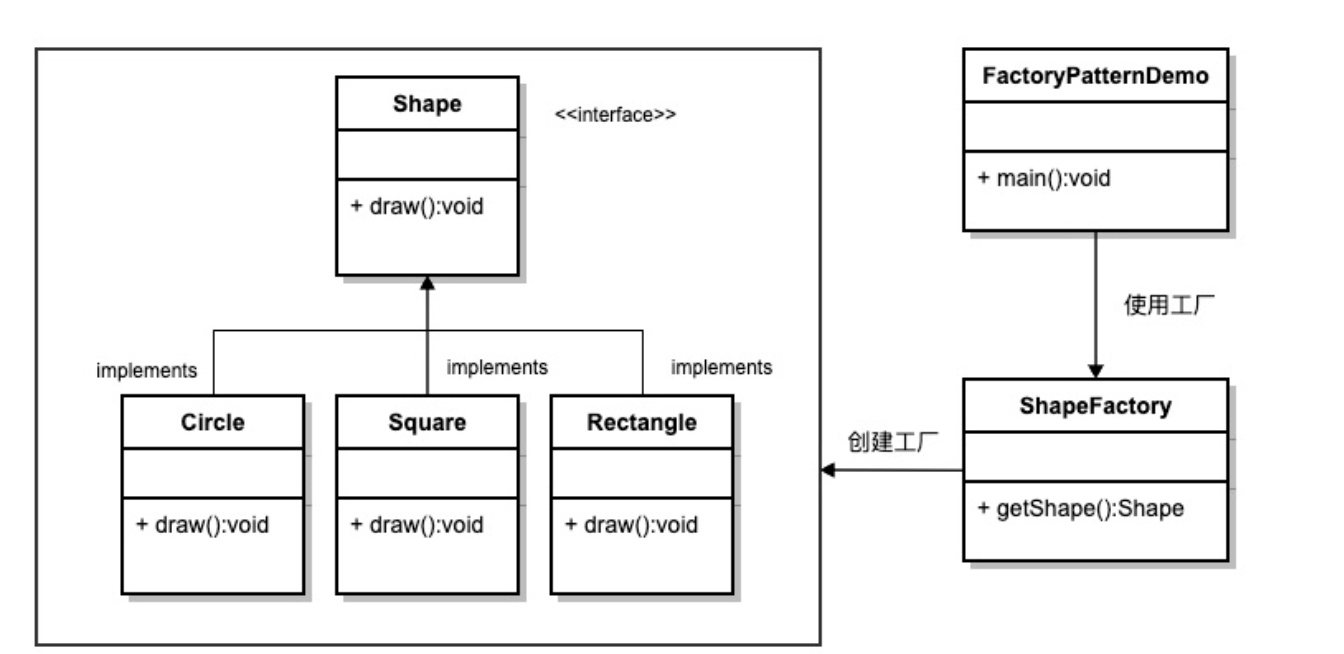
设计模式：

创造者模式：

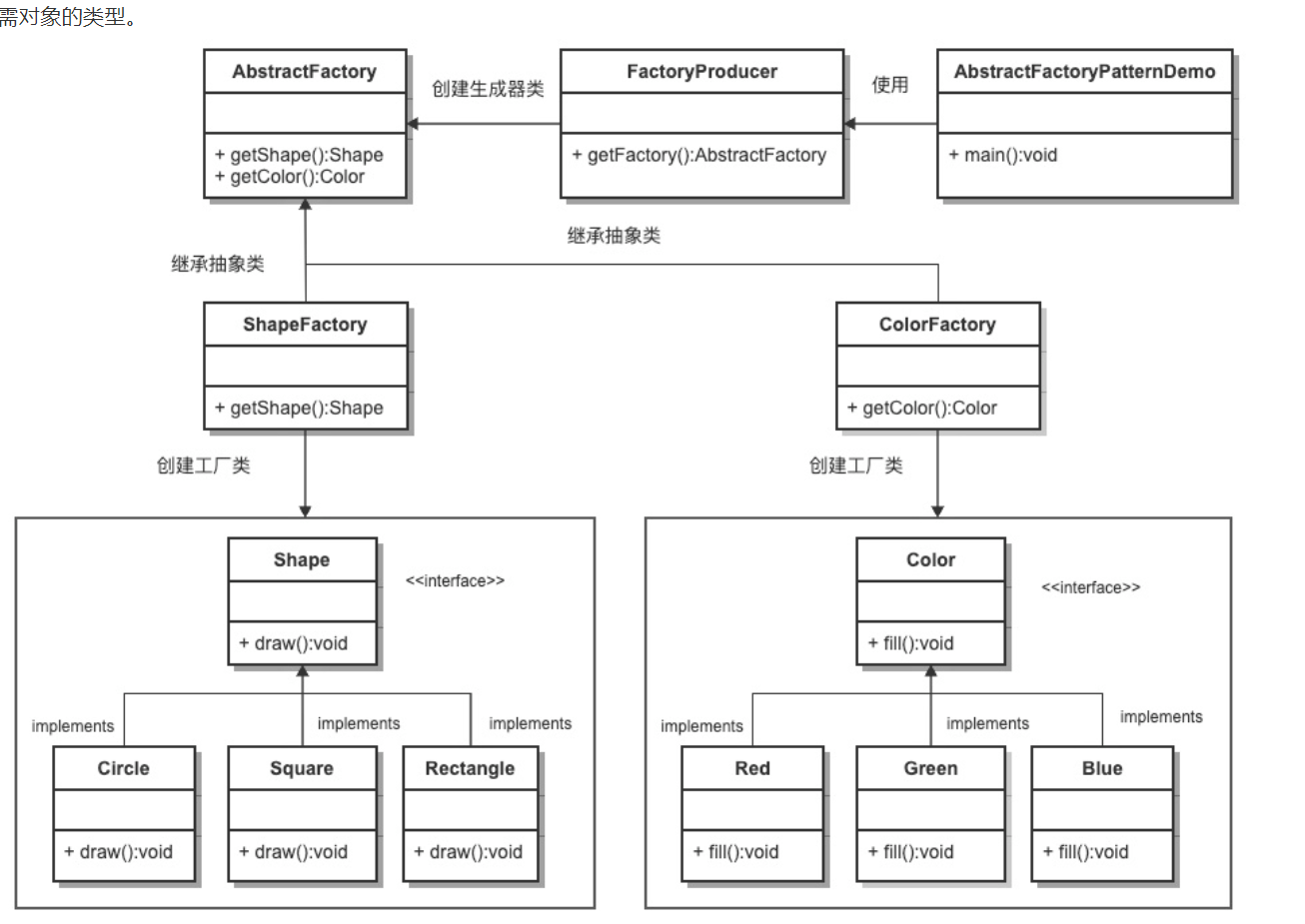
工厂模式



Shape s1=ShapeFactory.getShape(1);

根据数字 返回对应Circle();

抽象工厂模式：



ABSFactory=Factory.producer(1)

absFactory的使用 使用的是factory里面的静态方法；

abs.getColor(1);

