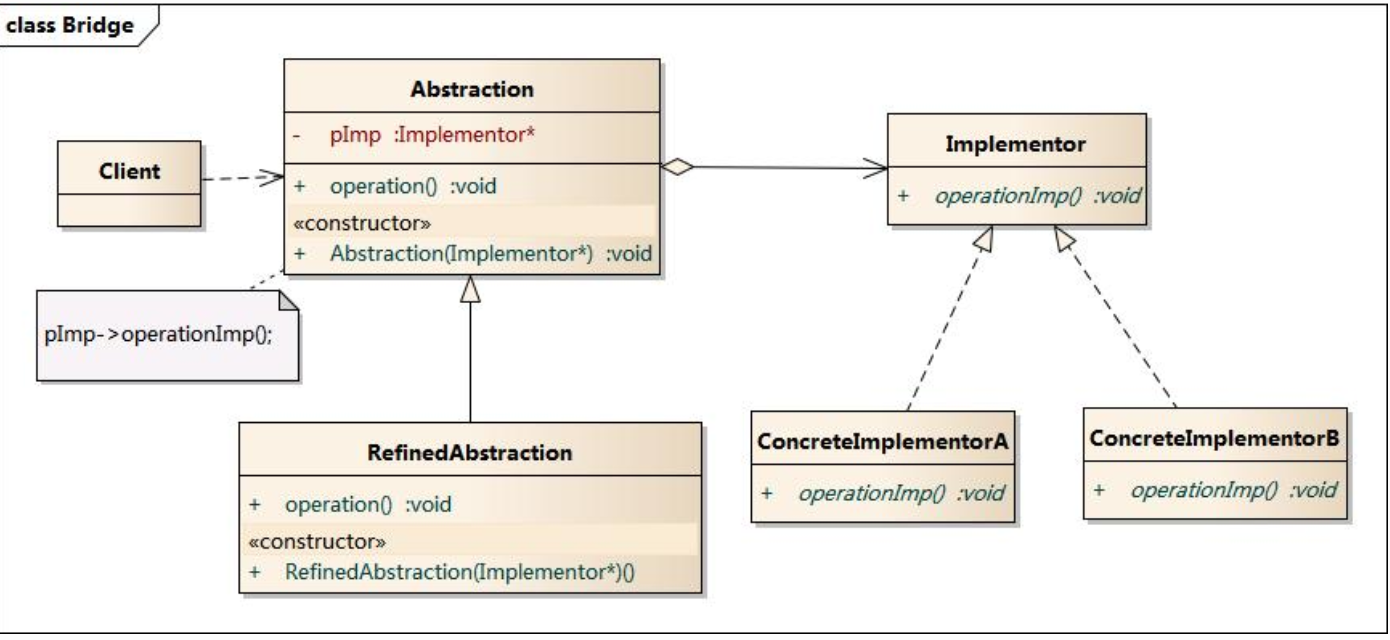
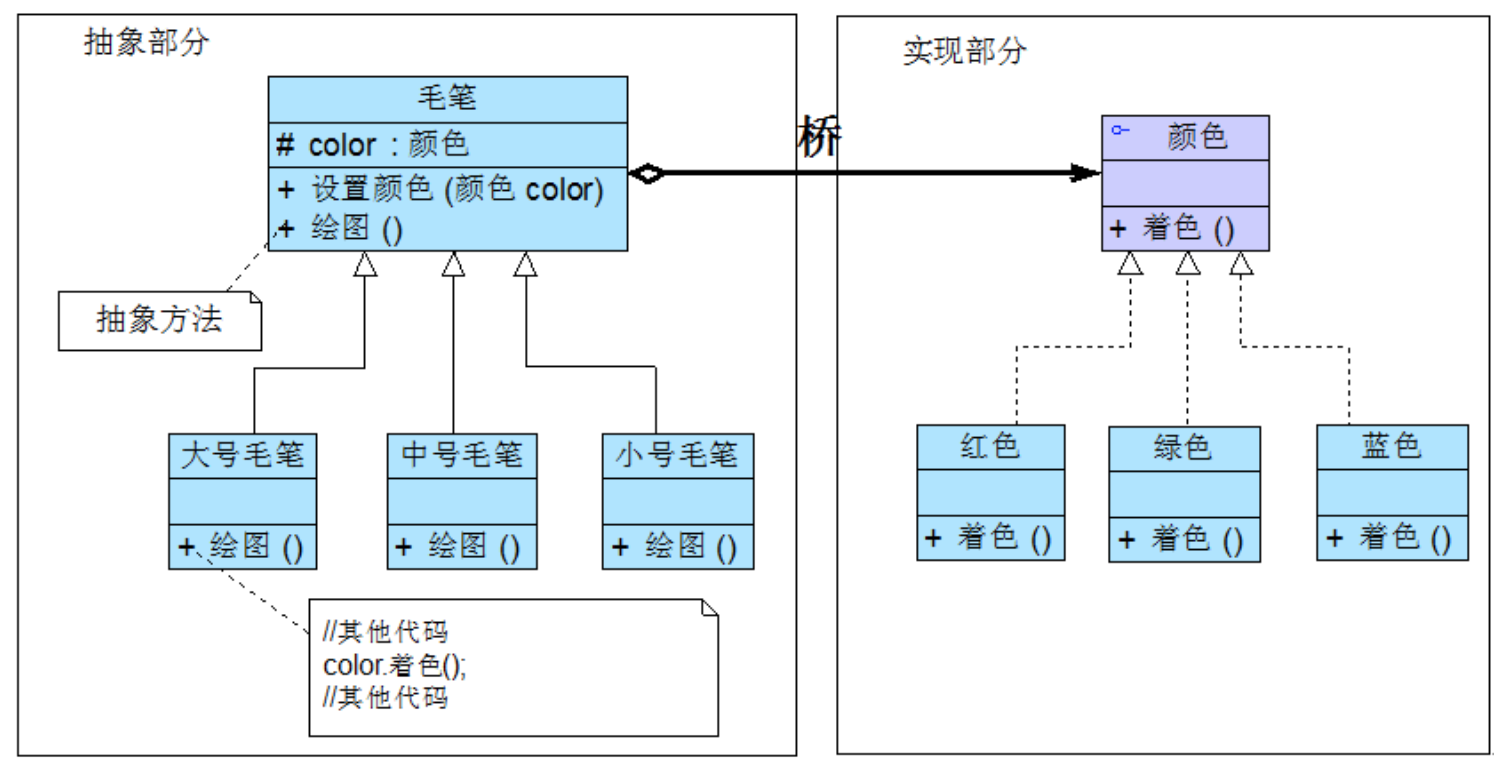
1. 桥接模式(Bridge Pattern)：将抽象与实现分离，放在两个不同的类层次中，使得两个层次可以独立改变，是一种结构型设计模式。如果软件系统中某个类存在多个独立变化的**维度**，通过该模式可以将这多个维度分离出来，使其可以独立扩展，让系统更加符合“单一职责原则”
2. 类图

