Class/Object Relationships

Inheritance & Compile Time Binding

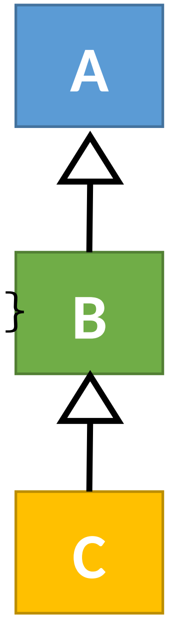
CS(217) Object Oriented Programming

Abeeda Akram

Inheritance (is-a)

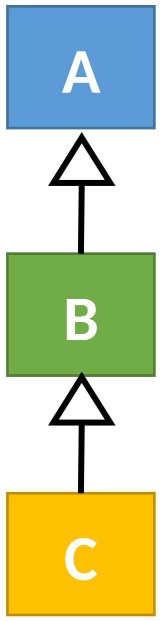
```
class A{
    int a;
public:
    A(int a=0){ this->a=a;}
    void print(){ cout<<a;}
};
```

```
class B: public A{
    int b;
public:
    B(int a=0, int b=0):A(a) { this->b = b;}
//override print function inherited from A
    void print() {
        A::print();
        cout<<b;
    }
//overload print function inherited from A
    void print(int x){ cout<<x+b; }
//Add New Function in class B
    void funb() { cout<<“funb”<<endl };
};
```

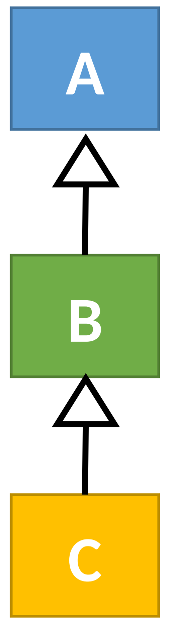


Inheritance (is-a)

```
class C: public B{
    int c;
public:
    C(int a=0, int b=0, int c=0) :B(a,b)
    { this->c = c;}
//override print function inherited from B
    void print() {
        B::print();
        cout<<c;
    }
//overload print function inherited from B
    void print(int x, int y){
        cout<<x+y+c;
    }
//Add New Function in class C
    void func(){ cout<< “func” <<endl; }
};
```

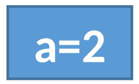


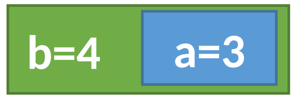
Inheritance (is-a) Compile Time Binding

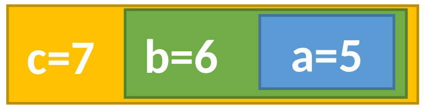


- Call the functions on object according to the type of object.
- Cannot change compile time binding of static objects.
- Can be changed for pointers or reference to objects.

```
void main(){
    A a1(2);
    a1.print(); //Base print called print a’s data

    B b1 (3,4);
    b1.print(); //Derived print called print a’s and b’s data
    b1.print(3); //overloaded print called
    b1.funb(); //new function defined in b called

    C c1 (5,6,7);
    c1.print(); //Derived print called print a’s and b’s and c’s data
    c1.print(3); //inherited print called
    c1.funb(); //inherited funb called
    c1.print(3, 5); //overloaded print called
    c1.func(); //new function defined in c called
}
```



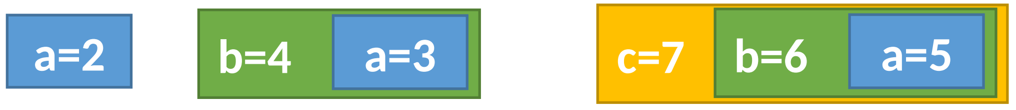


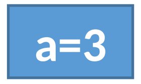


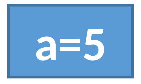
Inheritance (is-a) Object Slicing Problem

