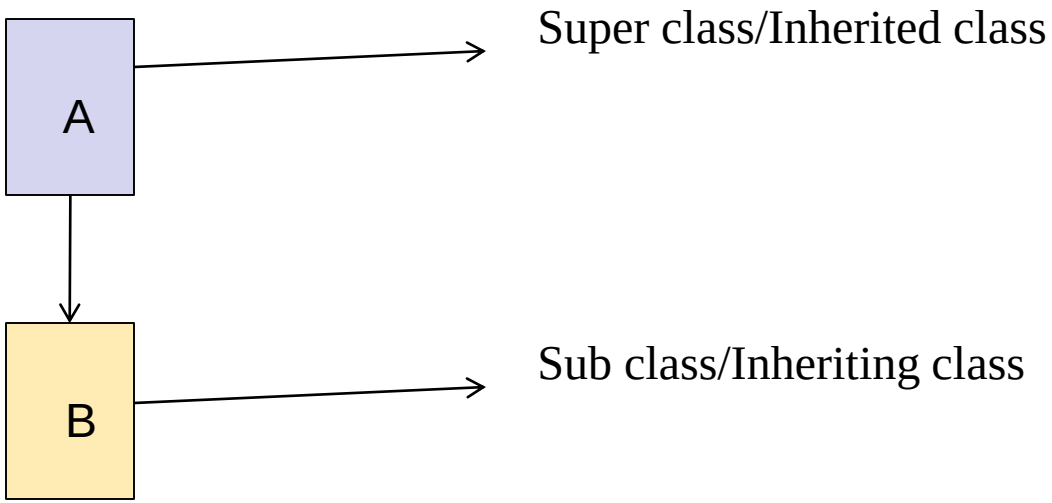
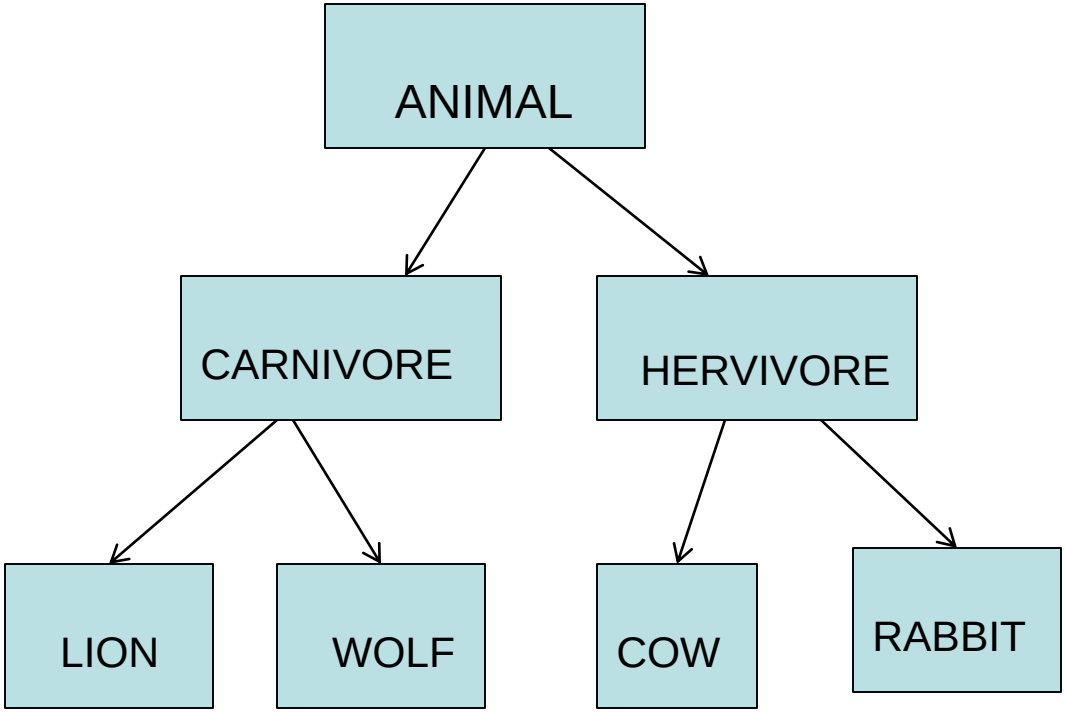
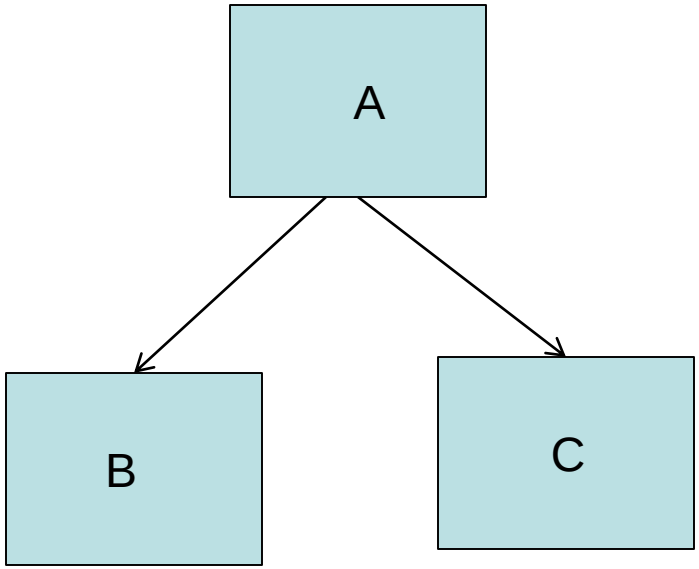
INHERITANCE

