INHERITANCE

Topic 1 –term 2

description

- This is the ability to define new classes from already existing class or classes
- Inheritance is an "is-a" relationship where a new class is referred to as a derived class/sub-class or a child class. The existing class is referred to as a base class/parent class or a super class. The derived class <u>inherits properties of the base class</u>
- In OOP , the concept of inheritance <u>provide the idea of reusability</u>, this means that we can add additional features to an existing class with out modifying it,
- This is possible by deriving a new class from the existing one, the class will have combined features of both classes hence the classes are reusable.
- Through inheritance we can <u>eliminate redundant</u> code and extend the use of existing code

- Inheritance is a process by which we can acquire the characteristics of an existing entity and form a new entity by adding more features to it.
- In terms of C++, inheritance is creating a new class by deriving it from an existing class so that this new class has the properties of its parent class as well as its own.

Forms of inheritance

- Extension – The sub class will add behavior to the inherited behavior (adds new functionality to parents class properties)
- Combination – The child class inherits properties from more than one parent
- Limitation – The child class restricts the use of some form of behavior
- Variance – The child class and the parent class relationship is arbitrary/random
- Whole – The child class inherits all properties from the parent class
- Partial - The child class inherits some of the parents properties

- Contraction- Because the sub-class can override some behavior from the super-class to make it fit a specialized situation, this sub class can be set to be a (contracted)shrinkage of the parent class

Cont..

- Note
  -Private members of base class can not be inherited by the derived class
  -Public members of base class can be inherited by derived class either as public or
  private members of the derived class
  -Derived class can have additional members
  -Derived class can redefine as public members of the base class

The summary of the different access types :

ACCESS	PUBLIC	PROTECTED	PRIVATE
MEMBERS OF THE SAME CLASS	**YES**	**YES**	**YES**
MEMBERS OF DERIVED CLASSES	**YES**	**YES**	**NO**
NOT MEMBERS	**YES**	**NO**	**NO**

Object slicing

Object slicing is used to describe the situation when you assign an object of a derived class to an instance of a base class. This causes a loss of methods and member variables for the derived class object. This is termed as information being sliced away. This is known as member slicing.

This is where you assign an object of a derived class to an instance of a base class, thereby losing part of the information - some of it is "sliced" away. For example

```
 class A
{
 int x;
};
class B : public A
{
 int y;
 }; // So an object of type B has two data members,  x and y .
```

nb

* Virtual base class –Used when there is need to implement multiple inheritance e.g. a child inherits properties of a grandparent through two parents

poy

Inheritance

Inheritance in C++

The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important feature of Object Or ented Programming.

Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.

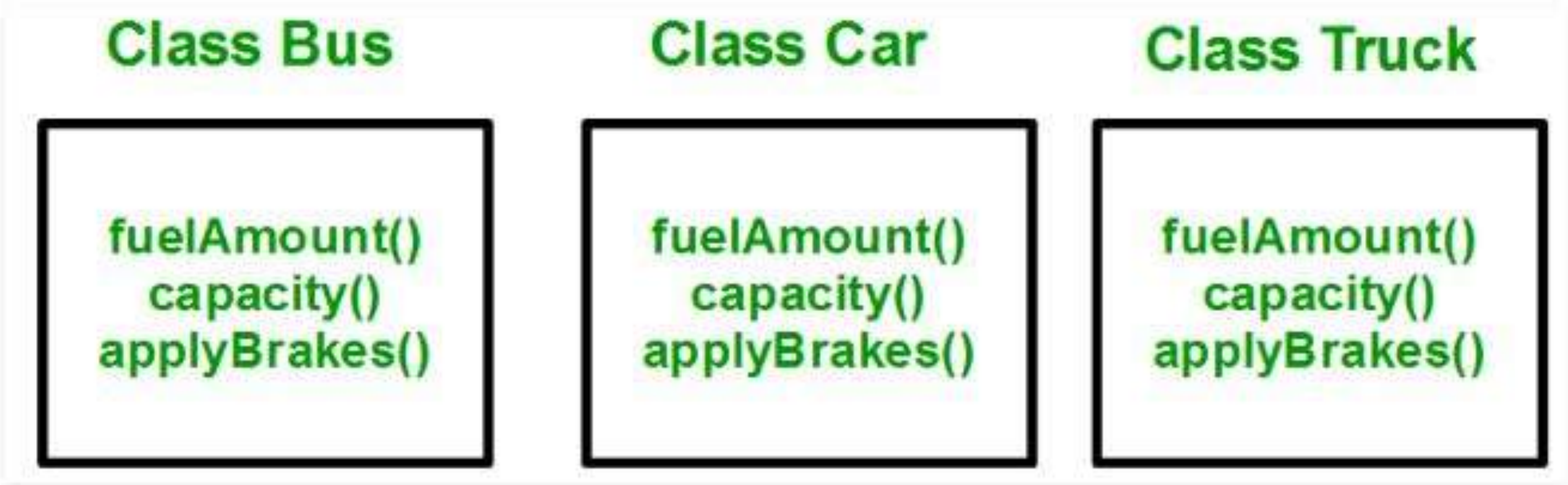
Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.

The article is divided into following subtopics:

1. Why and when to use inheritance?
2. Modes of Inheritance
3. Types of Inheritance

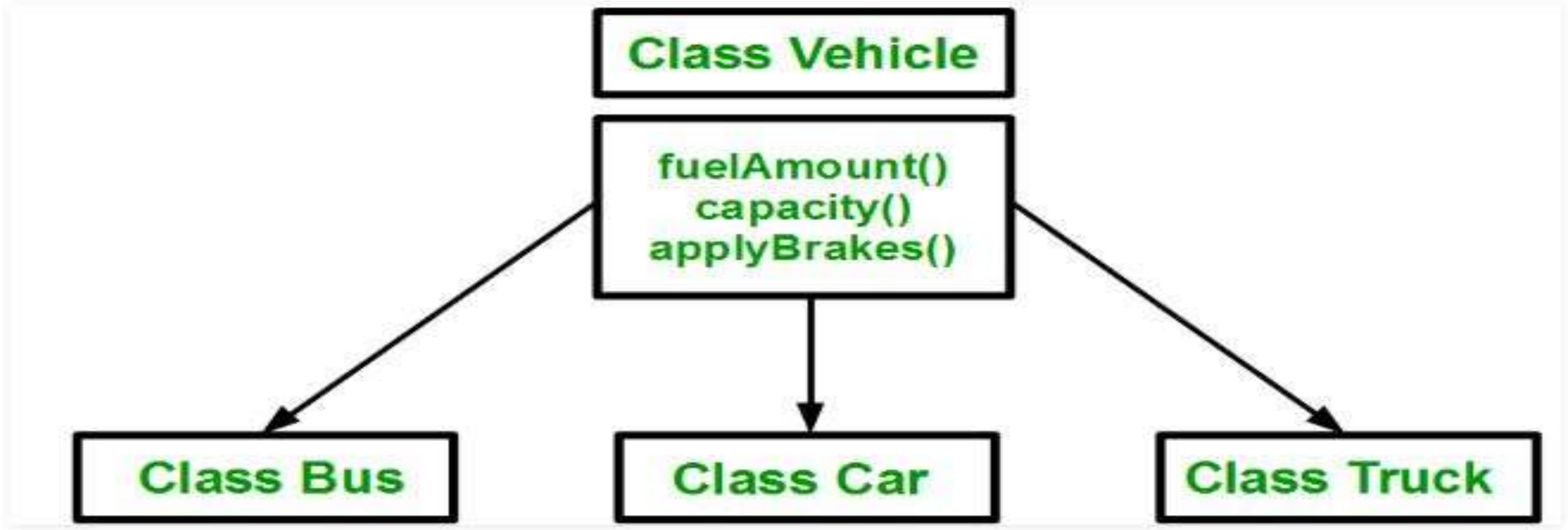
Why and when to use inheritance?