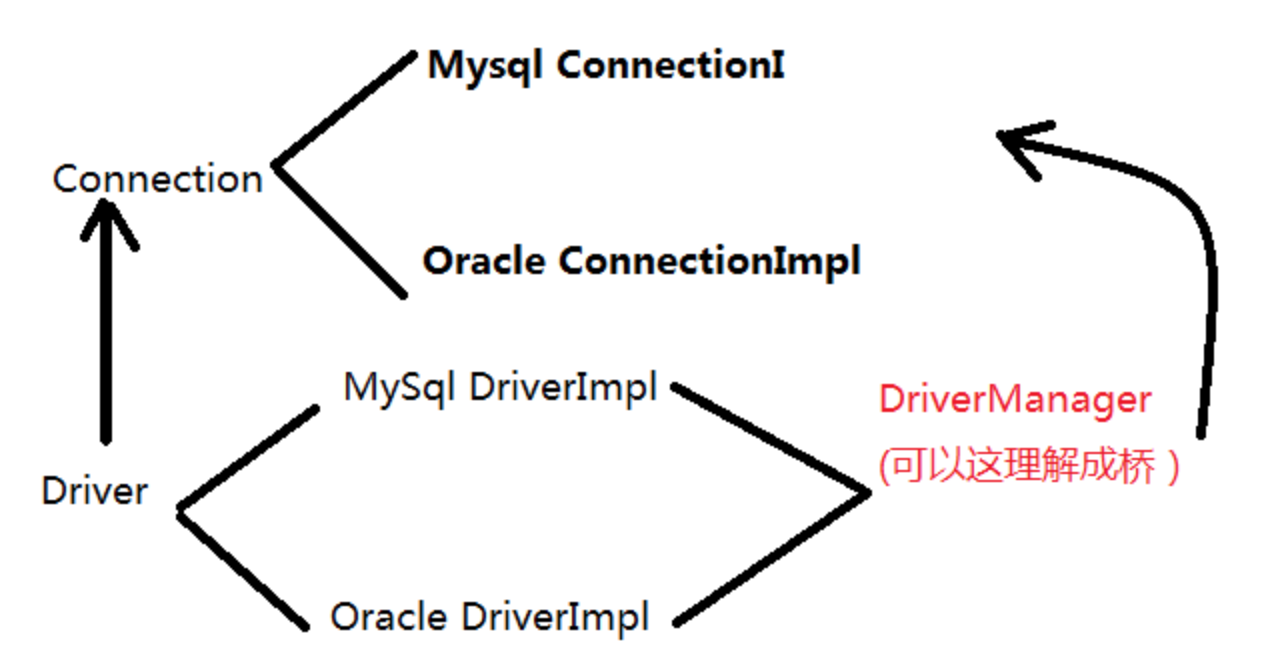


1. Abstraction: 抽象类，维护了Implementor，即它的实现类，二者是聚合关系，Abstraction充当桥接类
2. RefinedAbstraction：扩充抽象类
3. Implementor: 行为实现类接口
4. ConcreteImplementor: 行为具体实现类
5. 实例
   1. 对于毛笔而言，由于型号是其固有的维度，因此可以设计一个抽象的毛笔类，在该类中声明并部分实现毛笔的业务方法，而将各种型号的毛笔作为其子类；颜色是毛笔的另一个维度，由于它与毛笔之间存在一种“设置”的关系，因此我们可以提供一个抽象的颜色接口，而将具体的颜色作为实现该接口的子类。在此，型号可认为是毛笔的抽象部分，而颜色是毛笔的实现部分，结构示意图如图所示



* 1. 如果需要增加一种新型号的毛笔，只需扩展左侧的“抽象部分”，增加一个新的扩充抽象类；如果需要增加一种新的颜色，只需扩展右侧的“实现部分”，增加一个新的具体实现类。扩展非常方便，无须修改已有代码，且不会导致类的数目增长过快

1. 桥接模式应用
   1. JDBC DriverManager



* 1. 银行转账系统
     1. 转账分类: 网上转账、柜台转账、ATM转账
     2. 转账用户分类: 普通用户、银卡用户、金卡用户
  2. 消息管理
     1. 消息类型: 即时消息、延时消息
     2. 消息分类: 手机短信、邮件消息、QQ消息

1. 桥接模式注意事项/细节
   1. Bridge 模式基于类的最小设计原则，通过使用封装、聚合及继承等行为让不同的类承担不同的职责。它的主要特点是把抽象(Abstraction)与行为实现(Implementation)分离开来，从而可以保持各部分的独立性以及应对他们的功能扩展
   2. 桥接模式用一种巧妙的方式处理多层继承存在的问题，用抽象关联取代了传统的多层继承，将类之间的静态继承关系转换为动态的对象组合关系，使得系统更加灵活，并易于扩展，同时有效控制了系统中类的个数
   3. 桥接模式替代多层继承方案，可以减少子类的个数，避免”**类爆炸**”，降低系统的管理和维护成本
   4. 桥接模式要求正确识别出系统中两个独立变化的维度(抽象、和实现)，因此其使用范围有一定的局限性，即需要有这样的应用场景
   5. 桥接模式的引入增加了系统的理解和设计难度，由于聚合关联关系建立在抽象层，要求开发者针对抽象进行设计和编程