➢ Visibility Modes in C++ with Examples

What is the visibility mode?

In inheritance whenever the derived class inherits from the base class than which of the member of the parent class can be accessible in the child class is controlled by the visibility mode. By default visibility mode is always set to private.

Syntax is:

```
class derived_class_name :: visibility_mode base_class_name
{
//Lines of code
}
```

Types of Visibility Mode in C++

There are total 3 types of visibility mode in C++ that are:

1. Private visibility mode,
2. Protected visibility mode,
3. Public visibility mode

	Derived Class	Derived Class	Derived Class
Base Class	**Private Mode**	**Protected Mode**	**Public Mode**
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Private	Protected	Protected
Public	Private	Protected	Public

Access Modes of Inheritance

There are three Access Modes of Inheritance in C++:

1. Public Mode

When we derive a subclass from a public base class. Then the base class's public members become public in the derived class, and the base class's protected members become protected in the derived class.

2. Protected Mode

When a subclass is derived from a Protected base class. Then, in the derived class, both public and protected members of the base class will be protected.

3. Private Mode

When a subclass is derived from a Private base class. The base class's public and protected members will then become Private in the derived class.

We will learn more in detail using the examples:

1. Private visibility mode:

When we inherit a derived class from the base class with private visibility mode then the public and protected members of the base class become private members of the derived class.

```
#include <iostream>

class X{

private:

int a;

protected:

int b;

public:
```

int c;

};

class Y : **private** X{

//As the visibility mode is private none of the member of base class is inherited to derived class

};

int main**()**

{

//Nothing can be accessed using Class Y object because there are no members in class Y.

return 0;

}

2. Protected visibility mode:

When we inherit a derived class from a base class with protected visibility mode the protected and public members of the base class become protected members of the derived class

```
#include <iostream>
class X{
private:
int a;
protected:
int b;
public:
int c;
};
class Y : protected X{
```

```
//As the visibility mode is protected the protected and public members of class X becomes the protected members of Class Y
//protected: int b,int c inherited from class X
};
int main()
{
//As the members in the class Y are protected they cannot be accessed in main using Class Y object.
return 0;
}
```

3. Public mode:

When we inherit a derived class from a base class with public visibility mode, the public members and protected members of the base class will be inherited as public members and protected members respectively of the derived class.

```
#include <iostream>
class X{
private:
int a;
protected:
int b;
public:
int c;
};
class Y : public X{
//As the visibility mode is public the protected members of class X becomes protected member for class Y and public members of class X becomes public member for class Y
//protected: int b; inherited from class X
//public: int c; inherited from class X
};
int main()
{
```

```
//Only int c can be accessed in main function using Class Y object as it is public;
Y obj;
std::cout<<obj.c;
return 0;
}
```