Class/Object Relationships

Inheritance and Identifiers

CS(217) Object Oriented Programming

Abeeda Akram

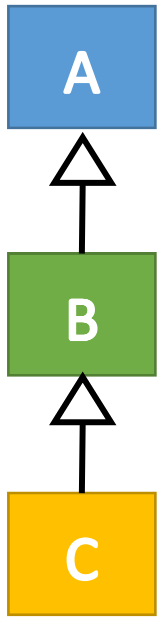
Inheritance (is-a) Identifier **override**

- A derived class can override, inherited virtual functions.
  - but the return type, name and parameters should same.
- If by mistake the programmer change return type, name or parameters the program may generate logical errors.
  - To avoid this issue the identifier **override** is added at end of virtual overridden function header.
  - Compiler will generate an error message, if function is not properly overridden in derived class.
- Programmer can visualize the overridden virtual functions directly by looking at derived class implementation.

Inheritance (is-a) Identifier **override**

```
class A{
   int a;
public:
   A(int a=0){ this->a=a;}
   virtual void print(){ cout<<a;}
   virtual ~A(){}
};
class B: public A{
   int b;
public:
   B(int a=0, int b=0):A(a)
   { this->b = b;}
   void print() override{
   A::print();
   cout<<b;
   }
   virtual ~B(){}
};
```

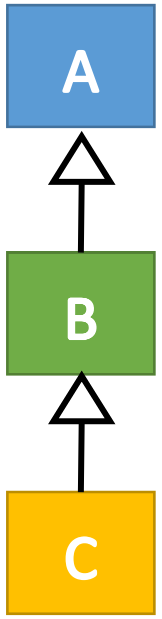
```
class C: public B{
    int c;
public:
    C(int a=0, int b=0, int c=0) :B(a,b)
    { this->c = c;}
// Compile Time Error: change return type
     int print() override{
       B::print();
       cout<<c;
    }
    virtual ~C(){}
};
```



Inheritance (is-a) Identifier **override**

```
class A{
   int a;
public:
   A(int a=0){ this->a=a;}
   virtual void print(){ cout<<a;}
   virtual ~A(){}
};
class B: public A{
   int b;
public:
   B(int a=0, int b=0):A(a)
   { this->b = b;}
   void print() override{
   A::print();
   cout<<b;
   }
   virtual ~B(){}
};
```

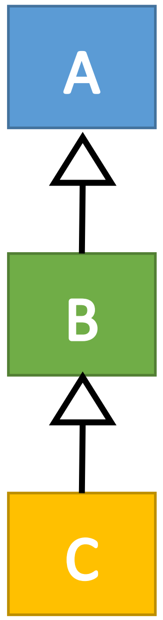
```
class C: public B{
    int c;
public:
    C(int a=0, int b=0, int c=0) :B(a,b)
    { this->c = c;}
// Compile Time Error: Not override
     void print(int x) override{
        B::print();
        cout<<c;
    }
    virtual ~C(){}
};
```



Inheritance (is-a) Identifier override

```
class A{
   int a;
public:
   A(int a=0){ this->a=a;}
   virtual void print(){ cout<<a;}
   virtual ~A(){}
};
class B: public A{
   int b;
public:
   B(int a=0, int b=0):A(a)
   { this->b = b;}
   void print() override{
   A::print();
   cout<<b;
   }
   virtual ~B(){}
};
```

```
class C: public B{
    int c;
public:
    C(int a=0, int b=0, int c=0) :B(a,b)
    { this->c = c;}
// Compile Time Error: Not override
     void print() const override{
        B::print();
        cout<<c;
    }
    virtual ~C(){}
};
```



Inheritance (is-a) Identifier **final**

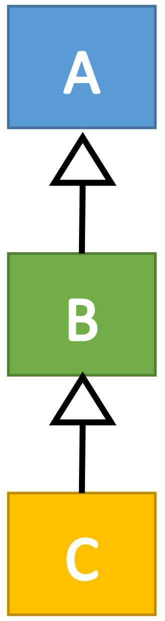
- We can stop a derive class to override an inherited function.
  - **Add final keyword at end of the function header.**
  - **Compiler will generate an error and will not allow to override a final function.**

- We can stop inheritance of a class.
  - Define the class as final
  - Compiler will generate an error and will not allow to derive a class from final class.

