**第七章 原型模式**

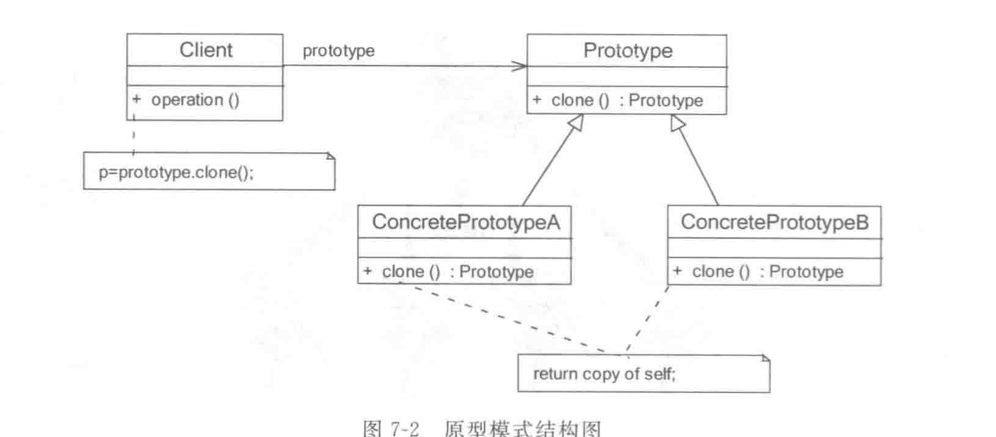
原型模式是一种特殊的创建型模式,它通过复制一个已有对象来获取更多相同或者相似的对象。原型模式可以提高相同类型对象的创建效率，简化创建过程。  
  
**原型模式概述**：

类比孙悟空拔毛变小猴，在面向对象系统中也可以通过复制一个原型)对象得到多个与原型对象一模一样的新对象，这就是原型模式的动机。

原型模式的定义如下:使用原型实例指定待创建对象的类型，并且通过复制这个原型来创建新的对象。  
原型模式是一种对象创建型模式，它的工作原理很简单:将一个原型对象传给要发动创建的对象(即客户端对象),这个要发动创建的对象通过请求原型对象复制自己来实现创建过程。

**原型模式结构与实现：**

**原型模式的结构：**



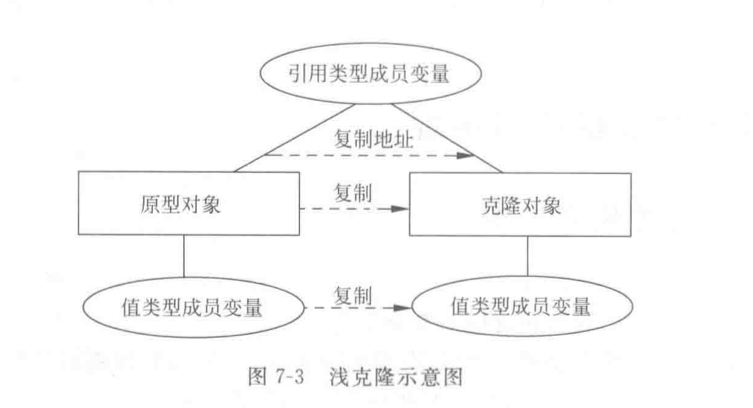
1. Prototype(抽象原型类)，它是声明克隆方法的接口,是所有具体原型类的公共父类,它可以是抽象类也可以是接口,甚至还可以是具体实现类。
2. ConeretePrototype(具体原型类):它实现在抽象原型类中声明的克隆方法,在克隆方法中返回自己的一个克隆对象。
3. Client(客户类): 在客户类中,让一个原型对象克隆自 身从而创建一个新的对象，只需要直接实例化或通过工厂方法等方式创建一个原型对象，再通过调用该对象的克隆方法即可得到多个相同的对象。由于客户类针对抽象原型类Prototype编程,因此用户可以根据需要选择具体原型类,系统具有较好的可扩展性,增加或更换具体原型类都很方便。

