# **一、 模式定义**

      桥接模式即将抽象部分与它的实现部分分离开来，使他们都可以独立变化。

      桥接模式将继承关系转化成关联关系，它降低了类与类之间的耦合度，减少了系统中类的数量，也减少了代码量。

      将抽象部分与他的实现部分分离这句话不是很好理解，其实这并不是将抽象类与他的派生类分离，而是抽象类和它的派生类用来实现自己的对象。这样还是不能理解的话。我们就先来认清什么是抽象化，什么是实现化，什么是脱耦。

      抽象化：其概念是将复杂物体的一个或几个特性抽出去而只注意其他特性的行动或过程。在面向对象就是将对象共同的性质抽取出去而形成类的过程。

      实现化：针对抽象化给出的具体实现。它和抽象化是一个互逆的过程，实现化是对抽象化事物的进一步具体化。

      脱耦：脱耦就是将抽象化和实现化之间的耦合解脱开，或者说是将它们之间的强关联改换成弱关联，将两个角色之间的继承关系改为关联关系。

      对于那句话：将抽象部分与他的实现部分分离套用《大话设计模式》里面的就是实现系统可能有多个角度分类，每一种角度都可能变化，那么把这种多角度分类给分离出来让他们独立变化，减少他们之间耦合。

      桥接模式中的所谓脱耦，就是指在一个软件系统的抽象化和实现化之间使用关联关系（组合或者聚合关系）而不是继承关系，从而使两者可以相对独立地变化，这就是桥接模式的用意。

# **二、 模式结构**

      桥接模式主要包含如下几个角色：

      Abstraction：抽象类。   
      RefinedAbstraction：扩充抽象类。   
      Implementor：实现类接口。   
      ConcreteImplementor：具体实现类 。

# **三、 模式优缺点**

## **优点**

      1、分离抽象接口及其实现部分。提高了比继承更好的解决方案。

      2、桥接模式提高了系统的可扩充性，在两个变化维度中任意扩展一个维度，都不需要修改原有系统。

      3、实现细节对客户透明，可以对用户隐藏实现细节。

## **缺点**

      1、桥接模式的引入会增加系统的理解与设计难度，由于聚合关联关系建立在抽象层，要求开发者针对抽象进行设计与编程。   
      2、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。

# **四、 模式使用场景**

       1、如果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，避免在两个层次之间建立静态的继承联系，通过桥接模式可以使它们在抽象层建立一个关联关系。

       2、对于那些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的系统，桥接模式尤为适用。

       3、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。

# **五、 模式总结**

      1、桥接模式实现了抽象化与实现化的脱耦。他们两个互相独立，不会影响到对方。

      2、对于两个独立变化的维度，使用桥接模式再适合不过了。

      3、对于“具体的抽象类”所做的改变，是不会影响到客户。