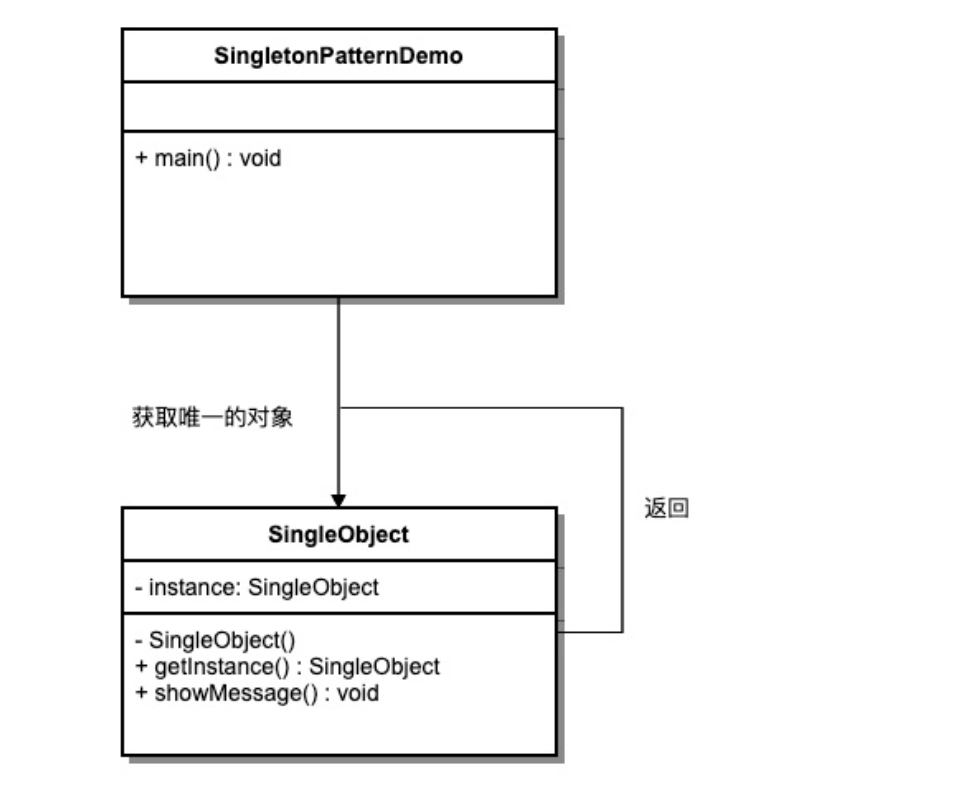
absFac

getShape(){

}

getColor(){

}两个抽象方法，Shape Factory执行自己的



public class A{

//要加statci

private static volatile A instance;

private A(){}

public static A getA(){

if(instance==null){

synchronized(A.class){

if(instance==null)

instance=new A();

