Inheritance: Extending Classes

Introduction

- Reusability is an important feature of OOP.
- C++ strongly supports the concept of reusability.

Introduction

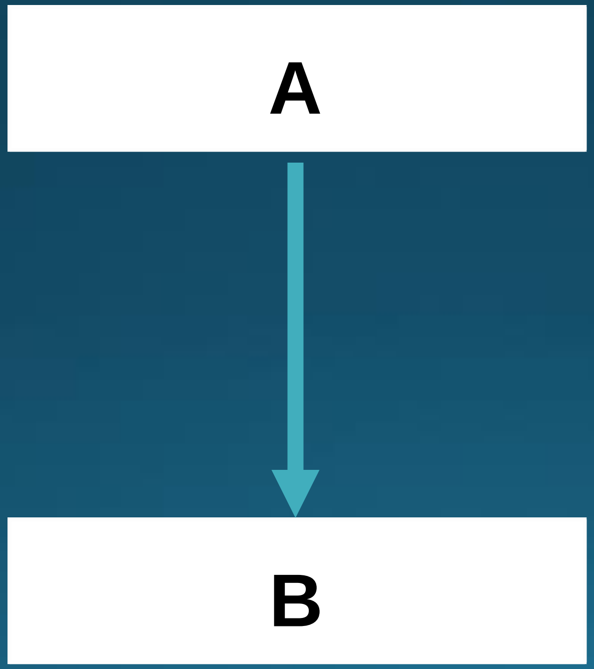
- The mechanism of deriving a new class from an old one is called inheritance (or derivation).
- The old class is referred to as base class.
- The new class is called the derived class or subclass.

Introduction

- The derived class inherits some or all of the properties from the base class.
- A class can also inherit properties from more than one class or from more than one level.

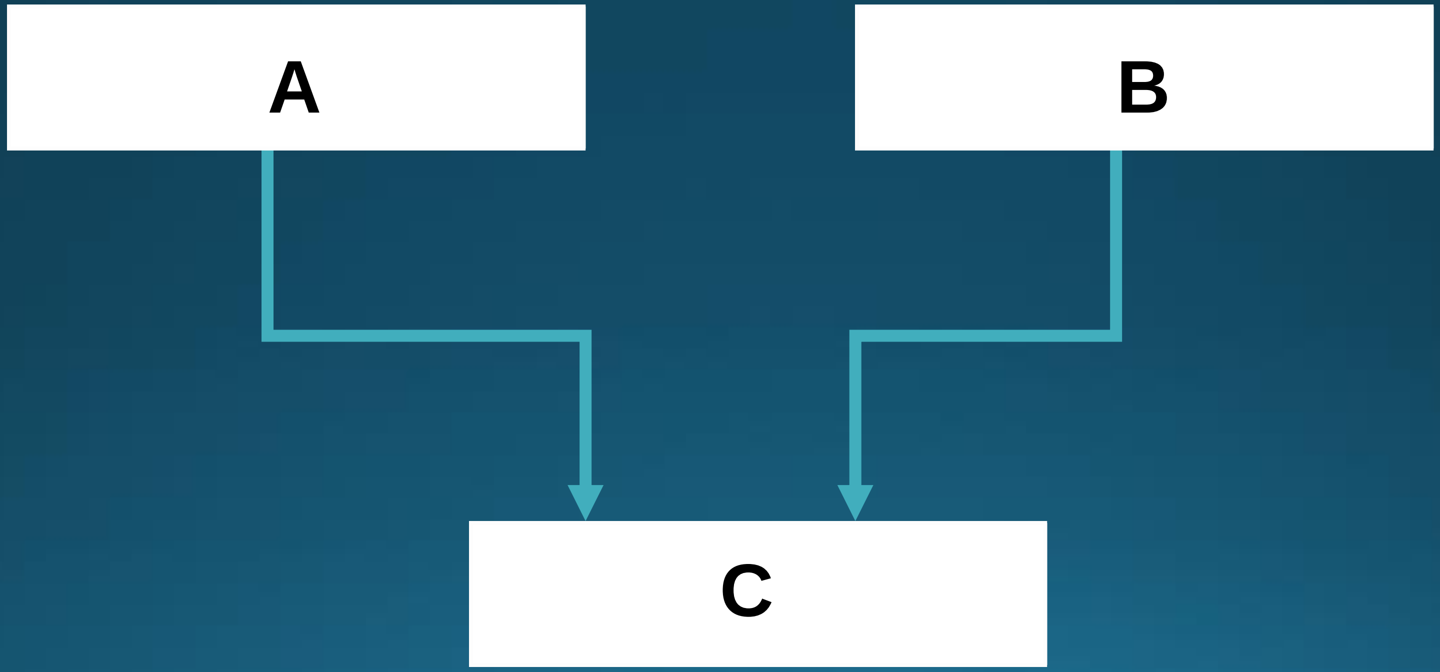
Single Inheritance

- A derived class with only one base class.



