National University of Computer and Emerging Sciences

Assignment

For

Object-Oriented Programming

FAST School of Computing

Submission deadline
04-Apr-2024 2:45 PM

Instructions:

- Make a word document with the naming convention "SECTION_ LAB#_ROLLNO" and put all your source code and snapshots of its output in it. Make sure your word file is formatted properly.
- Zip the word file with all Source code.
- Plagiarism is strictly prohibited.

Question#1

Implement a C++ program that defines a hierarchical class structure consisting of three classes: Base, Intermediate, and Derived.

The Base class should have the following attributes:

- Base1 (public)
- Base2 (protected)
- Base3 (private)

Implement constructors and methods for Base to handle default initialization and initialization with passed integer values. Additionally, provide a display method to showcase the attributes of the Base class.

- The Intermediate class should publicly inherit from Base and introduce the following attribute:

- Intermediate1 (public)
- Intermediate2 (protected)
- Intermediate3 (private)

Implement constructors and methods for Intermediate to handle default initialization and initialization with passed integer values. Also, include a display method to exhibit the attributes of the Intermediate class.

- The Derived class should publicly inherit from Intermediate and include the following attributes:

- Derived1 (public)
- Derived2 (protected)
- Derived3 (private)

Implement constructors and methods for Derived to handle default initialization and initialization with passed integer values. Introduce a display method to demonstrate the attributes of the Derived class.

Your implementation should demonstrate proper access specifiers and inheritance relationships. Ensure that each class maintains its own encapsulation while inheriting attributes from its base classes. Additionally, consider incorporating appropriate error handling and validation mechanisms where necessary.

Question#2

Design a class named **PersonData** with the following member variables:

- FirstName
- LastName
- Address
- City
- State
- Zip
- PhoneNo

Write the appropriate **accessor** (getter) and **mutator** (setter) functions for these member variables.

Next, design a class named **CustomerData**, which is derived from the **PersonData** class.
The **CustomerData** class should have the following member attributes

• CustomerNumber
• MailingList

The **CustomerNumber** attribute will be used to hold a **unique** integer for each customer.

The **MailingList** attribute should be a **bool**. It will be set to true if the customer wishes to be on a mailing list, or false if the customer does not wish to be on a mailing list.

Write appropriate Accessor and Mutator functions for these member attributes.

CustomerData class will have the

- **InputCustomerData** member function which will Input all the data for customer.
- **DisplayCustomerData** member function which will display all the data for customer.

Demonstrate an object of the **CustomerData** class in a simple program.

Question#3

A retail store has a **preferred customer** plan where customers may earn discounts on all their purchases.

The amount of a **customer's discount** is determined by the amount of the customer's cumulative purchases in the store.

- When a preferred customer spends $500, he or she gets a 5% discount on all future purchases.
- When a preferred customer spends $1,000, he or she gets a 6% discount on all future purchases.
- When a preferred customer spends $1,500, he or she gets a 7% discount on all future purchases.
- When a preferred customer spends $2,000 or more, he or she gets a 10% discount on all future purchases.

Design a class named **PreferredCustomer**, which is derived from the **CustomerData** class you created in **question 1 above**. The **PreferredCustomer** class should have the following member variables:

- purchasesAmount (a double)
- discountLevel (a double)

The **purchasesAmount** variable holds the total of a customer's purchases to date. The **discountLevel** variable should be set to the correct discount percentage, according to the store's preferred customer plan. Write appropriate member functions for this class and demonstrate it in a simple program.

Question#4

You are going to make a **Mini School Database System** to search Students. It contains **Person** class which contains following Attributes.

- Name
- Age
- CNIC
- Contact Number

In School there mainly two classes **Teacher** & **Student.**

Make two classes which have attributes of person. Teacher can have further attributes

- Emp Id
- Course
- Salary

While Student can have further attributes

- CGPA
- Degree

Make a struct names **Login** with attributes ID & Password. There will be a admin login. User can access the Database system with specific ID & Password.

Admin can Perform Following Tasks.

- Create (Teacher or Student)
- Modify (Teacher or Student)
- Search (Teacher or Student)

Search the Student or Teacher by following ways

- Search Student by name
- Search Student by roll no.
- Search all students of specific age
- Search all students of CGPA more than given by admin
- Search all students of CGPA less than given by admin

All these searching will be done by **Functions**. Pass the array or anything in which you store the data to the function and it will print the Data of Students in proper **Format.**

NOTE : Make Constructor, Destructor, and Copy Constructor in each class.

Question#5

This prompt challenges you to tackle the diamond problem in C++ object-oriented programming using advanced virtual inheritance. We'll explore a scenario involving shapes with color and texture attributes. The goal is to design a class hierarchy where TexturedColoredShape inherits from both ColoredShape and TexturedShape. However, this inheritance structure can lead to ambiguity if both base classes have a common function like draw().

Your task is to leverage virtual inheritance to resolve this ambiguity and achieve the following:

1. Create concrete base classes ColoredShape and TexturedShape inheriting virtually from Shape, each offering drawing functionality based on color or texture.
2. Design a derived class TexturedColoredShape that inherits from both ColoredShape and TexturedShape.
3. Implement the draw() function in TexturedColoredShape to combine drawing functionalities from both base classes.
4. Ensure that TexturedColoredShape can access color and texture information from its base classes without ambiguity.