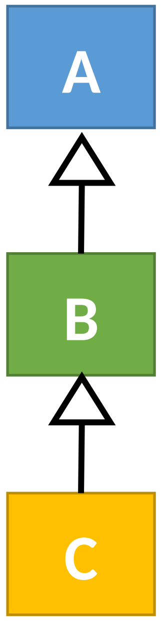
- We can assign derived class object to base class object.
- **Only copy data of base class, derived portion is discarded.**
- **Compile time binding system will call base class functions only.**

```
void main(){
   A a1 (2);
   B b1 (3, 4);
   C c1 (5,6,7);

   a1 = b1; //slice b1 and copy in a1 the A’s portion only.
   a1.print(); //Base print called print A’s data only.
   a1.funb(); a1.print(3); //Compile time Error

   a1 = c1; //slice c1 and copy in a1 the A’s portion only.
   a1.print(); //Base print called print A’s data only.
   a1.funb(); a1.func();    //Compile time Error
   a1.print(3); a1.print(3, 4); //Compile time Error

}
```



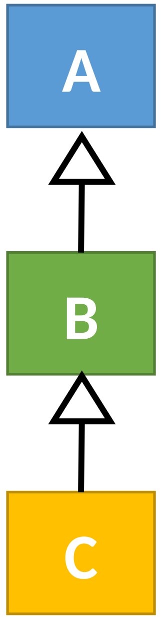


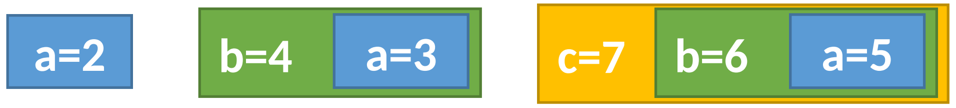


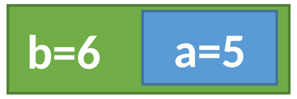


Inheritance (is-a) Object Slicing Problem

- We can assign derived class object to base class object.
- Only copy data of base class, derived portion is discarded.
- Compile time binding system will call base class functions only.



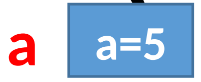


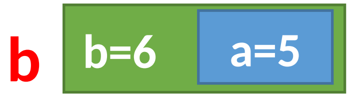


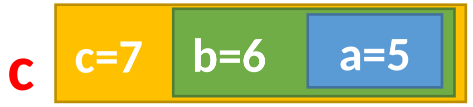
```
void main(){
   A a1 (2);
   B b1 (3, 4);
   C c1 (5,6,7);

   b1 = c1; //slice c1 and copy in b1 the B’s portion only.
   b1.print(); //Base print called print B’s and A’s only.
   b1.print(3); //overloaded print called.
    b1.func(); b1.print(3, 4); //Compile time Error

   c1 = b1; or c1 = a1;
   //Compile time Error: We cannot assign base class object to derived class.
   b1 = a1; //Every derived is a base but every base is not a derived.
}
```

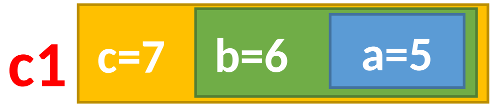
Inheritance (is-a) Object Slicing Problem

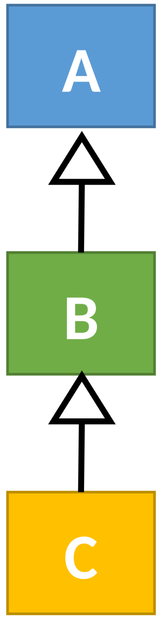
```
void fun(A a){
   a.print(); //Base print called print A’s data only.
}
void fun2(B b){
   b.print(); //Base print called print B’s data.
   b.print(5);
}
void fun3(C c){
   c.print(); //C’s print called.
   c.print(5,9);
}
void main(){
   C c1 (5,6,7);
   fun(c1);  //slice c1 to A’s object
   fun2(c1); //slice c1 to B’s object
   fun3(c1);
}
```