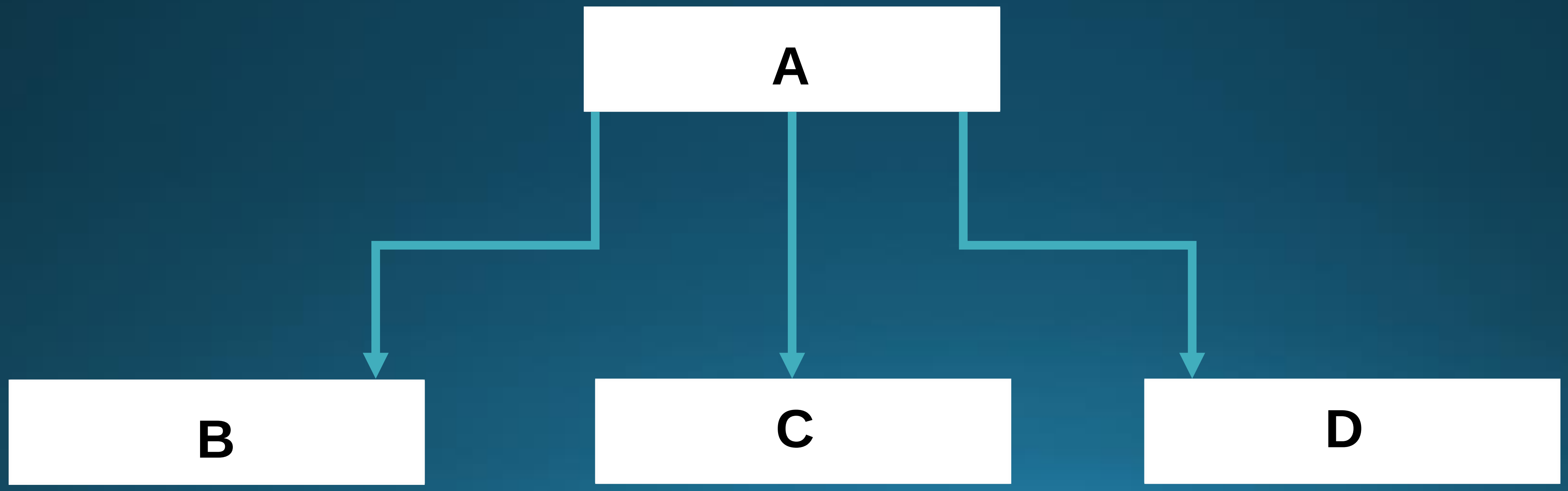
Multiple Inheritance

- A derived class with several base classes



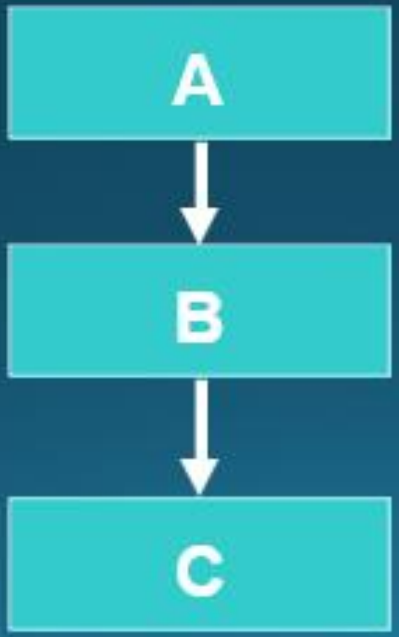
Hierarchical Inheritance

- **A traits of one class may be inherited by more than one class.**



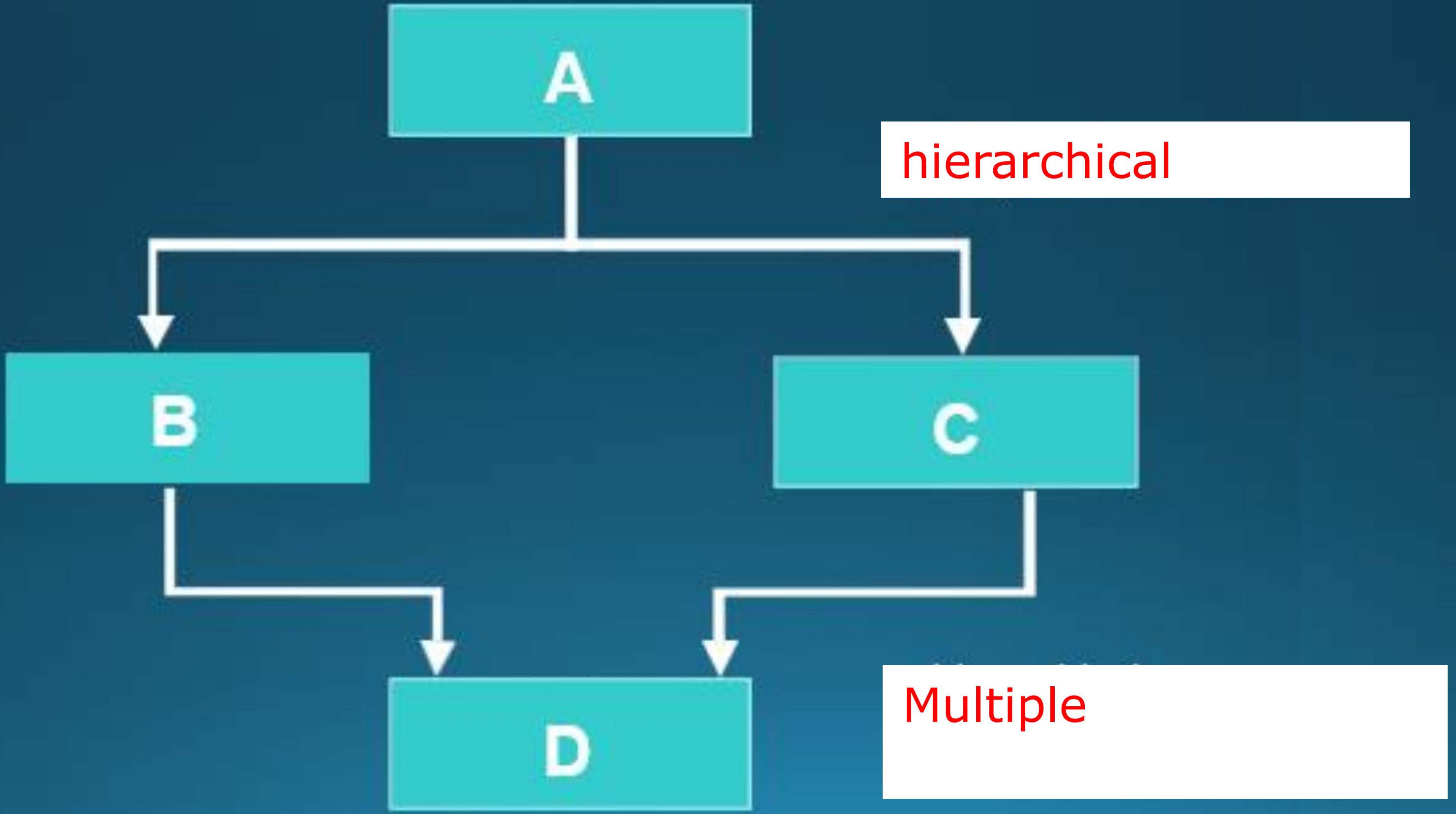
Multilevel Inheritance

- The mechanism of deriving a class from another derived class.



Hybrid Inheritance

- The mechanism of deriving a class by using a mixture of different methods.



Defining Derived Classes

- A derived class can be defined by specifying its relationship with the base class in addition to its own details.

```
class derived-class-name  : visibility-mode base-class-name
{
    ………//
    ………//    members of derived class
    ………//
};
```

Defining Derived Classes

continue …

class derived-class-name : visibility-mode base-class-name

The colon indicates that the **derived-class-name** is derived from the **base-class-name**

```
{
    .........//
    .........//    members of derived class

    .........//
};
```

The visibility mode is optional and , if present, may be either **private** or **public.**

The default visibility mode is **private.**

Visibility mode specifies whether the features of the base class are **derived privately or publicly.**

Defining Derived Classes

continue ...

- When a base class is privately derived by a derived class, “public members” of the base class becomes “private members” of the derived class.
- Therefore only the member function of derived class access the public members of the base class
- They are inaccessible to the objects of the derived class.
- As a result no member of the base class is accessible to the object of the derived class.