Consider a group of vehicles. You need to create classes for Bus, Car and Truck. The methods fuelAmount(), capacity(), applyBrakes() will be same for all of the three classes. If we create these classes avoiding inheritance then we have to write all of these functions in each of the three classes as shown in below figure:



2

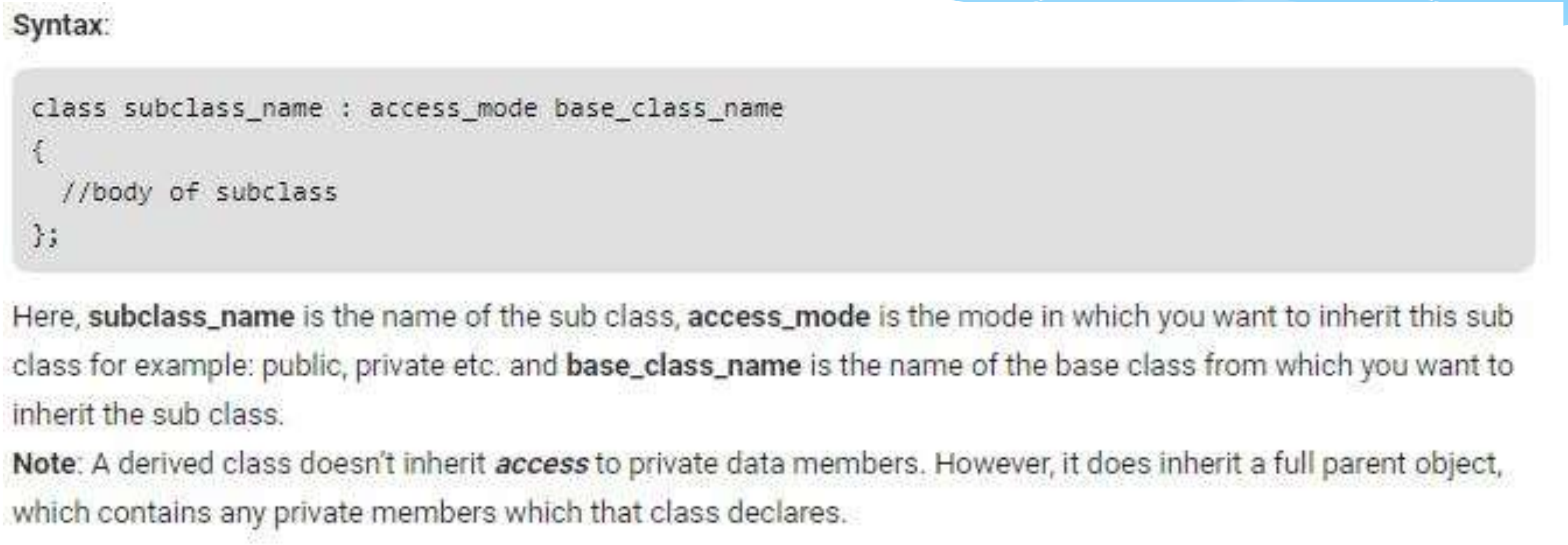
You can clearly see that above process results in duplication of same code 3 times. This increases the chances of error and data redundancy. To avoid this type of situation, inheritance is used. If we create a class Vehicle and write these three functions in it and inherit the rest of the classes from the vehicle class, then we can simply avoid the duplication of data and increase re-usability. Look at the below diagram in which the three classes are inherited from vehicle class:



Using inheritance, we have to write the functions only one time instead of three times as we have inherited rest of the three classes from base class(Vehicle).

Implementing inheritance in C++: For creating a sub-class which is inherited from the base class we have to follow the below syntax.

3

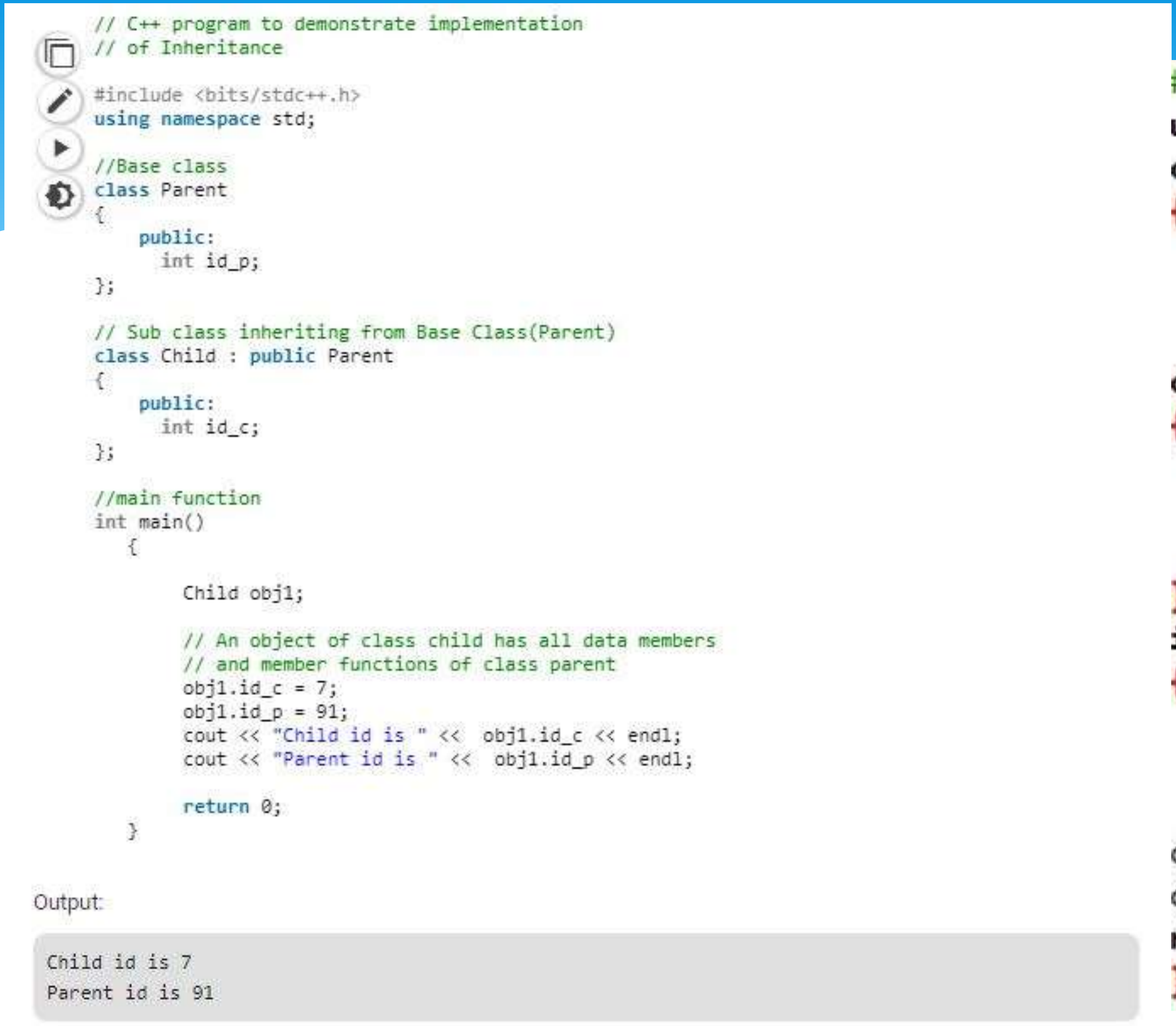


Syntax:

```
class subclass_name : access_mode base_class_name
{
  //body of subclass
};
```

Here, **subclass_name** is the name of the sub class, **access_mode** is the mode in which you want to inherit this sub class for example: public, private etc. and **base_class_name** is the name of the base class from which you want to inherit the sub class.

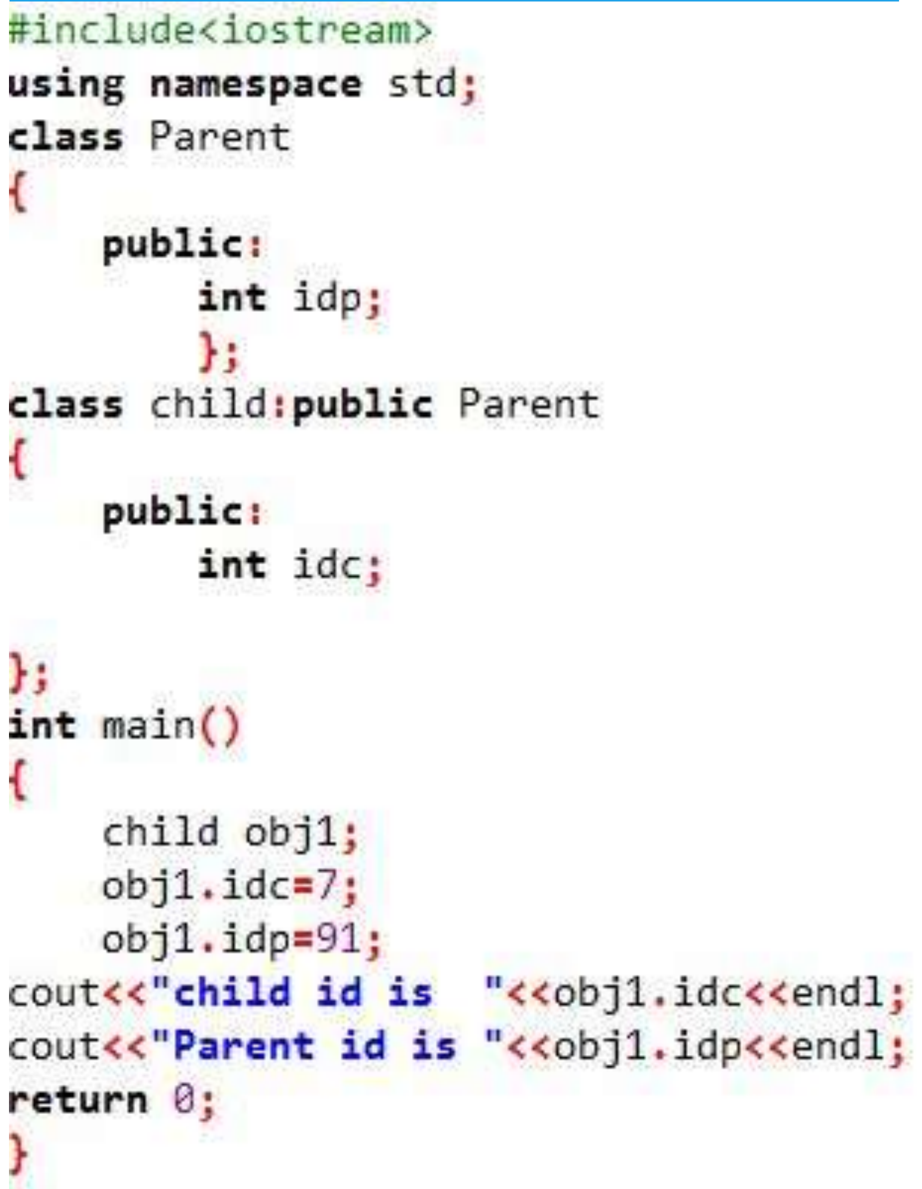
Note: A derived class doesn't inherit ***access*** to private data members. However, it does inherit a full parent object, which contains any private members which that class declares.



```cpp
// C++ program to demonstrate implementation
// of Inheritance

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

//Base class
class Parent
{
    public:
      int id_p;
};

// Sub class inheriting from Base Class(Parent)
class Child : public Parent
{
    public:
      int id_c;
};

//main function
int main()
   {

        Child obj1;

        // An object of class child has all data members
        // and member functions of class parent
        obj1.id_c = 7;
        obj1.id_p = 91;
        cout << "Child id is " <<  obj1.id_c << endl;
        cout << "Parent id is " <<  obj1.id_p << endl;

        return 0;
   }
```