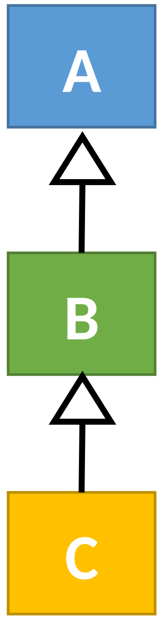






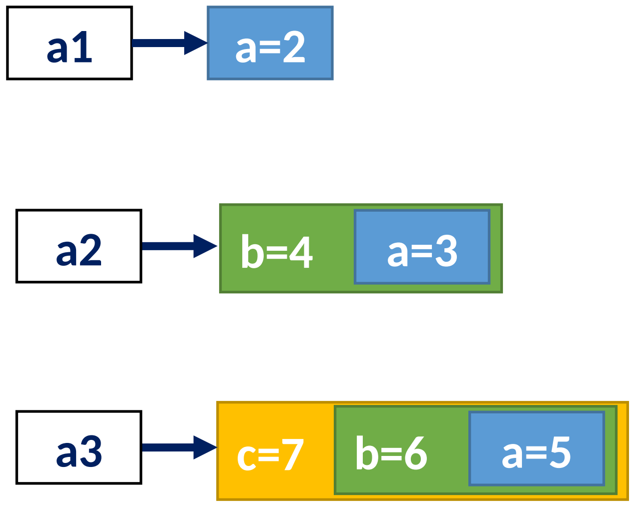


Inheritance (is-a) Base and Derived Pointers

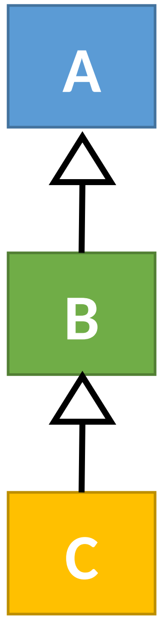


- Base class pointer can point to base or derived class objects.
- Base class pointer can only see base class members, derived part still exist.
- Can only call inherited functions not extra functions of derived class.
- **Compile time binding, system will call base class functions only according to type of pointer, not call overridden ones.**

```
void main(){
   A * a1 = new A(2); //A’s pointer to A’s object
   a1->print(); //A’s print called.

   A * a2 = new B(3, 4); //A’s pointer to B’s object
   a2->print(); //A’s print called.
   a2->funb(); a2->print(3); //Compile time Error

   A * a3 = new C(5, 6, 7); //A’s pointer to C’s object
   a3->print(); //A’s print called.
   a3->funb(); a3->print(3); //Compile time Error
   a3->func(); a3->print(3,8); //Compile time Error
}
```

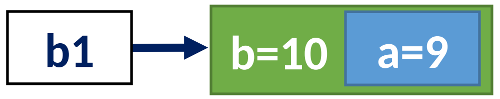


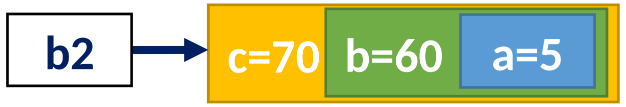
Inheritance (is-a) Base and Derived Pointers



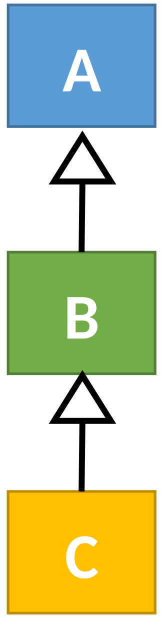
- Derived class pointer can only point to its own or further derived class objects.
- **Compile time binding, system will call derived class functions only according to type of pointer.**

```
void main(){
   B * b1 = new B(9, 10); //B’s pointer to B’s object
   b1->print(); //B’s print called.
   b1->funb(); b1->print(3);

   B * b2 = new C (5, 60, 70); //B’s pointer to C’s object
   b2->print(); //B’s print called.
   b2->funb(); b2->print(3);
   b2->func(); b2->print(3,8); //Compile time Error

   B * b3 = new A(2); //Error: B’s pointer to A’s object
   //Every derived is a base but every base is not a derived.
   //Allowed if explicit cast made
}
```



