* 设计模式 – 原始原型模式（Prototype）：

原型模式使用clone() 能够动态的抽取当前对象运行时的状态并且克隆到新的对象中，新对象就可以在此基础上进行操作而不损坏原有对象；而new只能得到一个刚初始化的对象，而在实际应用中，这往往是不够的。

用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象。

原型模式底层调用的 其实就是Object类继承下来的clone()方法。

在Java中有一种克隆技术，我有一个对象，在内存中，想再new一个的话，通过普通的new，可能要调用这个类的构造方法，构造方法可能没有无参或必须使用有参的，那么具体的参数值需要获取，有的时候可能你没有获取的权限，权限需要一步一步验证，那么耗费时间比较大。那么，这个时候，内存中恰巧有一个你所需要的对象，则可以通过克隆技术，把这个对象再创建一个副本。

原型模式用的主要是深克隆。克隆技术：以某个对象为原型，复制出新的对象，显然新 的对象具备原型对象的所有特点。

原型模式角色：

1，客户角色：完成一个原型对象来克隆自己，得到一个新对象。

2，抽象原型角色：实现自己的clone()方法，通常是抽象类，且具有许多具体的子类。

3，具体原型角色：被复制的对象，为抽象类的具体子类。

优势：效率高，直接克隆，避免重新执行构造过程等步骤。

缺陷：1，破坏了系统的安全性。2，每一个类都必须配备一个克隆方法。

由于clone() 方法在java实现中有着一定的弊端和风险，所以clone() 方法是不建议使用的。因此很少能在java应用中看到原型模式的使用。

* 原型模式的使用场景：

通过new产生一个对象，有时候需要很多步骤，比如说数据采集，比如说连接远程数 据库(数据库不在本地，在远程调用)，有的时候，这个数据还没有权限去获取，或者获 取这个权限需要很大的时间消耗，像权限验证等。不是说想能new出来就能出来的， 需要很多资源的耗费。即需要非常复杂的数据准备或者权限访问，则可以使用原型模式。

* 设计模式 – 原始原型模式Demo：