**浅克隆与深克隆**

原型模式的克隆机制分为两种，即浅克隆(ShallowClone)和深克隆(DeepClone)。

**1.浅克隆**

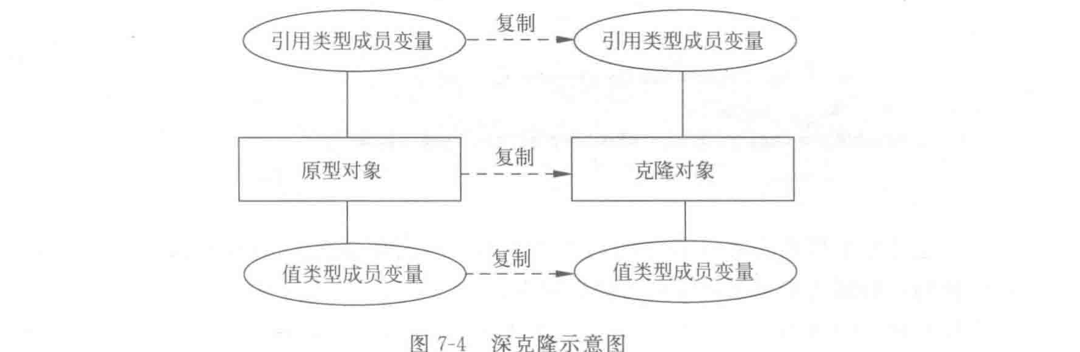
在浅克隆中，如果原型对象的成员变量是值类型(如int、double、byte、boolean、char等)基本数据类型)，将复制一份给克隆对象;如果原型对象的成员变量是引用类型(如类、接日、数组等复杂数据类型)，则将引用对象的地址复制一份给克隆对象,也就是说原型对象和克隆对象的成员变董指向相同的内存地址。简单来说,在浅克隆中，当原型对象被复制时只复制它本身和其中包含的值类型的成员变量,而引用类型的成员变量并没有复制。



**2.深克隆**

在深克隆中，无论原型对象的成员变量是值类型还是引用类型都将复制一份给克隆对

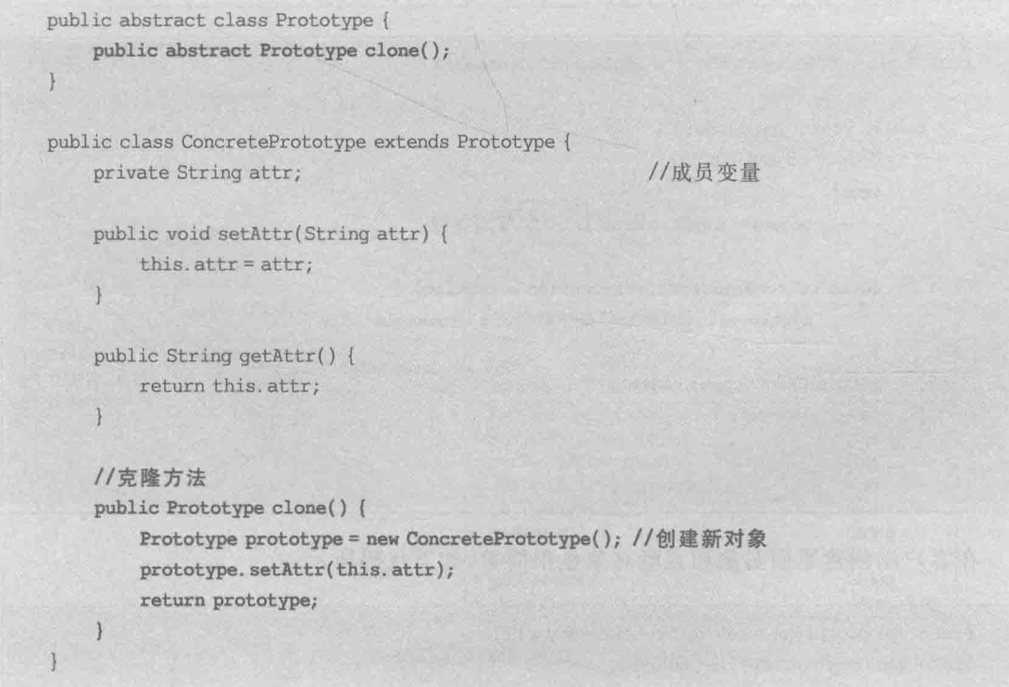
象,深克隆将原型对象的所有引用对象也复制一份给克隆对象。简单来说,在深克隆中，除了对象本身被复制外,对象所包含的所有成员变量也将被复制。