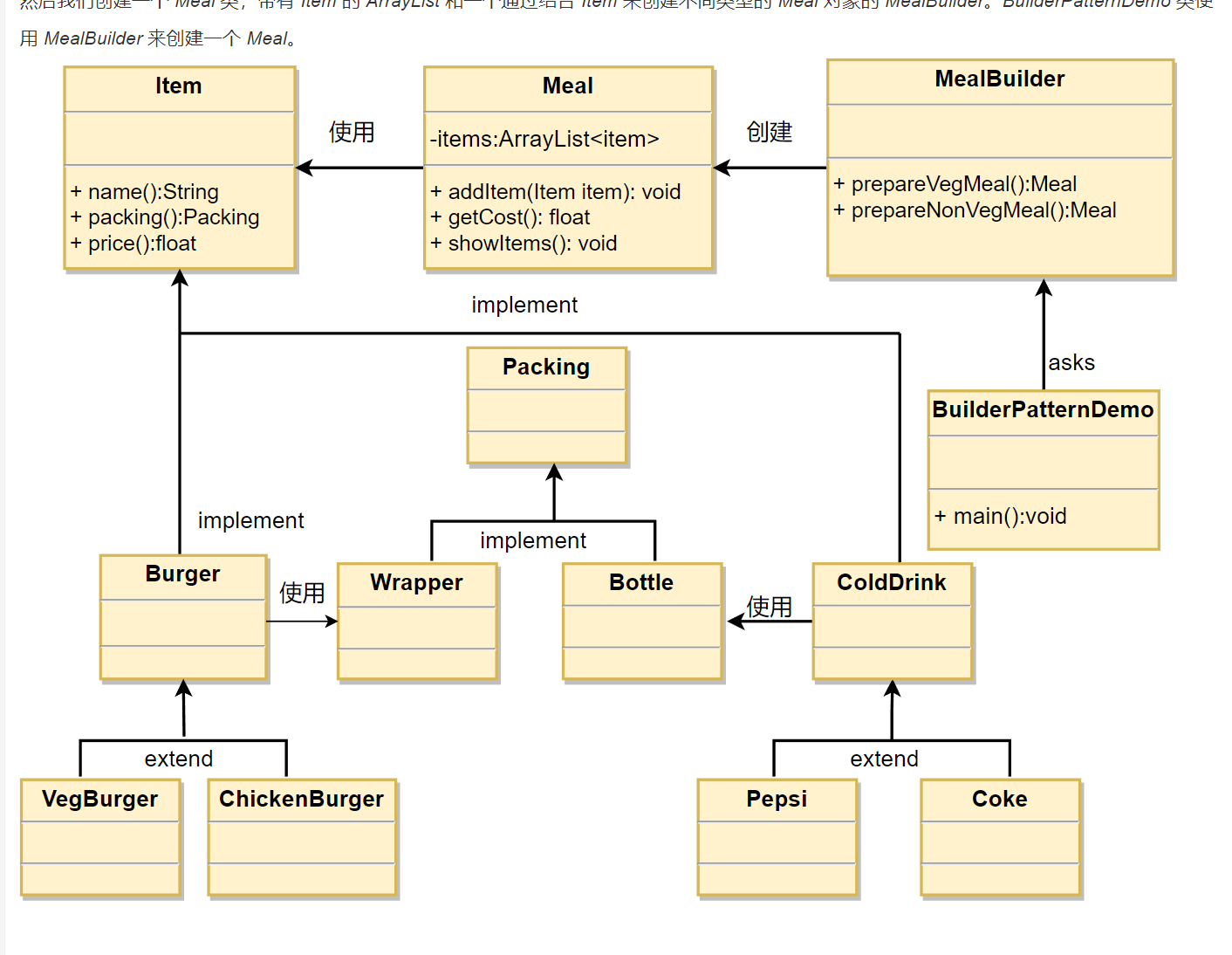
}

}

return instance;

}

}



MealBuilder mb1；

mb1.prepareMeat();

mb1.prepareNoMeat();

