Class/Object Relationships

Inheritance

CS(217) Object Oriented Programming

Abeeda Akram

Inheritance (is-a) Non-inherited Members

Members that are not Inherited from base class are

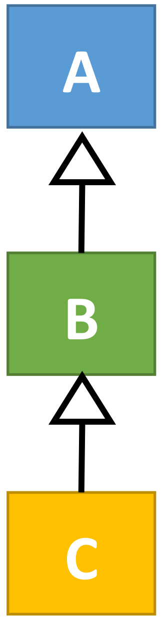
1. **Constructors**
2. **Destructor**
3. **Assignment operator**
4. **Non-member functions**

- Derived class constructors, destructor and assignment operators can call Base class constructors, destructor and assignment operators

Inheritance (is-a) Constructors in Derived Classes

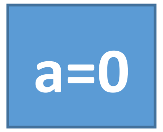
- Chain of constructor calls
  - **Derived-class constructor invokes base class constructor**
    - Implicitly by system default constructor
    - Explicitly by programmer parametrized or copy constructor.
  - **Base of inheritance hierarchy**
    - Last constructor called in chain
    - First constructor body to finish executing
  - **Initializing data members**
    - Each base-class constructor initializes its own data members
      - Inherited by derived class

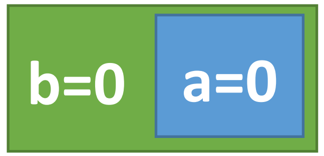
Inheritance (is-a) Default Constructors

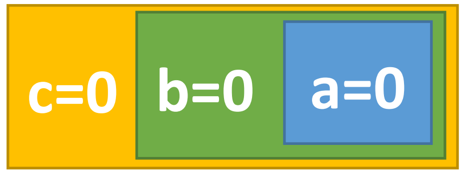


```
class A{
    int a;
public:
    A(){ this->a=0;}
    void print(){ cout<<a;}
};
class B: public A{
    int b;
public:
    B(){ this->b = 0;}
};
class C: public B{
    int c;
public:
    C(){ this->c = 0;}
};
```

```
void main(){
    A a1;
    //A default constructor called

    B b1;

    //B’s and A’s default constructor is
    implicitly called by system

    C c1;

    //C’s, B’s and A’s default
    constructor is implicitly called by
    system
}
```





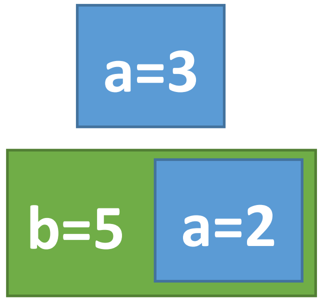


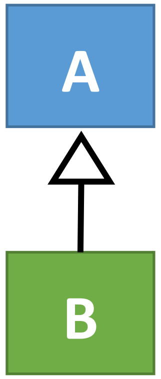
Inheritance (is-a) Parametrized Constructors

How to call specific constructors of base class?

```
class A{
    int a;
public:
    A(int a=0){ this->a=a;}
    void print(){ cout<<a;}
};
class B: public A{
    int b;
public:
    //call parametrized
    constructor of A
    B(int a=0, int b=0):A(a)
    { this->b = b;}
};
```

```
void main(){
    A a1(3);
    a1.print();

    B b1 (2,5);
    //Explicitly call parametrized
    constructor of B, A's constructor is
    called by B.
}
```





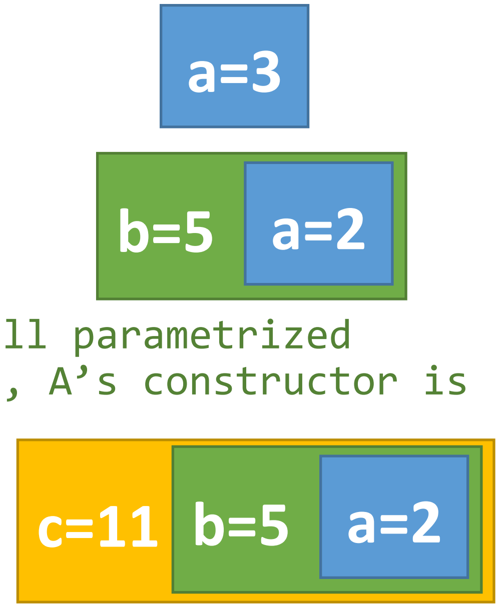
- **Member initializer syntax used to call the parameterized constructor of base class**

