**FAST School of Computing**

**Object Oriented Programming – Spring 2025**

**Software Engineering Department**

**LAB 13**

**Inheritance & its types in C++**

**Learning Outcomes**

In this lab you are expected to learn the following:

* Basic and Adanced Concept and Implementation of Inheritance

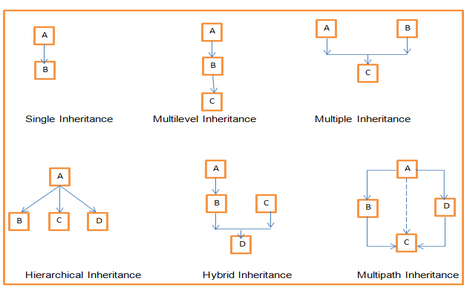
**Note:** Plagiarism(from some else or internet) in any 1 question will lead to zero marks in the whole lab task.

**Inheritance:**

Inheritance in C++ is a mechanism where a new class (called the derived class) acquires the properties and behaviors (data members and member functions) of an existing class (called the base class). It helps in code reusability and establishes a parent-child relationship between classes.

**Types of Inheritance:**

| **Type** | **Description** |
| --- | --- |
| **1. Single Inheritance** | A class inherits from only one base class. |
| **2. Multiple Inheritance** | A class inherits from more than one base class. |
| **3. Multilevel Inheritance** | A class inherits from a base class, and then another class inherits from that derived class (forming a chain). |
| **4. Hierarchical Inheritance** | Multiple classes inherit from a single base class. |
| **5. Hybrid Inheritance** | A combination of two or more types of inheritance (like combination of multiple and multilevel). |



**What can be done using Inheritance:**

| **Thing** | **Explanation** |
| --- | --- |
| **Code Reusability** | You can reuse existing code from the base class without rewriting it. |
| **Extension of Functionality** | You can add new features in the derived class, while keeping the old features. |
| **Polymorphism** | Using virtual functions, you can achieve **runtime polymorphism** (deciding behavior at runtime). |
| **Hierarchical Structure** | You can create a class hierarchy (parent-child-grandchild) easily. |
| **Overriding Methods** | You can override base class methods in derived classes to change behaviors. |
| **Simplified Maintenance** | Changes made to the base class automatically apply to derived classes if properly designed. |
| **Virtual Inheritance** | You can handle complex cases like the Diamond Problem through virtual inheritance. |
| **Encapsulation** | Inherited classes can access protected members (and public ones) while keeping them hidden from outside. |

**What cannot be done using inheritance:**

| Thing | Explanation |
| --- | --- |
| Multiple Copies of a Base Class | Without virtual inheritance, multiple inheritance can cause multiple copies of base class (Diamond Problem). |
| Inheritance of Constructors/Destructors | Constructors and destructors are not inherited automatically (but can be called explicitly). |
| Inheritance of Private Members | Private members of the base class are not directly accessible in derived classes (only through public/protected functions). |
| Crosswise Inheritance | One class cannot inherit from another class at runtime dynamically (inheritance is fixed at compile time). |
| Changing Access Specifiers of Base Class Members | You cannot make a base class's private member public directly just by inheritance. |
| Force "Different" Behavior Without Overriding | If you don’t override a virtual function, the base class behavior remains — you can't automatically "invent" new behavior. |
| Multiple Inheritance Complexity | C++ allows multiple inheritance, but it can make code very complex and error-prone if not handled carefully. |
| Cannot Replace Composition | Sometimes composition (having objects inside classes) is better than inheritance. Not everything should be modeled by inheritance. |

**Tasks**

1. Write a C++ program to demonstrate the concept of **Multiple Inheritance**. Create three classes: M, N, and P. The class M should have a protected integer variable m and a public member function get\_M(int) that assigns a value to m. Similarly, the class N should have a protected integer variable n and a public member function get\_N(int) that assigns a value to n. The class P should inherit publicly from both M and N. It should have a public member function display() that prints the values of m and n, and also displays the product of m and n. In the main() function, create an object of class P, set values for m and n using the respective functions (for example, 10 and 20), and then call the display() function to show the results. Define all the member functions outside the class definitions using the scope resolution operator. Make sure to follow proper access specifiers so that the derived class can access the inherited variables.

**2. Multilevel Inheritance (Grandparent ➔ Parent ➔ Child)**

Suppose you are designing a **vehicle registration system**. You begin by defining a basic Vehicle class, which contains common properties such as the registrationNumber and the ownerName. Next, you create a Car class that **inherits** from Vehicle and adds car-specific features like fuelType and seatingCapacity. Finally, you define an ElectricCar class that **inherits** from Car and introduces additional properties such as batteryCapacity (in kWh) and chargingTime (in hours).

To manage multiple electric cars, you create an **array of ElectricCar objects**. The program should allow adding electric cars to the array, displaying their details, and performing basic operations like **updating the battery capacity** or **finding the electric car with the lowest charging time**. This forms a simple three-level inheritance hierarchy: **Vehicle ➔ Car ➔ ElectricCar**.

