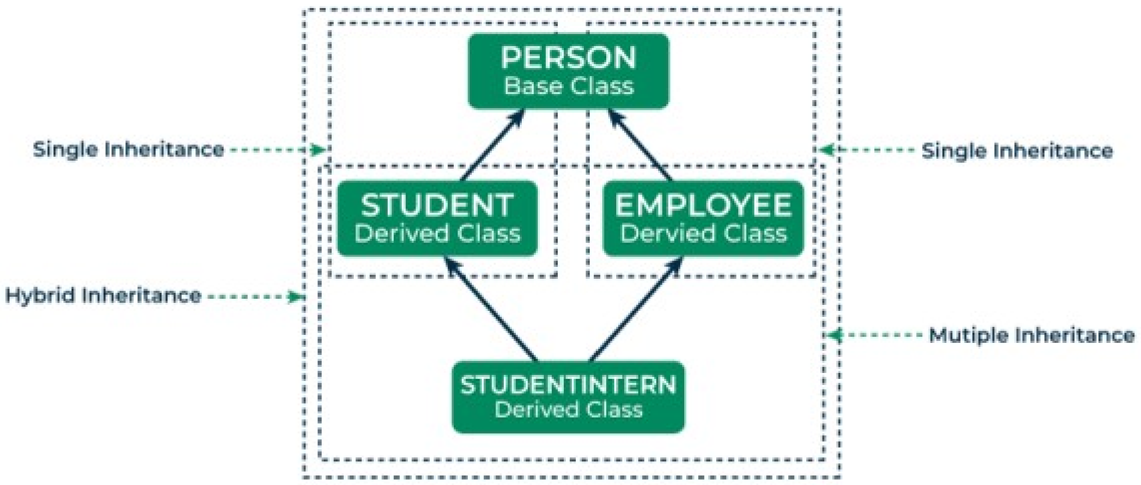
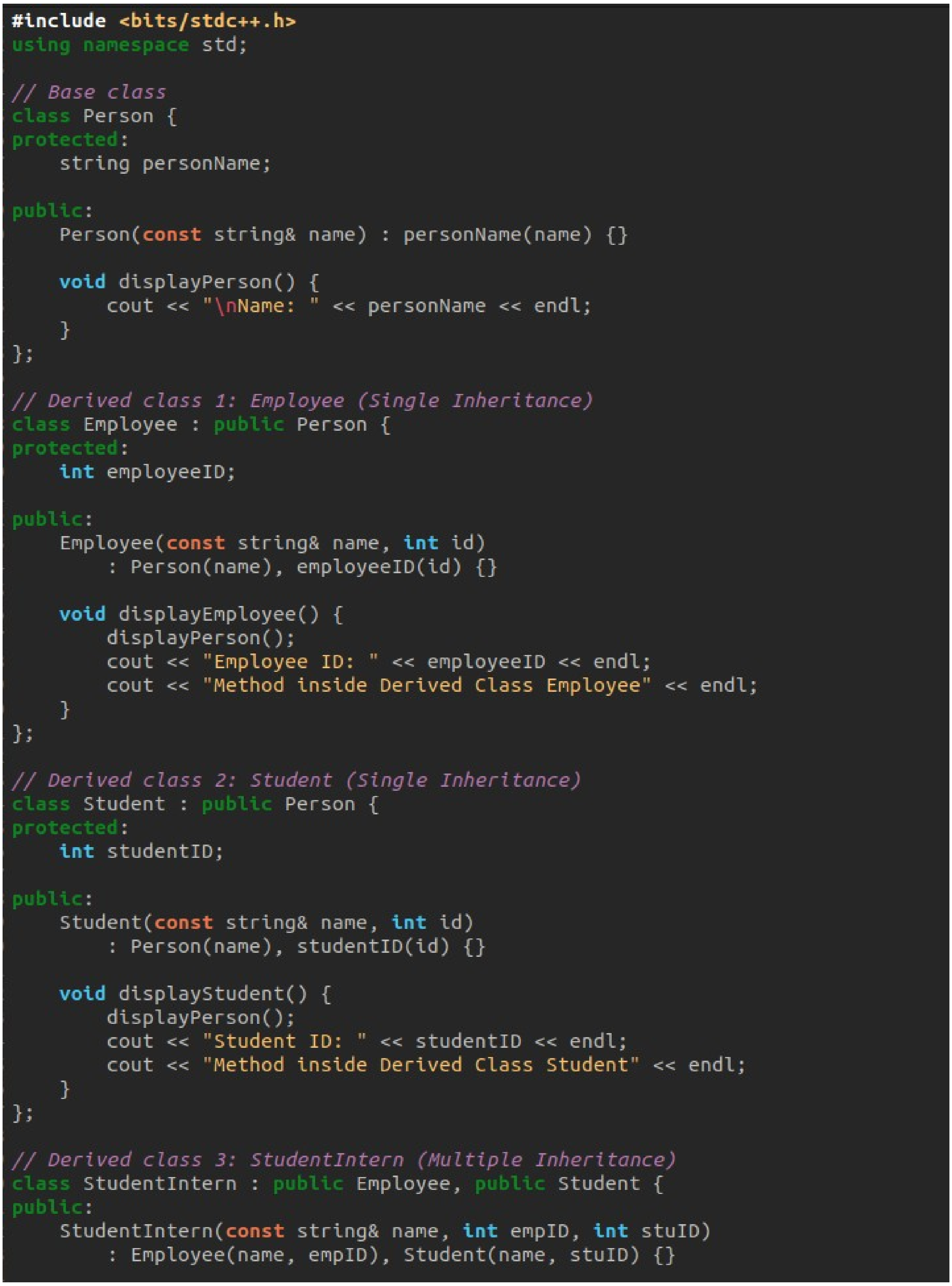
Nithin S
221IT085