An object can acquire the properties of another class




ANIMAL
CARNIVORE
HERVIVORE
LION
WOLF
COW
RABBIT



i+j+b

i+j+c

IMPLEMENT INHERITANCE USING 'extends'

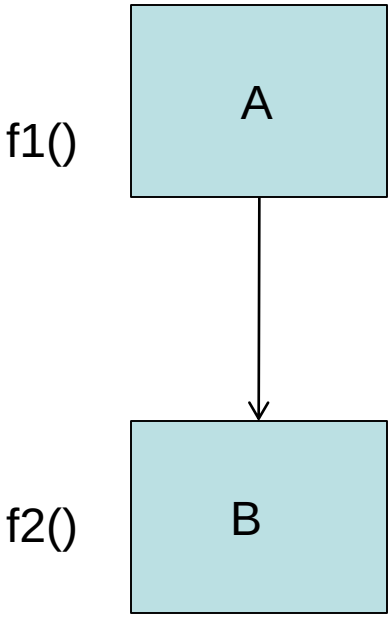
```
class A
{
int i;
int j;
 void setdata( int x, int y)
 {
   i=x;
   j=y;
 }
void print_A()
{
System.out.println(i+j);
}
}
```

```
class B extends A
{
int b=5;
  void print_B()
   {
     System.out.println(i+j+b);
   }
}
```

```
class C extends A
{
int c=6;
  void print_C()
   {
     System.out.println(i+j+c);
   }
}
```

```
class XYZ
{
    public static void main(String  args[])
            {
             B kb=new B();
             C kc=new C();
             kb.print_A();
             kb.print_B();
             kc.print_A();
             kc.print_C();
   }
}
```

SUPERCLASS VARIABLE CAN REFER SUBCLASS OBJECT

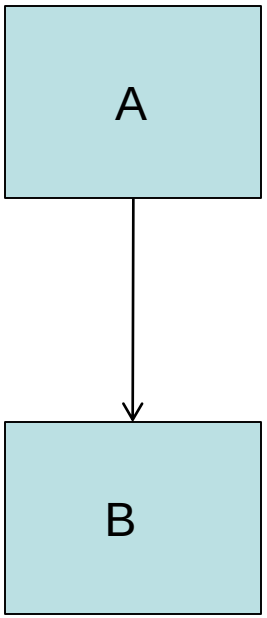


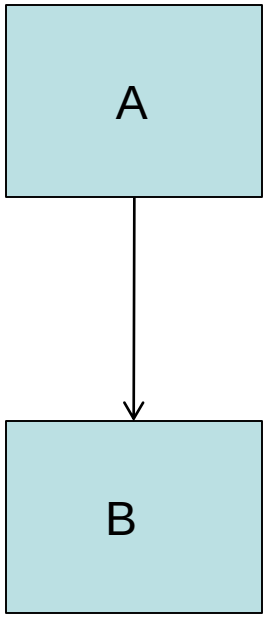

```
class XYZ
{
   public static void main(String  args[])
            {
             A ka=new A();
             B kb=new B();
             ka=kb;
             ka.f2();
             }
}
```

Keyword ‘super’

To initialize Superclass variable

super(parameter list);