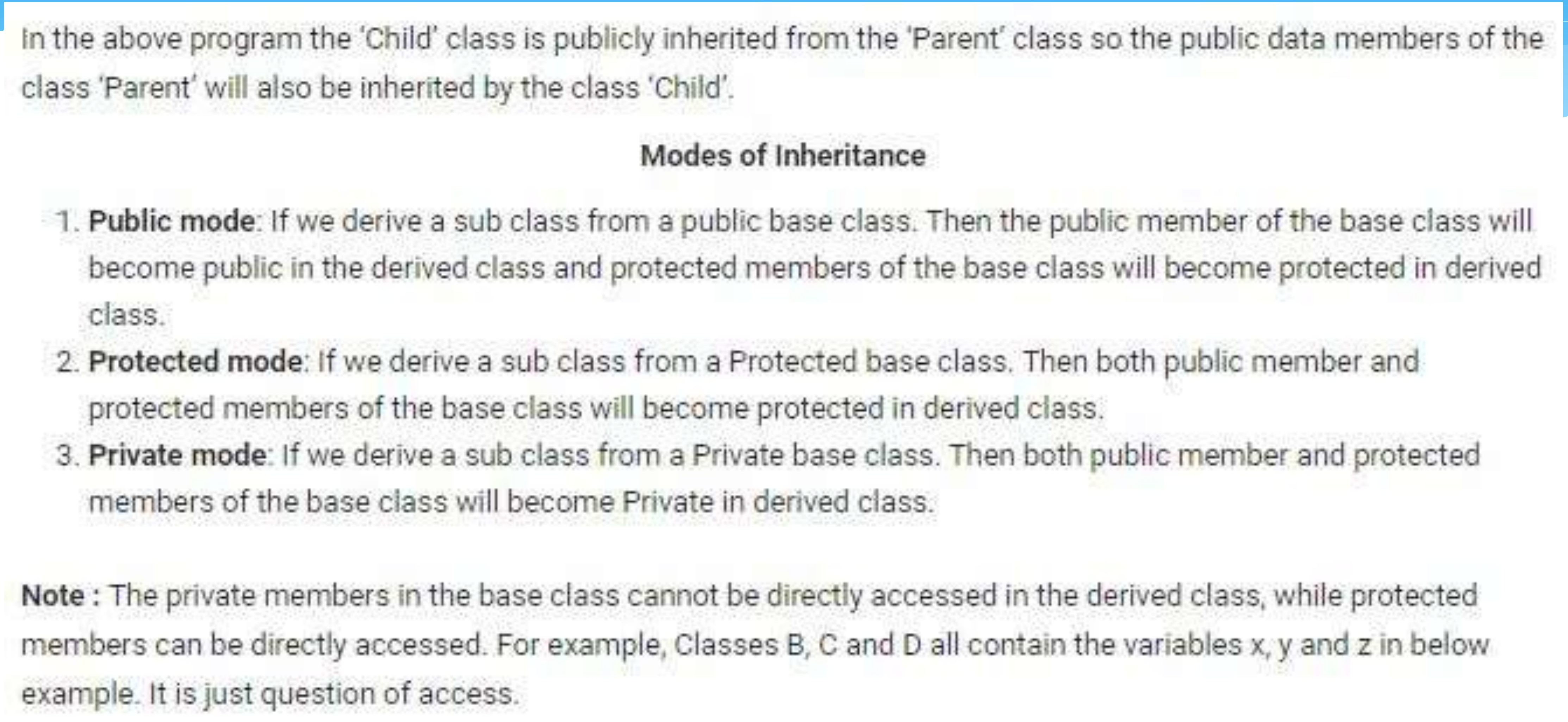
Output:

```
Child id is 7
Parent id is 91
```



```cpp
#include<iostream>
using namespace std;
class Parent
{
    public:
        int idp;
        };
class child:public Parent
{
    public:
        int idc;

};
int main()
{
    child obj1;
    obj1.idc=7;
    obj1.idp=91;
cout<<"child id is  "<<obj1.idc<<endl;
cout<<"Parent id is "<<obj1.idp<<endl;
return 0;
}
```

5

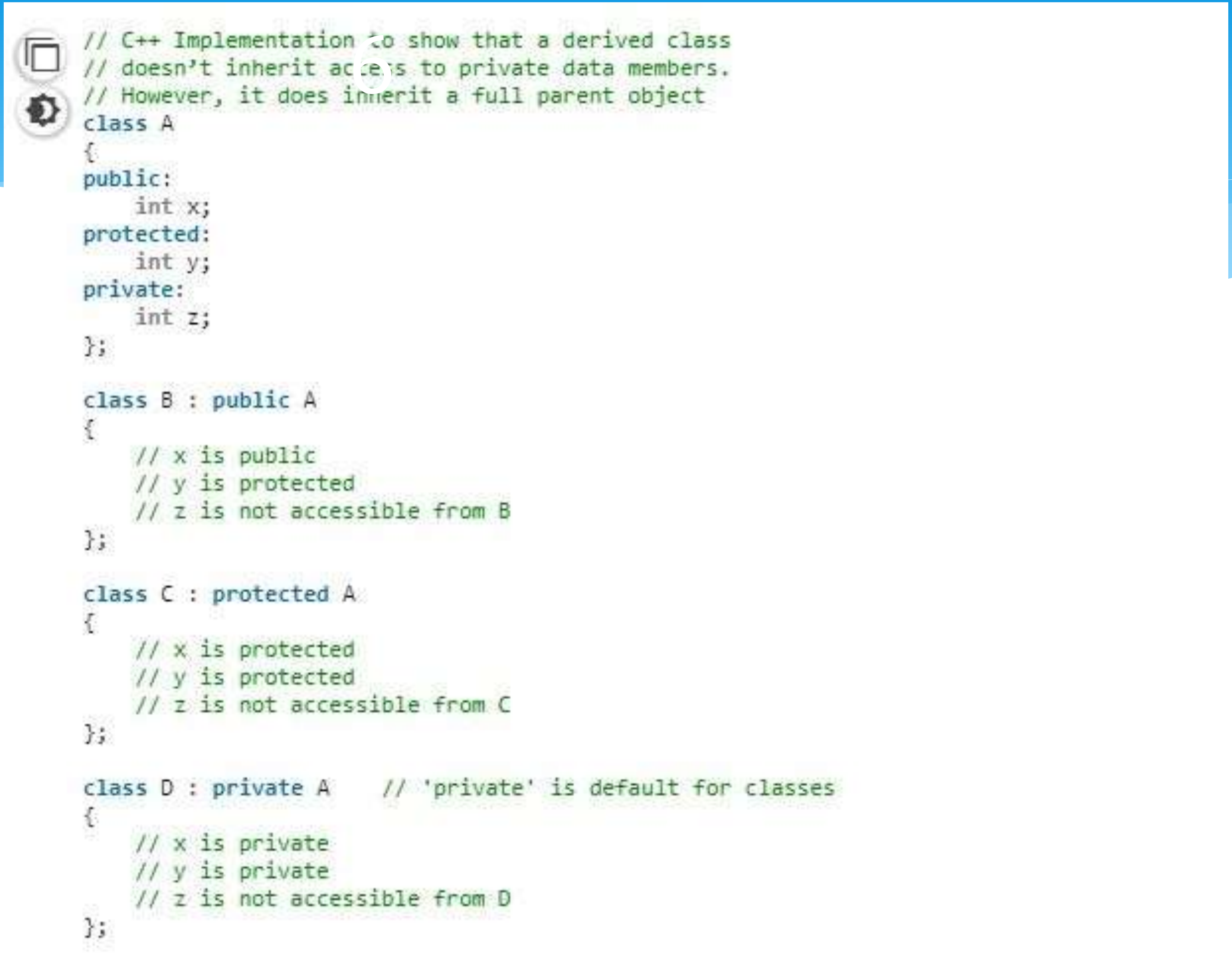


In the above program the 'Child' class is publicly inherited from the 'Parent' class so the public data members of the class 'Parent' will also be inherited by the class 'Child'.