IT150 Lab Assignment 2

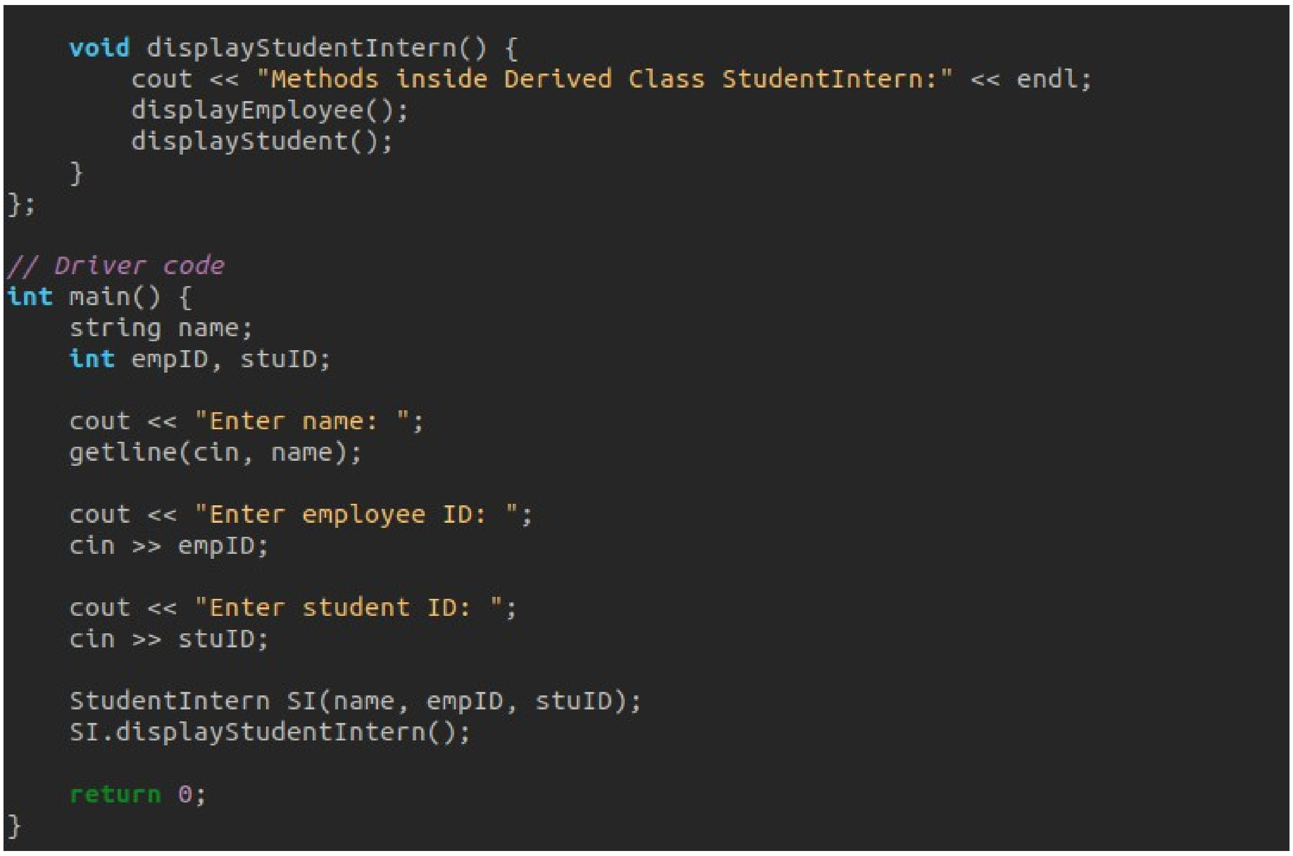
1. 1.Write a C++ program to display the details of the below figure. Use the concept of different types of inheritance techniques to implement the same.