Defining Derived Classes

continue …

- When a base class is publicly inherited, “public members” of the base class become the “public members” of the derived class.
- Hence, they are accessible to the objects of the derived class.

Defining Derived Classes

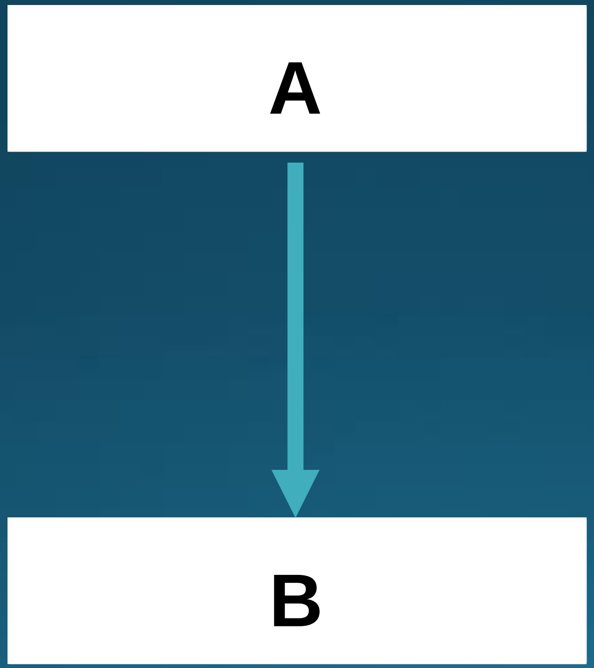
continue …

- The private members of a base class are not inherited in both private and public inheritance
- The private members of a base class will never become the members of its derived class.

Defining Derived Classes

- In inheritance, some of the base class data elements and member functions are inherited into the derived class.
- We can add our own data and member functions for extending the functionality of the base class.
- It is a powerful tool for incremental program development
- Can increase the capabilities of an existing class without modifying it.

Single Inheritance



```
// Single inheritance - Public  (all members are public in the base class)
class b

{
   public:
     int x;
   void getdata()
   {
     cout << "Enter the value of x = "; cin >> x;
   }
 };
class d : public b    //single derived class
{
   private:
    int y;
   public:
   void readdata()
   {
     cout << "Enter the value of y = "; cin >> y;
   }
   void product()
   {
     cout << "Product = " << x * y;
   }
 };
```

```
int main()
{
   d d1;     //object of derived class
   d1.getdata();
   d1.readdata();
   d1.product();
   return 0;
}
```

```
Enter the value of x = 3
Enter the value of y = 4
Product = 12
```

```
//Single inheritance - Public (with a private members in base class)
Class B
{
int a;  // not inheritable
   public:
int b;   //can be inherited
void get_ab() { a=5;b=10; }

int get_a() { return a;}  // since private member cant be inherited
void show_a() { count<< "a="<<a<< "\n" ;}
};

Class D: public B          //public derivation
{
int c;
   public:
void mul()   { c=b*get_a(); }
void display()
{
Count<< "a="<<get_a()
Count<< "b="<<b
Count<< "c="<<c
}
};
```

```
 int main()
{
D d;
d.get_ab();
d.mul();
d.show_a();
d.display();
d.b=20;
d.mul();
d.display();
return 0;
}
```

Output

```
a=5
a=5
b=10
c=50

a=5
b=20
c=100
```

```
//Single inheritance - Private
Class B
{
int a;  // not inheritable
    public:
int b;   //can be inherited
void get_ab() { a=5;b=10; }

int get_a() { return a;}
void show_a() { count<< “a=”<<a<< “\n” ;}
};

Class D: private B            //Private derivation
{
int c;
    public:
void mul()   { get_ab();   c=b*get_a(); }
void display()
{
show_a()
Count<< “b=”<<b
Count<< “c=”<<c
}
};
```

```
 int main()
{
D d;
//d.get_ab();   invalid
d.mul();
//d.show_a();  invalid
d.display();
//d.b=20;  invalid, b is now private
return 0;
}
Output
a=5
b=10
c=50
```

Making a Private Member Inheritable

- By making the visibility limit of the private members to the public.
  - But this takes away advantage of data hiding
- The visibility modifier “protected” can be used for this purpose.
- A member declared as “protected” is accessible by the member function within its class and any class immediately derived from it.
- It can not be accessed by the functions outside these two classes.