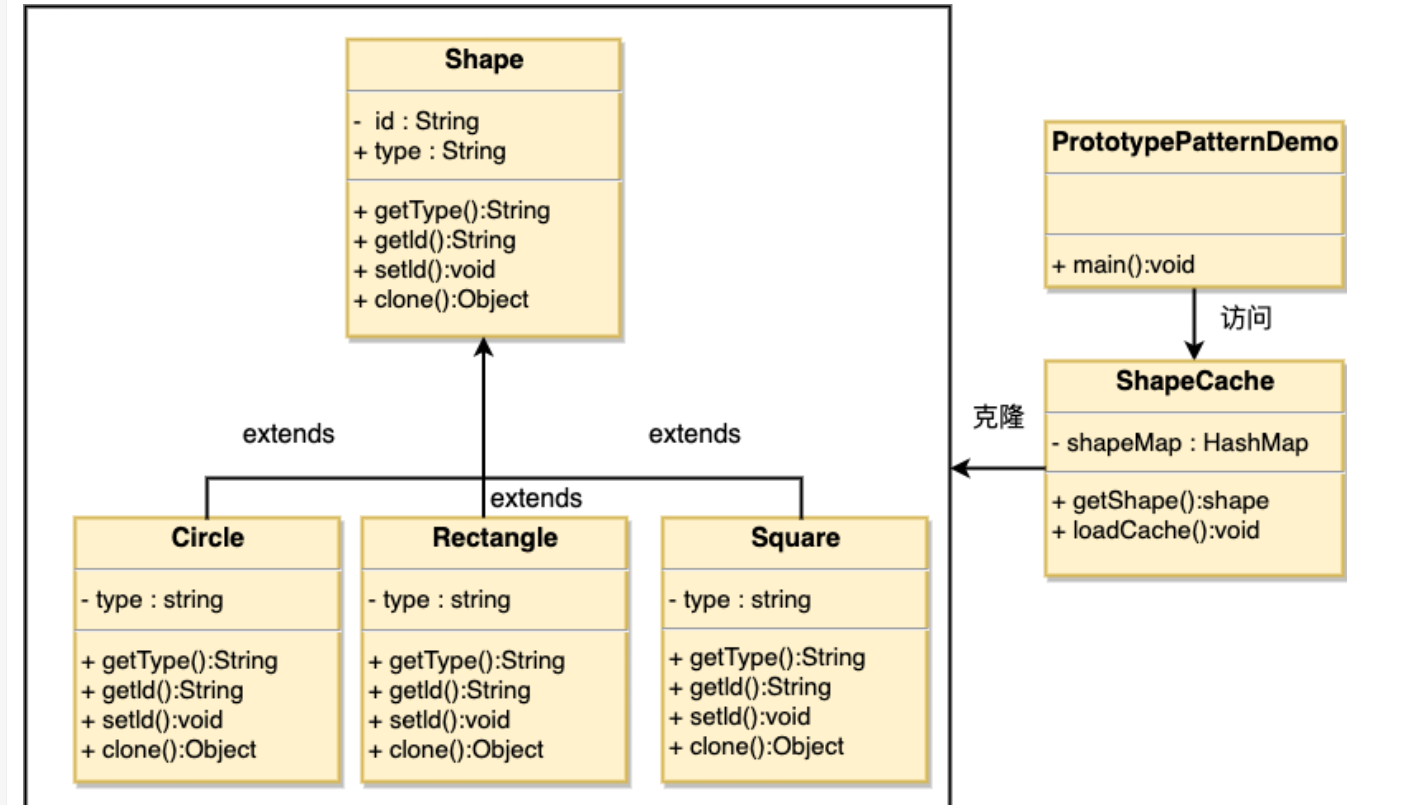
主要强调的是这个组装的顺序；

工厂模式强调的是 同一层次，如何用工厂的方式去创建;