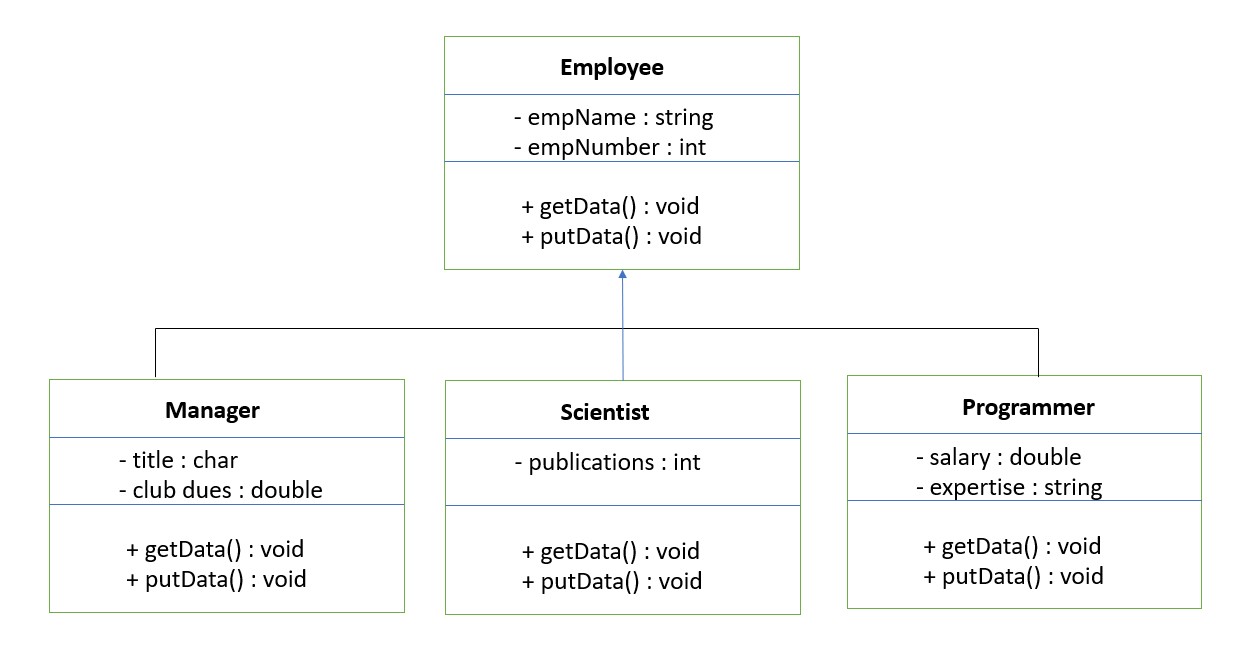
**3. Hierarchical Inheritance (One Parent, Many Children)**

You are designing a university database system, starting with a basic class called Person that stores general information like the person's name, age, and address. From this base class, you derive three more specialized classes: Student, Professor, and Staff. Each of these classes inherits from Person, using the protected access specifier, which means the personal details remain accessible within the child classes but are hidden from external access. Each specialized class adds its own unique properties: Student includes the GPA and student ID, Professor tracks the subjects they teach and their number of publications, and Staff holds details about their department and years of service. Additionally, each class defines functions that manage and manipulate these details, one is displaying full information, calculate ones student GPA till 4th semester and award scholarships to students based on CGPA if student have 3.5 CGPA he/she is eligible for scholarship add into scholarship awarding list otherwise add those students into a waiting list array and display how many are eligible and how many are not , in professor class create a function of promotion which check if they have more than 5 publication promote those professors and show a simple message like promotion granted, and calculate bonuses for staff based on their years of service(if they have worked more than 5 year) add 20% of the salary as a bonus. And make to create 10 students objects, 3 professors objects, 3 staff objects and each staff have different salaries based on their profession.

**Task\_2:**

Implement the following UML Diagram using C++ Inheritance. First identify inheritance type w.r.t level and then code accordingly.

Note: Each class must have parameterize constructors.



**Task\_3:**

**Case Study**

A local community library aims to modernize its material management system, currently relying on separate manual logs for books, magazines, and research papers. Each material is identified by a unique ID and a title, with books additionally having an author and ISBN, magazines an issue number and publication date, and research papers authors and a conference/journal name. The library requires a digital system capable of adding new materials (books, magazines, research papers), displaying the ID and title of any material, accessing and displaying specific identifying information for each type (e.g., author and ISBN for books, issue number for magazines, conference/journal for research papers), tracking the total number of items, determining if two materials are the same based on their unique ID, and generating a type-specific display string that combines the title with a key identifier (like ISBN for books). The design of this system should utilize object-oriented programming principles, focusing on organization and future extensibility for new material types.

**Submission Details:**

1. Save single .cpp file with your roll no and lab number e.g. i22-XXXX\_Lab#.cpp
2. Take screen shot of running test cases of tasks.
3. Zip the .cpp file and screen shots (Do not create .rar file) with roll no and lab no. e.g. i22-XXXX\_Lab#.zip.
4. Submit the zip file on google class room.