Making a Private member Inheritable

continue …

```
class alpha
{
    private : // optional
        .........     //  visible to the member within its class
        .........
    protected :
        ......... //  visible to member functions
        ......... //  of its own and immediate derived class
    public :
        ......... //  visible to all functions
        ......... //  in the program
};
```

Protected Member

continue …

- When a protected member is inherited in public mode, it becomes protected in the derived class.
- They are accessible by the member functions of the derived class.
- And they are ready for further inheritance.

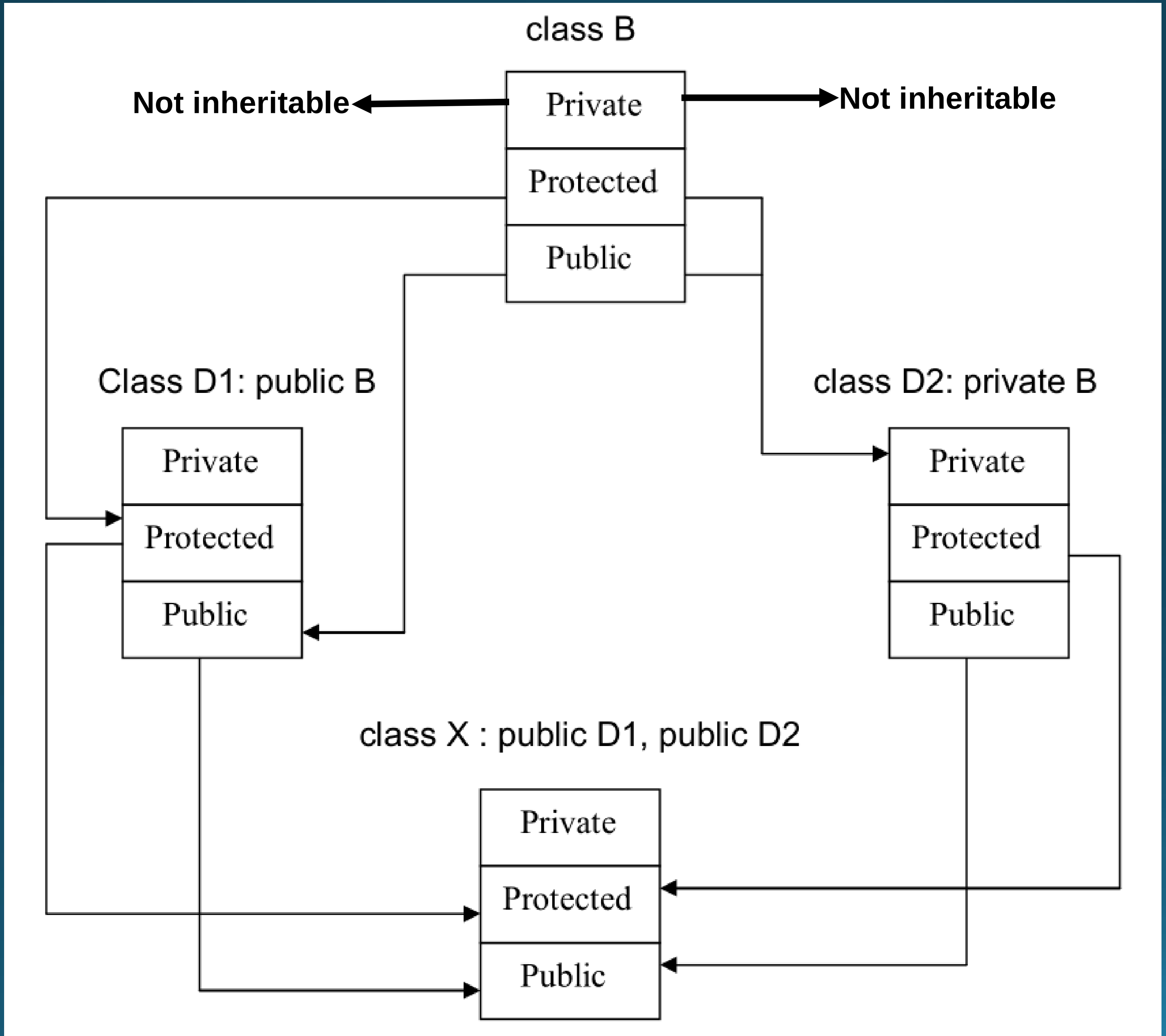
- When a protected member is inherited in private mode, it becomes private in the derived class.
- They are accessible by the member functions of the derived class.
- But, they are not available for further inheritance.

Protected Derivation

- It is possible to inherit a base class in protected mode – protected derivation.

