**Task: Implement a C++ Program Using Class Inheritance for Faculty and Student Management**

**Objective**: Write a C++ program that demonstrates the use of class inheritance and data manipulation through member functions to manage faculty and student information.

**Instructions:**

1. **Class Definitions**:
2. **Class faculty**:
3. **Private Data Members**:
4. int course\_c: Stores the count of courses taught by the faculty member.
5. int tn: Stores a 3-digit telephone number.
6. string fname, lname: Stores the first and last names of the faculty member.
7. **Protected Data Members**:
8. int age: Stores the age of the faculty member.
9. **Public Member Functions**:
10. void set(): Prompts the user to enter the faculty member's first name, last name, course count, and telephone number, and stores these values.
11. void display(): Outputs the stored information of the faculty member.
12. void setage(int no): Sets the age of the faculty member.
13. int getage() const: Returns the age of the faculty member.
14. **Class Gstudent** (Inherits publicly from faculty):
15. **Private Data Members**:
16. string first, last, thesis: Stores the first name, last name, and thesis topic of the graduate student.
17. float cgpa: Stores the CGPA of the graduate student.
18. **Public Member Functions**:
19. void set(): Prompts the user to enter the graduate student's first name, last name, thesis topic, and CGPA, and stores these values.
20. void display(): Outputs the stored information of the graduate student.
21. void setage(int a): Sets the age of the graduate student.
22. int getage() const: Returns the age of the graduate student.
23. **Class UNstudent** (Inherits publicly from Gstudent):
24. **Private Data Members**:
25. string fname, lname, fyp\_name: Stores the first name, last name, and final year project name of the undergraduate student.
26. float cgpa: Stores the CGPA of the undergraduate student.
27. **Public Member Functions**:
28. void set(): Prompts the user to enter the undergraduate student's first name, last name, final year project name, and CGPA, and stores these values.
29. void display(): Outputs the stored information of the undergraduate student.
30. void setage(int a): Sets the age of the undergraduate student.
31. int getage() const: Returns the age of the undergraduate student.
32. **Main Function**:
33. **Create an object** of the class UNstudent named obj.
34. **Use the object to**:
35. Call the set() method from the faculty class to gather and store the faculty member's details.
36. Call the display() method from the faculty class to output the faculty member's details.
37. Call the setage(19) method from the faculty class to set the age of the faculty member.
38. Output the age using getage() method from the faculty class.

**Key Concepts to Understand:**

1. **Class Inheritance**: The Gstudent and UNstudent classes inherit from the faculty class, allowing them to access its public and protected members.
2. **Data Encapsulation**: Private data members are used to protect sensitive information.
3. **Method Overriding**: Each class has its own set() and display() methods to manage their specific data.

**Expected Output:**

When the code is executed, it should display:

1. The details of the faculty member entered by the user, including their name, course count, and telephone number.
2. The age of the faculty member after being set to 19.

**Task: Implement a C++ Program to Manage Customer Data Using Class Inheritance**

**Objective**: Write a C++ program that demonstrates the use of class inheritance and encapsulation to manage personal and customer data effectively.

**Instructions:**

1. **Class Definitions**:
2. **Class persondata**:
3. **Private Data Members**:
4. string fname: Stores the first name of the person.
5. string lname: Stores the last name of the person.
6. string adress: Stores the address of the person.
7. **Public Member Functions**:
8. void setfname(const string& fn): Sets the first name of the person.
9. string getfname() const: Returns the first name of the person.
10. void setlname(const string& ln): Sets the last name of the person.
11. string getlname() const: Returns the last name of the person.
12. void setadress(const string& add): Sets the address of the person.
13. string getadress() const: Returns the address of the person.
14. **Class customerdata** (Inherits publicly from persondata):
15. **Private Data Members**:
16. int customerno: Stores the customer number.
17. bool mailinglist: Indicates whether the customer wants to be on the mailing list.
18. **Public Member Functions**:
19. void setcn(int no): Sets the customer number.
20. int getcn() const: Returns the customer number.
21. void setml(bool s): Sets the mailing list preference.
22. bool getml() const: Returns the mailing list preference.
23. void inputcustomerdata(): Prompts the user to enter the customer number and mailing list preference.
24. void displaycustomerdata() const: Outputs the customer number and mailing list preference.
25. **Main Function**:
26. **Create an object** of the class customerdata named obj.
27. **Use the object to**:
28. Set the first name, last name, and address of the person using the respective setter methods.
29. Display the first name, last name, and address using the getter methods.
30. Call the inputcustomerdata() method to gather the customer's specific information.
31. Call the displaycustomerdata() method to output the customer's number and mailing list preference.

