15·常用的设计模式（概率80%）

**单例设计模式(例如：Application)**

所谓单例设计模式简单说就是无论程序如何运行，采用单例设计模式的类（Singleton类）永远只会有一个实例化对象产生。具体实现步骤如下：

(1) 将采用单例设计模式的类的构造方法私有化（采用private修饰）。

(2) 在其内部产生该类的实例化对象，并将其封装成private static类型。

(3) 定义一个静态方法返回该类的实例。

**工厂设计模式**

程序在接口和子类之间加入了一个过渡端，通过此过渡端可以动态取得实现了共同接口的子类实例化对象

**代理设计模式**

指由一个代理主题来操作真实主题，真实主题执行具体的业务操作，而代理主题负责其他相关业务的处理。比如生活中的通过代理访问网络，客户通过网络代理连接网络（具体业务），由代理服务器完成用户权限和访问限制等与上网相关的其他操作（相关业务）。

**观察者设计模式(自定义控件的时候设置点击事件,)**

所谓观察者模式，举个例子现在许多购房者都密切观察者房价的变化，当房价变化时，所有购房者都能观察到，以上的购房者属于观察者，这便是观察者模式。

java中可以借助Observable类和Observer接口轻松实现以上功能。当然此种模式的实现也不仅仅局限于采用这两个类。

观察者模式本质上就是设置监听:通过监听来观察发生变化,通过回调返回,

首先需要定义一个接口,接口里面需要定义一个方法,方法的参数就是需要返回的参数,

其次需要定义一个监听设置的方法

最后在需要监听的地方设置监听

在需要调用的的地方首先new出对象,调用监听的方法,设置内部实现类,实现方法,从而达到监听的目的.

**适配器模式(listview,gridview)**

如果一个类要实现一个具有很多抽象方法的接口，但是本身只需要实现接口中的部分方法便可以达成目的，所以此时就需要一个中间的过渡类，但此过渡类又不希望 直接使用，所以将此类定义为抽象类最为合适，再让以后的子类直接继承该抽象类便可选择性的覆写所需要的方法，而此抽象类便是适配器类。