Modes of Inheritance

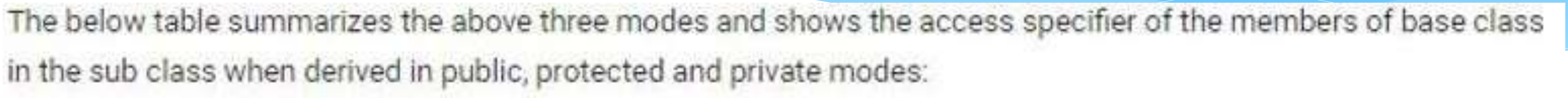
1. **Public mode**: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.
2. **Protected mode**: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.
3. **Private mode**: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class.

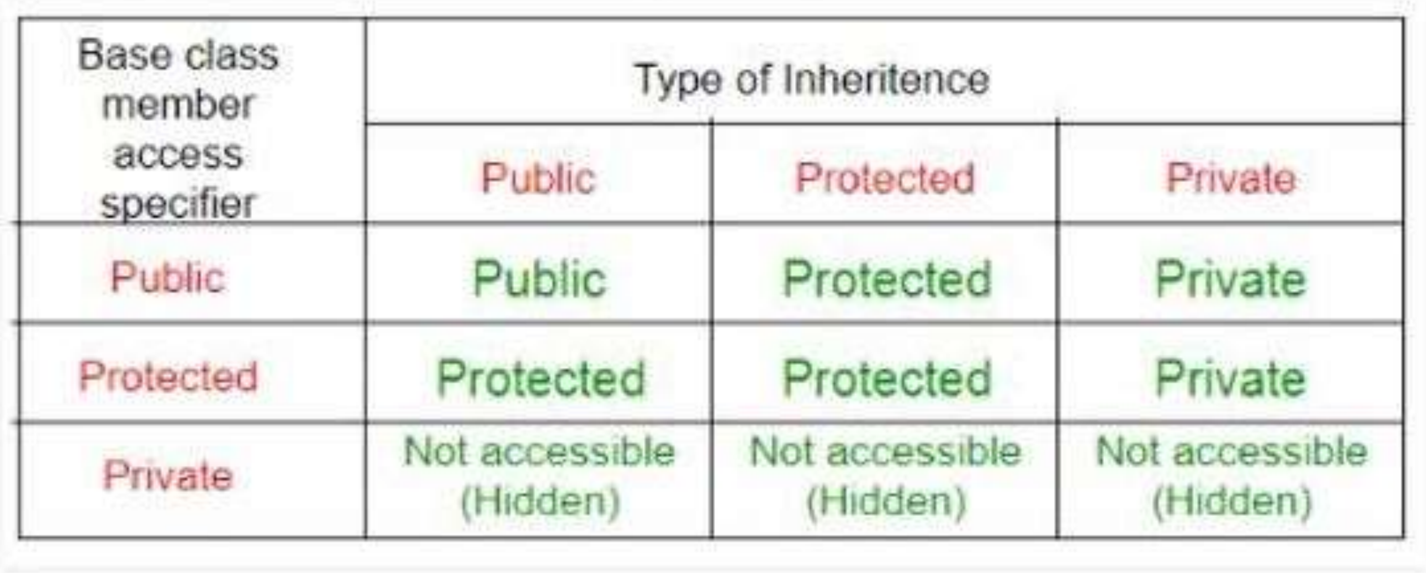
Note : The private members in the base class cannot be directly accessed in the derived class, while protected members can be directly accessed. For example, Classes B, C and D all contain the variables x, y and z in below example. It is just question of access.



```
// C++ Implementation to show that a derived class
// doesn't inherit access to private data members.
// However, it does inherit a full parent object
class A
{
public:
    int x;
protected:
    int y;
private:
    int z;
};

class B : public A
{
    // x is public
    // y is protected
    // z is not accessible from B
};

class C : protected A
{
    // x is protected
    // y is protected
    // z is not accessible from C
};

class D : private A    // 'private' is default for classes
{
    // x is private
    // y is private
    // z is not accessible from D
};
```

7

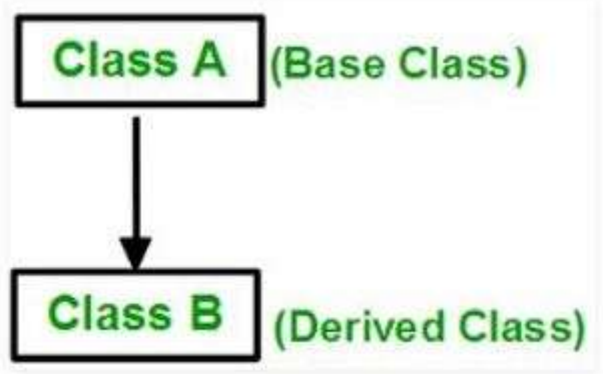

The below table summarizes the above three modes and shows the access specifier of the members of base class in the sub class when derived in public, protected and private modes:

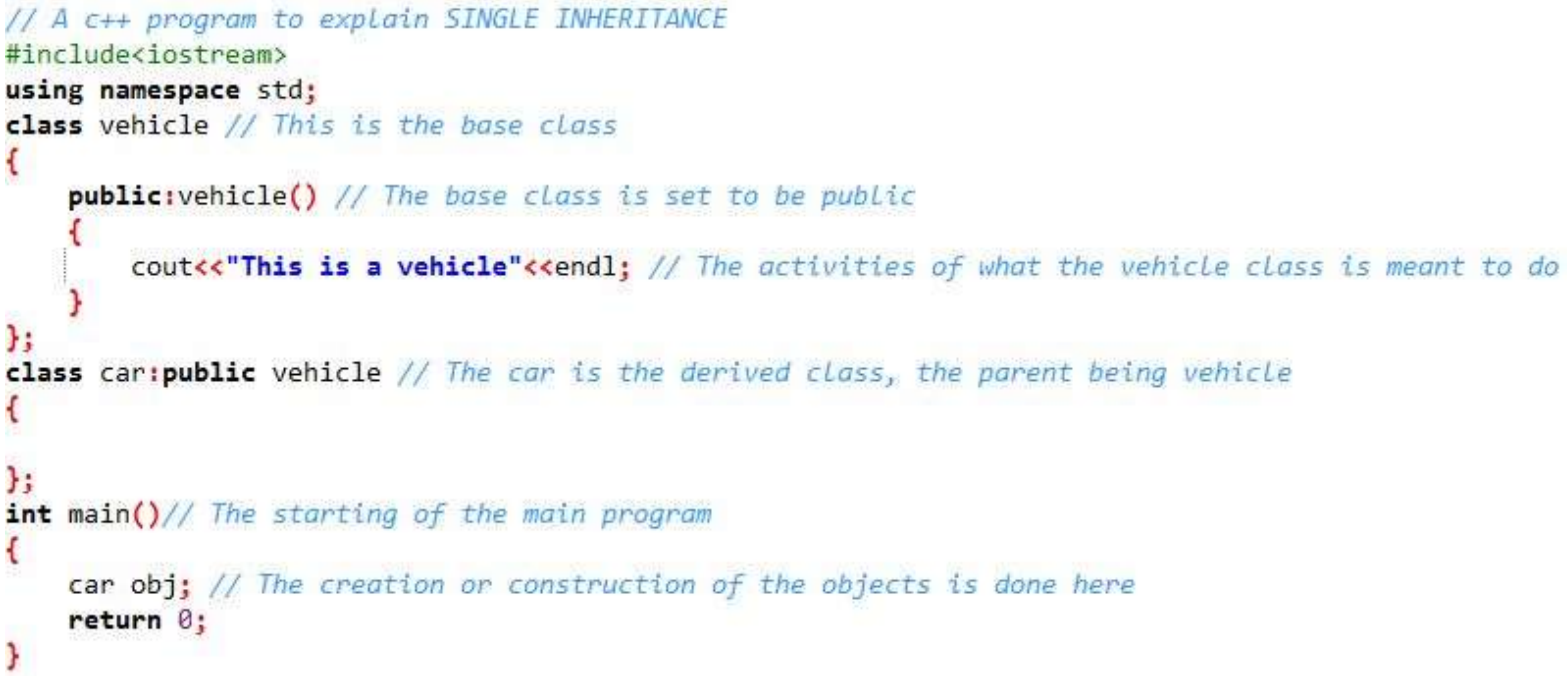


Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

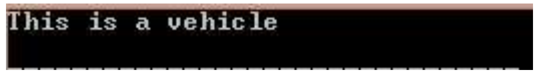
Single inheritance

1. **Single Inheritance**: In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.





```
// A c++ program to explain SINGLE INHERITANCE
#include<iostream>
using namespace std;
class vehicle // This is the base class
{
    public:vehicle() // The base class is set to be public
    {
        cout<<"This is a vehicle"<<endl; // The activities of what the vehicle class is meant to do
    }
};
class car:public vehicle // The car is the derived class, the parent being vehicle
{

};
int main()// The starting of the main program
{
    car obj; // The creation or construction of the objects is done here
    return 0;
}
```

