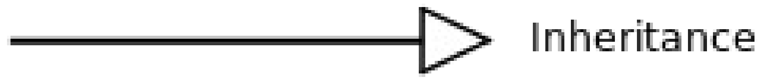
Class/Object Relationships

Inheritance

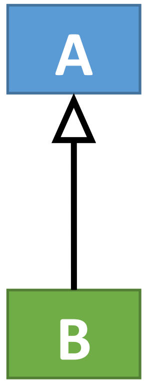
CS(217) Object Oriented Programming

Abeeda Akram

Inheritance (is-a)



- Used for reusability
- Create new class from existing class by absorbing existing class's
  1. Attributes, (Data members)
  2. Behaviors, (Functions)
- **Parent/Base/Super class**
  - The general class from which data and functions are inherited
- **Child/ Derived/Sub class**
  - The specialized class which inherits data and functions
  - It can customize inherited functions according to its need
  - It can also add more functions and data members
- **Unidirectional**
  - Every derived class is a base class type, but every base is not derived type
- **Derived class object is treated as base class object**



B is derived class inheriting data and functions from class A the base class

Inheritance (is-a) Types

1. **Direct**

   Base class is one level up

2. **Indirect (Multilevel)**

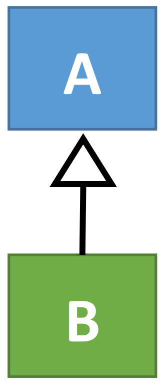
   Base class is two or more levels up

3. **Single**

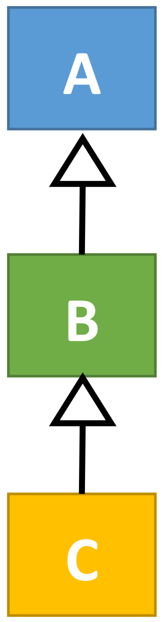
   Inherited directly from a single base class

4. **Multiple**

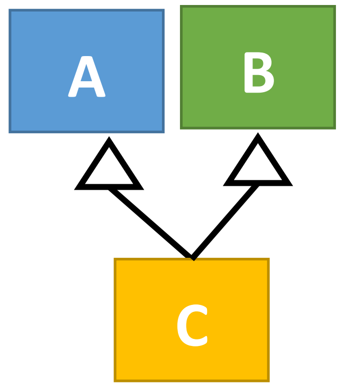
   - Inherited from multiple base classes.
   - Base classes possibly unrelated



Single, Direct



Multilevel, Indirect

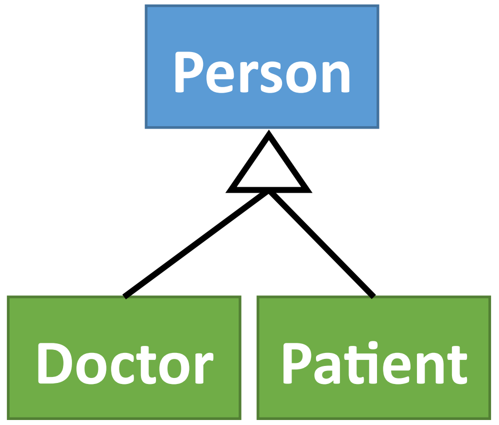


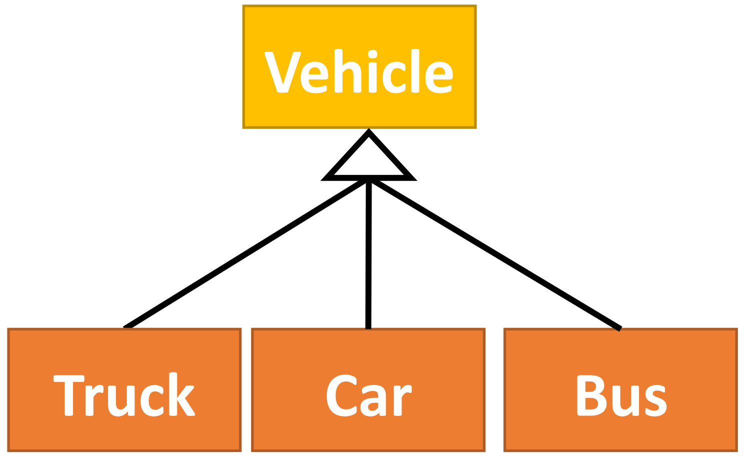
Multiple

Inheritance (is-a) Examples

- Every Doctor is a person
- Every Patient is a Person
- Every person is not a doctor or a patient

