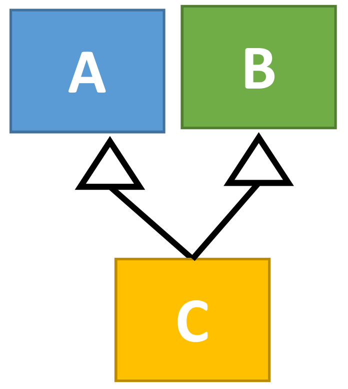
Class/Object Relationships

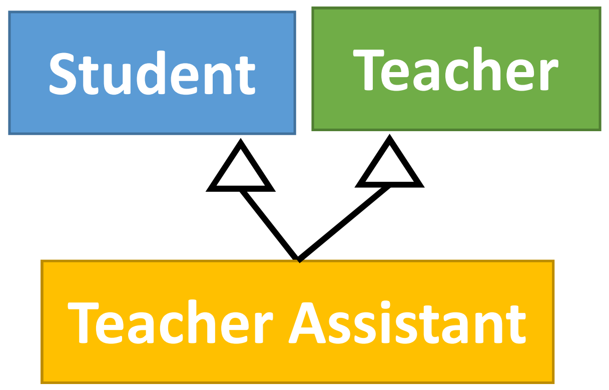
CS(217) Object Oriented Programming

Inheritance (is-a) **Multiple Inheritance**

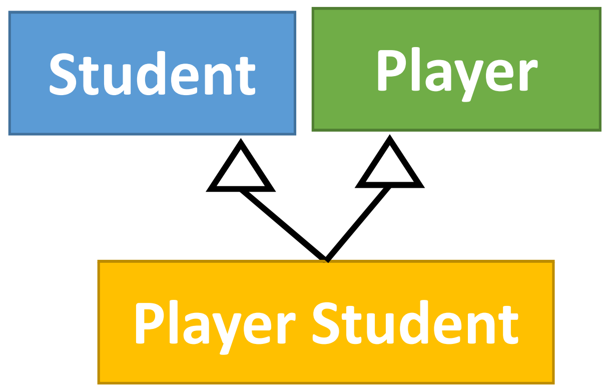
- Inherit directly from more than one base classes
- (Abstract or Concrete)
- Base classes possibly unrelated



Multiple

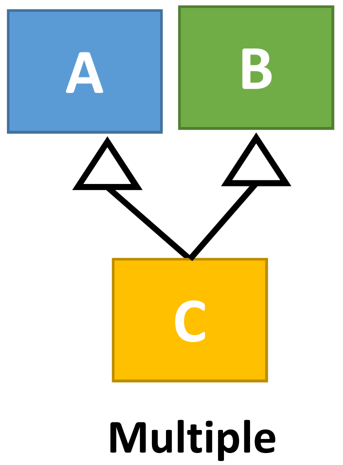


Multiple



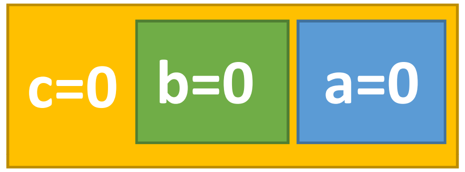
Multiple

Inheritance (is-a) **Multiple Inheritance**

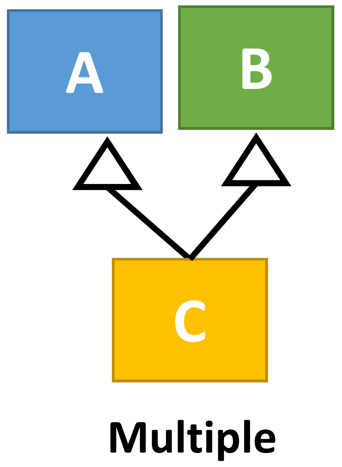


```
class A{
    int a;
public:
    A(int a=0){ this->a=a;}
};

class B{
    int b;
public:
    B(int b=0){ this->b = b;}
};
```

```
class C: public A, public B{
    int c;
public:
  //Call constructors of both base classes
    C(int a=0, int b=0, int c=0) :A(a), B(b)
    { this->c = c;}
};

void main(){
    C c1;
    //C inherited data from both A and B
}
```