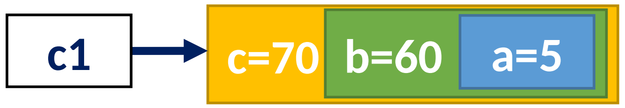


Inheritance (is-a) Base and Derived Pointers

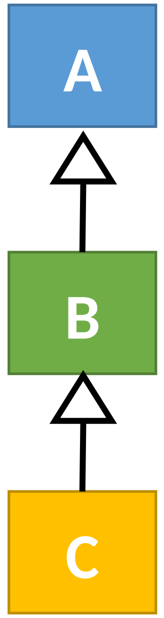


- Derived class pointer can only point to its own or further derived class objects.
- **Compile time binding, system will call derived class functions only according to type of pointer.**

```
void main(){
    C * c1 = new C(5, 60, 70); //C’s pointer to C’s object
    c1-> print(); //C’s print called.
    c1-> funb();
    c1-> print(3);
    c1-> func();
    c1-> print(3,8);

    C * c2 = new B(5,6); //Error: C’s pointer to B’s object
    C * c3 = new A(2);  //Error: C’s pointer to A’s object
    //Every derived is a base but every base is not a derived.
    //Allowed if explicit cast made
}
```

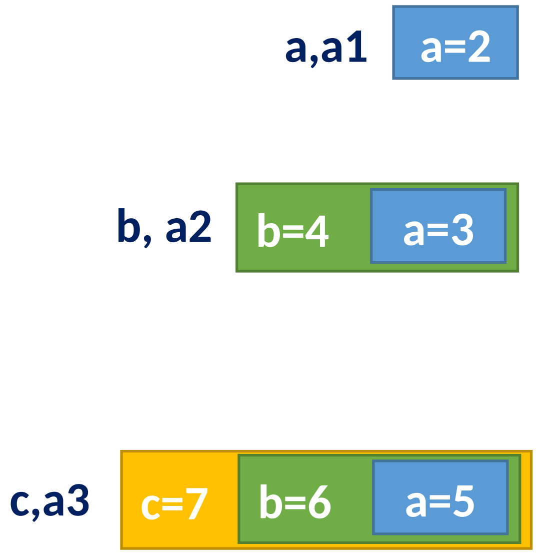


Inheritance (is-a) Base and Derived references



- Base class reference can be created for base or derived class objects.
- Base class reference can only see base class members, derived part still exist.
- Can only call inherited functions only not extra functions of derived class.
- **Compile time binding, system will call base class functions only according to type of reference , not call overridden ones.**

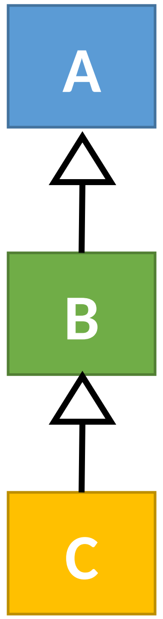
```
void main(){
   A a(2);  B b(3, 4);  C c(5,6,7);
   A & a1 = a;
   a1.print(); //A’s print called.

   A & a2 = b; //A’s reference to B’s object
   a2.print(); //A’s print called.
   b.print(3); //B’s print called
   b.funb(); //new function defined in B called
   a2.funb(); a2.print(3); //Compile time Error

   A & a3 = c; //A’s reference to C’s object
   a3.print(); //A’s print called.
   a3.funb(); a3.print(3); //Compile time Error
   a3.func(); a3.print(3,8); //Compile time Error
}
```

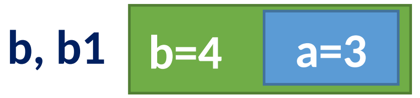


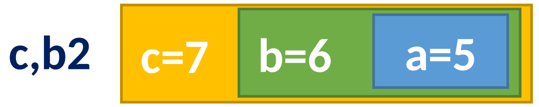
Inheritance (is-a) Base and Derived references

- Base class reference can be created for base or derived class objects.
- Base class reference can only see base class members, derived part still exist.
- **Compile time binding, system will call derived class functions only according to type of reference .**

```
void main(){
   A a(2);  B b(3, 4);  C c(5,6,7);

   B & b1 = b; //B’s reference to B’s object
   b1.print(); //B’s print called.
   b1.funb(); b1.print(3);

   B & b2 = c; //B’s reference to C’s object
   b2.print(); //B’s print called.
   b2.funb(); b2.print(3);
   b2.func(); b2.print(3,8); //Compile time Error

   B & b3 = a; //Error: B’s pointer to A’s object
   //Every derived is a base but every base is not a derived.
   //Allowed if explicit cast made
}
```