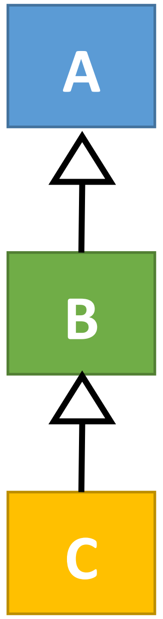
Inheritance (is-a) Parametrized Constructors

How to call specific constructors of base class?

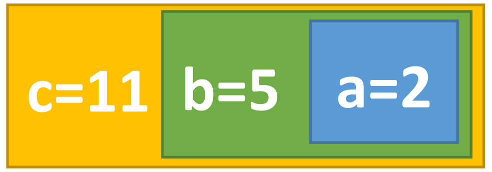
```
class C: public B{
    int c;
public:
    //call parametrized constructor
    C(int a=0, int b=0, int c=0)
    :B(a,b)
    { this->c = c;}
};
```

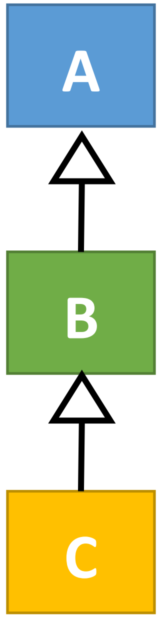
```
void main(){
    A a1(3);
    a1.print();

    B b1 (2,5);
    // Explicitly call parametrized
    constructor of B, A’s constructor is
    called by B too.

    C c1 (2,5,11);
    // Explicitly call parametrized
    constructor of C, B’s constructor is
    called by C, and A’s constructor is
    called by B.
}
```





Inheritance (is-a) Constructors in Derived Classes





```
void main(){
    C c1 (2,5,11);
    //call parametrized constructor of C, B’s constructor is called by C,
    and A’s constructor is called by B.
}
```

- **Constructor Calling Implicit or Explicit:**

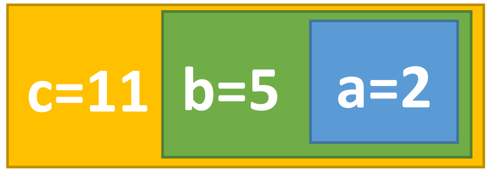
  1)C 2)B 3)A

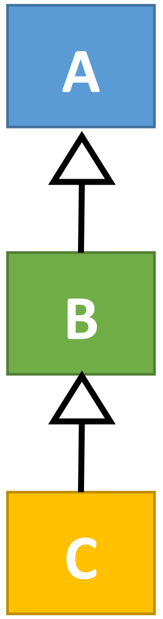
- **Constructor Execution:**

  in reverse order of inheritance from derived to base

  1)A 2)B 3)C

Inheritance (is-a) Destructor in Derived Classes





```
void main(){
    C c1 (2,5,11);
    //call parametrized constructor of C, B’s constructor is called by C,
    and A’s constructor is called by B.
}
```

- **Destructor Call and Execution:**
  - In reverse order of inheritance from derived to base
  - First destroy derived objects then base inherited objects