The Program Output: 

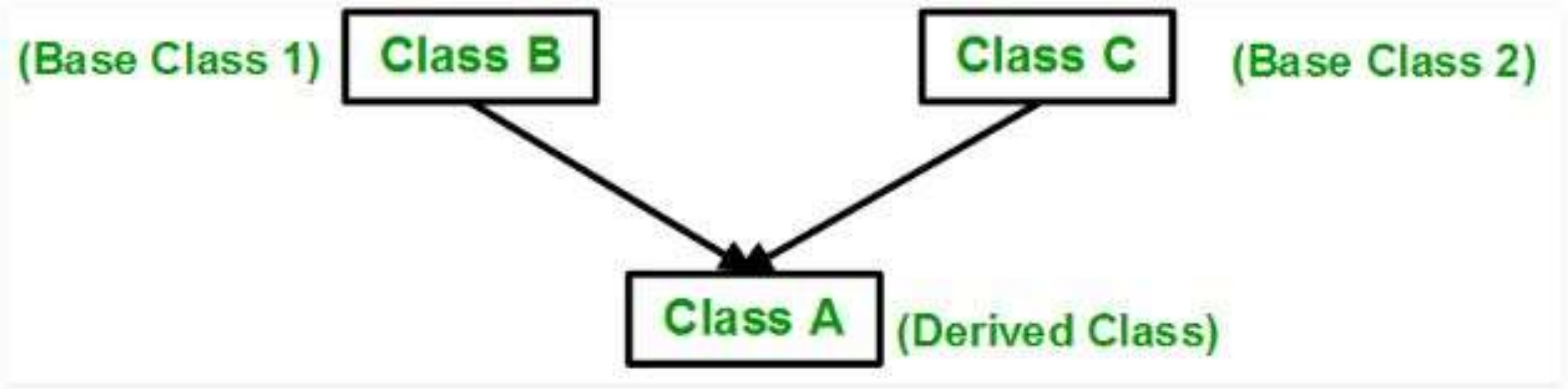
```
This is a vehicle
```

Question 1

a) Write a C++ program that illustrates the following statements in a passport processing Management information system, name the program as per the applied type of inheritance (6 Marks)

- A computer system performs the following with 2 classes
- 1. Citizenship class is public and is to display the following message "The applicant is a Kenyan Citizen with a valid Birth certificate"
- 2. Bcertificate class is a subclass to Citizenship class

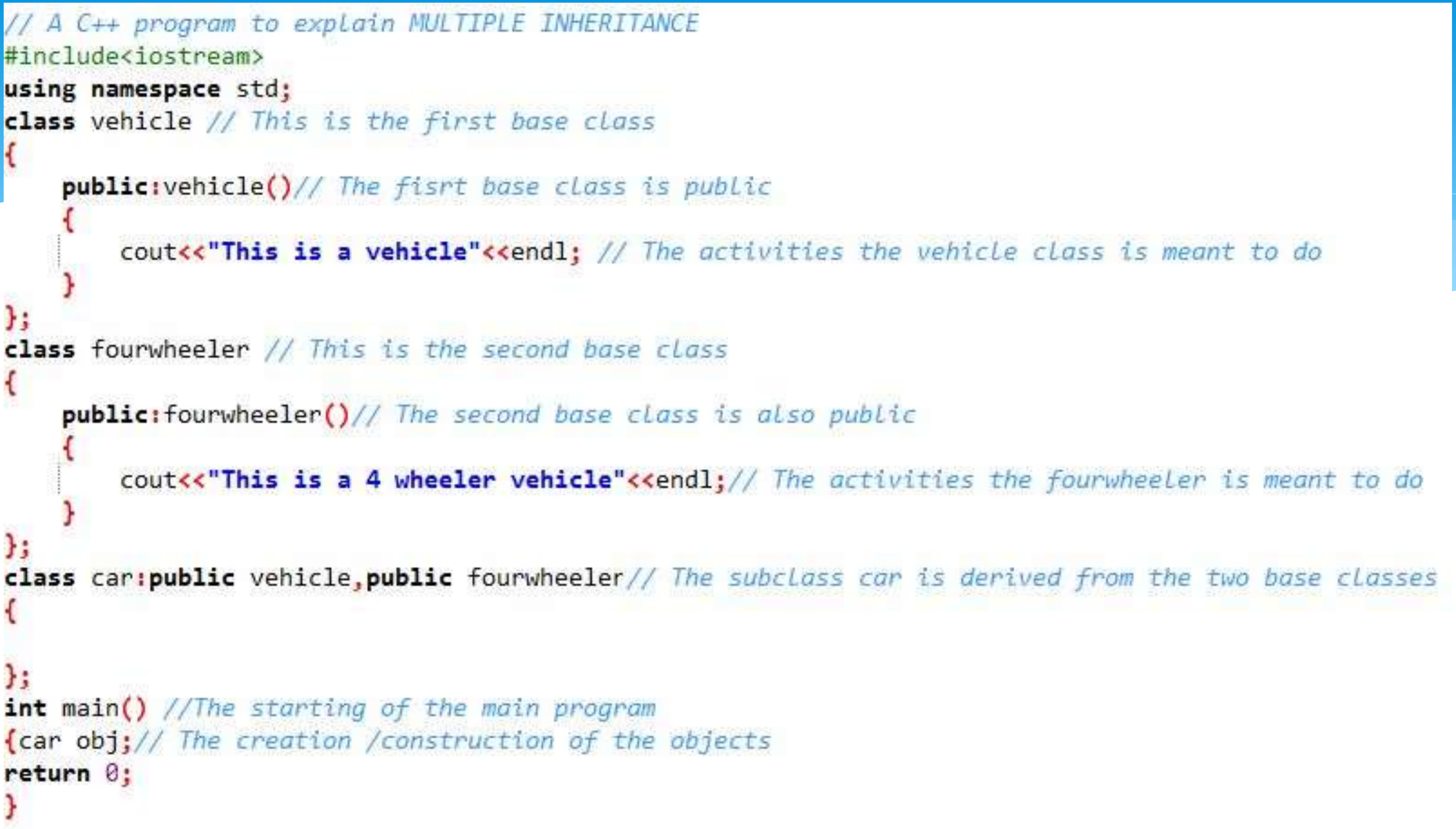
2. **Multiple Inheritance:** Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one **sub class** is inherited from more than one **base classes**.



Syntax:

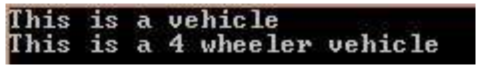
```
class subclass_name : access_mode base_class1, access_mode base_class2, ....
{
  //body of subclass
};
```

Here, the number of base classes will be separated by a comma (', ') and access mode for every base class must be specified.



```cpp
// A C++ program to explain MULTIPLE INHERITANCE
#include<iostream>
using namespace std;
class vehicle // This is the first base class
{
    public:vehicle()// The fisrt base class is public
    {
        cout<<"This is a vehicle"<<endl; // The activities the vehicle class is meant to do
    }
};
class fourwheeler // This is the second base class
{
    public:fourwheeler()// The second base class is also public
    {
        cout<<"This is a 4 wheeler vehicle"<<endl;// The activities the fourwheeler is meant to do
    }
};
class car:public vehicle,public fourwheeler// The subclass car is derived from the two base classes
{


};
int main() //The starting of the main program
{car obj;// The creation /construction of the objects
return 0;
}
```