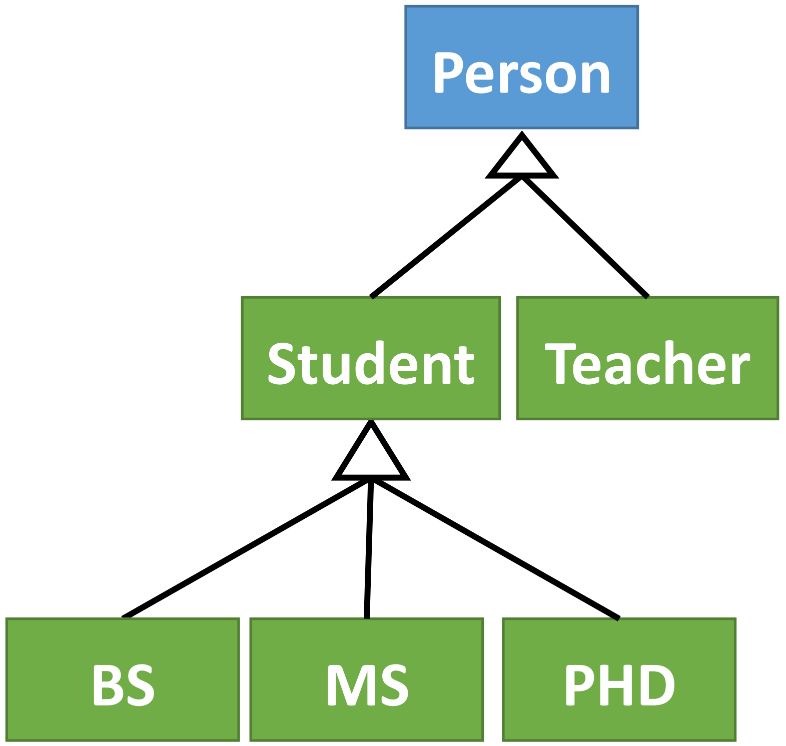
- Every Car, Truck and Bus is a vehicle but
- Every Vehicle is not a Car, Truck or a Bus.





Inheritance (is-a) Examples

- Every Student is a Person
- Every Teacher is a Person
- BS, MS, and PHD Student is a student and is a Person too.
- Every Student is not BS, MS or PHD
- Every Person is not a student or a teacher.



Inheritance (is-a) Members

For every class

- **Public members:**
  - accessible directly (by using member access operator on objects) in all classes and in all functions members or non-member
- **Private members:**
  - not directly accessible in any class even in derived class
  - only accessible in member functions of class
  - not directly accessible in non member functions except friend functions
  - call getter, setters for access

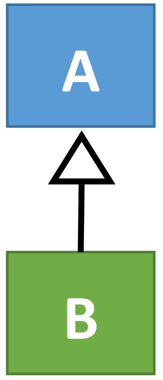
Inheritance (is-a) Protected Members

- **Protected member:**
  - not directly accessible in any class except derived classes
  - only accessible in member functions of class
  - not directly accessible in non member functions except friend functions
  - call getter, setters for access in all other classes and functions

Advantages	Disadvantages
Performance improved as getter, setters call avoided in derived class.	No validity check of base class is involved
Direct Modification of base class inherited members in derived class.	Implementation dependency increase, when base class change its member, derived class also need to change implementation accordingly

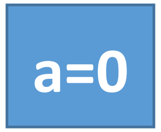
Inheritance (is-a) Type Public

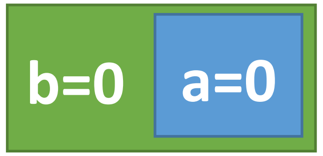
- **Public** members of base class inherited as **public**.
- **Private** members of base class are inherited as private, and need to call getter setters to access them from derived class.
- **Protected** members are inherited as protected and derived class can directly access them.



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

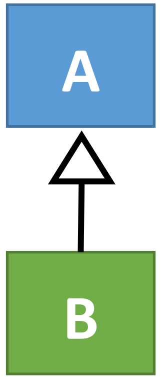
```
void main(){
   A a1;
   a1.print();
   //print is public in a

   B b1;
   b1.print();

   //print is inherited as public and
   can be accessed from object of class
   B
}
```





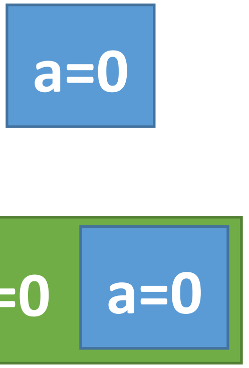
Inheritance (is-a) Type Protected

- **Public** members of base class inherited as **protected**.
- **Private** members of base class are inherited as private, and need to call getter setters to access them from derived class.
- **Protected** members are inherited as **protected** and derived class can directly access them.