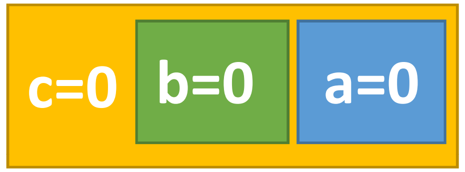


Inheritance (is-a) **Multiple Inheritance**



```
class A{
    int a;
public:
    A(int a=0){ this->a=a;}
    void print(){ cout<<a;}
};


class B{
    int b;
public:
    B(int b=0){ this->b = b;}
    void print(){ cout<<b;}
};
```

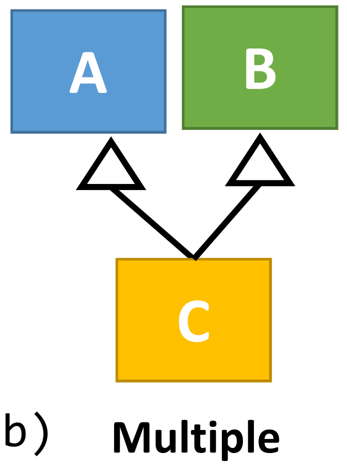
```
class C: public A, public B{
    int c;
public:
   C(int a=0, int b=0, int c=0)
    :A(a), B(b)
    { this->c = c;}
};
void main(){
    C c1;
    c1.print();
}
```



Issue 1: Base classes may have same functions.

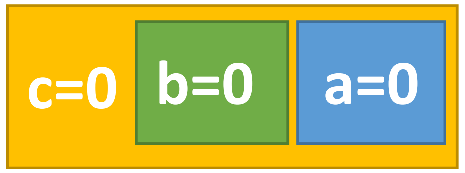
Which print function will be called A's or B's?

Inheritance (is-a) **Multiple Inheritance**



```
class A{
    int a;
public:
    A(int a=0){ this->a=a;}
    void print(){ cout<<a;}
};


class B{
    int b;
public:
    B(int b=0){ this->b = b;}
    void print(){ cout<<b;}
};
```

```
class C: public A, public B{
    int c;
public:
    C(int a=0, int b=0, int c=0) :A(a), B(b)
    { this->c = c;}
};
void main(){
    C c1;
    c1.A::print();
    c1.B::print();
}
```



Issue 1: Base classes may have same functions.

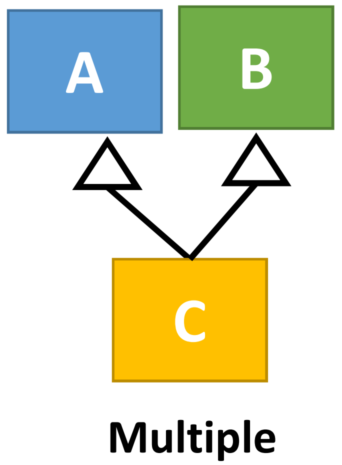
Provide explicit name of base class to call same function and to resolve the conflict.

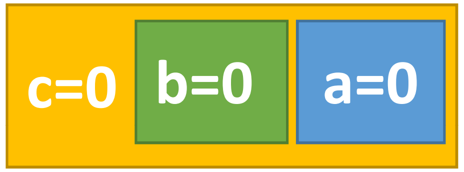
C’s data will not print!

Inheritance (is-a) **Function Overriding**

```
class A{
    int a;
public:
    A(int a=0){ this->a=a;}
    void print(){ cout<<a;}
};


class B{
    int b;
public:
    B(int b=0){ this->b = b;}
    void print(){ cout<<b;}
};
```

```
class C: public A, public B{
    int c;
public:
   C(int a=0, int b=0, int c=0) :A(a), B(b)
    { this->c = c;}
    void print(){
        A::print();
        B::print();
        cout<<c;
    }
};
void main(){
    C c1;
    c1.print();
}
```

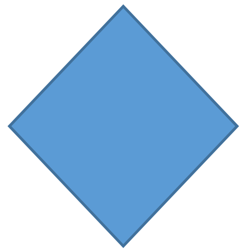


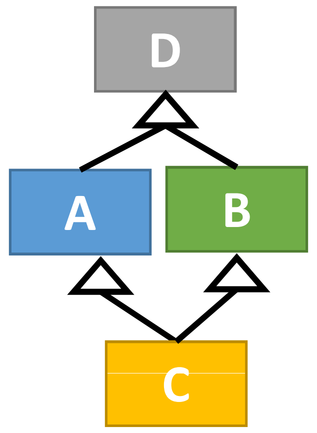


Issue 1: Base classes may have same functions.

