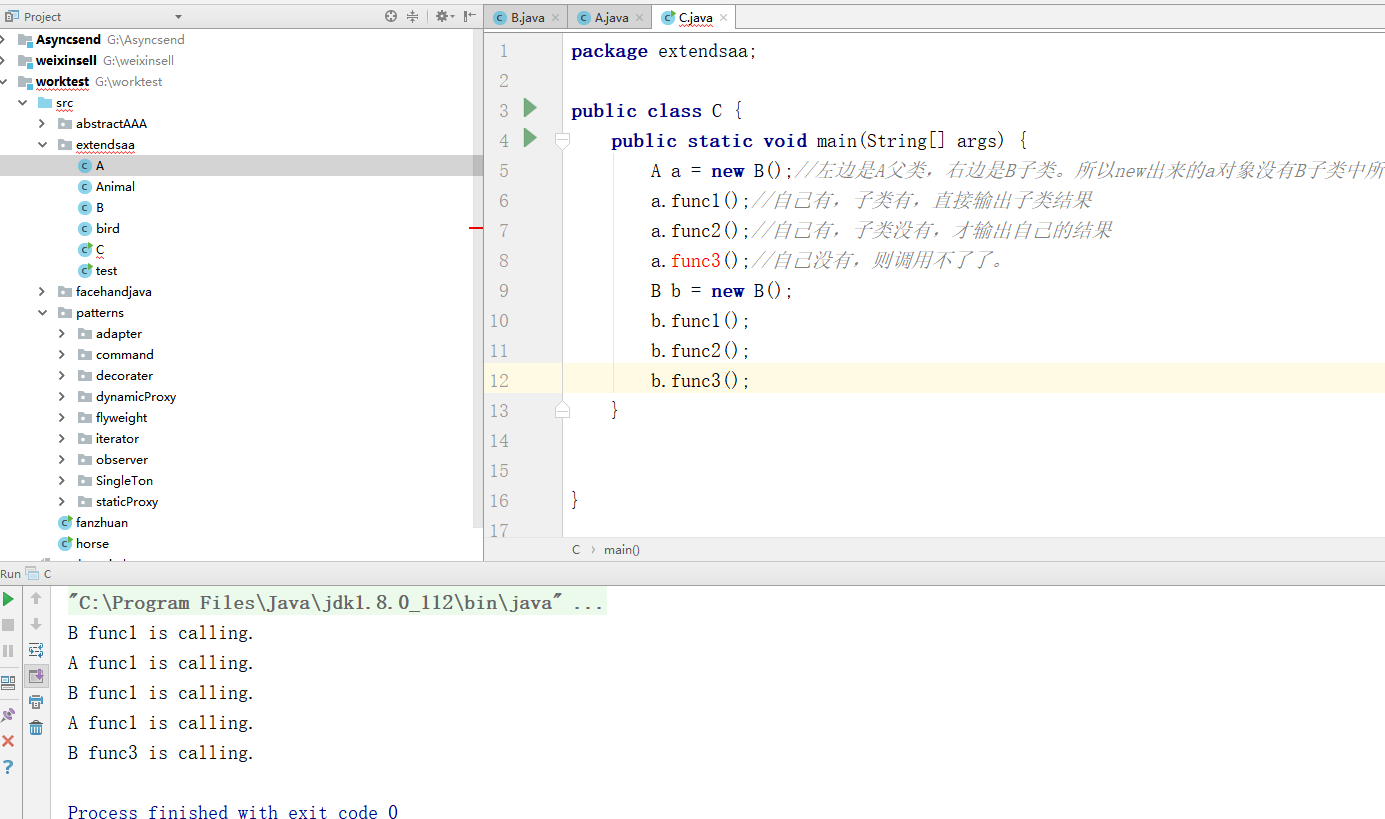


先上图，再解释哈。

1、子类转换成父类  
class A {  
     public void func1() {  
          System.out.println("A func1 is calling.");  
     }  
     public void func2() {  
          func1();  
     }  
}  
class B extend A {  
     public void func1() {  
          System.out.println("B func1 is calling.");           
     }  
     public void func3() {  
          System.out.println("B func3 is calling.");  
     }  
}  
class C {  
     public static void main(String[] args) {  
          B b = new B();  
**A a = b;**  
          callA(a);  
          callA(new B());  
     }  
     public void callA(A a) {  
          a.func1();  
          a.func2();  
     }  
}  
**编译器能够自动将类B的实例对象b直接赋值给A类的引用变量，也就是子类能够自动转换成父类类型**。如果方法两者都有，**跑的是B子类的结果（是否可以调用某方法看左边的类型，调用到方法的结果看右边的类型）。**



另外，程序可以直接创建一个类B的实例对象，传递给需要类A的实例对象作参数的callA()方法，在参数传递的过程中发生了隐式自动类型转换。子类能够自动转换成父类的道理非常容易理解。  
2、父类转换成子类  
如果知道callA方法中传递的参数a实际上就是子类B的一个引用对象，想在callA方法中调用子类的特有方法，如何做：  
     public void callA(A a) {  
          a.func1();  
          a.func2();  
          a.func3();//没有该方法  
     }  
编译有问题，因为对编译器来说，它只分析程序语法，它只知道变量a的引用类型是类A，**而类A又没有func3这个方法，所以编译不通过。**

代码改为：  
     public void callA(A a) {  
          B b = a;//这样也不可以   
          a.func1();  
          a.func2();  
          a.func3();  
     }  
编译还是有问题，**因为编译器是不能将父类对象自动转换成子类的。**     public void callA(A a) {  
          B b = (B)a;    **//强制类型转换（但是需要是子父类才可以这样转,并且a实际上就是子类B的一个引用对象）**          a.func1();  
          a.func2();  
          a.func3();  
     }  
   
－instanceof 操作符－  
可以用**instanceof判断是否一个类实现了某个接口**，也可以用它来判断一个实例对象是否属于一个类  
public void callA(A a) {  
     if(a instanceof B){  
          B b = (B)a;  
          a.func1();  
          a.func2();  
          a.func3();  
     } else {  
          a.func1();  
          a.func2();  
     }  
}  
只要记住：一个男人肯定也是人，一个人却不一定是男人的道理，就非常容易理解父类和子类之间的转换关系了。

－Object类－

Object类是所有类的父类，如果一个类没有使用extends关键字明确标识继承另外一个类，那么这个类就默认继承Object类。因此，Object类是Java类层中的最高层类，是所有类的超类。Java中任何一个类都是它的子类，由于所有的类都是由Object衍生出来的，所以Object的方法适用于所有类。

－面向对象的对态性－

当叫来一个中国人时，我们看到的是用筷子在吃饭，当叫来一个美国人时，我们看到的就是另外一番景象了，用的是叉子和刀子。同一段程序代码，却有两种截然不同的结果，这就是面向对象的多态性。特点：

* 应用程序不必为每一个子类编写功能调用，只需要对抽象基类进行处理即可。
* 子类的功能可以被父类的方法或引用变量调用，这叫向后兼容，可以提高程序的可扩充性和可维护性。