The Program Output:



```
This is a vehicle
This is a 4 wheeler vehicle
```

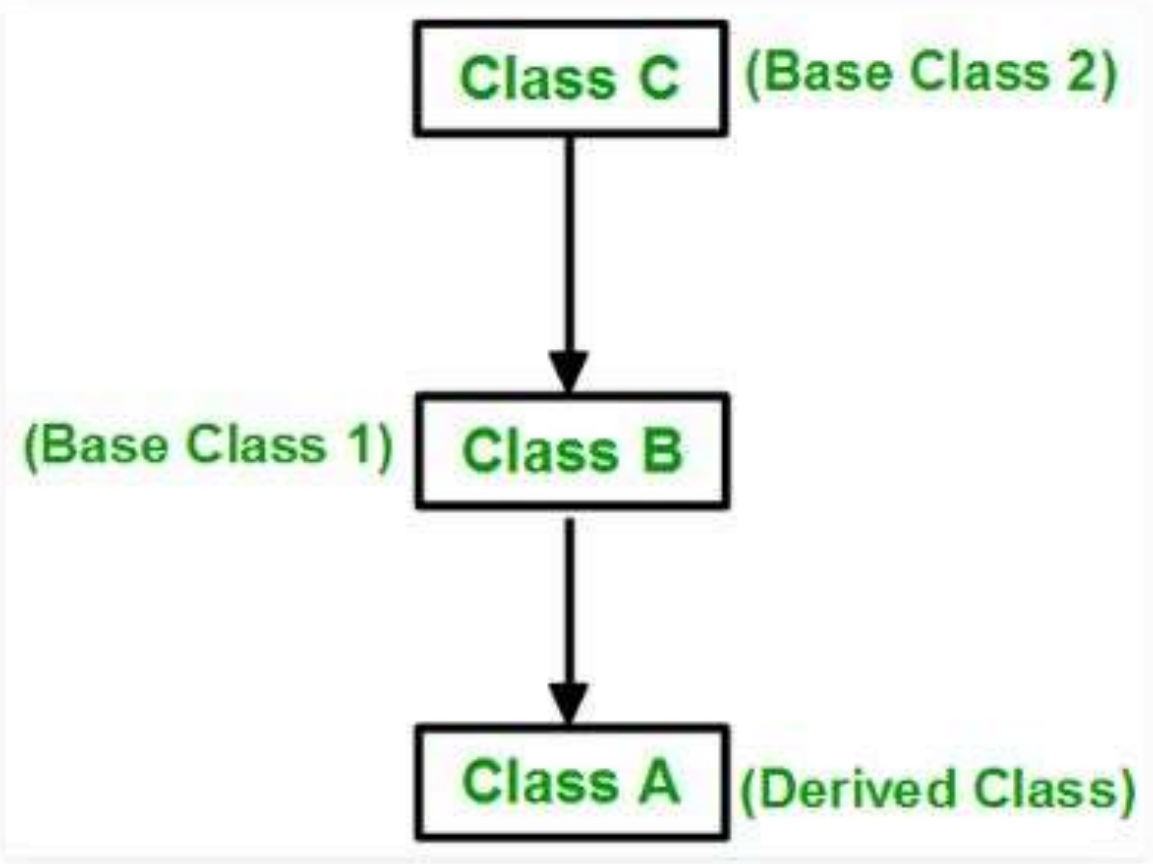
Question 2

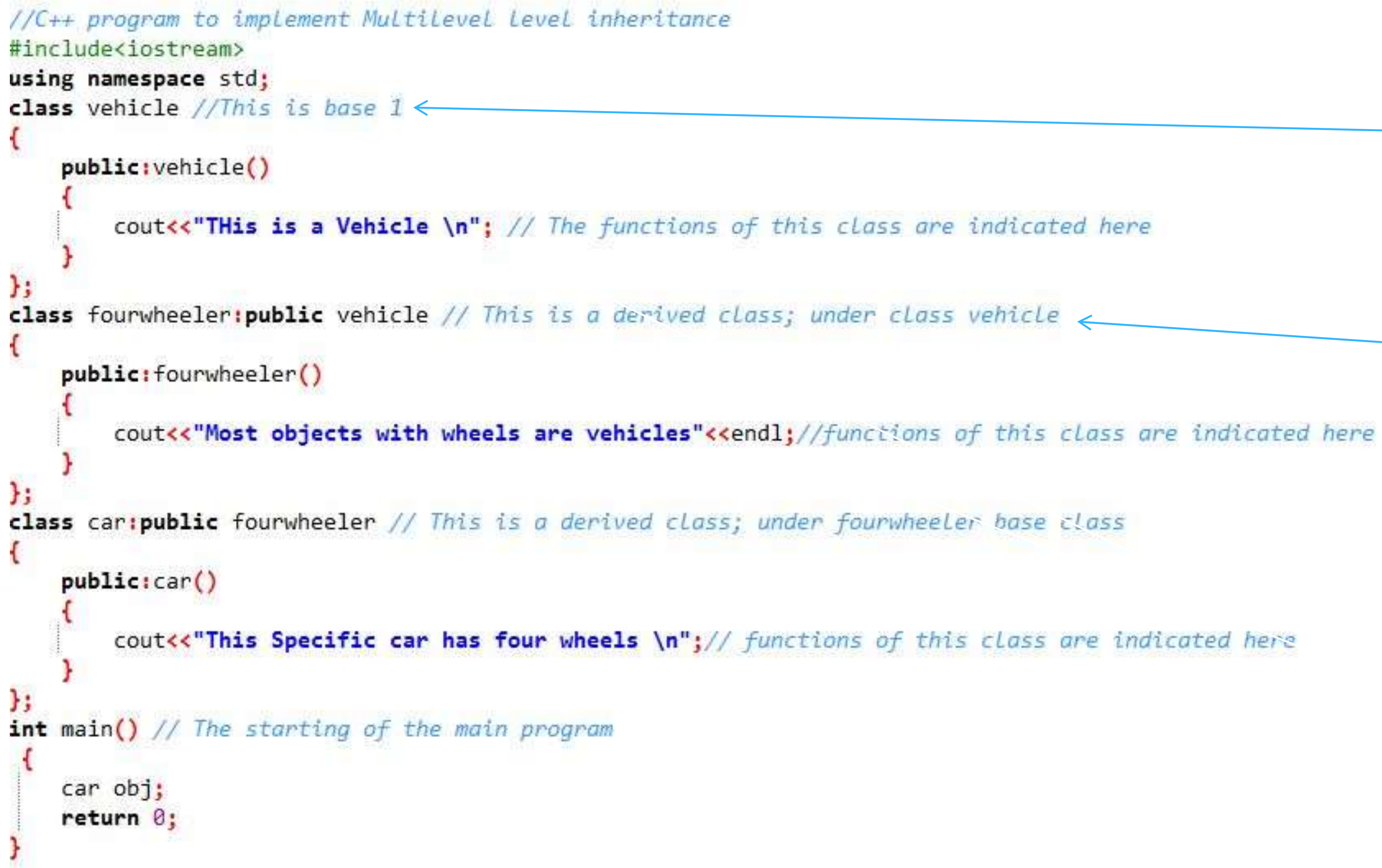
Write a C++ program that illustrates the following statements in a Government Inua jamii Management information system, name the program as per the applied type of inheritance (6 Marks)

The system performs the following with 3 classes

1. Citizenship class is public and is to display the following message "Is a Kenyan Citizen with a valid ID"
2. Vcitizen class is also a public class that displays the following message " He/She falls under the Vulnerable Category of citizens "
3. Gsupport class is a subclass to both the Citizenship and Vcitizen classes.

3. **Multilevel Inheritance**: In this type of inheritance, a derived class is created from another derived class.





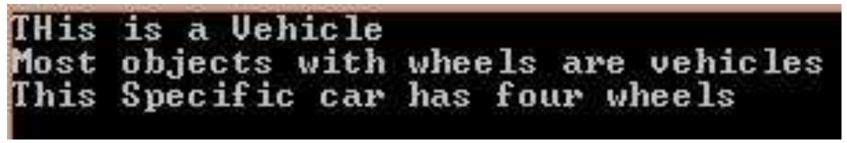
```cpp
//C++ program to implement Multilevel level inheritance
#include<iostream>
using namespace std;
class vehicle //This is base 1
{
    public:vehicle()
    {
        cout<<"THis is a Vehicle \n"; // The functions of this class are indicated here
    }
};
class fourwheeler:public vehicle // This is a derived class; under class vehicle
{
    public:fourwheeler()
    {
        cout<<"Most objects with wheels are vehicles"<<endl;//functions of this class are indicated here
    }
};
class car:public fourwheeler // This is a derived class; under fourwheeler base class
{
    public:car()
    {
        cout<<"This Specific car has four wheels \n";// functions of this class are indicated here
    }
};
int main() // The starting of the main program
 {
    car obj;
    return 0;
}
```

The Top base class of the system

The second class which is a Derived class to vehicle and is also Base class to car

Note: using this "\n" is the same as"endl"
Note the difference of where the semi colon is set when one uses \n and the inclusion of << when before using endln.