Override the inherited functions and call base class functions explicitly in overridden function.

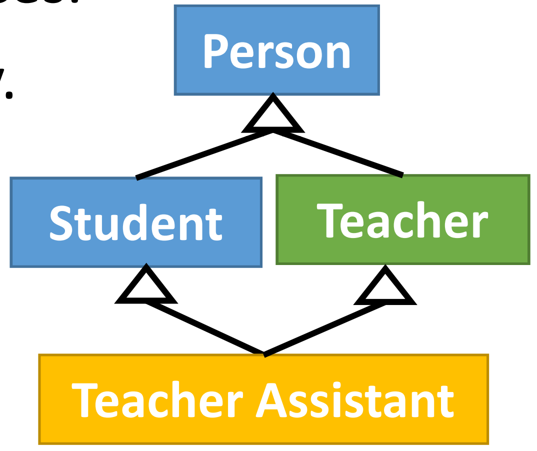
Inheritance (is-a) **Diamond Problem**



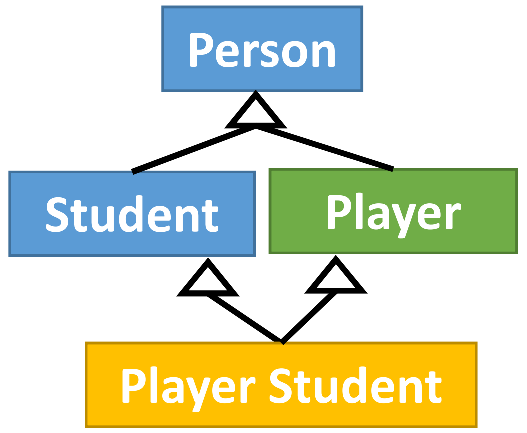


Multiple

- More than one base classes are inherited from a common base class.
  - System will inherit two Person objects in Teacher Assistant one from Student and one from Teacher?
  - Similarly system will inherit two Person objects in Player Student one from Student and other from Player class?
- Maintenance overhead to explicitly handle each class members.
- Causes more problems and ambiguous references.
- Resolve through virtual base classes in C++ only.



Multiple



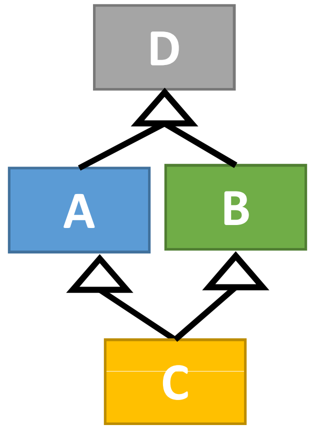
Multiple

Inheritance (is-a) **Virtual Class**

- Virtual base class has only one instance of base object in derived classes.
- The objects of virtual base class are created before derived classes
- All classes will share same object of base class.
- Only one time constructor is called on virtual base class object.

```
class D{};
class A: virtual public D{}; //Virtual inheritance of D in A
class B: virtual public D{}; //Virtual inheritance of D in B
class C: public A, public B{};
 //Now one copy of D’s object will be created in Class C
instead of two.
```

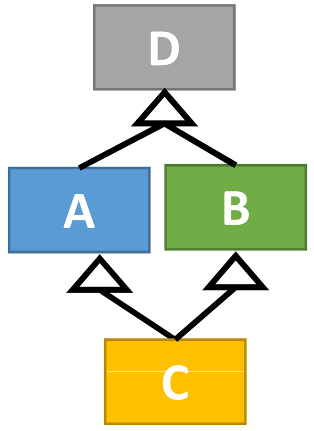
Issue: Which derived class is responsible for calling the constructor of class D?



Multiple

Inheritance (is-a) **Virtual Class**

- All derived classes should add constructer call for their base classes.
- A derived class can also call the constructor of Grand Parent base class.