CODE

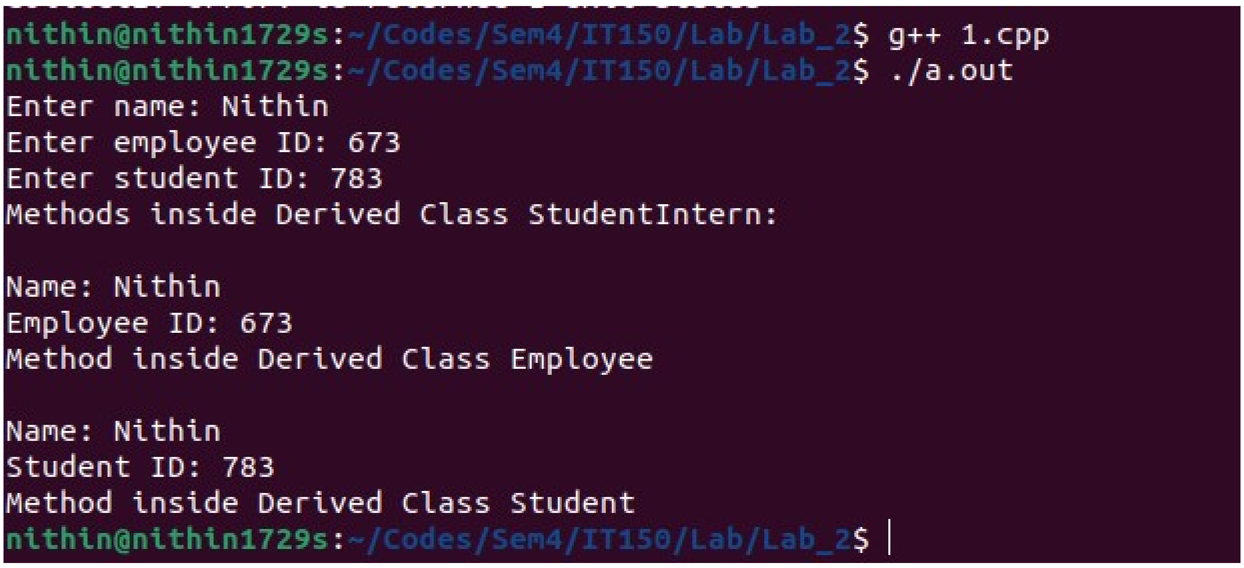


```cpp
#include <bits/stdc++.h>
using namespace std;

// Base class
class Person {
protected:
    string personName;

public:
    Person(const string& name) : personName(name) {}

    void displayPerson() {
        cout << "\nName: " << personName << endl;
    }
};

// Derived class 1: Employee (Single Inheritance)
class Employee : public Person {
protected:
    int employeeID;

public:
    Employee(const string& name, int id)
        : Person(name), employeeID(id) {}

    void displayEmployee() {
        displayPerson();
        cout << "Employee ID: " << employeeID << endl;
        cout << "Method inside Derived Class Employee" << endl;
    }
};

// Derived class 2: Student (Single Inheritance)
class Student : public Person {
protected:
    int studentID;

public:
    Student(const string& name, int id)
        : Person(name), studentID(id) {}

    void displayStudent() {
        displayPerson();
        cout << "Student ID: " << studentID << endl;
        cout << "Method inside Derived Class Student" << endl;
    }
};

// Derived class 3: StudentIntern (Multiple Inheritance)
class StudentIntern : public Employee, public Student {
public:
    StudentIntern(const string& name, int empID, int stuID)
        : Employee(name, empID), Student(name, stuID) {}
```



```cpp
    void displayStudentIntern() {
        cout << "Methods inside Derived Class StudentIntern:" << endl;
        displayEmployee();
        displayStudent();
    }
};

// Driver code
int main() {
    string name;
    int empID, stuID;

    cout << "Enter name: ";
    getline(cin, name);

    cout << "Enter employee ID: ";
    cin >> empID;

    cout << "Enter student ID: ";
    cin >> stuID;

    StudentIntern SI(name, empID, stuID);
    SI.displayStudentIntern();

    return 0;
}
```