The Output of the Program



```
THis is a Vehicle
Most objects with wheels are vehicles
This Specific car has four wheels
```

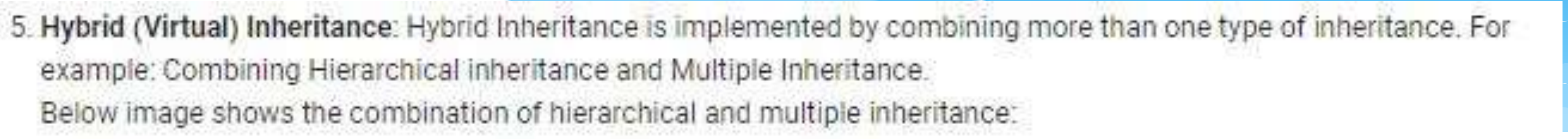
Question 3

An OOP system that clears a student at NTTI has the following classes, each having its output message as indicated below

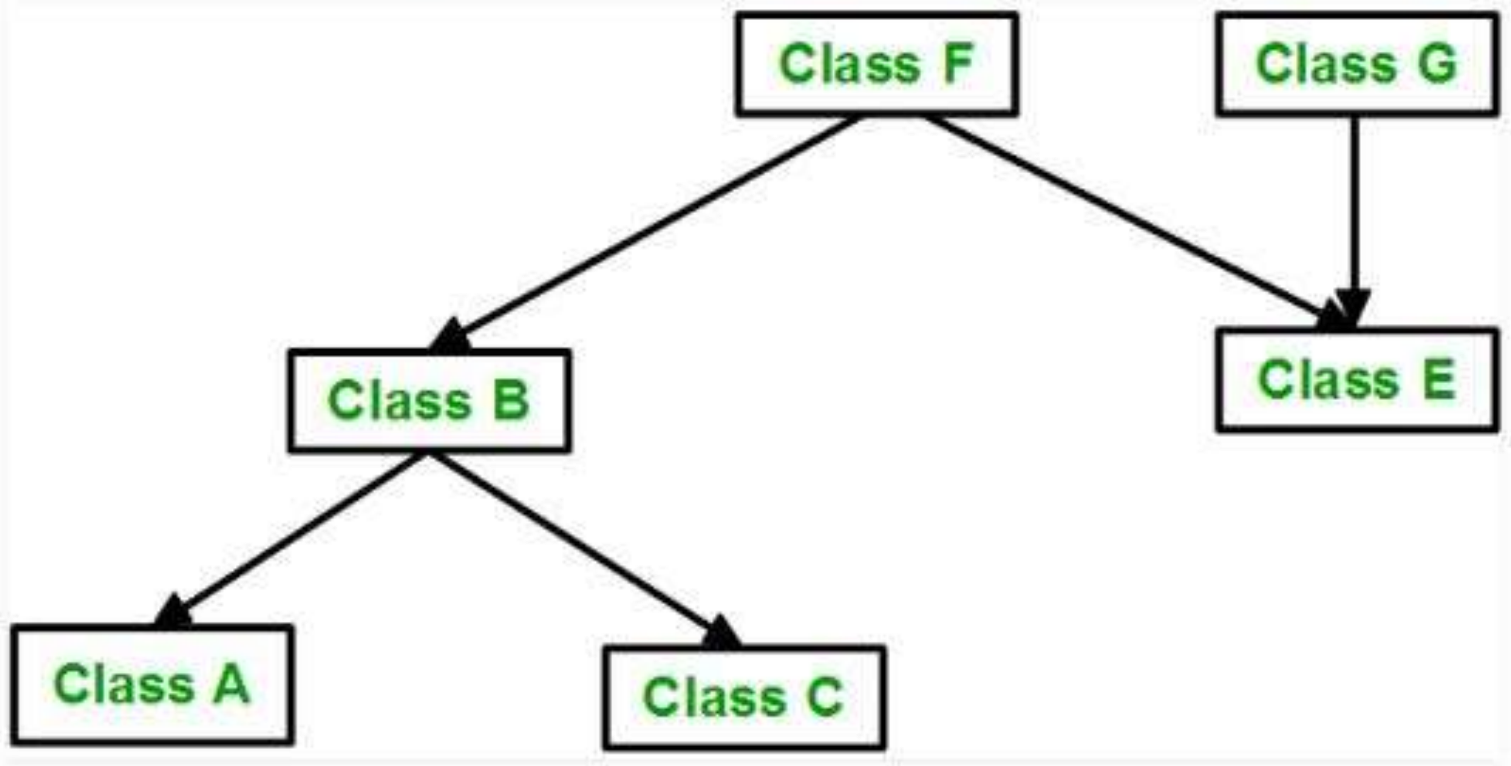
* Module1 “Met the relevant KCSE Entry criteria”
* Module2 “ Has Mean Grade of Pass, Credit or Distinction in Module 1”
* Module3 “Has Mean Grade of Pass, Credit or Distinction in Module 2”
* Diplomacourse “Has Mean Grade of Pass, Credit or Distinction in Module 3

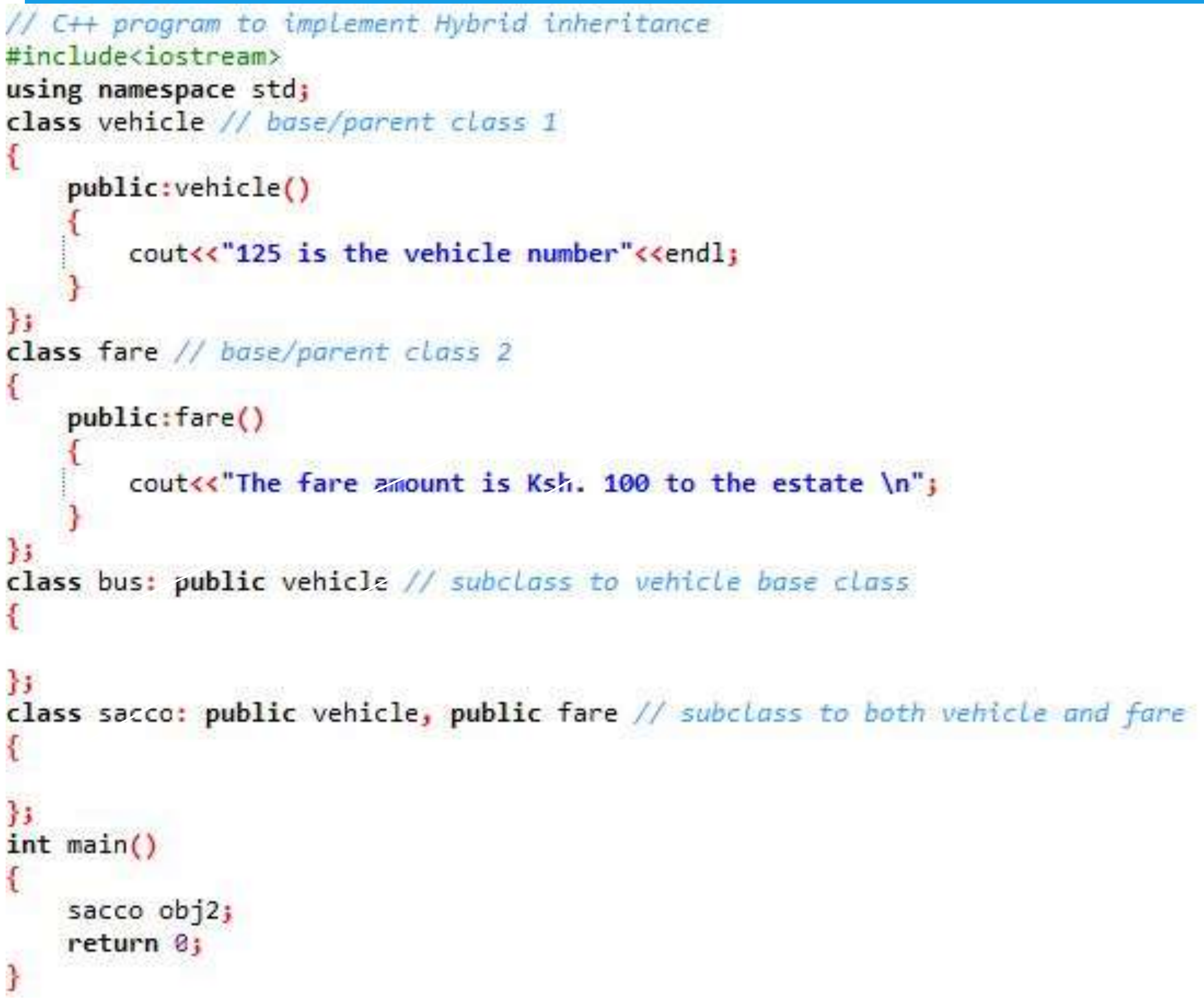
Required

i. Name the Base class (2 Marks)
ii. Use the appropriate inheritance type to write a C++ program that applies this concept. (6 Marks)



5. **Hybrid (Virtual) Inheritance**: Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.
Below image shows the combination of hierarchical and multiple inheritance:

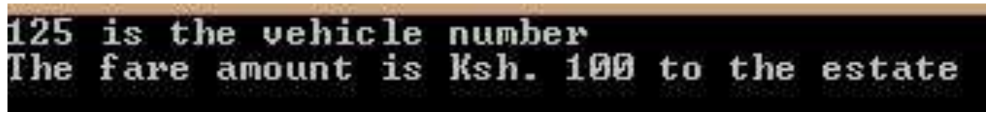




```
// C++ program to implement Hybrid inheritance
#include<iostream>
using namespace std;
class vehicle // base/parent class 1
{
    public:vehicle()
    {
        cout<<"125 is the vehicle number"<<endl;
    }
};
class fare // base/parent class 2
{
    public:fare()
    {
        cout<<"The fare amount is Ksh. 100 to the estate \n";
    }
};
class bus: public vehicle // subclass to vehicle base class
{

};
class sacco: public vehicle, public fare // subclass to both vehicle and fare
{

};
int main()
{
    sacco obj2;
    return 0;
}
```

Being blank indicates that there is NO new feature the subclass is meant to do other than just inheriting the features of the parent/base class

The Output



```
125 is the vehicle number
The fare amount is Ksh. 100 to the estate
```

