**National University of Computer & Emerging Sciences, Karachi** 

**Computer Science Department**

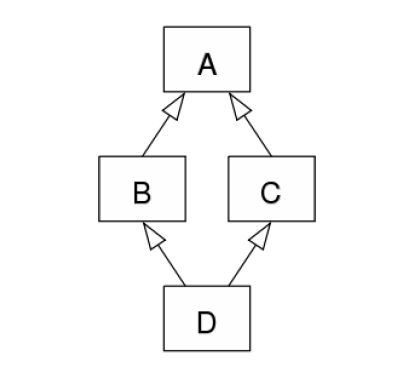
**Lab Manual - 01**

|  |  |
| --- | --- |
| **Course Code: CL-217** | **Course : Object Oriented Programming Lab** |
| **Instructor(s) :** | **Nida Munawar ,Muhammad Irfan Ayub, Mubashara Fayyaz, Aqsa Zahid** |

# Contents:

1. Diamond Problem in Hybrid Inheritance
2. Constructors

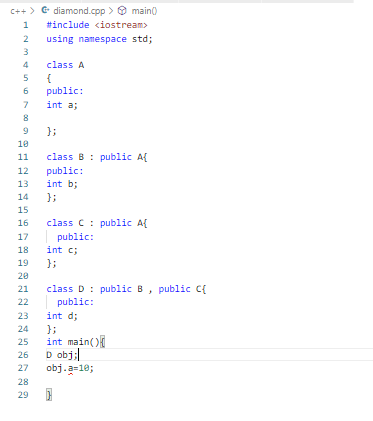
In case of hybrid inheritance, a Diamond problem may arise. The “dreaded diamond” refers to a class structure in which a particular class appears more than once in a class’s inheritance hierarchy.



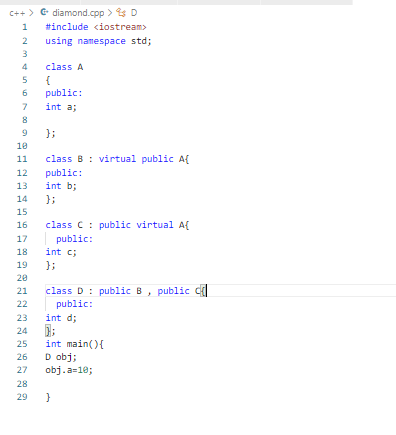
**Virtual base classes**

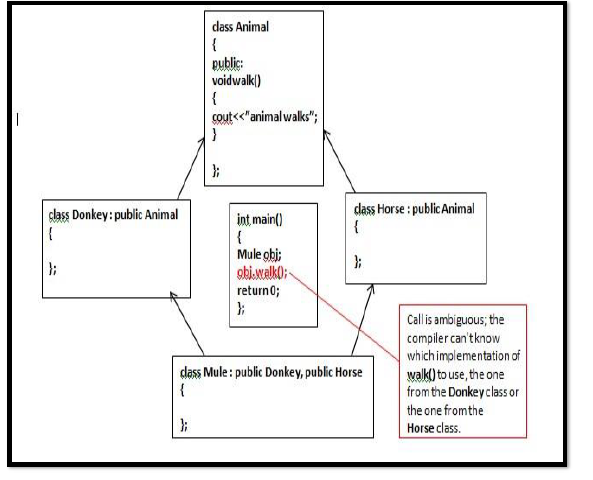
To share a base class, simply insert the “virtual” keyword in the inheritance list of the derived class. This creates what is called a **virtual base class**, which means there is only one base object. The base object is shared between all objects in the inheritance tree and it is only constructed once.

