**National University of Computer and Emerging Sciences**



**Lab Manual 12**

***for***

**Object Oriented Programming (OOP)**

|  |  |
| --- | --- |
| **Course Instructor** | **Ms. Hina Iqbal** |
| **Lab Instructor(s)** | **Amina Qaiser** |
| **Section** | **B** |
| **Semester** | **Fall 2024** |

**Department of Computer Science**

**FAST-NU, Lahore, Pakistan**

**Lab 12**

You are developing a system to manage employees in a company. The company has different types of employees, such as **Managers**, **Engineers**, and **Interns**. There is a base class Employee, with derived classes implementing their specific behaviors.

Additionally, the company allows multilevel inheritance for **Senior Managers**, and some employees hold dual roles, requiring **multiple inheritance**.

**Class Requirements**

**1. Employee (Abstract Base Class)**

* **Attributes**: name (string), ID (int).
* **Functions**:
  + virtual void displayInfo() - Pure virtual function to display employee details.
  + virtual void calculateSalary() - Virtual function for salary computation.
  + Constructor initializing name and ID.
  + Virtual destructor.

**2. Manager (Derived from Employee)**

* **Attributes**: numTeams (int).
* **Overrides**:
  + displayInfo() - Shows Manager-specific info.
  + calculateSalary() - Calculates salary based on team size.
* **New Member**: void planProject() - Specific to managers.

**3. Engineer (Derived from Employee)**

* **Attributes**: numProjects (int).
* **Overrides**:
  + displayInfo() - Shows Engineer-specific info.
  + calculateSalary() - Calculates salary based on project count.
* **New Member**: void writeCode() - Specific to engineers.

**4. Intern (Derived from Engineer)**

* Demonstrates **Multilevel Inheritance**.
* Adds stipend computation and overrides displayInfo().

**5. SeniorManager (Derived from Manager)**

* Demonstrates **Multilevel Inheritance**.
* Adds bonus and team-specific display.

**6. DualRoleEmployee (Multiple Inheritance)**

* Inherits from **Manager** and **Engineer**.
* Demonstrates **Diamond Problem**.
* Resolves ambiguity with virtual inheritance.

**Tasks**

1. **Constructor Call Sequence**:
   * Implement constructors for all classes.
   * Observe the sequence when creating objects in multilevel and multiple inheritance.
2. **Polymorphism with Virtual Functions**:
   * Use base class (Employee\*) pointers to access derived class objects.
   * Override displayInfo() and calculateSalary() to demonstrate runtime polymorphism.
3. **Static and Dynamic Binding**:
   * Observe static binding when calling non-virtual functions.
   * Observe dynamic binding when calling virtual functions.
4. **Type Casting**:
   * Use static\_cast and dynamic\_cast to:
     + Upcast Manager\* to Employee\*.
     + Downcast Employee\* to Manager\* or Engineer\*.
5. **Diamond Problem in Multiple Inheritance**:
   * Demonstrate ambiguity in method calls without virtual inheritance.
   * Resolve ambiguity using virtual inheritance.