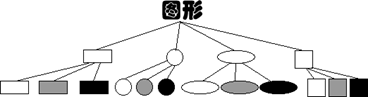
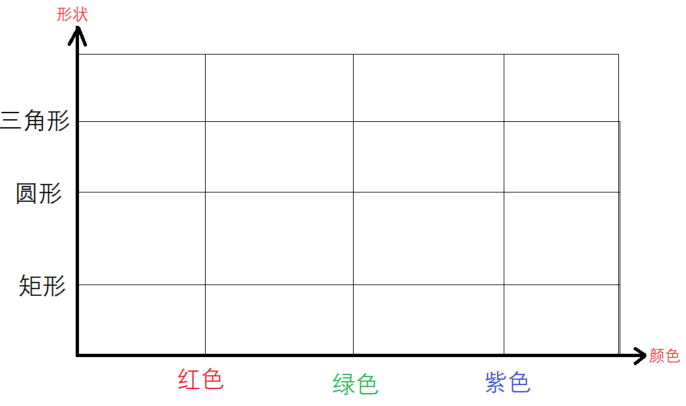
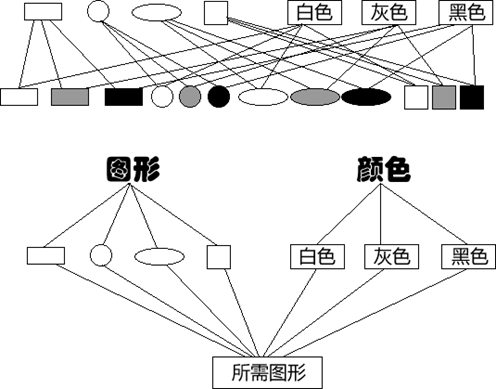
# 桥接模式

## 问题引入

现实生活中，大部分事物都有多种划分方式，以图形为例，可以按照形状划分，也可以按照颜色划分，我们称事物的一种划分方式为事物的一个变化维度。

事物在面向对象程序设计中以类的形式存在，如何对这些类进行组成？

1. 继承
2. 聚合

## 定义

桥接（Bridge）模式的定义如下：将抽象与实现分离，使它们可以独立变化。它是用组合关系代替继承关系来实现，从而降低了抽象和实现这两个可变维度的耦合度。

## 优点

桥接模式具有如下**优点**：

1. 抽象和实现的分离
2. 拓展性强
3. 具体实现细节多用户透明

## 缺点

桥接模式具有如下**缺点**：

1. 由于聚合关系建立在抽象层，要求开发者针对抽象化进行设计与编程，这增加了系统的理解与设计的难度。
2. 不易理解自然不易维护。

## 实现思路

**实现思路**：将抽象化部分和实现化部分分开，取消二者的继承关系，以组合方式替代。

## 结构

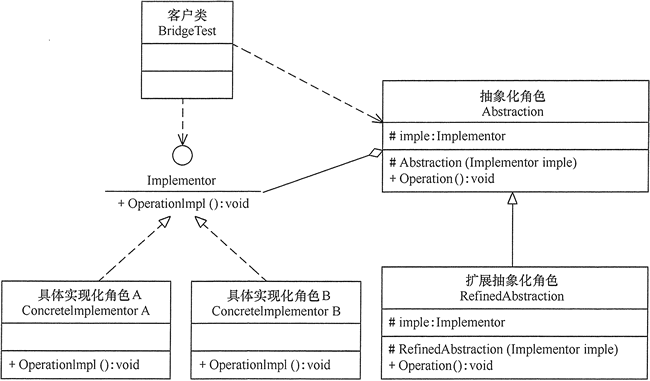
桥接（Bridge）模式包含以下主要角色。

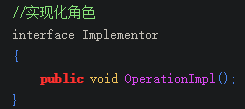
1）抽象化（Abstraction）角色：定义抽象类，并包含一个对实现化对象的引用。

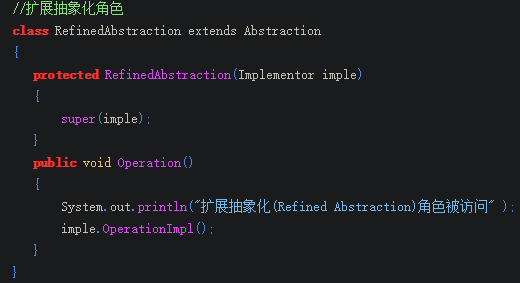
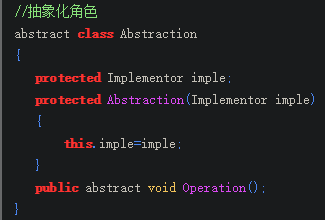
2）扩展抽象化（Refined Abstraction）角色：是抽象化角色的子类，实现父类中的业务方法，并通过组合关系调用实现化角色中的业务方法。

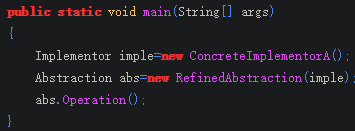
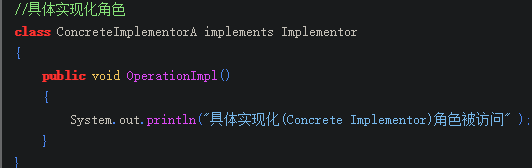
3）实现化（Implementor）角色：定义实现化角色的接口，供扩展抽象化角色调用。