OUTPUT



```
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_2$ g++ 1.cpp
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_2$ ./a.out
Enter name: Nithin
Enter employee ID: 673
Enter student ID: 783
Methods inside Derived Class StudentIntern:

Name: Nithin
Employee ID: 673
Method inside Derived Class Employee

Name: Nithin
Student ID: 783
Method inside Derived Class Student
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_2$
```

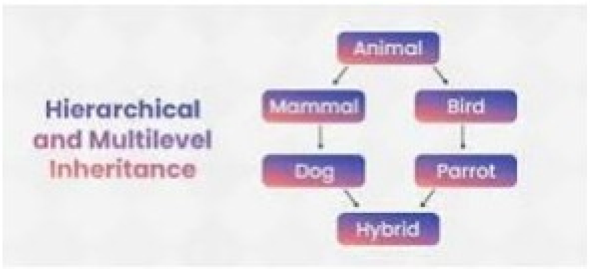
Explanation:

Inheritance: The Employee and Student classes inherit from the Person class, demonstrating single inheritance. This means that both Employee and Student inherit the attributes and methods of the Person class.

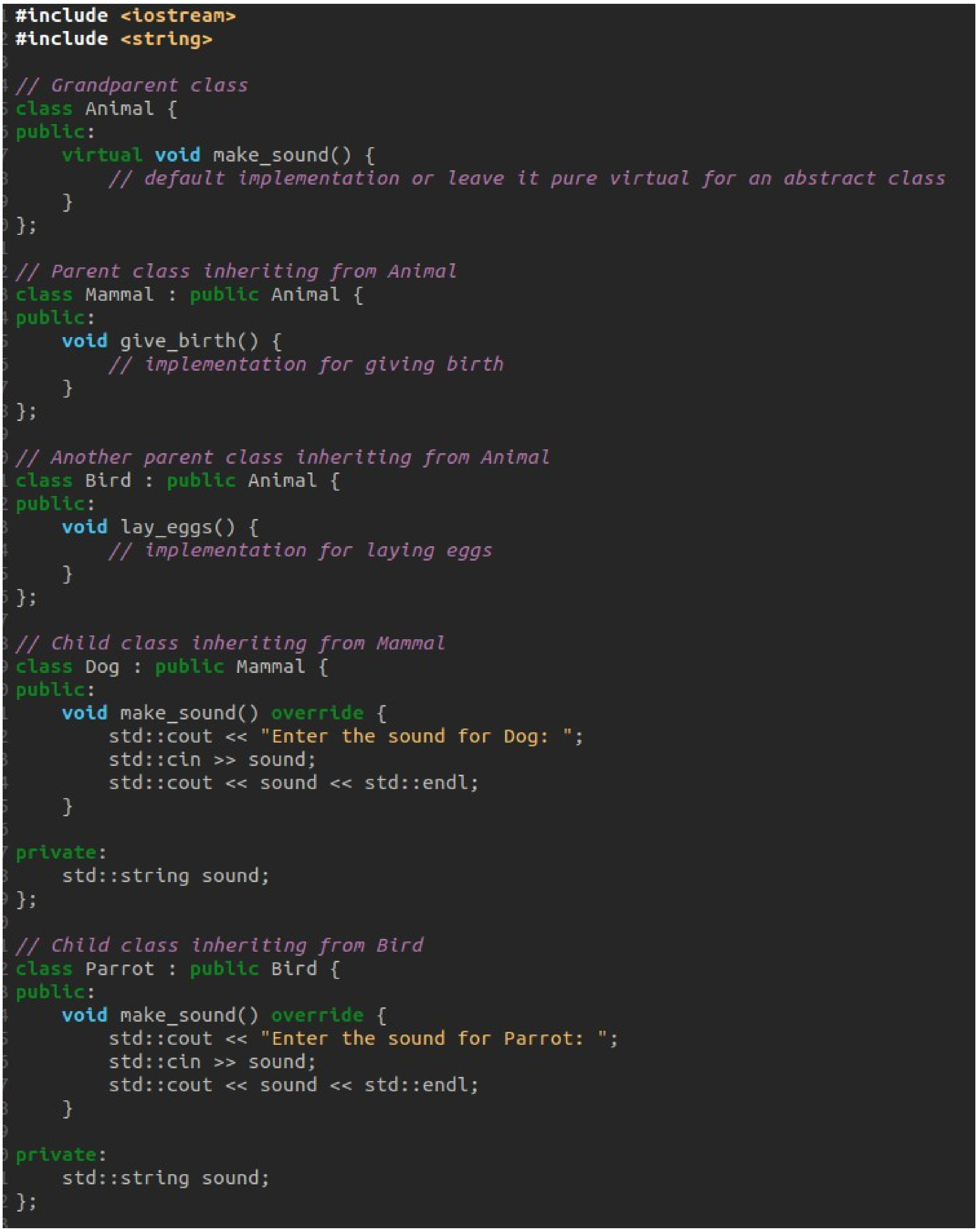
Single Inheritance: Both Employee and Student are derived from the Person base class, acquiring its member variables and functions, such as displayPerson(). This promotes code reusability by allowing the derived classes to use or override the common functionality defined in the base class.