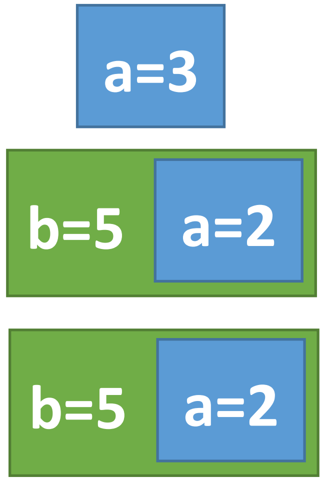
    1)~C 2)~B 3)~A

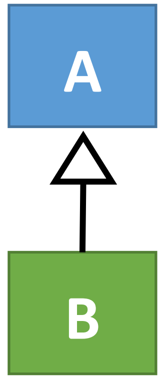
Inheritance (is-a) Copy Constructors

How to call specific constructors of base class?

```
class A{
    int a;
public:
    A(int a=0){ this->a=a;}
    A(const A& obj){ a = obj.a;}
    void print(){ cout<<a;}
};
class B: public A{
    int b;
public:
    //call parametrized constructor of A
    B(int a=0, int b=0):A(a)
    { this->b = b;}
    B(const B& obj):A(obj){
    b = obj.b;
    }
};
```

```
void main(){
    A a1(3);
    a1.print();

    B b1 (2,5);

    B b2 (b1);


    //Explicitly call copy constructor
    of B, A's copy constructor is
    called by B.
}
```



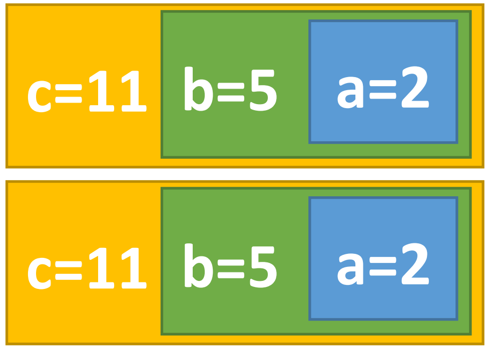


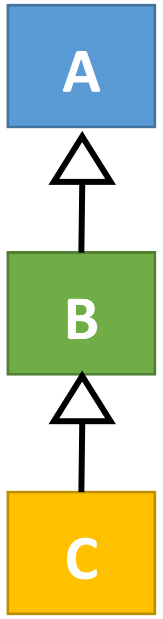
Inheritance (is-a) Copy Constructors

How to call specific constructors of base class?

```
class C: public B{
    int c;
public:
    //call parametrized constructor
    C(int a=0, int b=0, int c=0)
    :B(a,b)
    { this->c = c;}
    C(const C& obj):B(obj){
        c = obj.c;
    }
};
```

```
void main(){

    C c1 (2,5,11);
    C c2 (c1);

    // Explicitly call copy
    constructor of C, B’s copy
    constructor is called by C, and
    A’s copy constructor is called
    by B.
}
```





Inheritance (is-a) Function Overriding

- Many Inherited functions may have limited functionality related to base class members only
- Need to add more instructions in functions for derived class
- **Redefining inherited function in derived class with**
  - **Same Name**
  - **Same number, type, and order of parameters.**

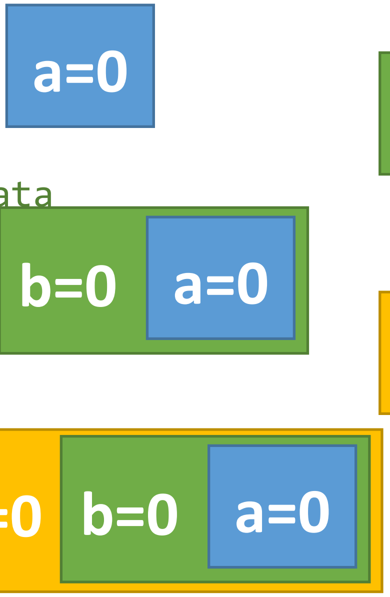
is called function overriding.

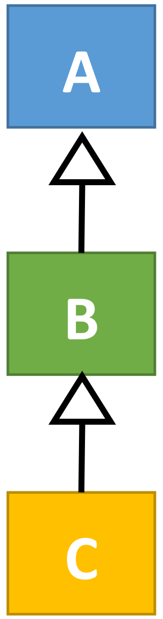
Inheritance (is-a) Function Overriding

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

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

   B b1;
   b1.print();
   //inherited print of A is called
   print a's data not b's

   C c1;
   c1.print();
   //inherited print of A is called
   print a's data not of c and b

