java初学者实践教程15－方法的重载与重写 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/269/2021_2022_java_E5_88_9D_E5_AD_A6_c67_269941.htm Java语言中的概念就是多，这回又有两个概念，重载和重写。这是两个新概念，也是两个令初学者容易混淆的概念。他们的概念截然不同，只不过都有个“ 重 ”字，就以为是很像的。 下面解释一下这两个概念：方法重载(overloading method) 是在一个类里面，方法名字相同，而参数不同。返回类型呢？可以相同也可以不同。 方法重写(overiding method) 子类不想原封不动地继承父类的方法，而是想作一定的修改，这就需要采用方法的重写。方法重写又称方法覆盖。如果还是搞混的话，就把“ 重写覆盖 ”，这个词多念几遍吧。知道是覆盖的话，就知道是子类覆盖父类的方法了。 实践： 重载的例子public class MethodOverloading { void recieve(int i) { System.out.println("接收一个int数据"). System.out.println("i=" i). } void recieve(float f) { System.out.println("接受一个float型的数据"). System.out.println("f=" f). } void recieve(String s) { System.out.println("接受一个String型数据"). System.out.println("s=" s). } public static void main(String[] args){ MethodOverloading m = new MethodOverloading(). m.recieve(3456). m.recieve(34.56).m.recieve(“ 百家拳软件项目研究室“ ).}} 大家看到了上面的例子方法receive()有三个，名字相同参数不同。这样的话，在main()调用的时候，参数用起来就很方便了。重写的例子似乎不用举了，记不住的话，就和“ 覆盖 ”。一起念。 有时候，重载和重写的方式有些复

杂，在jdk5里面。有一些方式能简化一些。我们来看看吧，jdk5的可变参数。 如果把相同参数类型的方法重载好几遍真的是很烦。就一个方法，pri(String args), pri(String arg0,String arg1), pri(String arg0,String arg1,String arg2), pri(String arg0,String arg1,String arg2,String arg3)。这样的话会写很多烦琐的代码。现在jdk5可以，用“…”来代替这些参数。 实践：public class overload { //若干个相同类型的参数，用“...”代替 public void pri(String... strings ){ for (String str : strings) //for这个循环语句也有迭代的意思 System.out.print(str). } public static void main(String[] args){ new overload().pri("100jq"," 百家拳软件项目研究室"," www.100jq.com"). }} jdk5的方法重写，比以前多了一个叫做协变返回的概念。在以往jdk的版本中，还有一个比较让人讨厌的地方。方法重写确实是比较不错的机制，如果想用父类的方法，写个super就可以了，如果不想用父类的方法就重写覆盖。但是重写覆盖的返回类型不能覆盖，父类的类型不够用怎么办，我们想在子类重写它的类型可以吗？现在可以了。看下面的例子：class Point2D { //定义二维的点 protected int x, y. public Point2D() { this.x=0. this.y=0.} public Point2D(int x, int y) { this.x = x. this.y = y. }}//定义三维的点，继承二维class Point3D extends Point2D { protected int z. public Point3D(int x, int y) { this(x, y, 0). } public Point3D(int x, int y, int z) { this.x = x. this.y = y. this.z = z. }}//定义二维的位置class Position2D { Point2D location. public Position2D() { this.location = new Point2D(). } public Position2D(int x, int y) { this.location = new Point2D(x, y). } public Point2D getLocation() { return location. }}//定义三维的位置，继承二维的位置class Position3D extends

Position2D { Point3D location. //在这里已经变成Point3D的类型了 public Position3D(int x, int y, int z) { this.location = new Point3D(x, y, z). } @Override //注释是重写方法 public Point3D getLocation() { return location. //返回是子类的类型而不是原来的类型了 }} 