链滴

设计模式之原型模式

作者：sologxl

原文链接：https://ld246.com/article/1543216628246

来源网站：链滴

这次来说说原型模式，原型模式相对就简单理解很多了。

其实就是克隆复制。比如我有一个对象，我可以从这个对象克隆一个新对象出来，但是克隆出来这个象会有两种情况：

一种是将内部的变量全都克隆出来，这样两个对象就相互之间完全不影响。我们把这种克隆称之为深隆。

一种就是克隆出来的对象和原型对象的变量共用。我们把这种称之为浅克隆。

下面是浅克隆的示例：

浅克隆中，当你在克隆对象中修改shuzu变量时，你会发现原型对象的shuzu变量值也发生了改变。是由于此时的克隆对象和原型对象是共用变量的，类似于共用静态变量。

```
package com.gxl.butt;

/**
* Created by guoxiaolin on 2018/11/26.
*/
public class CloneTest implements Cloneable {  //浅克隆需实现Cloneable接口
  private String[] shuzu = null;
  private String str = null;
  @Override
  protected CloneTest clone() throws CloneNotSupportedException {
    return (CloneTest) super.clone();
  }
  public void setShuzu(String[] shuzu){
    this.shuzu = shuzu;
  }
  public void setStr(String str){
    this.str = str;
  }
  public void printlnName(){
    System.out.println(this.shuzu[0]+","+this.shuzu[1]);
    System.out.println(this.str);
  }
  public static void main(String[] args) throws CloneNotSupportedException {
    CloneTest original = new CloneTest();
    original.setShuzu(new String[]{"shuzuA1","shuzuA2"});
    original.setStr("89");
    CloneTest cloneTest = original.clone();
      //cloneTest.setShuzu(new String[]{"shuzuB1","shuzuB2"});
      //cloneTest.setStr("12");
     original.printlnName();
    cloneTest.printlnName();
  }
}
```

下面是深克隆：

深克隆中，当你在克隆对象中修改shuzu变量时，你会发现原型对象的shuzu变量值不会发生改变，是因为shuzu变量也被独立克隆出来，克隆对象和原型对象此时是完全不相互影响的

```
package com.gxl.butt;
import java.io.*;
/**
 * Created by Administrator on 2018/11/26.
 */
 public class DeepCloneTest  implements Serializable { //深度克隆需实现序列化
private String[] shuzu = null;
private transient String str = null;//transient修饰的变量，拒绝复制克隆
  public DeepCloneTest deepClone() throws IOException, ClassNotFoundException {
    //将对象写到流里
       ByteArrayOutputStream bos = new ByteArrayOutputStream();
    ObjectOutputStream oos = new ObjectOutputStream(bos);
    oos.writeObject(this);
    //从流里读回来
    ByteArrayInputStream bis = new ByteArrayInputStream(bos.toByteArray());
    ObjectInputStream ois = new ObjectInputStream(bis);
    return (DeepCloneTest) ois.readObject();
}
public void setShuzu(String[] shuzu){
    this.shuzu = shuzu;
}
public String[] getShuzu(){
    return this.shuzu;
}
public void setStr(String str){
    this.str = str;
}
public void printlnName(){
    System.out.println(this.shuzu[0]+","+this.shuzu[1]);
    System.out.println(this.str);
}
public static void main(String[] args) throws Exception {
    DeepCloneTest original = new DeepCloneTest();
    original.setShuzu(new String[]{"shuzuA1","shuzuA2"});
    original.setStr("1000");
    DeepCloneTest cloneTest = original.deepClone();
    cloneTest.getShuzu()[0]="shuzuB1";
    cloneTest.setShuzu(cloneTest.getShuzu());
    //        cloneTest.setStr("2000");
        original.printlnName();
    cloneTest.printlnName();
}
}
```