Multiple Inheritance: The StudentIntern class showcases multiple inheritance by inheriting from both Employee and Student classes. This class combines features from both parent classes, emphasizing flexibility in designing class hierarchies.

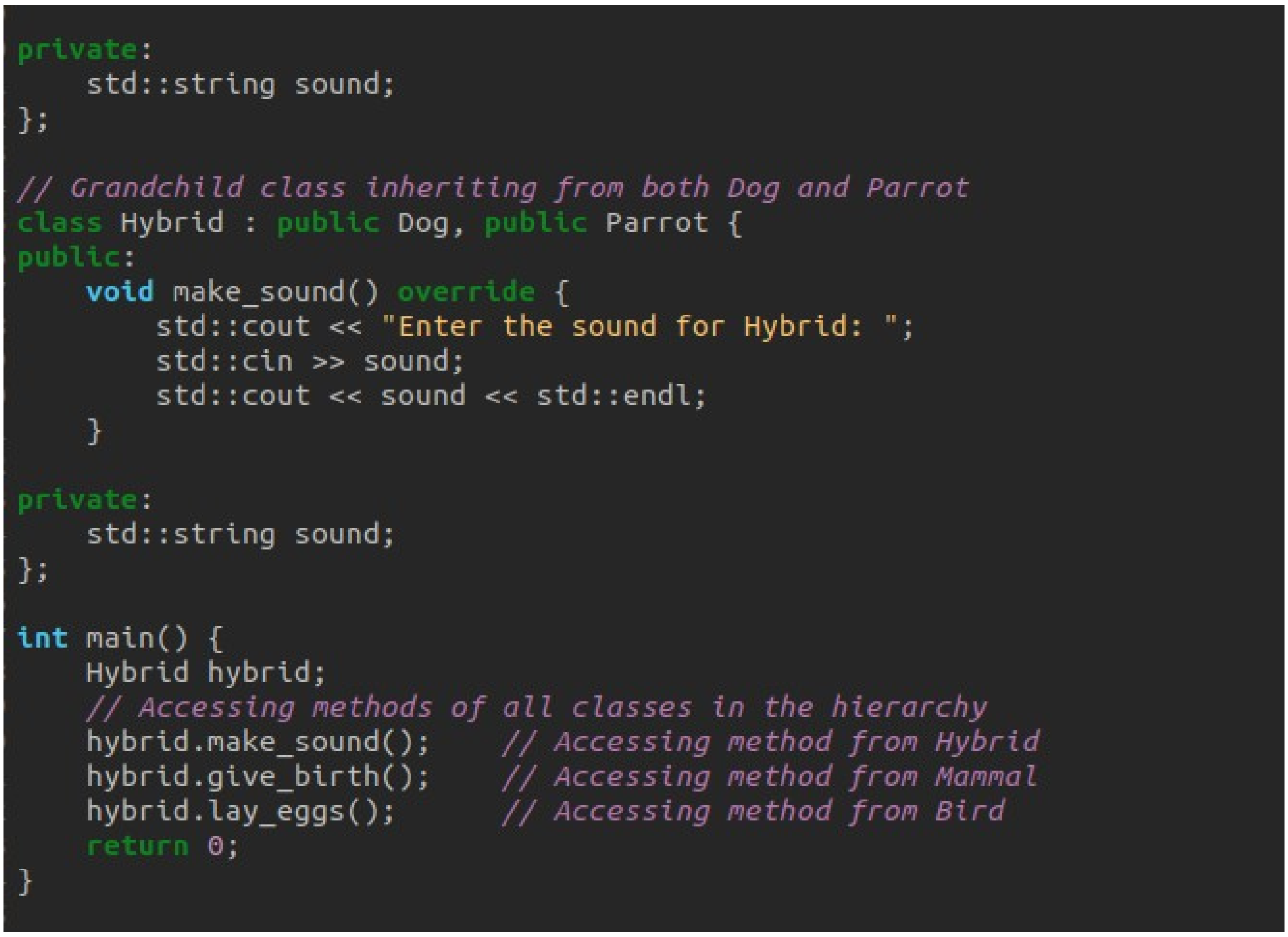
2. Write a C++ program to display the details in the below figure. Use hierarical and multilevel techniques to implement the same.