Question 4

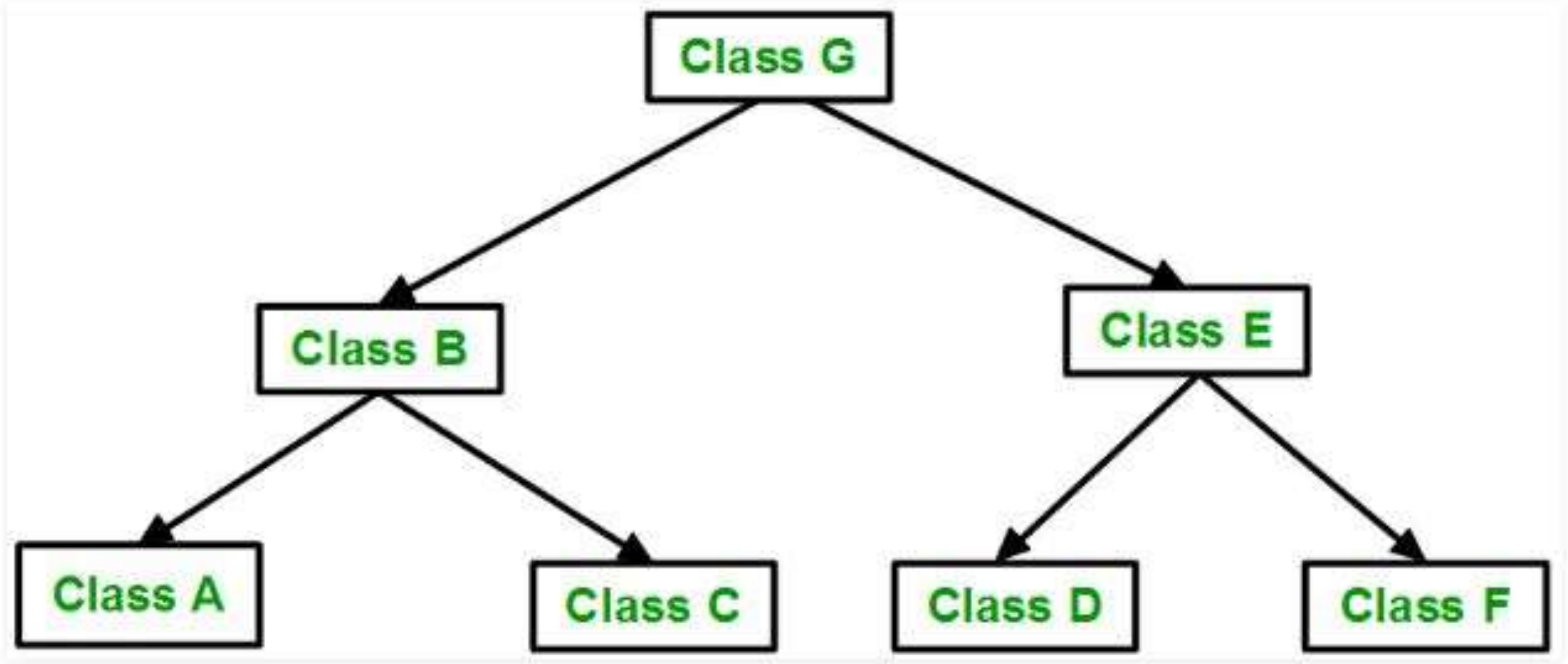
A Course Work Submit module is part of a college Management Information System. The module works as an OOP system with five classes and their respective procedures as follows.

1) CWSubmit class
2) CatScore " All cat exam score after conversion to 40%, Allocate CRNM if any is missing "
3) EndTermScore " All End of Term Exams score after conversion to 60%, Allocate CRNM if any is missing "
4) TotalExamscore
5) AttendanceScore "All students must score => 70%, Else Allocate CRNM"

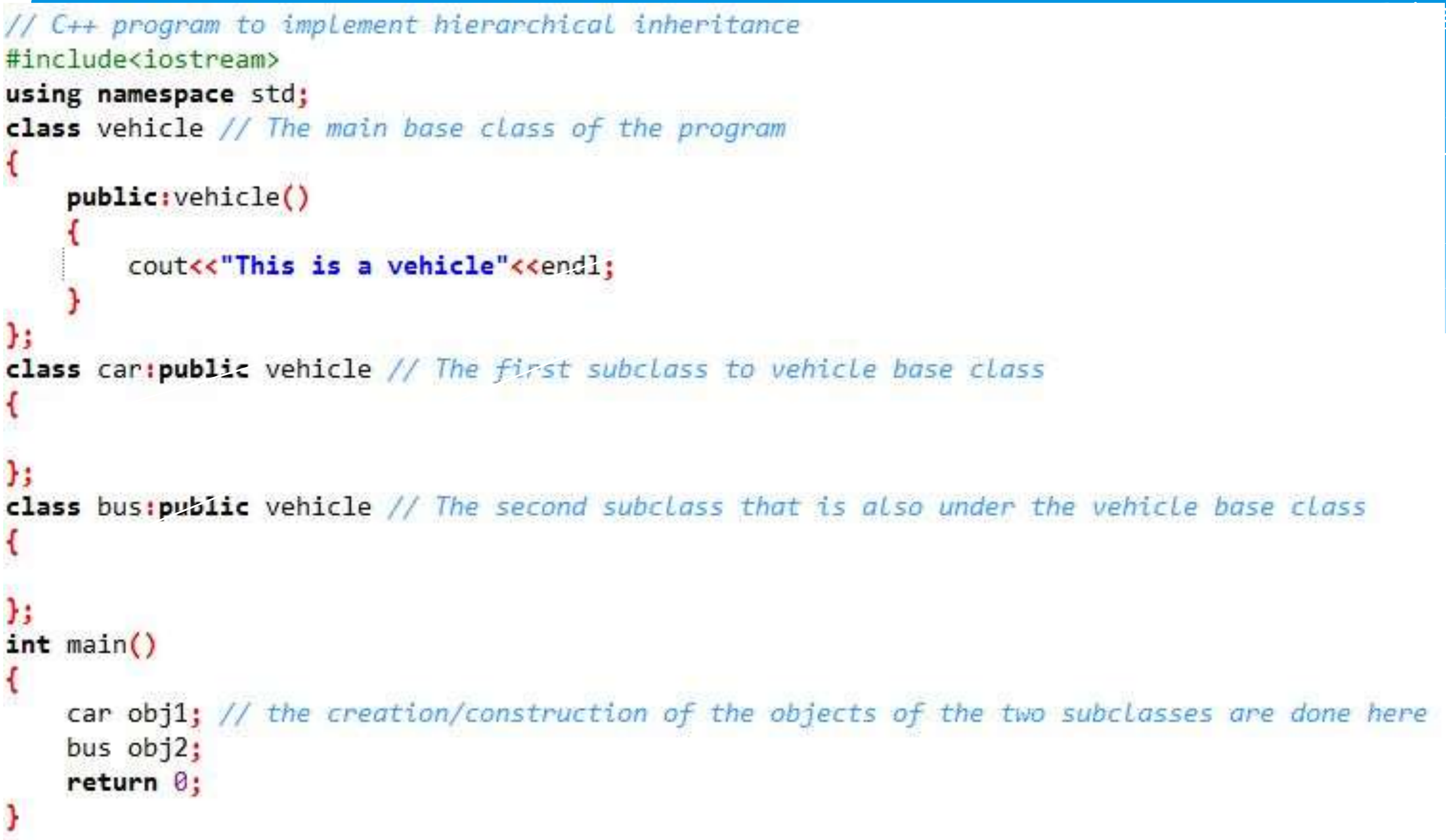
Required

i. With the aid of a diagram, determine the type of inheritance illustrated above (6 marks)
ii. write a C++ program for this concept. (8 Marks)

4. **Hierarchical Inheritance**: In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.

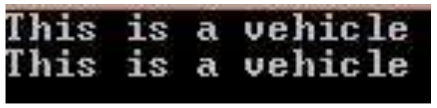


g blank indicates that the
new feature the subclas
ant to do other than just
iting the features of the
nt/base class



```
// C++ program to implement hierarchical inheritance
#include<iostream>
using namespace std;
class vehicle // The main base class of the program
{
    public:vehicle()
    {
        cout<<"This is a vehicle"<<endl;
    }
};
class car:public vehicle // The first subclass to vehicle base class
{

};
class bus:public vehicle // The second subclass that is also under the vehicle base class
{

};
int main()
{
    car obj1; // the creation/construction of the objects of the two subclasses are done here
    bus obj2;
    return 0;
}
```

The output of the program



```
This is a vehicle
This is a vehicle
```

Question 5

A section of NTTI MIS has an ICT Time Table Allocation module that has the following classes

* Core unitD2 – "All the DICTM2 core units"
* Theory – "Allocate classroom - SAD (System Analysis and Design), QT (Quantitative Methods)"
* Practical – "Allocate laboratory - OOP (Object Oriented Programming), VB(Visual Basic), DBMS (Database management System),CA2 (Computer Applications 2)"

Required

i. With the aid of a diagram illustrate the above inheritance type (2 Marks)

ii. Write a C++ program that applies this concept (6 Marks)