JAVA中浅复制与深复制 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/167/2021_2022_JAVA_E4_B8_AD_E6_B5_85_c104_167337.htm 1．浅复制与深复制概念⑴浅复制（浅克隆）被复制对象的所有变量都含有与原来的对象相同的值，而所有的对其他对象的引用仍然指向原来的对象。换言之，浅复制仅仅复制所考虑的对象，而不复制它所引用的对象。⑵深复制（深克隆）被复制对象的所有变量都含有与原来的对象相同的值，除去那些引用其他对象的变量。那些引用其他对象的变量将指向被复制过的新对象，而不再是原有的那些被引用的对象。换言之，深复制把要复制的对象所引用的对象都复制了一遍。 2．Java的clone（）方法⑴clone方法将对象复制了一份并返回给调用者。一般而言，clone（）方法满足：①对任何的对象x，都有x.clone() !=x//克隆对象与原对象不是同一个对象②对任何的对象x，都有x.clone().getClass()= =x.getClass()//克隆对象与原对象的类型一样③如果对象x的equals()方法定义恰当，那么x.clone().equals(x)应该成立。⑵Java中对象的克隆①为了获取对象的一份拷贝，我们可以利用Object类的clone()方法。②在派生类中覆盖基类的clone()方法，并声明为public。③在派生类的clone()方法中，调用super.clone()。④在派生类中实现Cloneable接口。 请看如下代码： class Student implements Cloneable{ String name. int age. Student(String name,int age) { this.name=name. this.age=age. } public Object clone() { Object o=null. try { o=(Student)super.clone().//Object中的clone()识别出你要复制的是哪一// 个对象。 }

catch(CloneNotSupportedException e) { System.out.println(e.toString()). } return o. }} public static void main(String[] args) { Student s1=new Student("zhangsan",18). Student s2=(Student)s1.clone(). s2.name="lisi". s2.age=20.System.out.println("name=" s1.name "," "age=" s1.age).//修改学生2后，不影响 //学生1的值。 } 说明：①为什么我们在派生类中覆盖Object的clone()方法时，一定要调用super.clone()呢？在运行时刻，Object中的clone()识别出你要复制的是哪一个对象，然后为此对象分配空间，并进行对象的复制，将原始对象的内容一一复制到新对象的存储空间中。②继承自java.lang.Object类的clone()方法是浅复制。以下代码可以证明之。 class Professor { String name. int age. Professor(String name,int age) { this.name=name. this.age=age. }}class Student implements Cloneable{ String name.//常量对象。 int age. Professor p.//学生1和学生2的引用值都是一样的。 Student(String name,int age,Professor p) { this.name=name. this.age=age. this.p=p. } public Object clone() { Student o=null. try { o=(Student)super.clone(). } catch(CloneNotSupportedException e) { System.out.println(e.toString()). } o.p=(Professor)p.clone(). return o. }}public static void main(String[] args) { Professor p=new Professor("wangwu",50). Student s1=new Student("zhangsan",18,p). Student s2=(Student)s1.clone(). s2.p.name="lisi". s2.p.age=30.System.out.println("name=" s1.p.name "," "age=" s1.p.age).//学生1的教授 //成为lisi,age为30。}那应该如何实现深层次的克隆，即修改s2的教授不会影响s1的教授？代码改进如下。 改进使学生1的Professor不改变（深层次的克隆）class

Professor implements Cloneable{ String name. int age. Professor(String name,int age) { this.name=name. this.age=age. } public Object clone() { Object o=null. try { o=super.clone(). } catch(CloneNotSupportedException e) { System.out.println(e.toString()). } return o. }}class Student implements Cloneable{ String name. int age. Professor p. Student(String name,int age,Professor p) { this.name=name. this.age=age. this.p=p. } public Object clone() { Student o=null. try { o=(Student)super.clone(). } catch(CloneNotSupportedException e) { System.out.println(e.toString()). } o.p=(Professor)p.clone(). return o. }}public static void main(String[] args) { Professor p=new Professor("wangwu",50). Student s1=new Student("zhangsan",18,p). Student s2=(Student)s1.clone(). s2.p.name="lisi". s2.p.age=30.System.out.println("name=" s1.p.name "," "age=" s1.p.age).//学生1的教授不改变。} 