Java常遇问题:简单明了判断对象类型的技巧Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文https://www.100test.com/kao_ti2020/644/2021_2022_Java_E5_B8_B8_E9_81_87_c104_644436.htm 一、向上转型与向下转型。 对象类型的转换在Java语言平台中经常遇到，主要包括向上转型与向下转型操作。程序开发人员需要熟练掌握这两个转型的方法以及其中容易出错的地方。如何来了解这两个转型的区别呢?笔者认为，以一个现实的例子作为比喻，可能会更加的容易理解。 如现在有动物、鸟类、燕子三个名词，他们之间有什么关系呢?通常我们都会首，燕子是特殊的鸟类，或者说燕子是鸟类的一种。为此，从对象的定义来看，鸟类就是一个父类，而燕子就是一个子类。或者说，燕子对象就是一个鸟类对象。笔者这里要强调的一点就是，由于燕子是鸟类的一个对象，所以鸟类所具有的特性燕子全部具有。而燕子所具有的特性(如迁徙)则鸟类不一定都具有。在这个例子中，燕子也是一种鸟类。为此可以将燕子的对象堪称是一个鸟类的对象。这种方法在Java语言环境中就叫做“ 向上转型 ”。从这个例子中可以看出，向上转型是一个从较抽象类型的类(鸟类)向比较具体的类(燕子)过度。由于具体类(燕子)具有抽象类 (鸟类)的全部特性，所以在这个转换过程中是不会有问题的。这就好像一个逻辑判断题说燕子是鸟类的一种，其具有鸟类的全部特性。这个命题至少到现在为止是完全正确的。但是，在实际工作中，我们还经常会遇到向下转型的情况。也就是说从一个抽象类中(鸟类)引用具体类(燕子)中的对象。也就是说，我们可以说燕子是鸟类的一种。但是现在反过来，如果说鸟类就是燕子，那显然就是以偏概全了，因为燕子

并不具有其他鸟类的特性。如鸽子的特性燕子就没有。所以，在应用程序开发中，如果将父类对象赋值给子类的对象，就可能有问题。如果硬要这么做的话，则很有可能发生编译器错误。因为父类对象并不一定是子类的实例。这是什么意思呢?即所说的鸟类(父类对象)并不一定是子类对象(燕子)。因为鸟类对象还有可能是鸽子、白鹭等等。所以，如果将父类对象给子类对象的话，那么就会出现问题。 二、如何实现向下转型? 由于向上转型一般都是安全的，即将一个子类对象直接赋值给父类对象，一般被认为是安全的，如燕子是鸟类在哪里都是成立的。所以在向下转型时不需要采用其他的关键字，我们常常把向下转换叫做隐式转换。但是在这里向上转换是一种不安全的转换方式，如说鸟类就是燕子，这种说法无论在哪里都说不过去。为此默认情况下，进行向下转型时，往往会发生编译器错误。 一般情况下，越是具体的对象所具有的特性越多。如燕子的特性就比鸟类的特性多的多。而越抽象的对象反而具有的特性越少，因为其只具有一些抽象对象的共性特征。在进行向下转型操作时，将特性范围小的对象转换为特性范围大的对象肯定会出现问题。为此在向下转型时，必须确保转换后不会出现问题，即具体对象的特性在抽象对象中也全部具备，只有如此才能够进行转换。而且即使满足这个条件，编译器也不不能够进行隐式转换。而是需要采用关键字进行强制转换。如子类对象名字=(子类名)父类对象名字。如果上面这个语法，就可以实现对象类型的强制转换。 百考试题在此强调一遍，在进行向下转型时一定要进行强制转换。即通过子类对象名字=(子类名)父类对象名字进行赋值，而不能够向向上转型那样进行隐式转换。 三、

确保向下转型的准确性。 从以上分析中可以看出，向下转型往往被认为是不安全的。当在程序中执行向下转型操作的时候，如果父类对象不是子类对象的实例，就会发生编译器错误。所以在执行向下转型之前要先作一件事情，就是判断父类对象是否为子类对象的实例。也就是说，先要想一想，燕子就是鸟类这个命题是否成立(在某些特定的情况下这个伪命题可能会成立，如燕子的特性与鸟类的特性完全一致)。只有如此，向下转型才不会出现问题。在进行向下转型操作时，将特性范围小的对象转换为特性范围大的对象肯定会出现问题。但是，如果两个转换的对象特性范围一样大的话，可那么就不会有问题了。 在应用程序开发中，往往通过操作符instanceof来完成这个判断。即可以利用这个操作符来判断是否一个雷实现了某个接口，也可以用来判断一个实例对象是否属于一个类。这个操作符的基本格式为：A(某个类的对象引用) instanceo(操作符号) B (某个类的名称)。这个操作符最后返回的是一个布尔值。如果是false的话，则说明A对象不是类B的实例对象。相反，如果返回的值是true的话，则说明对象A是类B的实例对象。 四、向下转型的注意事项。 在进行向下转型时，需要注意以下几方面的内容： 一是要慎用向下转型。由于向下转型容易出问题，为此不到万不得已的时候，最好不用使用向下转型。条条道路通罗马，如果在编程之前，合理规划类，往往可以避免向下转型的发生。只有其他路走不通的情况下，才考虑通过向下转型的技术来解决问题。 二是在进行向下转型的时候，需要做两件事情。一是一定要使用instanceof操作符来判断转型的合法性，即判断父类对象是否为子类对象的实例。这就好像在编写四则运算时，

要判断除数不为零一样。这是必须要做的。也是程序员必须要养成的一个习惯。在进行向下转型时，就自然而然会想到需要进行这个判断。只有如此，应用程序的错误才能够降低。而且还能够满足不同的需求。二是需要注意向上转型与向下转型的区别。一般情况下，向上转型往往被认为是安全的，所以在Java语言平台中向上转型采用的是隐式转型。而向下转型由于特性范围大小的不同，为此往往被认为是不安全的。故系统默认情况下进行向下转型时必须采用强制转型的方式。如果不采用强制转型，则即使满足向下转型的条件，其也会发生编译器错误。所以需要切记，向下转型必须要采用强制转型。 三是需要做好备注等注释工作。由于像向下转型等操作是容易出现问题的地方。为此在进行类似的操作时，最好在行注释或者块注释中能够进行说明。这对于后续的维护与代码的升级是很有帮助的。好记性不如烂笔头。如果没有做好相关注释的话，这次可能没有问题，但是下次再代码升级或者其他原因需要调整或者重写原有的代码时，就可能会因为疏忽而导致转型的失败。 最后笔者再次提醒各位程序员，向上转型大家可以放心大胆的用。但是在使用向下转型技术时，大家要慎重，要按部就班(先判断后使用)的进行操作。 