4）具体实现化（Concrete Implementor）角色：给出实现化角色接口的具体实现。





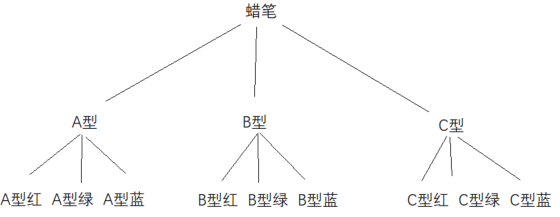
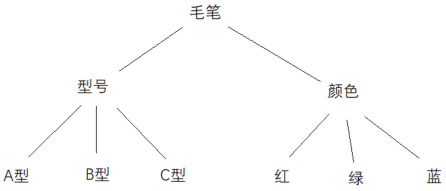




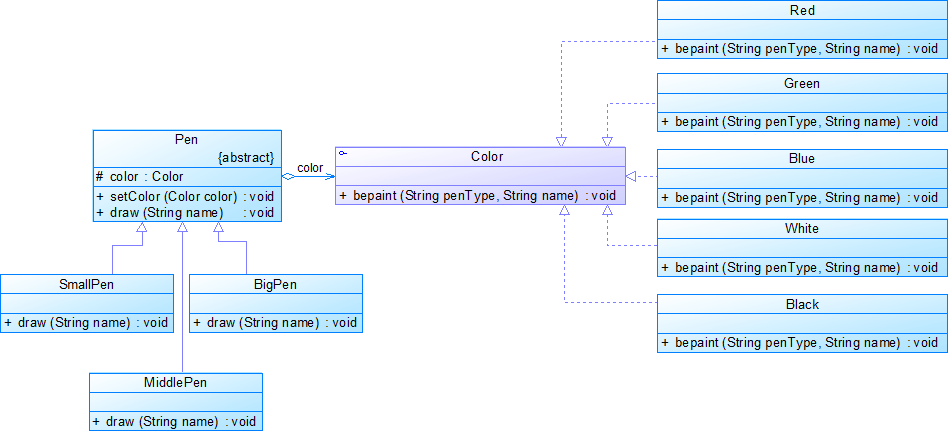
## 实例

### 蜡笔/毛笔



现需要提供大中小3种型号的画笔，能够绘制5种不同颜色，如果使用蜡笔，我们需要准备3\*5=15支蜡笔，也就是说必须准备15个具体的蜡笔类。而如果使用毛笔的话，只需要3种型号的毛笔，外加5个颜料盒，用3+5=8个类就可以实现15支蜡笔的功能。本实例使用桥接模式来模拟毛笔的使用过程。

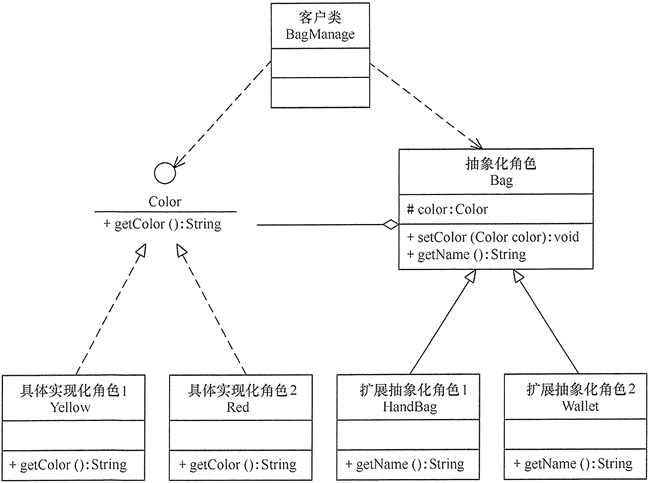


### 皮包选购

分析：女士皮包有很多种，可以按用途分、按皮质分、按品牌分、按颜色分、按大小分等，存在多个维度的变化，所以采用桥接模式来实现女士皮包的选购比较合适。

本实例按用途分可选钱包（Wallet）和挎包（HandBag），按颜色分可选黄色（Yellow）和红色（Red）。可以按两个维度定义为颜色类和包类。（点此下载本实例所要显示的包的图片）。

颜色类（Color）是一个维度，定义为实现化角色，它有两个具体实现化角色：黄色和红色，通过 getColor() 方法可以选择颜色；包类（Bag）是另一个维度，定义为抽象化角色，它有两个扩展抽象化角色：挎包和钱包，它包含了颜色类对象，通过 getName() 方法可以选择相关颜色的挎包和钱包。



## 应用场景

桥接模式通常适用于以下场景。

1）当一个类存在两个独立变化的维度，且这两个维度都需要进行扩展时。

2）当一个系统不希望使用继承或因为多层次继承导致系统类的个数急剧增加时。

3）当一个系统需要在构件的抽象化角色和具体化角色之间增加更多的灵活性时。

## 拓展

在软件开发中，有时桥接（Bridge）模式可与适配器模式联合使用。当桥接（Bridge）模式的实现化角色的接口与现有类的接口不一致时，可以在二者中间定义一个适配器将二者连接起来，其具体结构图如图：