**For Example:**

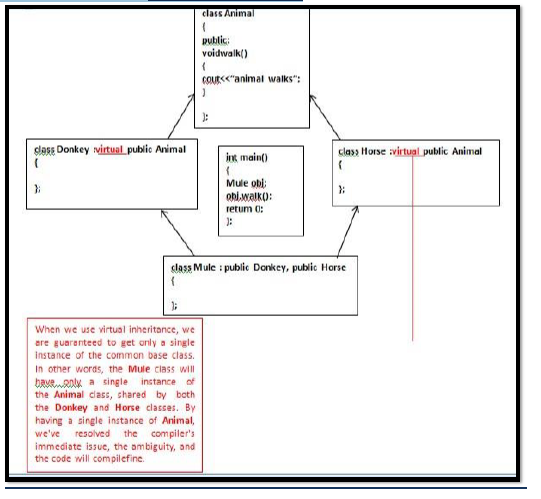


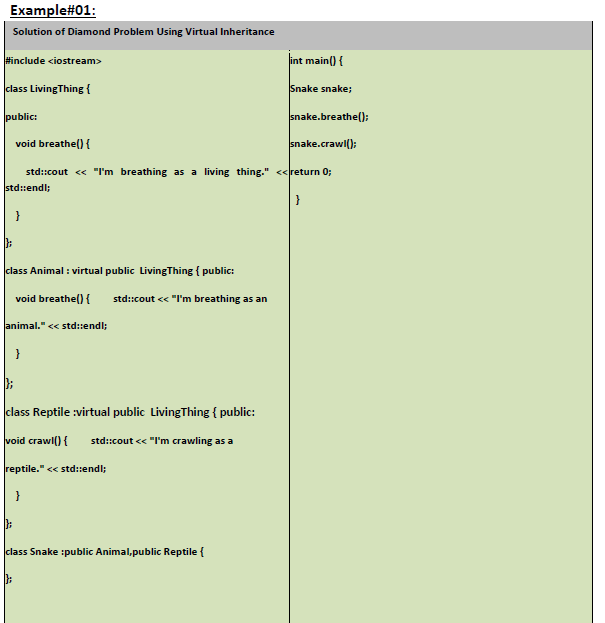
**How to solve this Problem??? Virtual Base Class Inheritance**



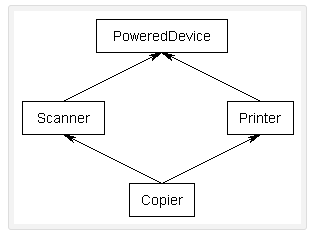


**How to solve this Problem??? Virtual Base Class Inheritance**





Here is another example (with some constructors) illustrating the diamond problem:



#include <iostream>

using namespace std;

class PoweredDevice

{

public:

    PoweredDevice(int power)

    {

        cout << "PoweredDevice: " << power << '\n';

    }

};

class Scanner: public PoweredDevice

{

public:

    Scanner(int scanner, int power)

        : PoweredDevice(power)

    {

        cout << "Scanner: " << scanner << '\n';

    }

};

class Printer: public PoweredDevice

{

public:

    Printer(int printer, int power)

        : PoweredDevice(power)

    {

        cout << "Printer: " << printer << '\n';

    }

};

class Copier: public Scanner, public Printer

{

public:

    Copier(int scanner, int printer, int power)

        : Scanner(scanner, power), Printer(printer, power)

    {

    }

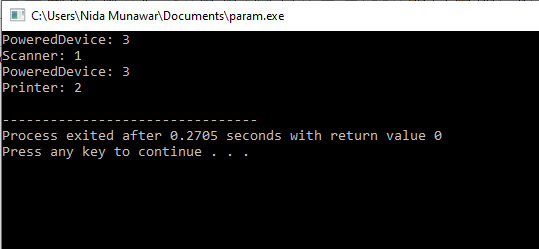
};

int main()

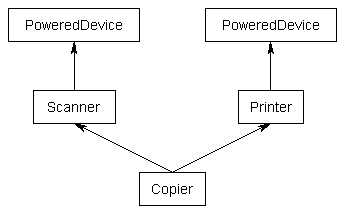
{

    Copier copier(1, 2, 3);

    return 0;}



If you were to create a Copier class object, by default you would end up with two copies of the PoweredDevice class -- one from Printer, and one from Scanner. This has the following structure:



