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

```
public class Prototype implements Cloneable {

    public Object clone() throws CloneNotSupportedException {
        Prototype proto = (Prototype) super.clone();
        return proto;
    }
}
```

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

```
public class Prototype implements Cloneable, Serializable {

    private static final long serialVersionUID = 1L;
    private String string;

    private SerializableObject obj;
```

```
    /* 浅复制 */
    public Object clone() throws CloneNotSupportedException {
        Prototype proto = (Prototype) super.clone();
        return proto;
    }

    /* 深复制 */
    public Object deepClone() throws IOException, ClassNotFoundException
{

        /* 写入当前对象的二进制流 */
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(bos);
        oos.writeObject(this);

        /* 读出二进制流产生的新对象 */
        ByteArrayInputStream bis = new ByteArrayInputStream(bos.toByteArr
ay());
        ObjectInputStream ois = new ObjectInputStream(bis);
        return ois.readObject();
    }

    public String getString() {
        return string;
    }

    public void setString(String string) {
        this.string = string;
    }

    public SerializableObject getObj() {
        return obj;
    }
```

```
    public void setObj(SerializableObject obj) {
        this.obj = obj;
    }

}

class SerializableObject implements Serializable {
    private static final long serialVersionUID = 1L;
}
```

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。