建造者模式 主要关注的是

原型模式：

自己有一个类的缓存机制， 使用clone的方式 对外提供新的对象使用；用于创建重复的对象，同时又能保证性能



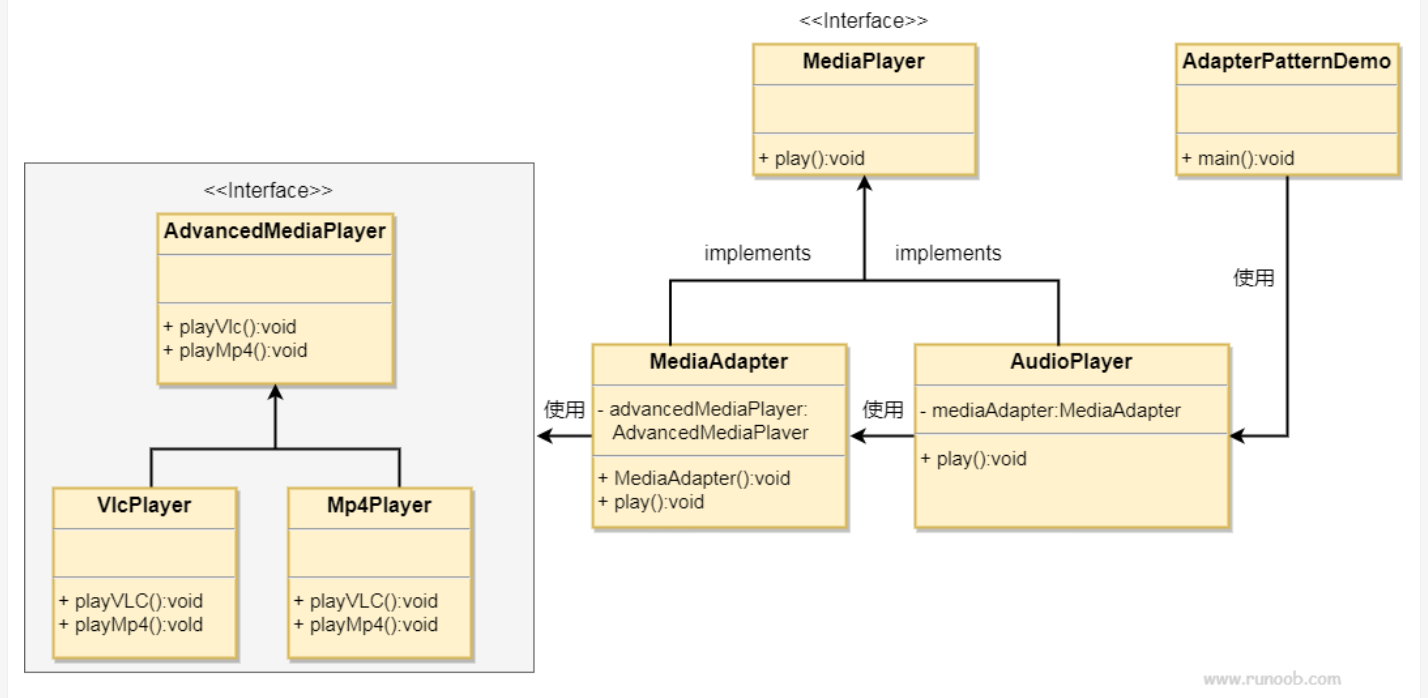
行为模式：

适配器模式：

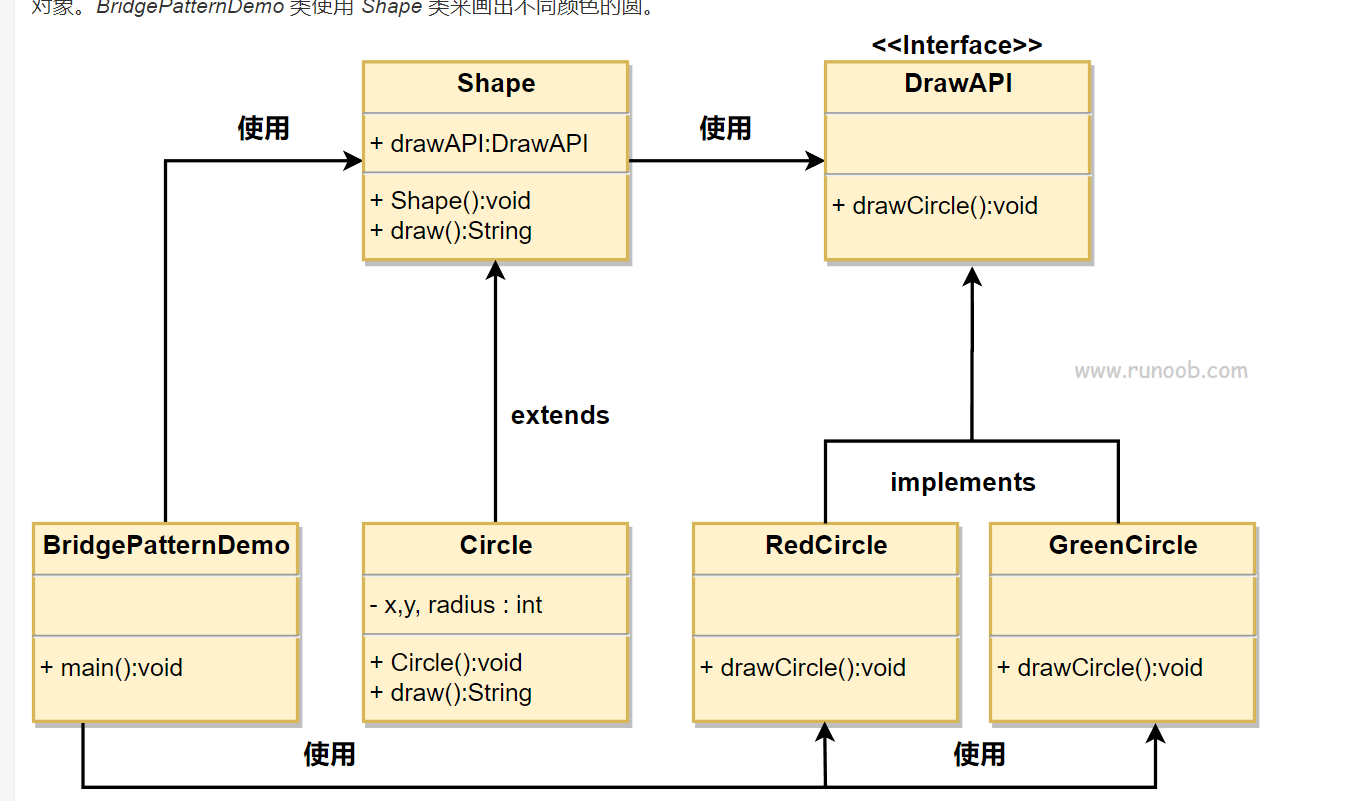
