5、原型模式（Prototype）

原型模式虽然是创建型的模式，但是与工程模式没有关系，从名字即可看出，该模式的思想就是将一个对象作为原型，对其进行复制、克隆，产生一个和原对象类似的新对象。本小结会通过对象的复制，进行讲解。在Java中，复制对象是通过clone()实现的，先创建一个原型类：

[java] view plaincopy

```
1. public class Prototype implements Cloneable {
2.
3.     public Object clone() throws CloneNotSupportedException {
4.         Prototype proto = (Prototype) super.clone();
5.         return proto;
6.     }
7. }
```

很简单，一个原型类，只需要实现Cloneable接口，覆写clone方法，此处clone方法可以改成任意的名称，因为Cloneable接口是个空接口，你可以任意定义实现类的方法名，如cloneA或者cloneB，因为此处的重点是super.clone()这句话，super.clone()调用的是Object的clone()方法，而在Object类中，clone()是native的，具体怎么实现，我会在另一篇文章中，关于解读Java中本地方法的调用，此处不再深究。在这儿，我将结合对象的浅复制和深复制来说一下，首先需要了解对象深、浅复制的概念：

浅复制：将一个对象复制后，基本数据类型的变量都会重新创建，而引用类型，指向的还是原对象所指向的。

深复制：将一个对象复制后，不论是基本数据类型还有引用类型，都是重新创建的。简单来说，就是深复制进行了完全彻底的复制，而浅复制不彻底。

此处，写一个深浅复制的例子：

[java] view plaincopy

```
1. public class Prototype implements Cloneable, Serializable {
2.
3.     private static final long serialVersionUID = 1L;
4.     private String string;
5.
6.     private SerializableObject obj;
7.
8.     /* 浅复制 */
9.     public Object clone() throws CloneNotSupportedException {
10.         Prototype proto = (Prototype) super.clone();
11.         return proto;
12.     }
```

```
13.
14.     /* 深复制 */
15.
    public Object deepClone() throws IOException, ClassNotFoundException {

16.
17.         /* 写入当前对象的二进制流 */
18.         ByteArrayOutputStream bos = new ByteArrayOutputStream();
19.         ObjectOutputStream oos = new ObjectOutputStream(bos);
20.         oos.writeObject(this);
21.
22.         /* 读出二进制流产生的新对象 */
23.
        ByteArrayInputStream bis = new ByteArrayInputStream(bos.toByteArra
y());
24.         ObjectInputStream ois = new ObjectInputStream(bis);
25.         return ois.readObject();
26.     }
27.
28.     public String getString() {
29.         return string;
30.     }
31.
32.     public void setString(String string) {
33.         this.string = string;
34.     }
35.
36.     public SerializableObject getObj() {
37.         return obj;
38.     }
39.
40.     public void setObj(SerializableObject obj) {
41.         this.obj = obj;
42.     }
43.
44. }
45.
46. class SerializableObject implements Serializable {
47.     private static final long serialVersionUID = 1L;
48. }
```

要实现深复制，需要采用流的形式读入当前对象的二进制输入，再写出二进制数据对应的对象。