

向上转型

即子类转型成父类

好处：向上转型体现了类的多态性，增强了程序的简洁性。（方法重载）

由于向上转型，对象会遗失和父类不同的方法，重载父类存在的方法

向下转型

public class A {

void aMthod() {

       System.out.println("A method");

}}

public class B extends A {

void bMethod1() {

       System.out.println("B method 1");

}

void bMethod2() {

       System.out.println("B method 2");

}

}

public class C {

     public static void main(String[] args) {

            A a1 = new B(); // 向上转型

            a1.aMthod();    // 调用父类aMthod()，a1遗失B类方法bMethod1()、bMethod2()

            B b1 = (B) a1; // 向下转型，编译无错误，运行时无错误

            b1.aMthod();    // 调用父类A方法

            b1.bMethod1(); // 调用B类方法

            b1.bMethod2(); // 调用B类方法

            A a2 = new A();

            B b2 = (B) a2; // 向下转型，编译无错误，运行时将出错

            b2.aMthod();

            b2.bMethod1();

            b2.bMethod2();

     }

}

向下转型需要使用强制转换。运行C程序，控制台将输出：

Exception in thread "main" java.lang.ClassCastException: a.b.A cannot be cast to a.b.B at  
                a.b.C.main(C.java:14)

因为a1指向一个子类B的对象，所以子类B的实例对象b1当然也可以指向a1。而a2是一个父类对象，子类对象b2不能指向父类对象a2。

要理解a1、b1都是引用

关于引用

|  |  |
| --- | --- |
| 1  2  3 | Person person;  person = new Person("张三");  person = new Person("李四"); |

　　这里让person先指向了“张三”这个对象，然后又指向了“李四”这个对象。也就是说，Person person，这句话只是声明了一个Person类的引用

关于强转

java程序运行后,对象的类型就不会发生改变。而强制类型转换只不过是用某种类型去引用内存中的实体,这种类型只是编译时类型,与运行后的实体并没有联系