**By using Virtual Base class**

#include <iostream>

class PoweredDevice {

public:

PoweredDevice(int power)

{ std::cout << "PoweredDevice: " << power << '\n';

}};

class Scanner: virtual public PoweredDevice // note: PoweredDevice is now a virtual base class

{

public:

Scanner(int scanner, int power)

: PoweredDevice(power) // this line is required to create Scanner objects, but ignored in this case

{ std::cout << "Scanner: " << scanner << '\n';

}

};

class Printer: virtual public PoweredDevice // note: PoweredDevice is now a virtual base class

{

public:

Printer(int printer, int power)

: PoweredDevice(power) // this line is required to create Printer objects, but ignored in this case

{ std::cout << "Printer: " << printer << '\n';

}};

class Copier: public Scanner, public Printer

{

public:

Copier( int scanner ,int printer, int power)

: PoweredDevice(power), // PoweredDevice is constructed here

Scanner(scanner, power),Printer(printer, power)

{

}

};

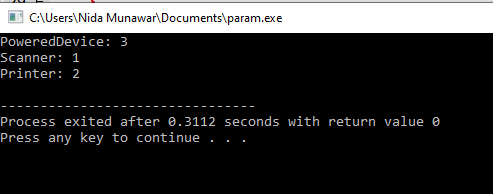
int main()

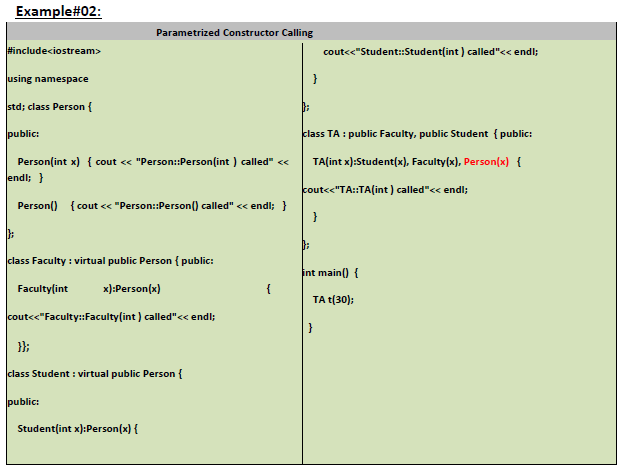
{

Copier copier(1, 2, 3);

return 0;}

As you can see, PoweredDevice only gets constructed once.

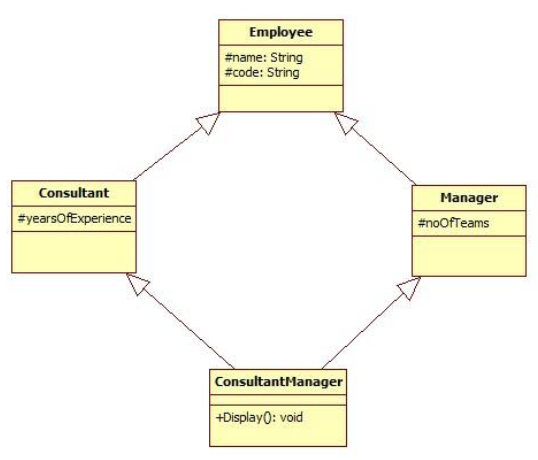




**Exercise**

**QUESTION#1**

Implement the following scenario in C++:



1. No accessors and mutators are allowed to be used.
2. The Display() function in “ConsultantManager” should be capable of displaying the values of all the data members declared in the scenario (name,code,yearsOfExperience,noOfTeams) without being able to alter the values.
3. The “int main()” function should contain only three program statements which are as follows:
4. In the first statement, create object of “ConsultantManager” and pass the values for all the data members:

**ConsultantManager obj("Ali",”S-123”,17,5);**

1. In the second statement, call the Display() function.
2. In the third statement, return 0.