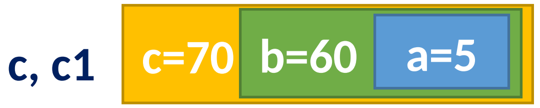


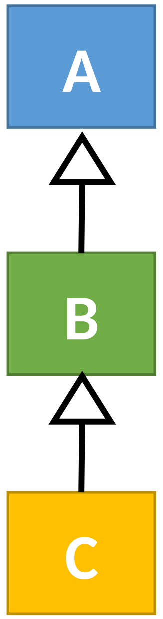


Inheritance (is-a) Base and Derived references

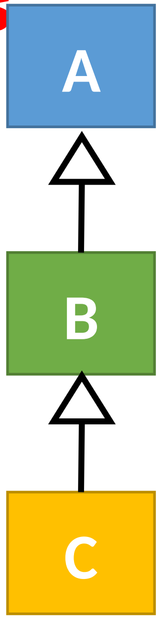
- Derived class reference can be created for derived class objects.
- **Compile time binding, system will call derived class functions only according to type of reference .**

```
void main(){
   A a(2);  B b(3, 4);  C c(5,6,7);

   C & c1 = c; //C's reference to C's object
   c1.print(); //C's print called.
   c1.funb();
   c1.print(3);
   c1.func();
   c1.print(3,8);

   C & c2 = b; //Error: C's reference to B's object
   C & c3 = a;  //Error: C's reference to A's object
   //Every derived is a base but every base is not a derived.
   //Allowed if explicit cast made
}
```



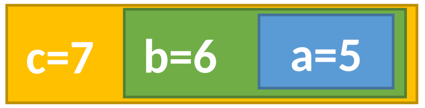


Inheritance (is-a) Base and Derived references



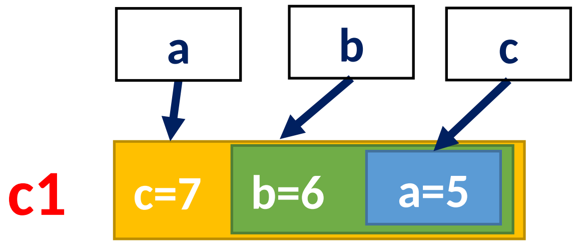
```
void fun(A & a){
    a.print(); //A’s print called print A’s data only.
}
void fun2(B & b){
    b.print(); //B’s print called.
    b.print(5);
}
void fun3(C & c){
    c.print(); //C’s print called.
    c.print(5,9);
}
void main(){
    C c1 (5,6,7);
    fun(c1);
    fun2(c1);
    fun3(c1);
}
```

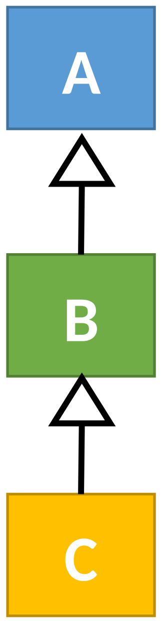
a, b, c, c1



Inheritance (is-a) Base and Derived Pointers

```
void fun(A * a){
    a->print(); //A’s print called print A’s data only.
}
void fun2(B * b){
    b->print(); //B’s print called.
    b->print(5);
}
void fun3(C * c){
    c->print(); //C’s print called
    c->print(5,9);
}
void main(){
    C c1 (5,6,7);
    fun(&c1);
    fun2(&c1);
    fun3(&c1);
}
```





Inheritance (is-a) Compile Time Binding

Base class static object		Call base class functions only
Derived class static object		Call derived class functions & inherited functions. Overridden and overloaded functions.
Base class static object	Derived class static object	Call base class functions **Slicing Issue** only copies base data in base object
Derived class static object	Base class static object	Error: Explicit cast required
Base class pointer or reference	Base class object	Call base class functions
Base class pointer or reference	Derived class object	Call base class functions only
Derived class pointer or reference	Base class object	Error: Explicit cast required
Derived class pointer or reference	Derived class object	Call derived class functions & inherited functions. Overridden and overloaded functions.