**Key Concepts to Understand:**

1. **Class Inheritance**: The customerdata class inherits from the persondata class, allowing it to access its public and protected members.
2. **Data Encapsulation**: Private data members protect sensitive information and ensure that they can only be accessed or modified through public member functions.
3. **Input/Output Operations**: Using cin for input and cout for output to interact with the user.

**Expected Output:**

When the code is executed, it should display:

1. The first name, last name, and address of the person as set in the program.
2. A prompt for the customer number and mailing list preference.
3. The customer number and mailing list preference as entered by the user.

**Task: Implement a C++ Program to Manage Customer Data with Preferred Discounts**

**Objective**: Write a C++ program that demonstrates class inheritance, data encapsulation, and the implementation of a discount system for preferred customers.

**Instructions:**

1. **Class Definitions**:
2. **Class CustomerData**:
3. **Protected Data Members**:
4. string customerName: Stores the name of the customer.
5. int customerID: Stores the unique ID of the customer.
6. string PhoneNumber: Stores the customer's phone number.
7. **Public Member Functions**:
8. void setCustomerName(std::string name): Sets the name of the customer.
9. void setCustomerID(int id): Sets the customer ID.
10. void setPhoneNumber(std::string phone): Sets the phone number of the customer.
11. std::string getCustomerName() const: Returns the customer's name.
12. int getCustomerID() const: Returns the customer's ID.
13. std::string getPhoneNumber() const: Returns the customer's phone number.
14. **Class PreferredCustomer** (Inherits publicly from CustomerData):
15. **Private Data Members**:
16. double purchasesAmount: Stores the total purchase amount of the preferred customer.
17. double discountLevel: Stores the discount level based on the purchase amount.
18. **Private Member Functions**:
19. void updateDiscountLevel(): Updates the discount level based on the purchasesAmount.
20. **Public Member Functions**:
21. void setPurchasesAmount(double amount): Sets the purchase amount and calls updateDiscountLevel() to adjust the discount level.
22. void display(): Outputs the purchase amount and discount level.
23. **Main Function**:
24. **Create an object** of the class PreferredCustomer named obj.
25. **Use the object to**:
26. Set the customer name, customer ID, and phone number using the respective setter methods.
27. Display the customer information using the getter methods.
28. Set the purchase amount using setPurchasesAmount() and then call display() to show the purchase amount and discount level.

**Key Concepts to Understand:**

1. **Class Inheritance**: The PreferredCustomer class inherits from the CustomerData class, allowing it to access its protected members and public member functions.
2. **Data Encapsulation**: Private data members protect sensitive information, ensuring that they can only be accessed or modified through public member functions.
3. **Discount System Logic**: The program includes a mechanism to update the discount level based on the total purchases made by the customer.

**Expected Output:**

When the code is executed, it should display:

1. The customer's name, ID, and phone number as set in the program.
2. The purchase amount and the calculated discount level based on the purchase amount.

**Task: Implement a C++ Program to Manage Person, Employee, and Businessman Data**

**Objective**: Write a C++ program that demonstrates class inheritance, data encapsulation, and the use of constructors to initialize class members.