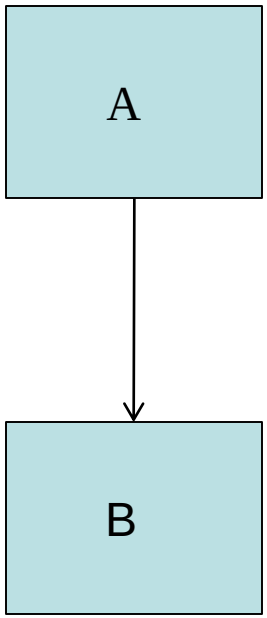

```
class A
{
  int i;
  int j;
          A(int x,int y)
                {
               i=x;
               j=y;
                }
}
```

```
class B extends A
{
  int b;
                B(int x, int y, int z)
                {
                 super(100,200);
                b=z;
                }
        void  display()
            {
              System.out.println(i+j+b);
             }
}
```

```
class XYZ
{
   public static void main(String  args[])
            {
             B k=new B(100,200,300);
             k.display();
}
```

Access the members of superclass

This approach is applicable when subclass and superclass have the member with same name.



i

i

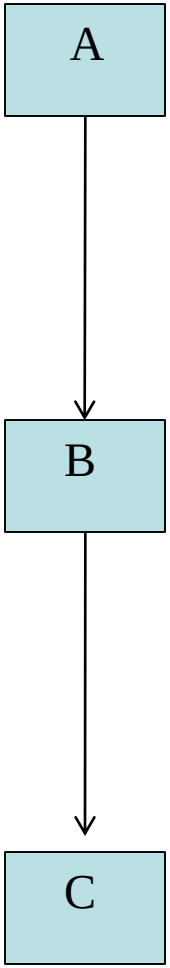
super.i = super.i +1

```
class A
{
   protected int i;
   protected int j;
              A(int x,int y)
              {
                i=x;
                j=y;
              }
}
```

```
class B extends A
{
   int b;
   int i=1000;
                B(int x, int y, int z)
                {
                   super(100,200);
                   b=z;
                }
          void  display()
              {
                System.out.println(i+j+b);
                System.out.println(super.i+j+b);
               }
}
```

```
class XYZ
{
    public static void main(String  args[])
            {
             B k=new B(100,200,300);
             k.display();
}
```

CREATING MULTILEVEL HIERARCHY



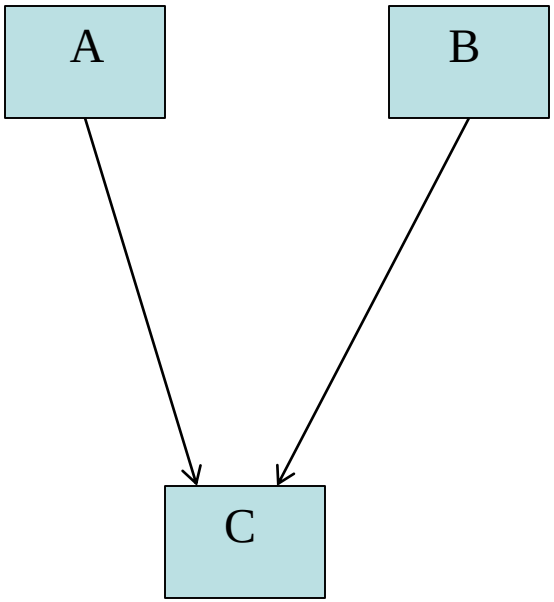

```
class A
{
   protected int i=100;

}
```

```
class B extends A
{
   protected int j=200;

}
```

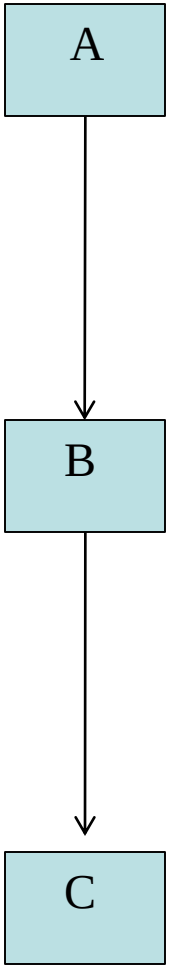
```
class C extends B
{
   int m=300;
  void  display()
              {

               System.out.println(i+j+m);
              }
}
```

```
class XYZ
{
   public static void main(String  args[])
             {
              C k=new C();
              k.display();
             }
}
```