Adapter 实现了接口 当执行play的时候 会有新的对象来使用；

保证替代；

已有的对象 使用adapter 就可以了；



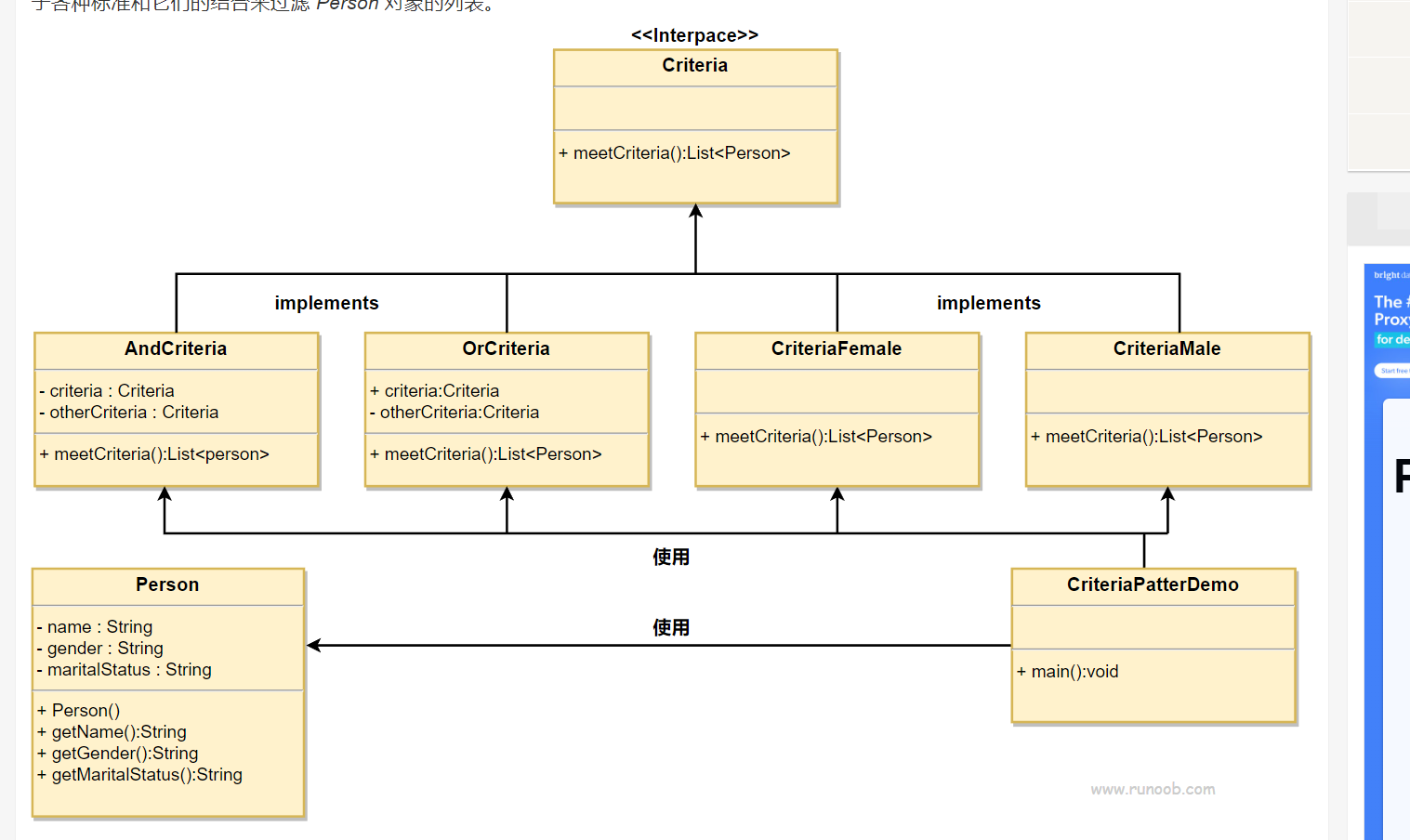
桥接模式



把对象的一个方法 使用对象的形式抽象出去

