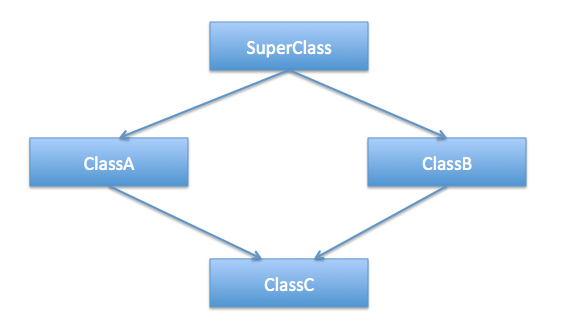
Implement the class hierarchy given below:

1. Class **Shape** has following protected data members: **length, breadth, height.**
2. Class **Square, Rectangle,** and **Triangle** all have **getArea()** method.
   1. Area of square = length \* lenth
   2. Area of rectangle = length\* breadth
   3. Area of triangle = ½ \* breadth \* height
3. Class **Painting** has method **getCost(area)** which returns the cost of painting a shape by multiplying paint cost with area of the shape.
4. For simplicity, you only create **one .h** file which contains the headers of all classes and **one .cpp** file which contains the implementation of all classes. Also, you are not required to use polymorphism in this question
5. Consider the following output(You can input your own values):



# **Exercise 2: 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. But it still doesn’t solve the ambiguity of the **AdministratorTeacher** object as now it has holds two copies of the **Faculty** class. Now, use virtual inheritance i.e. declare **Faculty** inheritance using public virtual keyword for **Teacher** and **Administrator** classes.
9. Execute the program again. This time you will observe that correct class names are displayed on console.

**Note:**

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