   }
```

Base class function is limited to its members printing only.

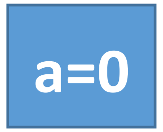


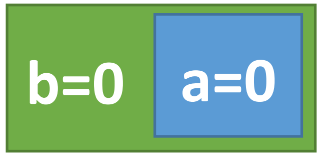


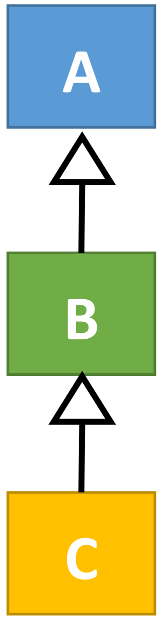
Inheritance (is-a) Function Overriding

```
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(){
    cout<<b;
    }
};
```

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

    B b1;
    b1.print();
    //overridden function called print b's
    data only not a's

    }
```





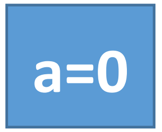


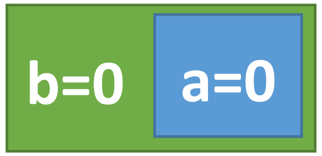
Redefine code only no change in function name and parameters.

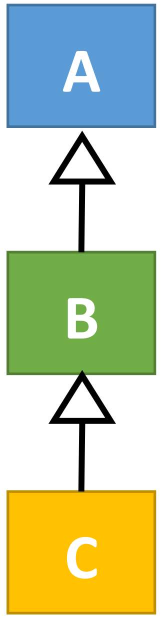
Inheritance (is-a) Function Overriding

```
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 inherited function from A
    void print(){
//calls base class print
A::print();
        cout<<b;
    }
};
```

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


    B b1;
    b1.print();
    //overridden function called, first calls A's
    print to print a's data then print b's data

    }
```







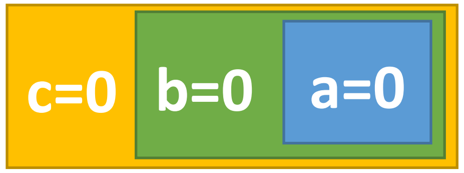
Can call inherited function of base class.

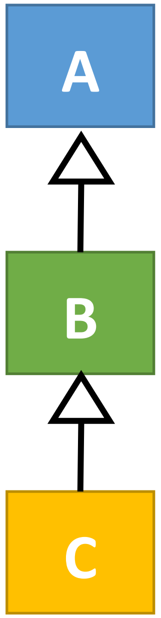
Name of base class, scope resolution operator :: , name of function

Inheritance (is-a) Function Overriding

```
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(){
//calls base class print for base
class data
    B::print();
    cout<<c;
    }

};
```

```
void main(){

    C c1;
    c1.print();
    //overridden function called, first
    calls B's print to print B's data
    then print c's data

    }
```





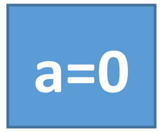
Inheritance (is-a) Function Overloading

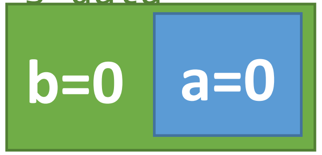
- Can overload base class inherited function in derived class to add some functionality
- **Overload function with**
  - **Same Name**
  - **Change parameters type, number or order**

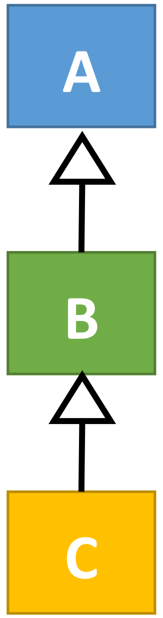
Inheritance (is-a) Function Overloading

```
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 inherited function from A
    void print(){ A::print(); cout<<b; }
//overload inherited function from A
    void print(int x){ cout<<x+b; }
};
```

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

    B b1;
    b1.print();
    //overridden function called, first calls A’s
    print to print a’s data then print b’s data

    b1.print(3);
    //overloaded function called
    a1.print(10);
    //overloaded function is not part of base class
    error
    }
```







Inheritance (is-a) Function Overloading

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

```
void main(){

    C c1;
    c1.print();
    //overridden function called, first calls B's
    print to print B's data
    then print c's data

    c1.print(9);
    //inherited function of B is called

    c1.print(9, 10);
    //overloaded function called
    //overloaded function is not part of B and A
    class
    }
```