CODE

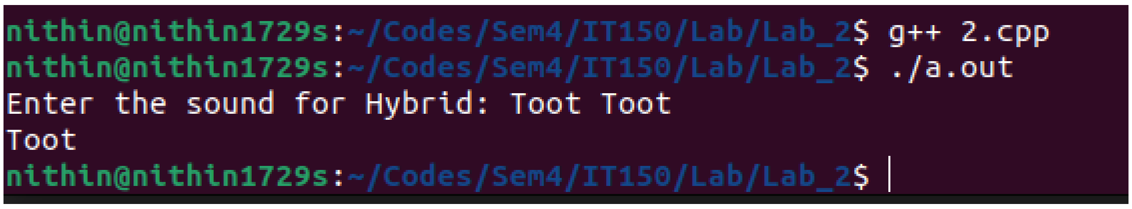


```cpp
#include <iostream>
#include <string>

// Grandparent class
class Animal {
public:
    virtual void make_sound() {
        // default implementation or leave it pure virtual for an abstract class
    }
};

// Parent class inheriting from Animal
class Mammal : public Animal {
public:
    void give_birth() {
        // implementation for giving birth
    }
};

// Another parent class inheriting from Animal
class Bird : public Animal {
public:
    void lay_eggs() {
        // implementation for laying eggs
    }
};

// Child class inheriting from Mammal
class Dog : public Mammal {
public:
    void make_sound() override {
        std::cout << "Enter the sound for Dog: ";
        std::cin >> sound;
        std::cout << sound << std::endl;
    }

private:
    std::string sound;
};

// Child class inheriting from Bird
class Parrot : public Bird {
public:
    void make_sound() override {
        std::cout << "Enter the sound for Parrot: ";
        std::cin >> sound;
        std::cout << sound << std::endl;
    }

private:
    std::string sound;
};
```



```
private:
    std::string sound;
};

// Grandchild class inheriting from both Dog and Parrot
class Hybrid : public Dog, public Parrot {
public:
    void make_sound() override {
        std::cout << "Enter the sound for Hybrid: ";
        std::cin >> sound;
        std::cout << sound << std::endl;
    }

private:
    std::string sound;
};

int main() {
    Hybrid hybrid;
    // Accessing methods of all classes in the hierarchy
    hybrid.make_sound();     // Accessing method from Hybrid
    hybrid.give_birth();     // Accessing method from Mammal
    hybrid.lay_eggs();       // Accessing method from Bird
    return 0;
}
```

OUTPUT



```
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_2$ g++ 2.cpp
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_2$ ./a.out
Enter the sound for Hybrid: Toot Toot
Toot
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_2$ 
```

Explanation:

Animals serve as the grandparent class.

Mammal and Bird are parent classes, both directly inheriting from Animal.

Dog is a child class inheriting from Mammal.

Parrot is a child class inheriting from Bird.

Hybrid is a grandchild class inheriting from both Dog and Parrot.

Here’s how the inheritance works:

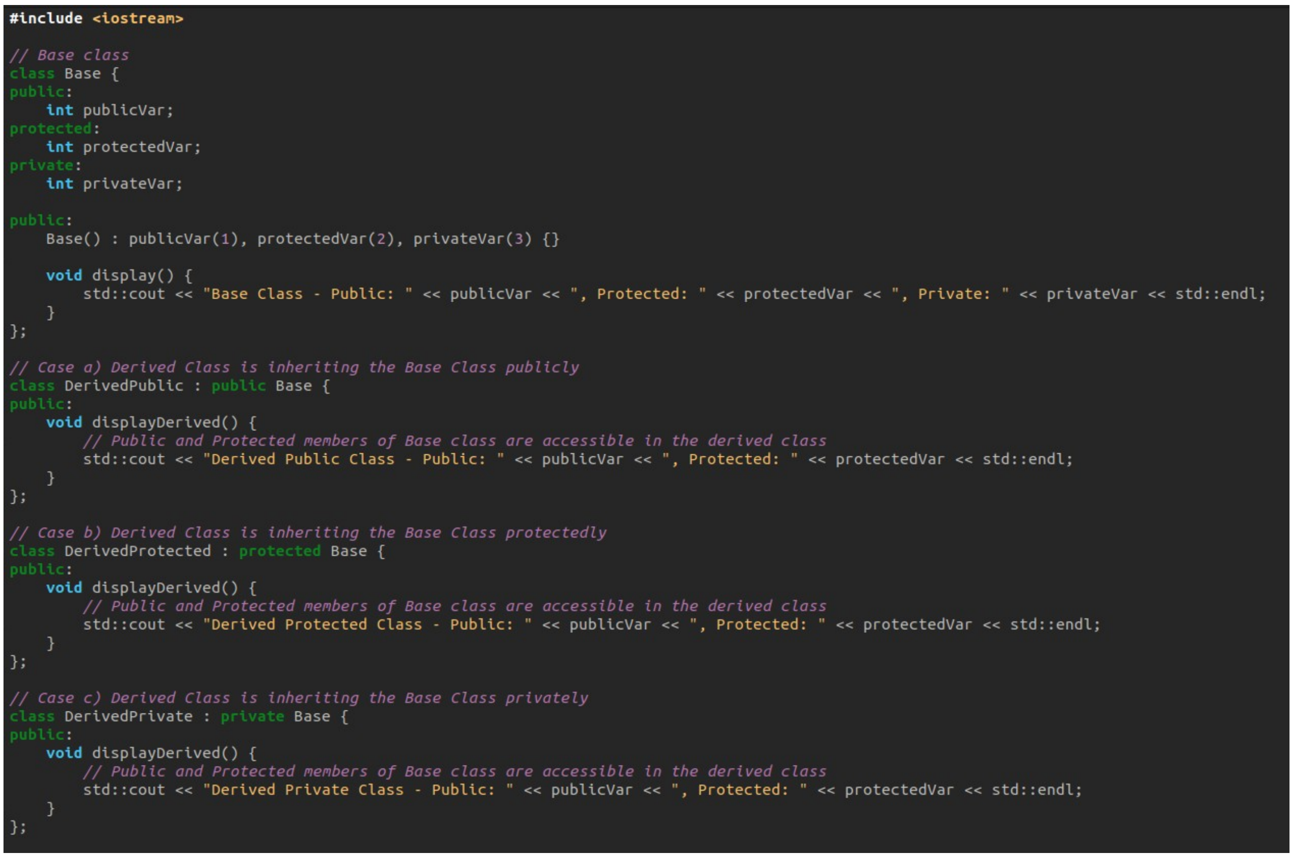
Mammal and Bird classes form a hierarchical inheritance, as they both inherit directly from Animal.

Dog and Parrot classes represent multilevel inheritance, inheriting from Mammal and Bird respectively.

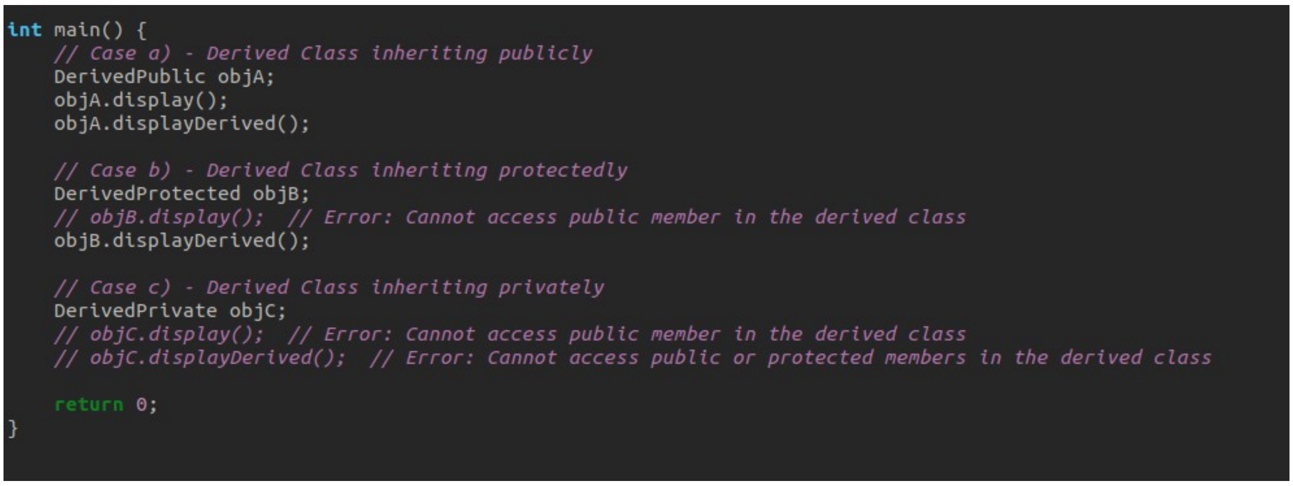
Hybrid class demonstrates a combination of multilevel and hierarchical inheritance by inheriting from both Dog and Parrot.

