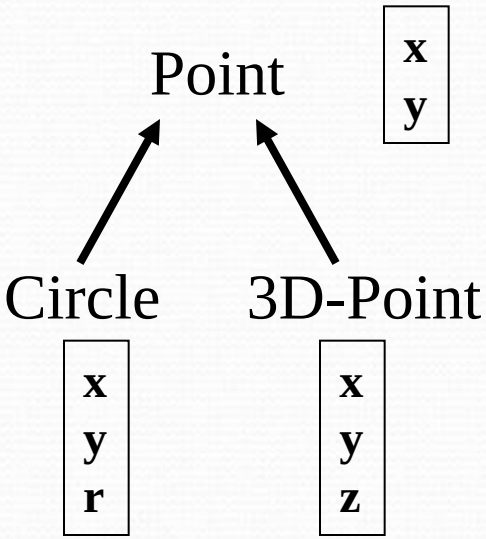
Section 3-Lecture 10

Introduction

- Introduction to Inheritance, Base Classes And Derived Classes, Protected Members, Casting
- Base- Class Pointers to Derived- Class Pointers, Using Member Functions, Overriding Base –
- Class Members in a Derived Class, Public, Protected and Private Inheritance.

.

Inheritance Concept




```
class Point{
    protected:
        int x, y;
    public:
        void set (int a, int b);
};
```

```
class Circle : public Point{
    private:
            double r;
};
```

```
class 3D-Point: public Point{
    private:
            int z;
};
```

Introduction

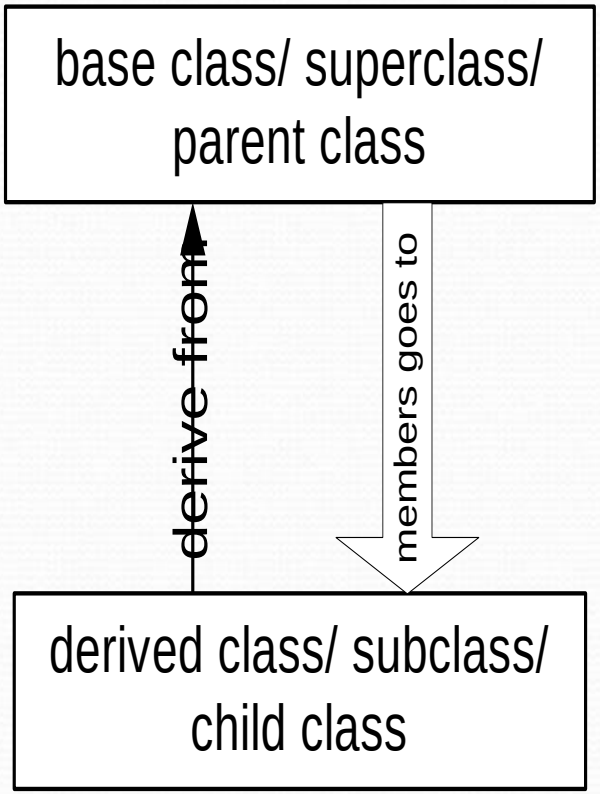
- Inheritance
  - Single Inheritance
    - Class inherits from one base class
    - -- Multiple Inheritance
    - Class inherits from multiple base classes
  - Three types of inheritance:
    - `public`: Derived objects are accessible by the base class objects (focus of this chapter)
    - `private`: Derived objects are inaccessible by the base class
    - `protected`: Derived classes and `friend`s can access protected members of the base class

single Inheritance

```
class Shape {
  double x, y;              // Base coordinates of shape
public:
  void translate(double dx, double dy) {
     x += dx; y += dy;
  }
};

class Line : public Shape {
void translate(double dx, double dy) {
     x += dx; y += dy;
  }

};

Line l;
l.translate(1,3);                  // Invoke Shape::translate()
```

Line inherits both the representation and member functions of the Shape class

Access Control Over the Members



- Two levels of access control over class members
  - class definition
  - inheritance type

```
class Point{
    protected: int x, y;
    public: void set(int a, int b);
};
```

```
class Circle : public Point{
    … …
};
```

Controlling Inheritance

Inheritance Type	Base class member access	Derived class member access
public	public	public
	protected	protected
	private	inaccessible
protected	public	protected
	protected	protected
	private	inaccessible
private	public	private
	protected	private
	private	inaccessible

Ambiguity resolution in Inheritance(multiple inheritance)

```
Class M
{
Public:
  void display(void)
{  cout<<“class M”;  }
};
Class N
{
Public:
Void display(void)
{  cout<<“class N”;}
};
```