圆可以画出来， 用红笔画和 用黑笔画 都是一样的 和策略模式有什么区别？

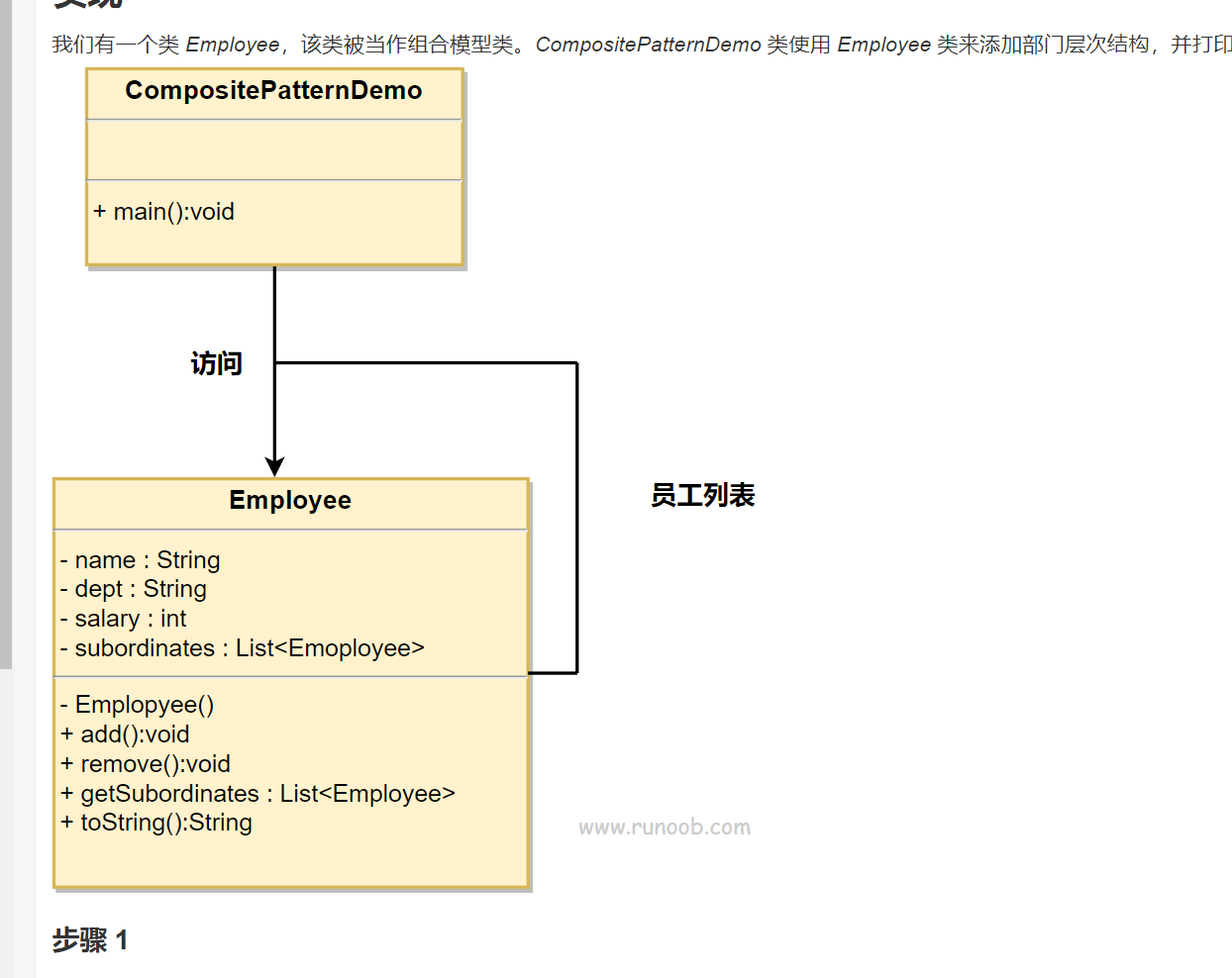
过滤器模式：



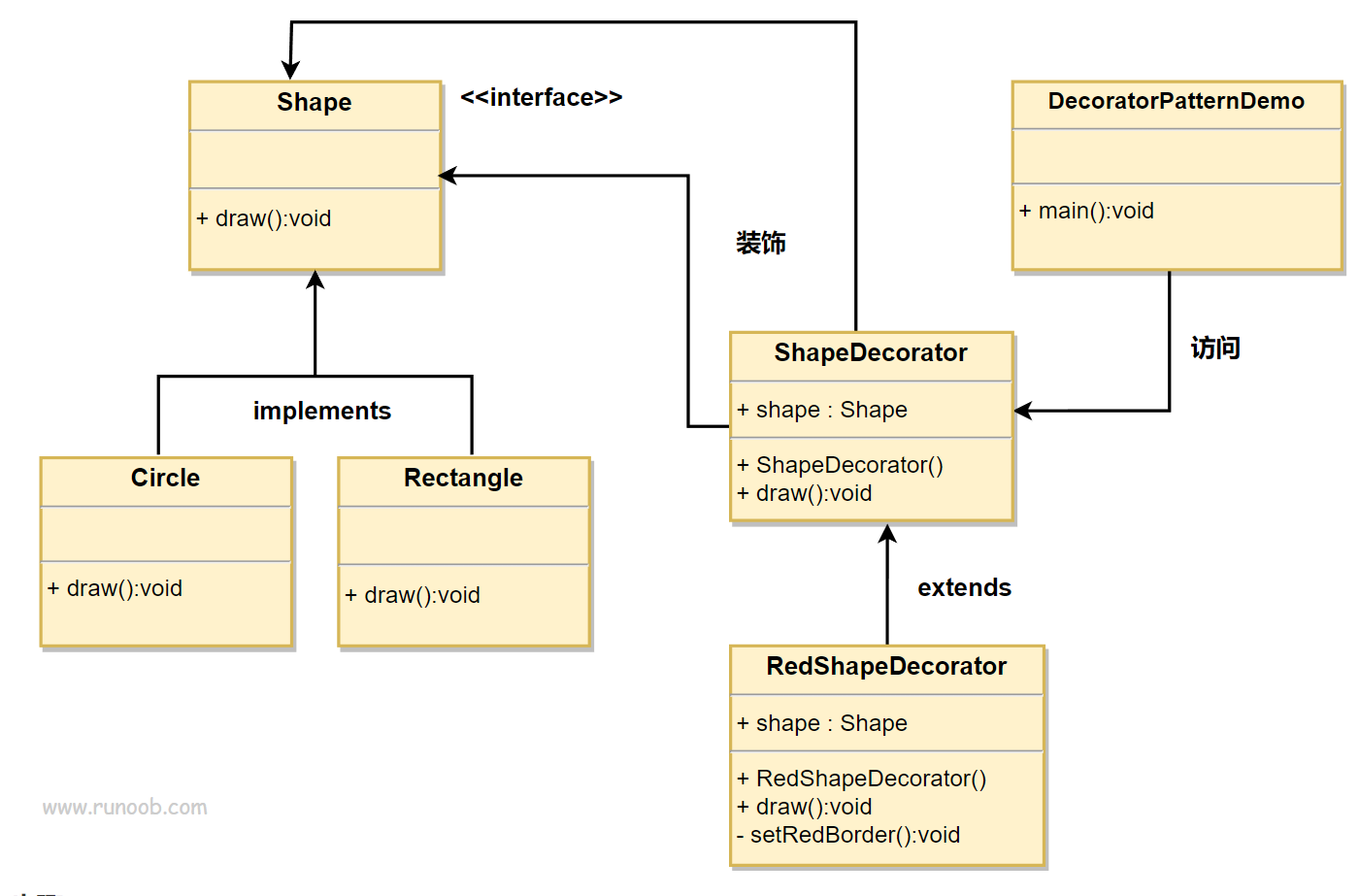
不同的类表示不同的过滤器； 执行不同的过滤逻辑；