3. 3. Derive a child class from the base class in different access modes. Write a C++ Program to show access
to Private, Public and Protected using Inheritance for the following 3 cases:
a) Derived Class is inheriting the Base Class publicly
b) Derived Class is inheriting the Base Class protectedly
c) Derived Class is inheriting the Base Class privately

CODE

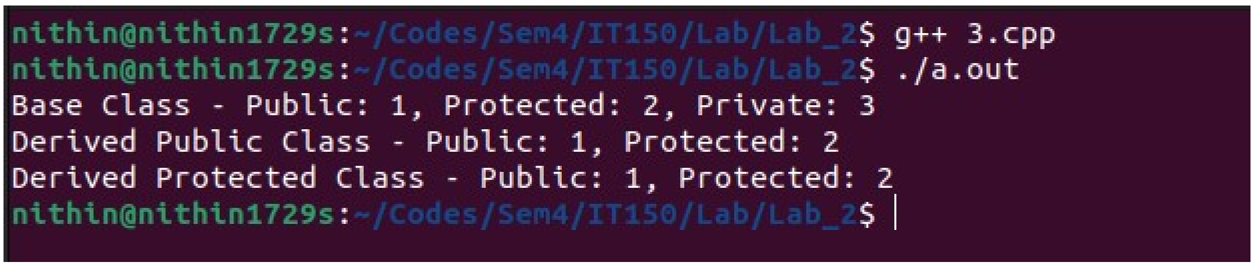


```cpp
#include <iostream>

// Base class
class Base {
public:
    int publicVar;
protected:
    int protectedVar;
private:
    int privateVar;

public:
    Base() : publicVar(1), protectedVar(2), privateVar(3) {}

    void display() {
        std::cout << "Base Class - Public: " << publicVar << ", Protected: " << protectedVar << ", Private: " << privateVar << std::endl;
    }
};

// Case a) Derived Class is inheriting the Base Class publicly
class DerivedPublic : public Base {
public:
    void displayDerived() {
        // Public and Protected members of Base class are accessible in the derived class
        std::cout << "Derived Public Class - Public: " << publicVar << ", Protected: " << protectedVar << std::endl;
    }
};

// Case b) Derived Class is inheriting the Base Class protectedly
class DerivedProtected : protected Base {
public:
    void displayDerived() {
        // Public and Protected members of Base class are accessible in the derived class
        std::cout << "Derived Protected Class - Public: " << publicVar << ", Protected: " << protectedVar << std::endl;
    }
};

// Case c) Derived Class is inheriting the Base Class privately
class DerivedPrivate : private Base {
public:
    void displayDerived() {
        // Public and Protected members of Base class are accessible in the derived class
        std::cout << "Derived Private Class - Public: " << publicVar << ", Protected: " << protectedVar << std::endl;
    }
};
```



```cpp
int main() {
    // Case a) - Derived Class inheriting publicly
    DerivedPublic objA;
    objA.display();
    objA.displayDerived();

    // Case b) - Derived Class inheriting protectedly
    DerivedProtected objB;
    // objB.display();  // Error: Cannot access public member in the derived class
    objB.displayDerived();

    // Case c) - Derived Class inheriting privately
    DerivedPrivate objC;
    // objC.display();  // Error: Cannot access public member in the derived class
    // objC.displayDerived();  // Error: Cannot access public or protected members in the derived class

    return 0;
}
```

OUTPUT



```
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_2$ g++ 3.cpp
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_2$ ./a.out
Base Class - Public: 1, Protected: 2, Private: 3
Derived Public Class - Public: 1, Protected: 2
Derived Protected Class - Public: 1, Protected: 2
nithin@nithin1729s:~/Codes/Sem4/IT150/Lab/Lab_2$ 
```

Explanation:

The Base class has three member variables: publicVar, protectedVar, and privateVar with different access specifiers.

Three derived classes (DerivedPublic, DerivedProtected, DerivedPrivate) inherit from Base class with different access modes.

In the main function, instances of each derived class are created to demonstrate the access to base class members in each case.

The display method of the base class is used to show the access to the members in each derived class. Note that in cases b) and c), accessing the public member directly in the derived class results in an error, but the derived class can still access it through its member functions.