JAVA DOES NOT SUPPORT MULTIPLE INHERITANCE



CONSTRUCTORS ARE CALLED IN ORDER OF DERIVATION FROM SUPERCLASS TO SUBCLASS



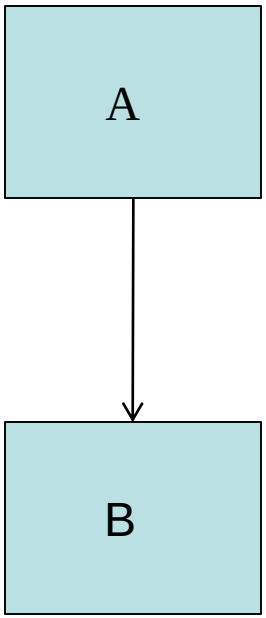

```
class A
{
      A()
        {
         System.out.println("Inside A");
        }
}
```

```
class B extends A
{
       B()
         {
          System.out.println("Inside B");
         }


}
```

```
class C extends B
{
      C()
        {
          System.out.println("Inside C");
        }
}
```

```
class XYZ
{
   public static void main(String  args[])
            {
             C k=new C();
            }
}
```

METHOD OVERRIDING

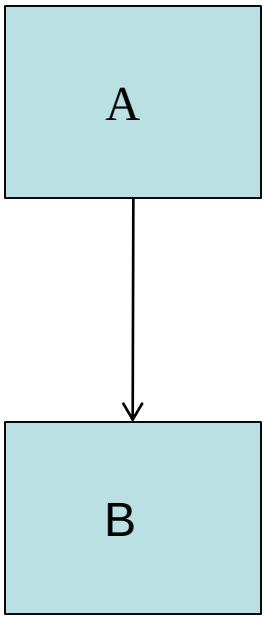



```
class A
{
      void fA()
        {
         System.out.println("Inside A");
        }
}
```

```
class B extends A
{
      void fA()
        {
          System.out.println("Inside B");
        }


}
```

```
class XYZ
{
   public static void main(String  args[])
            {
             B k=new B();
             k.fA();
            }
}
```

IS IT METHOD OVERRIDING ?



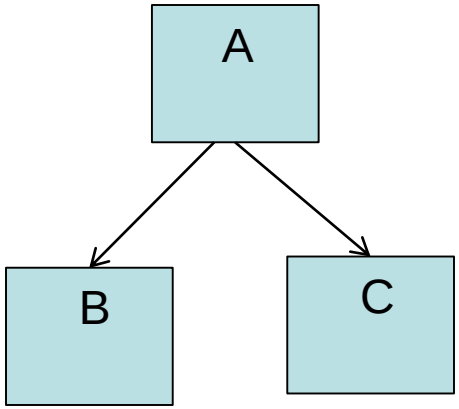

```
class A
{
      void fA()
         {
          System.out.println("Inside A");
         }
}
```

```
class B extends A
{
        void fA(int x)
          {
            System.out.println(x);
          }


}
```

```
class XYZ
{
   public static void main(String  args[])
            {
             B k=new B();
             k.fA(5);
            }
}
```

DYNAMIC METHOD DISPATCH

```
class A
{
      void f()
         {
          System.out.println("Inside A");
         }
}
```



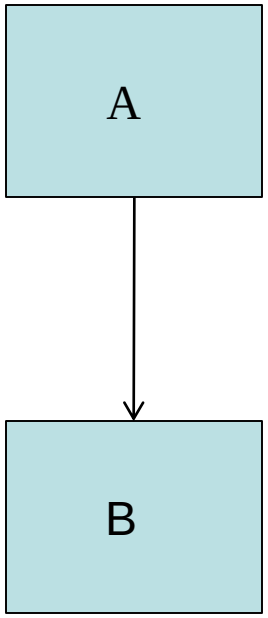

```
class XYZ
{
   public static void main(String  args[])
            {
             A ka=new A();
             B kb=new B();
             C kc=new C();
             A r;
             r=ka;
             r.f();
             r=kb;
             r.f();
             r=kc;
             r.f();
            }
}
```

```
class B
{
      void f()
         {
          System.out.println("Inside B");
         }
}
```

```
class C
{
      void f()
         {
          System.out.println("Inside C");
         }
}
```

ABSTRACT CLASS

A superclass that does not provide the complete implementation of every method.




```
abstract class A
{
     abstract void f1();
      void f2();
          {
           System.out.println("Inside A");
          }

}
```

```
class B extends A
{
     void f1()
       {
       System.out.println("Developed Inside B");
       }
}
```

```
class XYZ
{
    public static void main(String  args[])
            {
             B k=new B();
             k.f1();
             k.f2();
            }
}
```