组合模式：

简单的组装而已;



装饰器模式

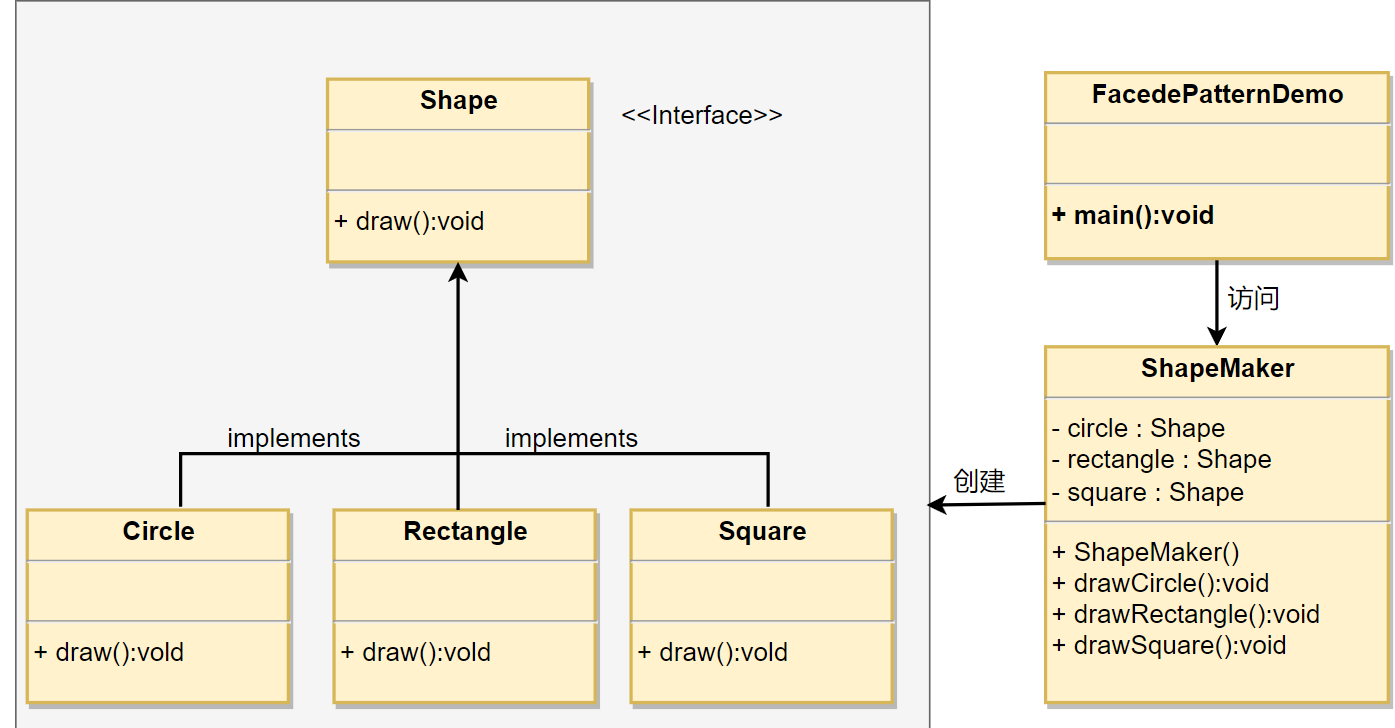


适配器模式 使用的是 原对象

装饰器 使用的是装饰器， 获得了原来的对象；

特别鲜明的在构造的时候 使用了被装饰的类；

外观模式：



就是维护了一个关心的对象的列表；