Java泛型的理解与等价实现 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/273/2021_2022_Java_E6_B3_9B_E5_9E_8B_c104_273317.htm 泛型是Java SE 1.5的新特性，泛型的本质是参数化类型，也就是说所操作的数据类型被指定为一个参数。这种参数类型可以用在类、接口和方法的创建中，分别称为泛型类、泛型接口、泛型方法。 Java语言引入泛型的好处是安全简单。 在Java SE 1.5之前，没有泛型的情况的下，通过对类型Object的引用来实现参数的“ 任意化 ”，“ 任意化 ”带来的缺点是要做显式的强制类型转换，而这种转换是要求开发者对实际参数类型可以预知的情况下进行的。对于强制类型转换错误的情况，编译器可能不提示错误，在运行的时候才出现异常，这是一个安全隐患。 泛型的好处是在编译的时候检查类型安全，并且所有的强制转换都是自动和隐式的，提高代码的重用率。 泛型在使用中还有一些规则和限制： 1、泛型的类型参数只能是类类型（包括自定义类），不能是简单类型。 2、同一种泛型可以对应多个版本（因为参数类型是不确定的），不同版本的泛型类实例是不兼容的。 3、泛型的类型参数可以有多个。 4、泛型的参数类型可以使用extends语句，例如。习惯上成为“ 有界类型 ”。 5、泛型的参数类型还可以是通配符类型。例如Class classType = Class.forName(java.lang.String). 泛型还有接口、方法等等，内容很多，需要花费一番功夫才能理解掌握并熟练应用。在此给出我曾经了解泛型时候写出的两个例子（根据看的印象写的），实现同样的功能，一个使用了泛型，一个没有使用，通过对比，可以很快学会泛型的应用，学会这个基本上学会

了泛型70%的内容。 例子一：使用了泛型public class Gen { private T ob. //定义泛型成员变量 public Gen(T ob) { this.ob = ob. } public T getOb() { return ob. } public void setOb(T ob) { this.ob = ob. } public void showTyep() { System.out.println("T的实际类型是: " ob.getClass().getName()). }}public class GenDemo { public static void main(String[] args){ //定义泛型类Gen的一个Integer版本 Gen intOb=new Gen(88). intOb.showTyep(). int i= intOb.getOb(). System.out.println("value= " i). System.out.println("--------------------------------"). //定义泛型类Gen的一个String版本 Gen strOb=new Gen("Hello Gen!"). strOb.showTyep(). String s=strOb.getOb(). System.out.println("value= " s).}} 例子二：没有使用泛型public class Gen2 { private Object ob. //定义一个通用类型成员 public Gen2(Object ob) { this.ob = ob. } public Object getOb() { return ob. } public void setOb(Object ob) { this.ob = ob. } public void showTyep() { System.out.println("T的实际类型是: " ob.getClass().getName()). }}public class GenDemo2 { public static void main(String[] args) { //定义类Gen2的一个Integer版本 Gen2 intOb = new Gen2(new Integer(88)). intOb.showTyep(). int i = (Integer) intOb.getOb(). System.out.println("value= " i). System.out.println("--------------------------------"). //定义类Gen2的一个String版本 Gen2 strOb = new Gen2("Hello Gen!"). strOb.showTyep(). String s = (String) strOb.getOb(). System.out.println("value= " s). }} 运行结果： 两个例子运行Demo结果是相同的,控制台输出结果如下： T的实际类型是:java.lang.Integervalue= 88--------------------------------T的

实际类型是: java.lang.Stringvalue= Hello Gen!Process finished with exit code 0 看明白这个，以后基本的泛型应用和代码阅读就不成问题了。 