Types of Inheritance

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Objectives

In this chapter, you will:

- Learn types of inheritance
- Explore three types of visibility of inherited members: public, protected, and private

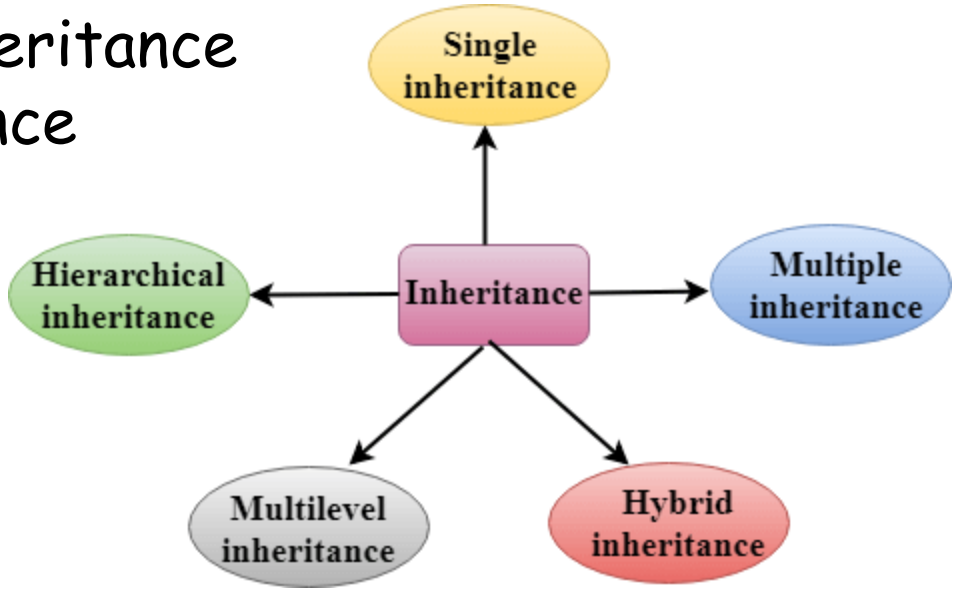
Advantage of Inheritance

Code reusability: Now you can reuse the members of your parent class. So, there is no need to define the member again. So less code is required in the class.

Types of Inheritance

C++ supports five types of inheritance:

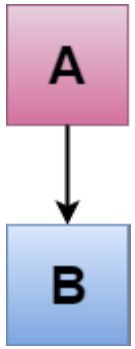
1. Single inheritance
2. Multilevel inheritance
3. Multiple inheritance
4. Hierarchical inheritance
5. Hybrid inheritance



Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Single Inheritance

Single inheritance is defined as the inheritance in which a derived class is inherited from the only one base class.



Where 'A' is the base class, and 'B' is the derived class.

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Single Inheritance/Inheriting Fields

When one class inherits another class, it is known as single level inheritance. Let's see the example of single level inheritance which inherits the fields only.

```
#include <iostream>
using namespace std;
 class Account {
   public:
   float salary = 60000;
 };
   class Programmer: public Account {
   public:
   float bonus = 5000;
   };
int main(void) {
     Programmer p1;
     cout<<"Salary: "<<p1.salary<<endl;
     cout<<"Bonus: "<<p1.bonus<<endl;
    return 0;
}
```

```
Salary: 60000
Bonus: 5000
```

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Single Inheritance/Inheriting Methods

Let's see another example of inheritance in C++ which inherits methods only.

```
#include <iostream>
using namespace std;
 class Animal {
   public:
 void eat() {
    cout<<"Eating..."<<endl;
 }
   };
   class Dog: public Animal
   {
      public:
     void bark(){
    cout<<"Barking...";
     }
   };
int main(void) {
    Dog d1;
    d1.eat();
    d1.bark();
    return 0;
}
```

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Single Inheritance/Inheriting Methods

Let's see a simple example.

```
#include <iostream>
using namespace std;
class A
{
    int a = 4;
    int b = 5;
    public:
    int mul()
    {
        int c = a*b;
        return c;
    }
};

class B : private A
{
    public:
    void display()
    {
        int result = mul();
        std::cout <<"Multiplication of a and b is : "<<result<< std::endl;
    }
};
int main()
{
   B b;
   b.display();
  // b.mul();
    return 0;
}
```

In the this example, class A is privately inherited. Therefore, the mul() function of class 'A' cannot be accessed by the object of class B. It can only be accessed by the member function of class B.

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

How to make a Private Member Inheritable

The private member is not inheritable. If we modify the visibility mode by making it public, but this takes away the advantage of data hiding.

C++ introduces a third visibility modifier, i.e., protected. The member which is declared as protected will be accessible to all the member functions within the class as well as the class immediately derived from it.

