自我理解：**（感觉就是协议模式+组合模式）**

**只要符合协议，按照条件配置好整条链路的话，我最终实现就能确定好整个链路，一笔画完整**

1、场景：多维情况

【1】桥接模式的优点：

* 桥接模式使用聚合关系，解耦了抽象和实现之间固有的绑定关系，使得抽象和实现可以沿着各自的维度来变化。// 多维度，能够欧自行更具参数匹配
* 提高了系统的可扩展性，可以独立地对抽象部分和实现部分进行扩展。// 抽象部分就是整条链路，具体实现是最后那个绘制的动作
* 可减少子类的个数，这个在前面讲手机示例的时候进行分析了。// 自行匹配就好了，不用多个情况进行判断

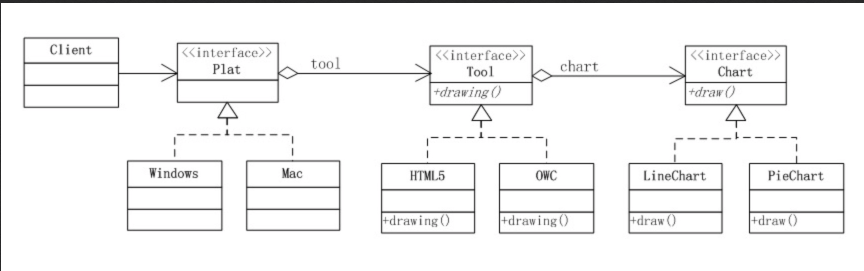
【2】桥接模式的缺点：

* 桥接模式的引入会增加系统的理解与设计难度，由于聚合关系建立在抽象层，要求开发者针对抽象进行设计与编程。// 考虑上需要谨慎仔细
* 桥接模式要求正确识别出系统中两个独立变化的维度，因此其使用范围具有一定的局限性。// 纬度变化，是否都能匹配，如果不能匹配，是不是就会有问题，这个出问题的环节如何取发现？？？？？（思考）

　通过优缺点的分析，我们可以在如下的情形下使用桥接模式：

* 不想在抽象与其实现之间形成固定的绑定关系；//绑定关系自己去搜索（配置好）
* 抽象及其实现都应可以通过子类化独立进行扩展；// 同个纬度，多个情况的话，多扩展一个类即可，遵守协议即可
* 对抽象的实现进行修改不应影响客户端代码；
* 如果每个实现需要额外的子类以细化抽象，则说明有必要把它们分成两（多）个部分；
* 想在带有不同抽象接口的多个对象之间共享一个实现 //最终是只有一个抽象实例去执行操作，前提是已经配置好整条链路了

　　总的来说，桥接模式的本质在于“**分离抽象和实现**”。



2、使用方式（这边先不考虑windows还是mac平台）

【1】这边不同的工具，可以绘制不同的形状，图形工具是最低级别，这边以它为第一维度；

【2】使用chart会画出线还是圆，通过实现协议去自行处理



【3】创建一个tool协议，定义一个Chart类型的属性，和一个绘图的方法，只要调属性去绘图，更具你使用的平台绘制出你想要的平台图形（二维数据：H5+线条、H5+圆型、OWC+线条、OWC+圆型）

【4】设置不同平台，都实现tool方法，这样就能使用tool在该平台上绘制不同的图形

【5】客户端调用：通过传入不同的平台，和绘制的图形，调用tool工具，画出想要的图形



【6】如果是三维的数据（增加操作系统的话）就对System平台进行区分（win 和 Mac），去实现不同的平台工具，再增加一个控制三维数据的类 System类（遵守Tool协议）

【7】控制流程：一维的图形，通过Chart（linChart、pieChart）来控制，二维的系那是页面，通过tool（H5Tool、OWCTool）来控制，如果是三维的平台，通过System（winSystem、MacSystem）来控制，再增加纬度，在桥接即可，能更具不同的参数：H5、LinChart、MacSystem来实现在mac系统下，H5页面上绘制一条线的形状