Inheritance (is-a) Identifier final function

```
class A{
  int a;
public:
  A(int a=0){ this->a=a;}
  virtual void print(){ cout<<a;}
  virtual ~A(){}
};
class B: public A{
  int b;
public:
  B(int a=0, int b=0):A(a)
  { this->b = b;}
  void print() override final{
  A::print();
  cout<<b;
  }
  virtual ~B(){}
};
```

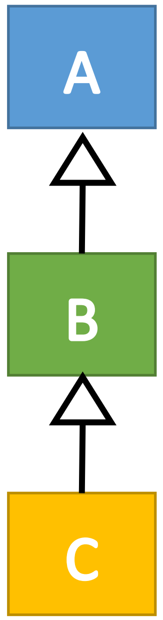
```
class C: public B{
   int c;
public:
   C(int a=0, int b=0, int c=0) :B(a,b)
   { this->c = c;}
// Compile Time Error: Cannot override
print function inherited from class B as
declared final in class B
    void print(){
      B::print();
      cout<<c;
   }
   virtual ~C(){}
};
```



Inheritance (is-a) Identifier final Class

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

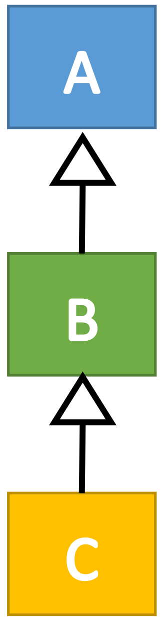
```
// Compile Time Error: Cannot
derive from final class A
class B: public A{
    int b;
public:
    B(int a=0, int b=0):A(a)
    { this->b = b;}
    void print() override {
        A::print();
        cout<<b;
    }
    virtual ~B(){}
};
```



Inheritance (is-a) Identifier **final** Class

```
class A{
   int a;
public:
   A(int a=0){ this->a=a;}
   virtual void print(){ cout<<a;}
   virtual ~A(){}
};
class B final : public A{
   int b;
public:
   B(int a=0, int b=0):A(a)
   { this->b = b;}
   void print() override{
   A::print();
   cout<<b;
   }
   virtual ~B(){}
};
```

```
// Compile Time Error: Cannot derive from
final class 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 print(){
        B::print();
        cout<<c;
    }
    virtual ~C(){}
};
```



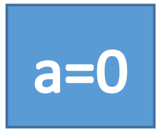
Inheritance (is-a) Identifier **using**

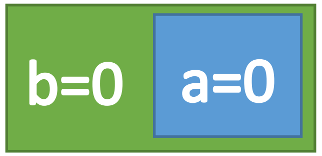
- We can change the specific inherited members access specifiers (private, protected or public) in derived class.
  - This is done by adding a using declaration under the new access specifier.
- Usage:
  1. Can make inherited members **public** in derived class to provide access through derived class object.
  2. Can make inherited members **private** or **protected** to restrict the user access on inherited members from derived class objects.

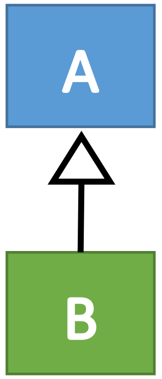
Inheritance (is-a) Identifier using

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

class B: public A{
   int b;
public:
   using A::print;
// note: no parenthesis here
   B(int a=0, int b=0):A(a)
   { this->b = b;}
   virtual ~B(){}
};
```

```
void main(){
    A a1;
    a1.print();
    //Cannot call A's print


    B b1;
    b1.print();
    //can call A's print through
    b's object made public in
    Derived class
    }
```



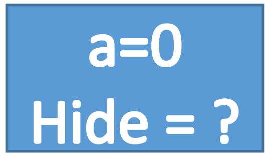


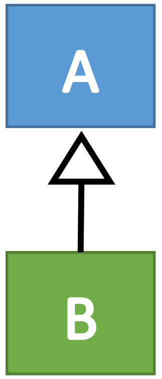


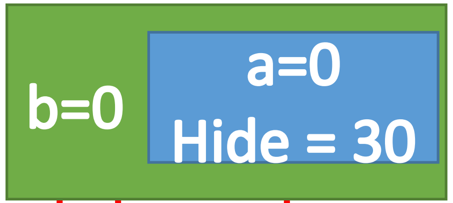
Inheritance (is-a) Identifier using

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

class B: public A{
   int b;
protected:
   using A::print;
public:
   using A::hide;
   B(int a=0, int b=0):A(a)
   { this->b = b;}
   virtual ~B(){}
};
```

```
void main(){
    A a1;
    a1.print();
    b1.hide = 50;
    //Cannot access hide from A's Object

    B b1;
    b1.print();
    //Cannot call A's inherited  print
    hidden in class B

    b1.hide = 30;
    //can access hide through b's object
    made public in Derived class
    }
```






b=0
a=0
Hide = 30