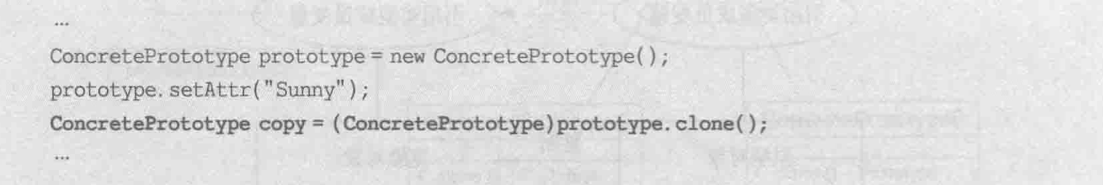
**原型模式实现：**

**（1）通用实现方法**

通用的克隆实现方法是在具体原型类的克隆方法中实例化一个与自身类型相同的对象(并将其返回.同时将相关的参数传入新创建的对象中。保证它们的成员变量相同。示意代码如下：



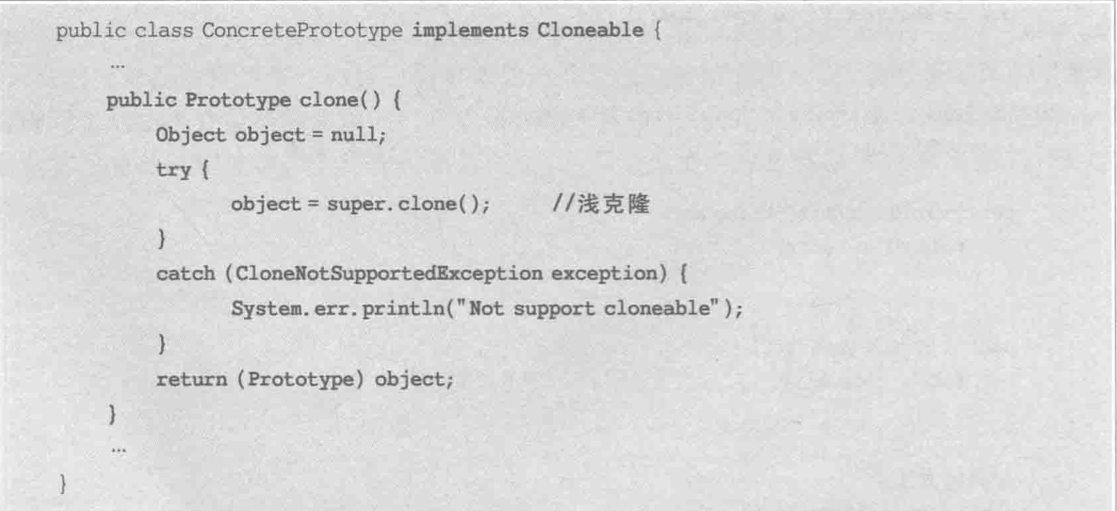
在客户类中只需要创建一个ConcretePrototype对象作为原型对象，然后调用其clone(方法即可得到对应的克隆对象，例如:



