

向下转型和 instanceof 运算符的应用


知识梳理

一：向下转型

1. 向下转型

向下转型是与向上转型相对的概念，它是用子类引用指向父类实例。

如：下图，在进行转换是会报错



```
Animal a=new Dog();  
Dog d=a;
```

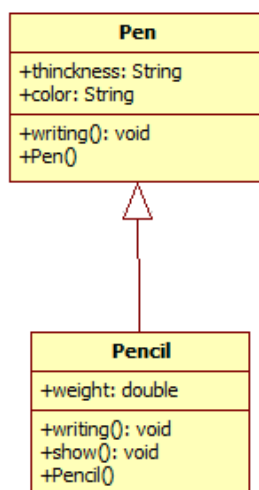
这时就告诉我们向下转型不能自动转换，我们需要强转，所以向下转型又叫做强制类型转换。

正确的转换语句为：

```
Animal a=new Dog();  
Dog d=(Dog)a;
```

2. 向下转型后，可以调用子类自己独有的方法。

例如：(承接上一文档例子)



测试类中通过强制类型转换后，可调用 Pencil 类中独有的方法 show()了。

```
Pen pc=new Pencil();//向上转型
Pencil p=(Pencil)pc;//强制类型转换
p.show();
```

3. 兄弟类之间不能进行强制类型转换。

如：父类 Pen 类派生出另一个子类 Brush。

```
Pen b=new Brush();
Pencil p=(Pencil)b;
```

将指向 Brush 对象的父类引用 b 强制转换为 Brush 的兄弟类 Pencil 的对象，此时编译器并没有报错，但在运行时会报出如下错误：

```
Exception in thread "main" java.lang.ClassCastException:
com.imooc.paper.Brush cannot be cast to com.imooc.paper.Pencil
```

二：instanceof 运算符

1. 基本概念

instanceof 运算符用来判断对象是否可满足某个特定类型实例特征。返回值为 true/false。一般用于 if 语句中。

表示方法为：



如：

```
boolean result;
Pen pc=new Pencil();
result=pc instanceof Pencil;
```

如果左边对象是右边类的实例则返回 true,否则返回 false。

2.instanceof 运算符的应用

1) 用 instanceof 运算符用来判断对象是否可满足某个特定类型实例特征

例子：

父类 Parents 类，Father 类和 Mother 类分别为它的两个子类：

```
//对象实例化
Parents f=new Father();
Parents m=new Mother();
//用instanceof运算符判断对象是否满足某个特定对象实例特征
System.out.println(m instanceof Father);
System.out.println(m instanceof Mother);
System.out.println(m instanceof Object);
System.out.println(f instanceof Father);
```

运行结果为：

```
false
true
true
true
```

注：java 中所有类都直接或间接继承于 Object 类。