- In protected derivation, both the public and protected members of the base class become protected members of the derived class.

Effect of inheritance on the visibility of members



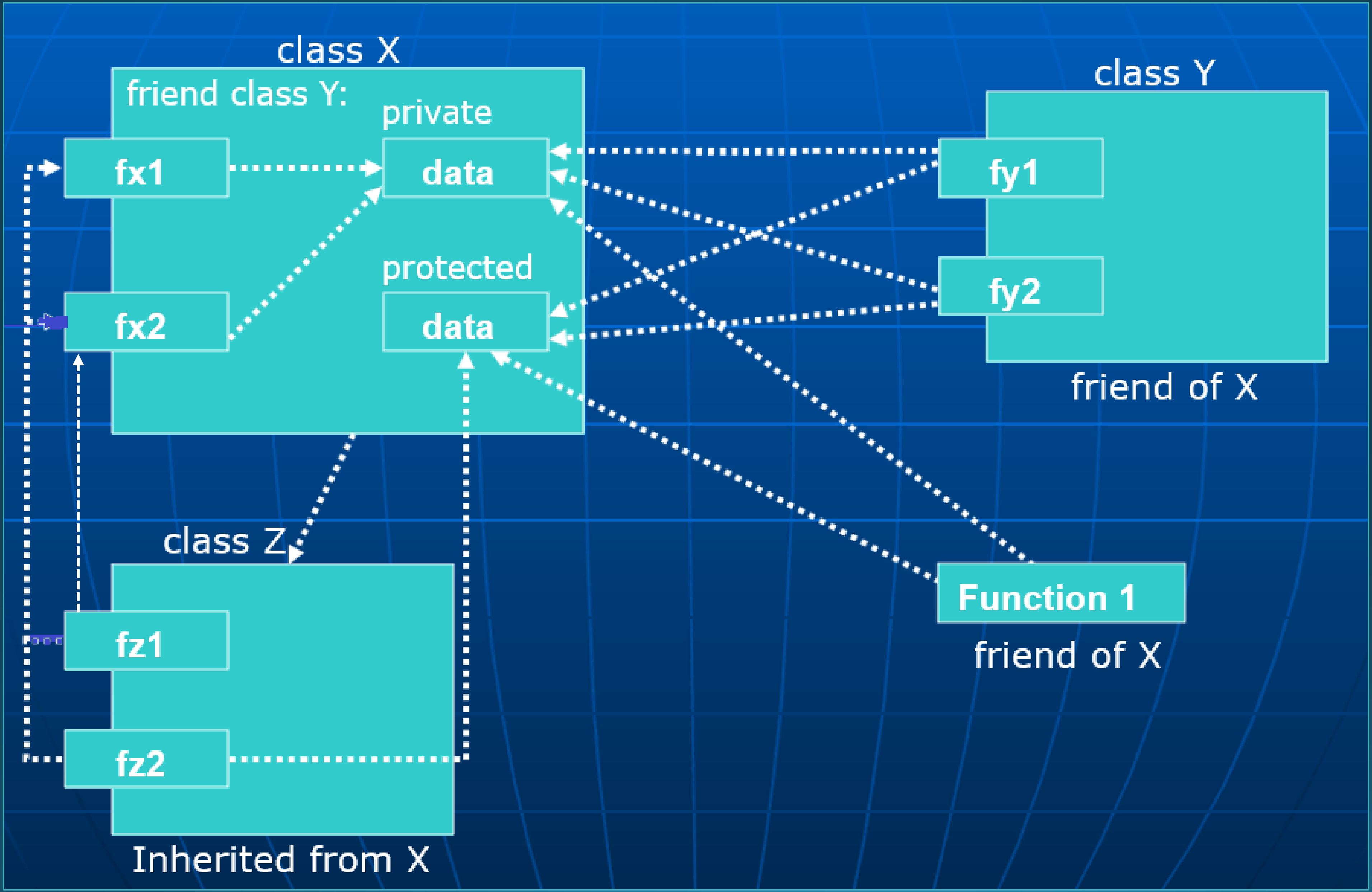
Visibility

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

Access Control to Data Members

- Functions that can have access to the private and protected members of class are:
  - A function that is a friend of the class.
  - A member function of a class that is a friend of the class.
  - A member function of a derived class.

Access mechanism in classes


class X
friend class Y:
private
fx1
data
fx2
protected
data
class Y
fy1
fy2
friend of X
class Z
fz1
fz2
Inherited from X
Function 1
friend of X