**National University of Computer and Emerging Sciences**



**Lab Manual 10**

**Object Oriented Programming**

|  |  |
| --- | --- |
| Course Instructor | Ms. Arooj Khalil |
| Lab Instructor (s) | Ms. Fariha Maqbool  Ms. Amara Nasir |
| Section | BSE-2A |
| Semester | Spring 2023 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

## **Objectives**

After performing this lab, students will practice:

* Inheritance (Multilevel, Multiple)
* Function overriding, polymorphism

# **TASK 1: Multilevel Inheritance & Polymorphism**

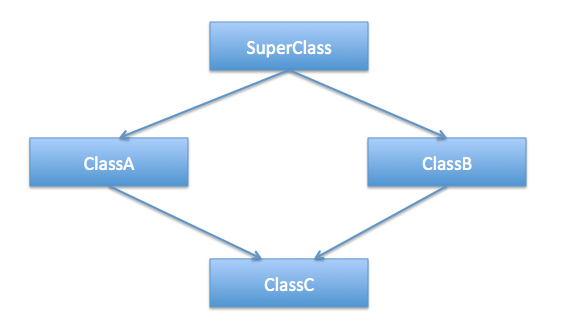
Design a Ship class that has the following members:

* A private member variable for the name of the ship (a string)
* A private member variable for the year that the ship was built (a string)
* A constructor and appropriate accessors and mutators
* A virtual print function that displays the ship’s name and the year it was built.
* Design a CruiseShip class that is derived from the Ship class. The CruiseShip class should have
* the following members:
* A member variable for the maximum number of passengers (an int)
* A constructor and appropriate accessors and mutators
* A print function that overrides the print function in the base class. The CruiseShip class’s print
* function should display only the ship’s name and the maximum number of passengers.
* Design a CargoShip class that is derived from the Ship class. The CargoShip class should have
* the following members:
* A member variable for the cargo capacity in tonnage (an int).
* A constructor and appropriate accessors and mutators.
* A print function that overrides the print function in the base class. The CargoShip class’s print function should display only the ship’s name and the ship’s cargo capacity.

Demonstrate the classes in a program that has an array of Ship pointers. The array elements should be initialized with the addresses of dynamically allocated Ship, CruiseShip, and CargoShip objects. The program should then step through the array, calling each object’s print function.

# **Task-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.
3. Add a **print()** method to **Faculty**, **Administrator**, and **Teacher** which displays the class name.
4. In the driver, create a pointer array of 3 **Faculty** objects.
5. Create one object for each of the remaining three classes as well and assign these three object to the **Faculty** object array.
6. Now, in a loop call the print method on the **Faculty** object array and observe the code behavior.
7. You may observe that “Faculty” is displayed on the console 3 times which is wrong.
8. To make corrections, use polymorphism. Make the **print()** method virtual and execute again.
9. 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)
10. 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.
11. Execute the program again. This time you will observe that correct class names are displayed on console.