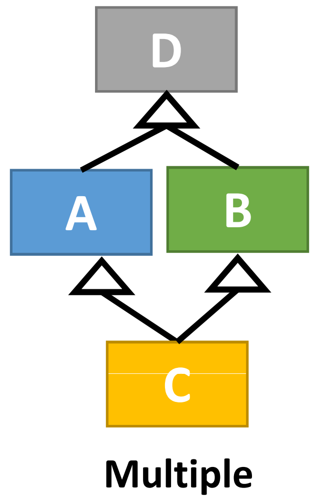


Multiple

```
class D{ int d
    int d;
public:
    D(int d=0){ this->d=d;}
};
class A: virtual public D{
    int a;
public:
    A(int a=0, int d=0) :D(d)
    { this->a=a;}
};
```

```
class B: virtual public D{
    int b;
public:
    B(int b=0, int d=0) :D(d)
    { this->b=b;}
};
class C: public A, public B{
    int c;
public:
  //Call constructors of all base classes
    C(int a=0, int b=0, int c=0, int d=0)
    :D(d), A(a), B(b),
    { this->c = c;}
};
```

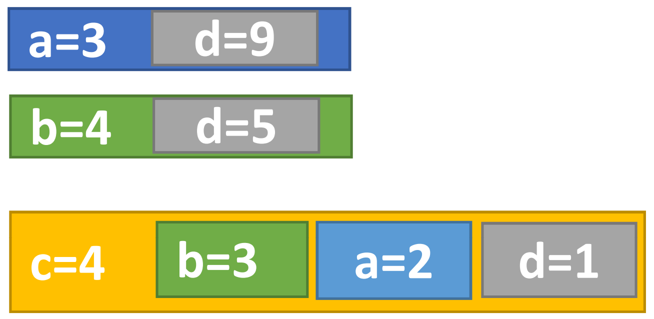
Inheritance (is-a) **Virtual Class**



Which derived class is responsible for calling the constructor of class D?

- Only one time constructor is called on virtual base class object.
- Constructor call depends on type of the object that is crated.
- The last derived class is responsible for constructor call.

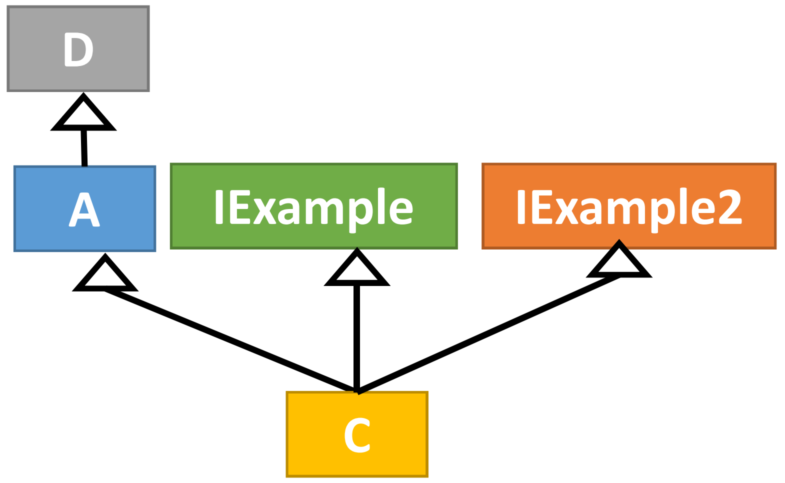
```
void main(){
    A a1(3, 9); //Call constructor of D from A
    B b1(4, 5); //Call constructor of D from B

    C c1 (3, 2, 4, 1); //Call constructor of D from C

    //Now constructor call from A and B is skipped by system as
    one copy of D’s object is already created in C’s object
}
```



Inheritance (is-a) **Multiple Inheritance and Interface**

- Some Object oriented languages such as java and c# does not allow multiple inheritance of concrete and abstract classes.
- A derived class can inherit data from only single base class (Abstract or Concrete).
- But, they allow to inherit multiple Interfaces.
  - Interface classes has no data members, and have all pure virtual functions.
  - Therefore issue of diamond problem is avoided completely by using them.

```cpp
class IExample{
public:
// Pure virtual functions
    virtual void Function1() = 0;
    virtual void Function2() = 0;
    virtual void Function3() = 0;
    virtual ~IExample();
};
```


D
A
IExample
IExample2
C