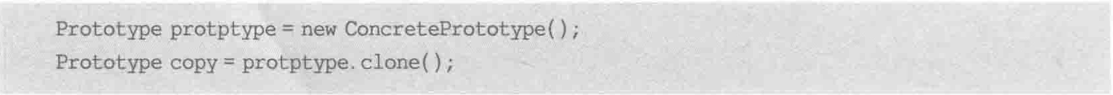
**（2）Java语言中的clone()方法和Cloneable接口**

在Java中可以直接使用Object提供的one()方法来实现对象的浅克隆。

需要注意的是能够实现克隆的Java类必须实现一个标识接口Cloneable,表示这个 Java类支持被复制。如果一个类没有实现这个接口但是调用了clone()方法,Java 编译器.将抛出一个CloneNotSupportedException异常。如下代码所示:



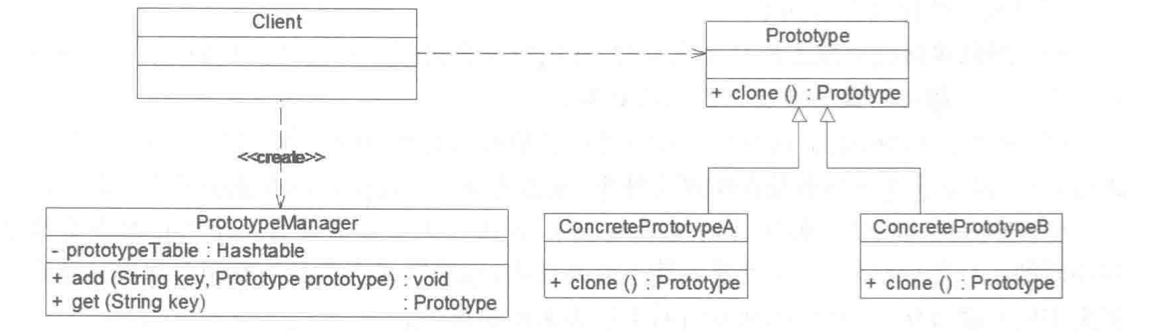
在客户端创建原型对象和克隆对象代码如下：



***Java语言中的clone()方法满足以下几点:***  
(1)对任何对象x,都有x.clone()!=x,即克隆对象与原型对象不是同一个对象。  
(2)对任何对象x,都有x.clone().getClass()==x.getClass(),即克隆对象与原型对象的类型一样。  
(3)如果对象x的equals()方法定义恰当,那么x.clone().equals(x)应该成立。为了获取对象的一个克隆,可以直接利用Object类的clone()方法,具体步骤如下:

1. 在派生类中覆盖基类的clone()方法,并声明为public。
2. 在派生类的clone()方法中调用super.clone()。
3. 派生类需实现Cloneable接口。  
   此时,Object类相当于抽象原型类,所有实现了Cloneable接口的类相当于具体原型类。

**原型管理器**  
原型管理器(Prototype Manager)将多个原型对象存储在一个集合中供客户端使用，它是一个专门负责克隆对象的工厂.其中定义了一个集合用于存储原型对象.如果需要某个原型对象的一个克隆，可以通过复制集合中对应的原型对象来获得。在原型管理器中针对抽象原型类进行编程，以便扩展，其结构如图7-7所示。



**原型模式优缺点以及适用环境**

***原型模式优点***  
(1)当创建新的对象实例较为复杂时,使用原型模式可以简化对象的创建过程,通过复制一个已有实例可以提高新实例的创建效率。  
(2)扩展性较好,由于在原型模式中提供了抽象原型类,在客户端可以针对抽象原型类进行编程,而将具体原型类写在配置文件中,增加或减少产品类对原有系统没有任何影响。

(3)原型模式提供了简化的创建结构,工厂方法模式常常需要有一个与产品类等级结构相同的工厂等级结构,而原型模式就不需要这样,原型模式中产品的复制是通过封装在原型类中的克隆方法实现的，无须专门的工厂类来创建产品。  
(4)可以使用深克隆的方式保存对象的状态,使用原型模式将对象复制一份并将其状态保存起来,以便在需要的时候使用(例如恢复到某一历史状态),可辅助实现撤销操作。  
  