**Instructions:**

1. **Class Definitions**:
2. **Class person**:
3. **Protected Data Members**:
4. string name: Stores the name of the person.
5. string gender: Stores the gender of the person.
6. **Private Data Member**:
7. int age: Stores the age of the person.
8. **Public Constructor**:
9. person(string n, string s, int a): Initializes the name, gender, and age of the person.
10. **Class employe** (Inherits publicly from person):
11. **Protected Data Member**:
12. double number: Stores the employee's identification number.
13. **Public Constructor**:
14. employe(string n, string s, int a, double num): Initializes the employee's name, gender, age, and identification number.
15. **Public Member Function**:
16. void employefun(): Displays a greeting message using the employee's name.
17. **Class unemploye** (Inherits publicly from person):
18. **Public Constructor**:
19. unemploye(string n, string s, int a): Initializes the unemployed person's name, gender, and age and displays a greeting message.
20. **Class businessman** (Inherits publicly from employe):
21. **Public Constructor**:
22. businessman(string n, string s, int a, double num): Initializes the businessman's name, gender, age, and identification number.
23. **Public Member Function**:
24. void display(): Calls employefun() to display a greeting message and then displays the employee's identification number.
25. **Main Function**:
26. **Create Objects**:
27. Create an object of the person class named obj1 with appropriate parameters.
28. Create an object of the employe class named obj2 with appropriate parameters.
29. Create an object of the businessman class named obj3 with appropriate parameters.
30. **Call display() on the businessman object** (obj3) to demonstrate the functionality of the classes.

**Key Concepts to Understand:**

1. **Class Inheritance**: The employe, unemploye, and businessman classes inherit properties from the person class, showcasing the concept of base and derived classes.
2. **Access Specifiers**: Understand how protected and private members are accessed within derived classes.
3. **Constructor Initialization**: Learn how to use constructors to initialize data members when creating objects of a class.

**Expected Output:**

When the code is executed, it should display:

1. A greeting message from the businessman object, along with the employee's identification number.

**Task: Implement a C++ Program for Vehicle Management**

**Objective**: Write a C++ program that demonstrates class inheritance and encapsulation through the creation of a vehicle management system that includes different types of vehicles.

**Instructions:**

1. **Class Definitions**:
2. **Class vehicle**:
3. **Protected Data Members**:
4. double speed: Represents the speed of the vehicle.
5. double distance: Represents the distance traveled by the vehicle.
6. float duration: Represents the time taken to cover the distance.
7. **Public Member Functions**:
8. void setspeed(double s): Sets the speed of the vehicle.
9. double getspeed() const: Returns the speed of the vehicle.
10. void setdistance(double d): Sets the distance traveled by the vehicle.
11. double getdistance() const: Returns the distance traveled by the vehicle.
12. void getcomputeduration(): Computes and displays the duration taken to cover the distance based on speed.
13. **Class wheelvehicle** (Inherits publicly from vehicle):
14. **Protected Data Member**:
15. int wheels: Represents the number of wheels on the vehicle.
16. **Public Member Functions**:
17. void setwheels(int w): Sets the number of wheels.
18. int getwheels() const: Returns the number of wheels.
19. **Class wingvehicle** (Inherits publicly from vehicle):
20. **Data Member**:
21. int wings: Represents the number of wings.
22. **Public Member Functions**:
23. void setwings(int w): Sets the number of wings.
24. int getwings() const: Returns the number of wings.
25. **Class truck** (Inherits publicly from wheelvehicle):
26. **Data Member**:
27. double carryingload: Represents the load capacity of the truck.
28. **Public Member Functions**:
29. void setload(double l): Sets the carrying load of the truck.
30. double getload() const: Returns the carrying load of the truck.
31. **Main Function**:
32. **Create an Object**:
33. Create an object of the truck class named obj.
34. **Set and Get Values**:
35. Use the member functions to set the number of wheels, speed, and distance covered.
36. Display the number of wheels, speed, distance covered, and computed duration based on the speed and distance.

**Key Concepts to Understand:**

1. **Class Inheritance**: The wheelvehicle and wingvehicle classes inherit from the vehicle class, demonstrating the concept of base and derived classes.
2. **Encapsulation**: Understand how data is encapsulated within classes and how member functions provide access to private/protected data members.
3. **Computational Functions**: Learn how to perform calculations based on the input data.

**Expected Output:**

When the code is executed, it should display:

1. The number of wheels on the truck.
2. The speed of the truck.
3. The distance covered by the truck.
4. The computed duration taken to cover that distance.