Java中重载和重写的区别 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/474/2021_2022_Java_E4_B8_AD_E9_87_8D_c67_474978.htm （1）方法重载是让类以统一的方式处理不同类型数据的一种手段。多个同名函数同时存在，具有不同的参数个数/类型。重载Overloading是一个类中多态性的一种表现。 （2）Java的方法重载，就是在类中可以创建多个方法，它们具有相同的名字，但具有不同的参数和不同的定义。调用方法时通过传递给它们的不同参数个数和参数类型来决定具体使用哪个方法, 这就是多态性。 （3）重载的时候，方法名要一样，但是参数类型和个数不一样，返回值类型可以相同也可以不相同。无法以返回型别作为重载函数的区分标准。 下面是重载的例子： package c04.answer.//这是包名 //这是这个程序的第一种编程方法，在main方法中先创建一个Dog类实例，然后在Dog类的构造方法中利用this关键字调用不同的bark方法。不同的重载方法bark是根据其参数类型的不同而区分的。 //注意：除构造器以外，编译器禁止在其他任何地方中调用构造器。 package c04.answer. public class Dog { Dog() { this.bark(). } void bark()//bark()方法是重载方法 { System.out.println("no barking!"). this.bark("female", 3.4). } void bark(String m,double l)//注意：重载的方法的返回值都是一样的， { System.out.println("a barking dog!"). this.bark(5, "China"). } void bark(int a,String n)//不能以返回值区分重载方法，而只能以“ 参数类型 ”和“ 类名 ”来区分 { System.out.println("a howling dog"). } public static void main(String[] args) { Dog dog = new Dog(). //dog.bark().

//dog.bark("male", "yellow"). //dog.bark(5, "China"). 然后我们再来谈谈 重写（Overriding） （1） 父类与子类之间的多态性，对父类的函数进行重新定义。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。在Java中，子类可继承父类中的方法，而不需要重新编写相同的方法。但有时子类并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。 （2） 若子类中的方法与父类中的某一方法具有相同的方法名、返回类型和参数表，则新方法将覆盖原有的方法。如需父类中原有的方法，可使用super关键字，该关键字引用了当前类的父类。 （3） 子类函数的访问修饰权限不能少于父类的；下面是重写的例子： 概念：即调用对象方法的机制。 动态绑定的内幕： 1、编译器检查对象声明的类型和方法名，从而获取所有候选方法。试着把上例Base类的test注释掉，这时再编译就无法通过。 2、重载决策：编译器检查方法调用的参数类型，从上述候选方法选出唯一的那一个（其间会有隐含类型转化）。如果编译器找到多于一个或者没找到，此时编译器就会报错。试着把上例Base类的test(byte b)注释掉，这时运行结果是1 1。 3、若方法类型为priavte static final ，java采用静态编译，编译器会准确知道该调用哪个方法。 4、当程序运行并且使用动态绑定来调用一个方法时，那么虚拟机必须调用对象的实际类型相匹配的方法版本。在例子中，b所指向的实际类型是TestOverriding，所以b.test(0)调用子类的test。但是，子类并没有重写test(byte b)，所以b.test((byte)0)调用的是父类的test(byte b)。如果把父类的(byte b)注释掉，则通过第二步隐含类型转化为int,最终调用

的是子类的test(int i)。 学习总结：多态性是面向对象编程的一种特性，和方法无关。 简单说，就是同样的一个方法能够根据输入数据的不同，做出不同的处理，即方法的重载有不同的参数列表（静态多态性） 而当子类继承自父类的相同方法，输入数据一样，但要做出有别于父类的响应时，你就要覆盖父类方法，即在子类中重写该方法相同参数，不同实现（动态多态性） OOP三大特性：继承，多态，封装。 public class Base { void test(int i) { System.out.print(i). } void test(byte b) { System.out.print(b). } } public class TestOverriding extends Base { void test(int i) { i . System.out.println(i). } public static void main(String[]agrs) { Base b=new TestOverriding(). b.test(0) b.test((byte)0) } } 这时的输出结果是1 0，这是运行时动态绑定的结果。动态绑定 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