泛型类型的子类及通配符的使用 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/253/2021_2022__E6_B3_9B_E5_9E_8B_E7_B1_BB_E5_c104_253552.htm 本文讲述以下几个方面的内容，试图说明泛型类型的子类及通配符的使用。（1） 子类及替换原则；（2） 使用extends关键字的通配符；（3） 使用super关键字的通配符；（1） 子类及替换原则 在java语言中，我们通俗讲一个类是另一个类的子类型，是通过使用extends关键字去继承某一个类或者使用implements关键字去实现某些接口。这样我们在编程时就可以面向接口或基类进行编程，如： Number num1 = new Integer（1）；Number num2 = new Double（2.1d）；这个就是所谓的替换原则，替换原则的定义是： Substitution Principle： a variable of a given type may be assigned a value of any subtype of that type， and a method with a parameter of a given type may be invoked with an argument of any subtype of that type. 大概的意思是说某种类型的变量可以被该类型的任何子类所赋值，一个方法中的参数也可以被该参数的任何子类进行调用。 现在我们再来看看泛型中替换原则的适用性： Example2.1 List listNums = new ArrayList（）；nums.add（10）；nums.add（8.88）；在example2.1中，可以看出替换原则被很好地应用在这里，ArrayList是List的子类，我们提供给listNums变量的类型参数为Number，往listNums中添加元素时，10被封箱为Integer类型，而Integer是Number的子类，第三行的情况类似。 Example2.2 List intList = new ArrayList（）；List numList = intList；//compile error … numList.add（1.35）；// can‘ t do that 根据替换原则，我们会

很容易想到，既然Integer是Number的子类，则我们应该可以将List的变量赋给 List的变量，但从实际情况表明，List并不是List的子类。 我们不妨试想想，若果List类型的变量可以成功赋值给List类型的变量，会出现什么情况？我们可能在程序的某个位置添加一个double类型的元素进去numList中，而实质上在numList中其它元素都是Integer 的类型的元素，这样就违背了泛型的初衷了。 有时，我们确实希望将形如List的List对象赋给List的变量，这时就要使用extends关键字的通配符。 （2） 使用extends关键字的通配符 Example2.3 List intList = new ArrayList（）；List numList = intList（）；… numList.add（1.35）；//compile error （can‘ t do that） 从Example2.3看到numList这个变量，我们可以将类型参数为Number及其Number子类的List赋给它。 记住一条规则如果你使用了“？ extends T ”，一般情况下，你不能往该数据结构中put元素，而你可以做的就是get元素。 如果要往内put元素，就需要使用下面提到的super关键字的通配符。 （3） 使用super关键字的通配符 Example2.4 List intList = new ArrayList List numList = intList；numList.add（3）；//can put integer or null 在example2.4 我们可以看到的意思为，我们可以将类型参数为Integer或Integer超类的List赋给 numList变量，并且可以put元素到列表中（注意：在该例子中put进的元素只能为Integer或null类型）。 一条比较通用的规则：如果要往List中put元素则用，如果要从List中get元素则用，如果既要get又要put则不使用通配符。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com