**Android源码中设计模式分析**

## 一.Builder模式

## 定义：使用多个简单的对象一步一步构建成一个复杂的对象。 这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。Builder 类是独立于其他对象的。

## 使用场景：主要是在创建某个对象时，需要设定很多的参数（通过setter方法），但是这些参数必须按照某个顺序设定，或者是设置步骤不同会得到不同结果。

## UML图

## 

## 例子：

public class MyData{

private int id;

private String num;

public void Test(){

}

public void setId(int id){

this.id=id;

}

public void setNum(String num){

this.num=num+"id";

}

}

1. Builder例子
2. public class MyBuilder{
3. private int id;
4. private String num;
5. public MyData build(){
6. MyData d=new MyData();
7. d.setId(id);
8. d.setNum(num);
9. return t;
10. }
11. public MyBuilder setId(int id){
12. this.id=id;
13. return this;
14. }
15. public MyBuilder setNum(String num){
16. this.num=num;
17. return this;
18. }
19. }
20. public class Test{
21. public static void main(String[] args){
22. MyData d=new MyBuilder().setId(10).setNum("hc").build();
23. }
24. }

6. 解释：回到主题，就是如果是上面的例子，该怎么办呢？你可能会说，那还不简单，先调用setId函数，再调用setNum函数。是的，没错。可是，万一你一不小心先调用了setNum呢？这是比较简单的示例，如果是比较复杂的，有很多变量之间依赖的关系，那你每次都得小心翼翼的把各个函数的执行步骤写正确。而Builder中例子注意到，Builer类的setter函数都会返回自身的引用this，这主要是用于链式调用，这也是Builder设计模式中的一个很明显的特征，

7. Android中例子在创建对话框时，用到了Builder设计模式，这里的create()函数就想到上面代码中的build函数。

AlertDialog.Builer builder=new AlertDialog.Builder(context);

builder.setIcon(R.drawable.icon)

.setTitle("title")

.setMessage("message")

.setPositiveButton("Button1",

new DialogInterface.OnclickListener(){

public void onClick(DialogInterface dialog,int whichButton){

setTitle("click");

}

})

.create()

.show();

8. 总结：

Builder 模式和 AbstractFactory 模式在功能上很相似，因为都是用来创建大的复杂的对象，它们的区别是：Builder 模式强调的是一步步创建对象，并通过相同的创建过程可以获得不同的结果对象，一般来说 Builder 模式中对象不是直接返回的。而在 AbstractFactory 模式中对象是直接返回的，AbstractFactory 模式强调的是为创建多个相互依赖的对象提供一个同一的接口。

Builder 模式的主要优点在于客户端不必知道产品内部组成的细节，将产品本身与产品的创建过程解耦，使得相同的创建过程可以创建不同的产品对象，每一个具体建造者都相对独立，而与其他的具体建造者无关，因此可以很方便地替换具体建造者或增加新的具体建造者，符合“开闭原则”，还可以更加精细地控制产品的创建过程；其主要缺点在于由于建造者模式所创建的产品一般具有较多的共同点，其组成部分相似，因此其使用范围受到一定的限制，如果产品的内部变化复杂，可能会导致需要定义很多具体建造者类来实现这种变化，导致系统变得很庞大。

Builder 模式将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。建造者模式是一步一步创建一个复杂的对象，它允许用户只通过指定复杂对象的类型和内容就可以构建它们，用户不需要知道内部的具体构建细节。建造者模式属于对象创建型模式。