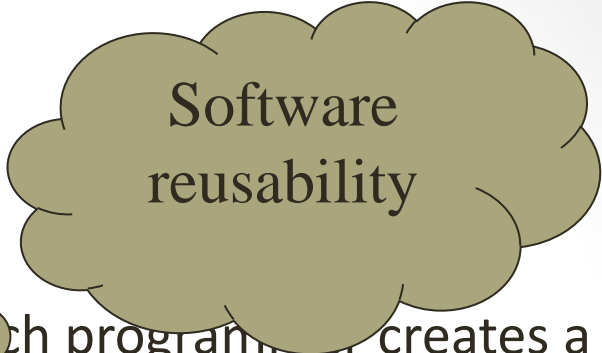


Inheritance

Lecture 16

Inheritance



Software
reusability

- It is a form of software reuse in which programmer creates a class that absorbs an existing class's data and behavior and enhances them with new capabilities.
- New class (derived) can inherit the members of existing class (base)

Example

```
class base
```

```
{ int x;
```

```
  public: int y;
```

```
  base(int i=0,int j=0) : x(i),y(j) { }
```

```
  void display() { cout<<"\n X : "<<x<<" Y : "<<y; };
```

```
class derived : public base
```

```
{ int z;
```

```
  public:
```

```
  derived(int s) : z(s) { }
```

```
  void d() { display(); cout<<"\n Z : "<<z; } };
```

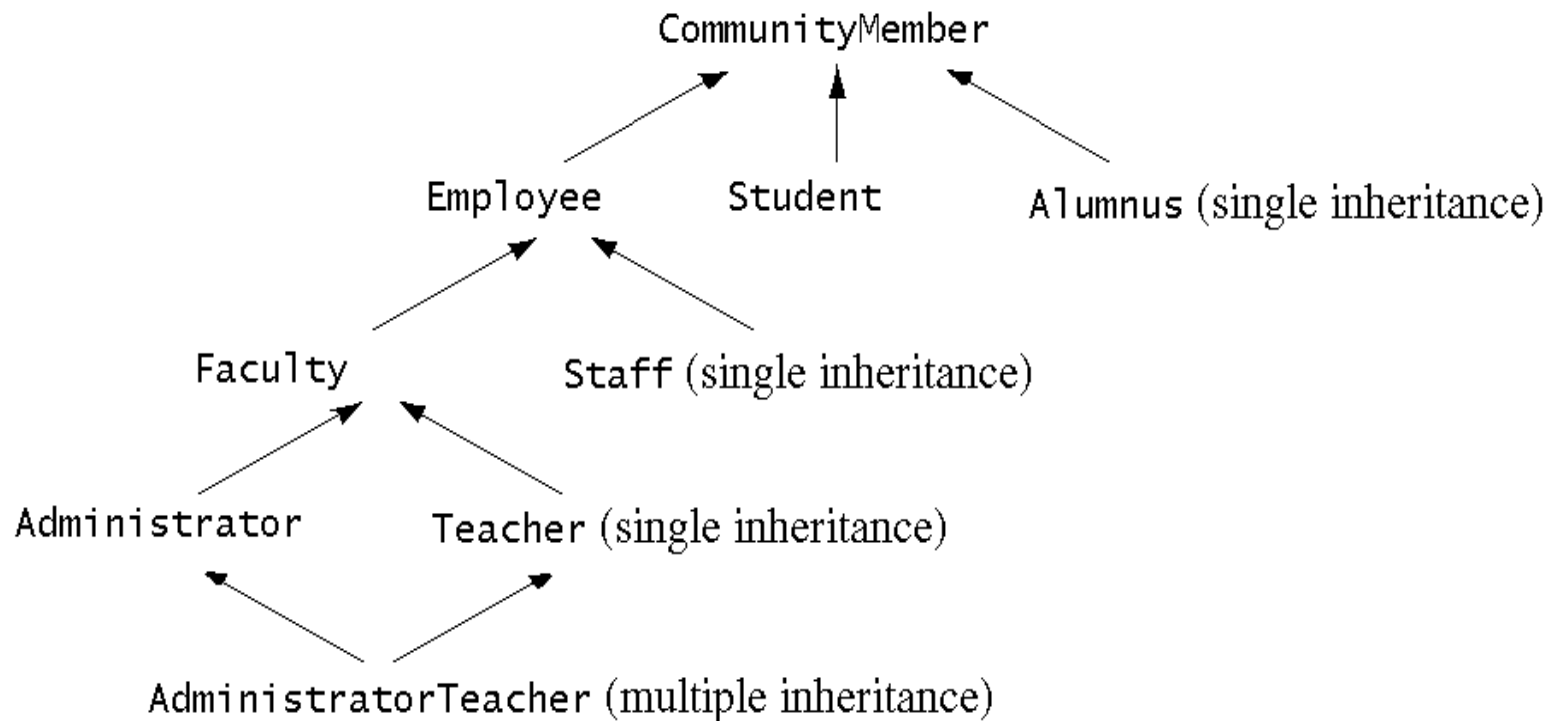
```
void main()
```

```
{ derived d(5); d.d(); d.display(); }
```

Inheritance

- Direct base class – from which derived is directly inherited from base class
- Indirect base class – inherited from two or more levels
- Single inheritance – one base class
- Multiple inheritance – multiple base classes

Example



Access specifiers

- Derived class can access the non-private members of its base class
- Though derived class can change the private members through non-private member functions of base class

Is-a relationship

- Is-a relationship represent inheritance
- Object of derived class also can be treated as an object of the base class

protected access specifier

A base class protected members can be accessed

- within the body of that base class
- by members and friends of that base class
- and by members and friends of any classes derived from that base class

Access specifiers

Base class member access specifier	Type of inheritance		
	public inheritance	protected inheritance	private inheritance
public	public in derived class. Can be accessed directly by any non-static member functions, friend functions and non-member functions.	protected in derived class. Can be accessed directly by all non-static member functions and friend functions.	private in derived class. Can be accessed directly by all non-static member functions and friend functions.
	protected in derived class. Can be accessed directly by all non-static member functions and friend functions.	protected in derived class. Can be accessed directly by all non-static member functions and friend functions.	private in derived class. Can be accessed directly by all non-static member functions and friend functions.
protected	Hidden in derived class. Can be accessed by non-static member functions and friend functions through public or protected member functions of the base class.	Hidden in derived class. Can be accessed by non-static member functions and friend functions through public or protected member functions of the base class.	Hidden in derived class. Can be accessed by non-static member functions and friend functions through public or protected member functions of the base class.

```
class base
{
    private: int x;
    protected: int y;
    public: int z;
    base() { x=0; y=0; z=0;}
};
```

```
void main()
{ base b; derived1 d1;
  derived2 d2; derived3 d3;
  cout<<b.z;
  cout<<d1.z;
  //cout<<d2.z; ERROR
  //cout<<d3.z; ERROR
  d1.display(); d2.display();
  d3.display();
}
```

```
class derived1 : public base
{ public:
    void display()
    { cout<<"\n Public inheritance
      Y:"<<y<<" Z:"<<z; };
```

```
class derived2 : protected base
{ public:
    void display()
    { cout<<"\n Public inheritance
      Y:"<<y<<" Z:"<<z; };
```

```
class derived3 : private base
{ public:
    void display()
    { cout<<"\n Public inheritance
      Y:"<<y<<" Z:"<<z; } };
```

Assignment

- Explain various types of Inheritance.