GRANTING ACCESS IN CPP

Granting Access

When a base class is inherited as **private**, all public and protected members of that class become private members of the derived class. However, in certain circumstances, you may want to restore one or more inherited members to their original access specification. For example, you might want to grant certain public members of the base class public status in the derived class even though the base class is inherited as **private**. In Standard C++, you have two ways to accomplish this. First, you can use a **using** statement, which is the preferred way. The **using** statement is designed primarily to support namespaces and is discussed in Chapter 23. The second way to restore an inherited member's access specification is to employ an *access declaration* within the derived class. Access declarations are currently supported by Standard C++, but they are deprecated. This means that they should not be used for new code. Since there are still many, many existing programs that use access declarations, they will be examined here.

An access declaration takes this general form:

base-class::member;

The access declaration is put under the appropriate access heading in the derived class' declaration. Notice that no type declaration is required (or, indeed, allowed) in an access declaration.

To see how an access declaration works, let's begin with this short fragment:

```
class base {
public:
int j; // public in base
};
// Inherit base as private.
 class derived: private base {
 public:
 // here is access declaration
 base::j; // make j public again
 .
 .
 .
 };
```

Because **base** is inherited as **private** by **derived**, the public member **j** is made a private member of **derived**. However, by including base::j;
as the access declaration under **derived**'s **public** heading, **j** is restored to its public status.
You can use an access declaration to restore the access rights of public and protected members. However, you cannot use an access declaration to raise or lower a member's access status. For example, a member declared as private in a base class cannot be made public by a derived class. (If C++ allowed this to occur, it would destroy its encapsulation mechanism!) The following program illustrates the access declaration; notice how it uses access declarations to restore **j**, **seti( )**, and **geti( )** to **public** status.

```
#include <iostream>
using namespace std;
class base {
int i; // private to base
public:
int j, k;
void seti(int x) { i = x; }
int geti() { return i; }
};
// Inherit base as private.
class derived: private base {
public:
/* The next three statements override
base's inheritance as private and restore j,
 seti(), and geti() to public access. */
 base::j; // make j public again - but not k
 base::seti; // make seti() public
 base::geti; // make geti() public
 // base::i; // illegal, you cannot elevate access
 int a; // public
 };
```

```
int main()
{
derived ob;
//ob.i = 10; // illegal because i is private in derived
ob.j = 20; // legal because j is made public in derived
//ob.k = 30; // illegal because k is private in derived
ob.a = 40; // legal because a is public in derived
ob.seti(10);
cout << ob.geti() << " " << ob.j << " " << ob.a;
return 0;
}
```

Access declarations are supported in C++ to accommodate those situations in which most of an inherited class is intended to be made private, but a few members are to retain their public or protected status.

While Standard C++ still supports access declarations, they are deprecated. This means that they are allowed for now, but they might not be supported in the future. Instead, the standard suggests achieving the same effect by applying the ***using*** *keyword.*

Source : http://elearningatria.files.wordpress.com/2013/10/cse-iii-object-oriented-programming-with-c-10cs36-notes.pdf