java的深复制与浅复制

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。

JAVA的 CLONE()方法

x.clone()!=x //它们并不是同一个对象；

x.clone().getClass()=x.getClass() //它们是同一个类型；

x.clone().equals(x) //返回 true 它们的值相同；

JAVA的 cloneAble接口

为了获取对象的一份拷贝，我们可以利用 Object类的 clone()方法。
在派生类中覆盖基类的 clone()方法，并声明为 public。
在派生类的 clone()方法中，调用 super.clone()。
在派生类中实现 Cloneable接口。

代码；

```
public class Student implements Cloneable
{
        String name;
        int age;
        public 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(Exception e )
        {
            System.out.println(e.toString());
         }
}
public static void mian(String[] args)
{
        Student s1=new Student("daiwang","21")
        Student s2=(Student)s1.clone();
        s2.name="wang";
        s2.age=23;
         System.out.println("name:"+s1.name+"age;"+s1.age);//修改s2的值不会影响s1的
值
}
```

做深层复制：

```
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("wang",50);
      Student s1=new Student("daiwang",18,p);
      Student s2=(Student)s1.clone();
      s2.p.name="wangdai";
```

```
        s2.p.age=23;
System.out.println("name="+s1.p.name+","+"age="+s1.p.age);//学生 1的教授不改变。
}
```