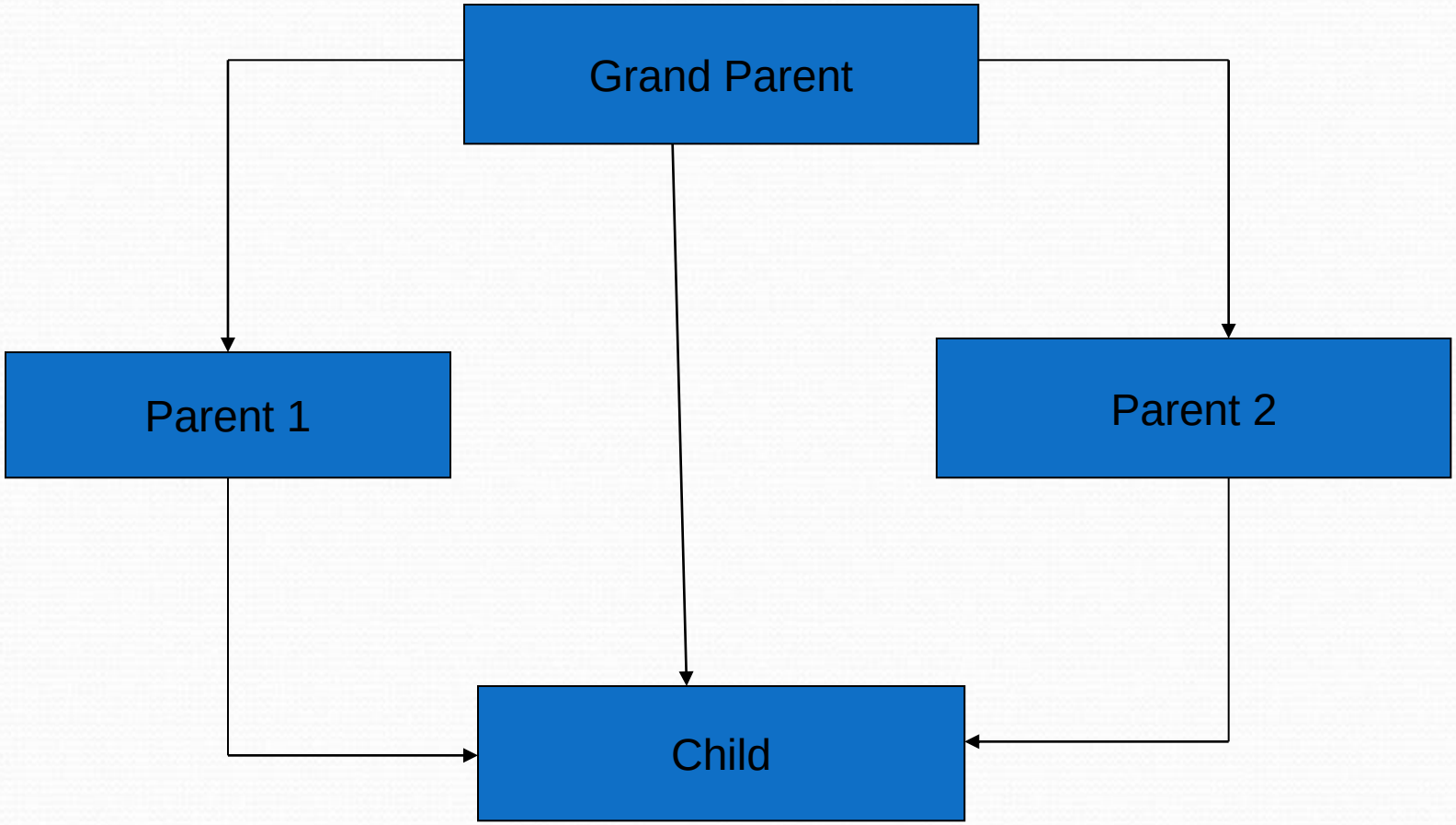
```
class P:public M,public N
{
public:
        void display(void)
        {
        M::display();
        }
        };
int main()
        {
        P p;
        p.display();
        }
```

Ambiguity resolution in Inheritance (single inheritance)

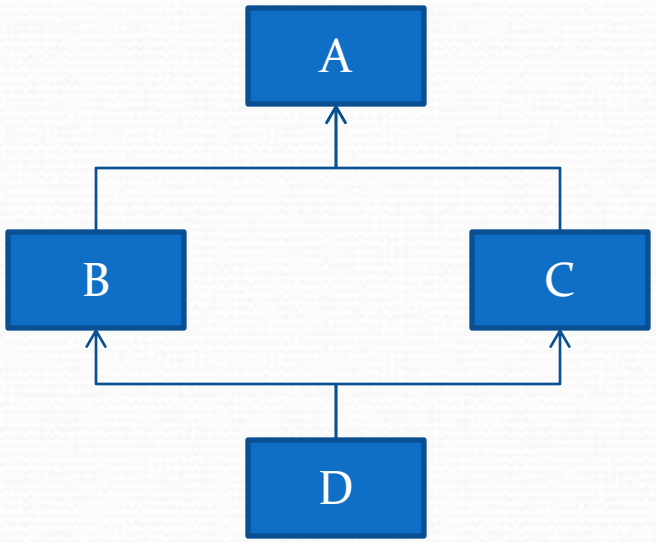
```
Class A
{
Public:
   void display(void)
{   cout<<“class A”;  }
};
Class B:public A
{
Public:
Void display(void)
{   cout<<“class B”;}
};
```

```
void main()
        {
                B b;
                b.display();
                b.A::display()
                //invokes display in A
        }
```

Virtual base class



Virtual Inheritance




```
class A
{
 public:
 int a;
};
```

```
class B : public virtual A
{
 public:
 int b;
};
```

```
class C : public virtual A
{
 public:
 int c;
};
```

- Multiple copy of same base class sub-object eliminated

```
class D : public B, public C
{
 public:
 int d;
};
```

Virtual base class

```
Class A
{ ....};                                  //grandparent
Class B1: virtual public A        //parent1
{.....};
Class B2: virtual public A        //parent2
{.....};
Class C:  public B1,public B2     //child
{.....};                                  //only one copy of A
                                          will be inherited
```