***原型模式缺点***  
(1)需要为每一个类配备一个克隆方法,而且该克隆方法位于一个类的内部,当对已有的类进行改造时需要修改源代码,违背了开闭原则。  
(2)在实现深克隆时需要编写较为复杂的代码,而且当对象之间存在多重的嵌套引用时,为了实现深克隆,每一层对象对应的类都必须支持深克隆,实现起来可能会比较麻烦。  
***原型模式适用环境***  
(1)创建新对象成本较大(例如初始化需要占用较长的时间、占用太多的CPU资源或网络资源),新对象可以通过复制已有对象来获得，如果是相似对象,则可以对其成员变量稍作修改。  
(2)系统要保存对象的状态,而对象的状态变化很小。  
(3)需要避免使用分层次的工厂类来创建分层次的对象,并且类的实例对象只有一个.或很少的几个组合状态,通过复制原型对象得到新实例可能比使用构造函数创建一个新实例更加方便。

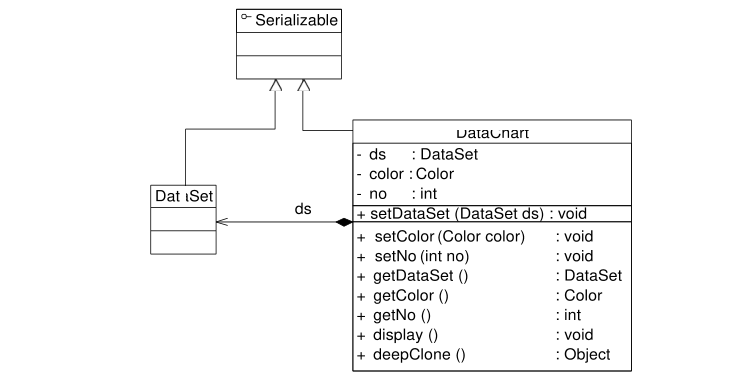
**本章习题参考答案如下：**

1.A

2.D

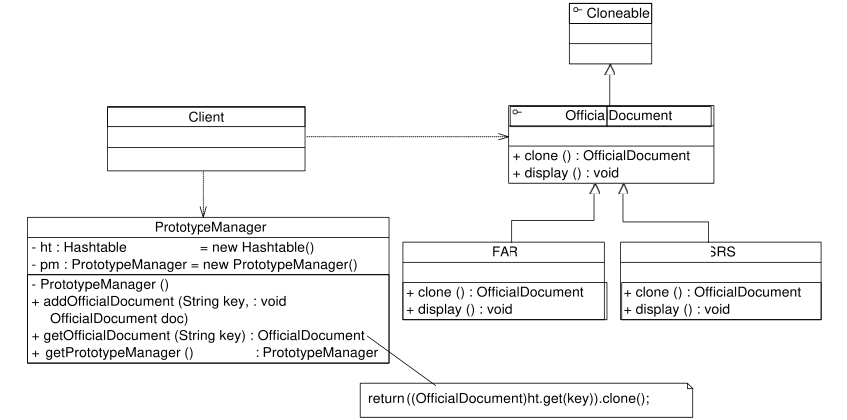
3.C

4.参考类图如下：

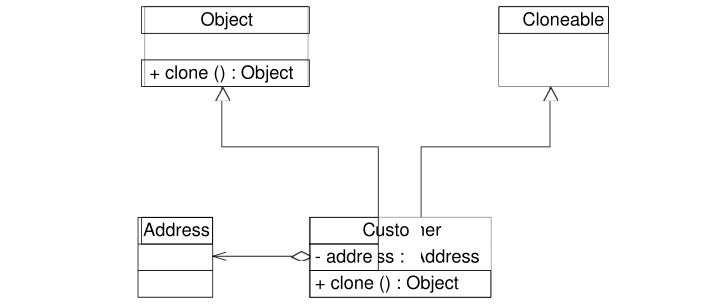


在该设计方案中，DataChart类包含一个DataSet对象，在复制DataChart对象的同时将复制DataSet对象，因此需要使用深克隆技术，可使用流来实现深克隆。

5.参考类图如下：



6.浅克隆类图参考如下：



深克隆类图参考如下：