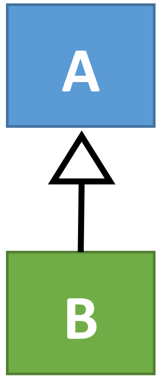
```
class A{
    int a;
public:
    A(int a=0){ this->a=a;}
    void print(){ cout<<a;}
};
class B: protected A{};
//protected inheritance
```

```
void main(){
   A a1;
   a1.print();
   //print is public in a

   B b1;
   b1.print();

   //print is inherited as protected and
   cannot be accessed from object of class B
   outside
}
```



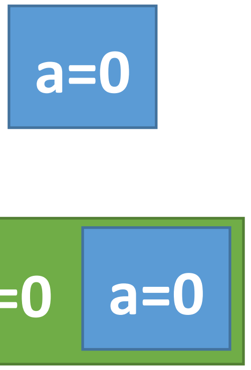
Inheritance (is-a) Type Private

- **Public** members of base class inherited as **private**.
- Private members of base class are inherited as **private**, and need to call getter setters to access them from derived class.
- **Protected** members are inherited as **private** and derived class can directly access them.



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

```
void main(){
   A a1;
   a1.print();
   //print is public in a

   B b1;
   b1.print();

   //print is inherited as private and cannot
   be accessed from object of class B outside
}
```



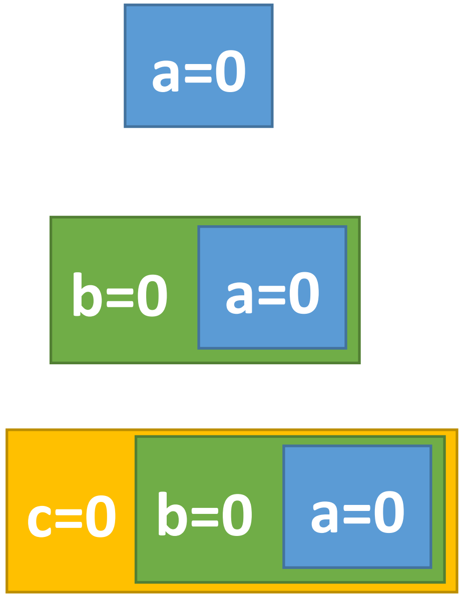
Inheritance (is-a) Types and Members

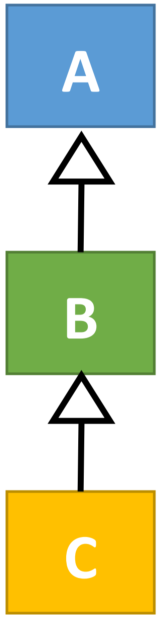
Base class Members	Inheritance Types and member access changes in derived class		
	Public	**Protected**	**Private**
Private	Private	Private	Private
Protected	Protected	Protected	Private
Public	Public	Protected	Private

Inheritance (is-a) Type Examples

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

```
void main(){
   A a1;
   a1.print();
   //print is public in a

   B b1;
   b1.print();
   //print is public in b

   C c1;
   c1.print();
   //print is inherited as protected and
   cannot be accessed from object of class C
   outside
}
```

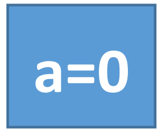


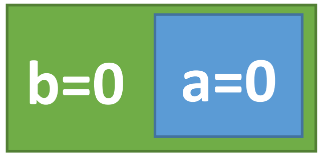


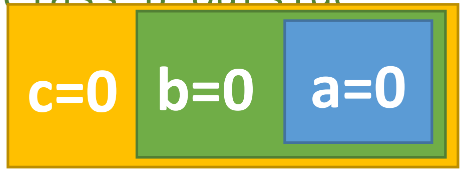
Inheritance (is-a) Type Examples

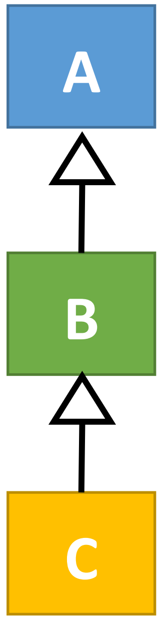
```
class A{
   int a;
public:
   A(int a=0){ this->a=a;}
   void print(){ cout<<a;}
};
//private inheritance
class B: A{
   int b;
public:
   B(int b=0){ this->b = b;}
};
//protected inheritance
class C: protected B{
   int c;
public:
   C(int c=0){ this->c = c;}
};
```

```
void main(){
   A a1;
   a1.print();
   //print is public in a

   B b1;
   b1.print();
   //print is now private in b cannot be
   accessed from object of class B outside

   C c1;
   c1.print();
   //print is inherited as private and
   cannot be accessed from object of class
   C outside
}
```









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