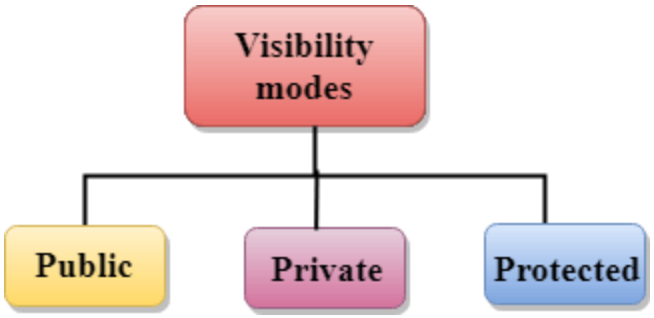
Visibility Modes

Visibility modes can be classified into three categories:

Public: When the member is declared as public, it is accessible to all the functions of the program.

Private: When the member is declared as private, it is accessible within the class only.

Protected: When the member is declared as protected, it is accessible within its own class as well as the class immediately derived from it.



Visibility of Inherited Members

Base class visibility	Derived class visibility		
	Public	Private	Protected
Public	Public	Private	Protected
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Protected	Private	Protected

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Multilevel Inheritance

Multilevel inheritance is a process of deriving a class from another derived class.

When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C++. Inheritance is transitive so the last derived class acquires all the members of all its base classes.

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Multilevel Inheritance/Example

```
#include <iostream>
using namespace std;

class Animal {
    public:
    void eat() {
        cout<<"Eating..."<<endl;
    }
};

class Dog: public Animal{
    public:
    void bark(){
       cout<<"Barking..."<<endl;
    }
};
```

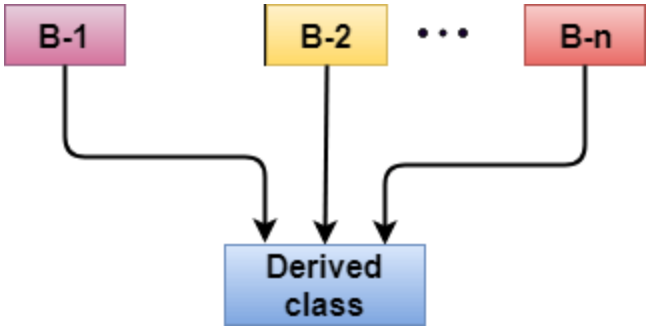
```
class BabyDog: public Dog{
    public:
    void weep() {
        cout<<"Weeping...";
    }
};

int main(void) {
    BabyDog d1;
    d1.eat();
    d1.bark();
    d1.weep();
    return 0;
}
```

```
Eating...
Barking...
Weeping...
```

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Multiple Inheritance

Multiple inheritance is the process of deriving a new class that inherits the attributes from two or more classes.



Syntax of the Derived class:

```
class D : visibility B-1, visibility B-2, ?
{
    // Body of the class;
}
```

Multiple Inheritance

C++ class can inherit members from more than one class and here is the extended syntax:

class derived-class: access baseA, access baseB....

Example:

class D: public A, private B, protected C

Multiple Inheritance/Example

```
#include <iostream>
using namespace std;
class A
{
    protected:
     int a;
    public:
    void get_a(int n)
    {
        a = n;
    }
};

class B
{
    protected:
    int b;
    public:
    void get_b(int n)
    {
        b = n;
    }
};
```

```
class C : public A,public B
{
   public:
    void display()
    {
        std::cout << "The value of a is : " <<a<< std::endl;
        std::cout << "The value of b is : " <<b<< std::endl;
        cout<<"Addition of a and b is : "<<a+b;
    }
};
int main()
{
   C c;
   c.get_a(10);
   c.get_b(20);
   c.display();

    return 0;
}
```

```
The value of a is : 10
The value of b is : 20
Addition of a and b is : 30
```

In the above example, class 'C' inherits two base classes 'A' and 'B' in a public mode.

Md. Sharif Hossen
Lecturer, Dept. of ICT, Comilla University, Bangladesh

Inheritance

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

Type of Inheritance

When deriving a class from a base class, the base class may be inherited through **public**, **protected** or **private** inheritance. The type of inheritance is specified by the access-specifier

We hardly use **protected** or **private** inheritance but **public** inheritance is commonly used. While using different type of inheritance, following rules are applied:

1. Public Inheritance: When deriving a class from a **public** base class, **public** members of the base class become public members of the derived class and **protected** members of the base class become **protected** members of the derived class. A base class's **private** members are **never accessible** directly from a derived class, but can be accessed through calls to the **public** and **protected** members of the base class.

Type of Inheritance

2.Protected Inheritance: When deriving from a **protected** base class, **public** and **protected** members of the base class become protected members of the derived class.

3.Private Inheritance: When deriving from a private base class, **public** and **protected** members of the base class become private members of the derived class.