**Content 38**

**Ambguity Resolution In Inheritance**

If there are two class and when used one of its function in your derived class then the compiler would get confused that from both the classes which function he had to run or execute the nit would through an ambiguity.

class derived : public base1, public base2

{   public:

    void greet()

    {

        base1::greet();    //this is a resolving of ambiguity that which greeat will be used by derived class.

    }

};

This is the way or Syntax to call that particular class function and execute it without any ambiguity.

#include <iostream>

using namespace std;

class base1

{

public:

    void greet()

    {

        cout << "Good Morning" << endl;

    }

};

class base2

{

public:

    void greet()

    {

        cout << "Have a Nice day!!!" << endl;

    }

};

class derived : public base1, public base2

{   public:

    void greet()

    {

        base1::greet();    //this is a resolving of ambiguity that which greeat will be used by derived class.

    }

};

int main()

{   base1 b1;

    base2 b2;

    b1.greet();

    b2.greet();

    derived d;

    d.greet();

    return 0;

}

***Output:***

Good Morning

Have a Nice day!!!

Good Morning

**Code2:**

// //Ambiguity 2.... Which has base class and derived class only. Inwhich compile reads

// // both the function in both classes and gives overriden one.

#include <iostream>

using namespace std;

class base1{

    public:

    void show(){

        cout<<"This is me."<<endl;

    }

};

class derived : public base1{

    public:

    void show(){

        cout<<"And I am working with C++...."<<endl;

    }

};

int main()

{   base1 b;

    b.show();

    derived d;

    d.show();

    return 0;

}

**Output:**

This is me.

And I am working with C++....