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。

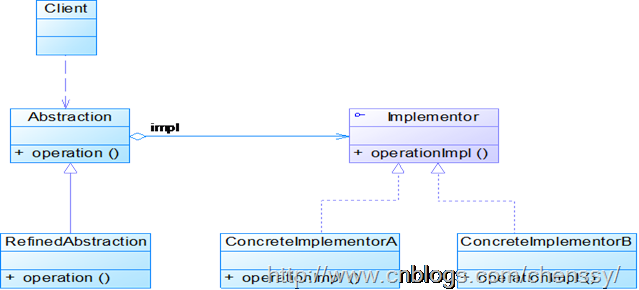
      脱耦：**脱耦就是将抽象化和实现化之间的耦合解脱开**，或者说是将它们之间的强关联改换成弱关联，将两个角色之间的继承关系改为关联关系。

      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**

      下图是桥接模式的UML结构图：

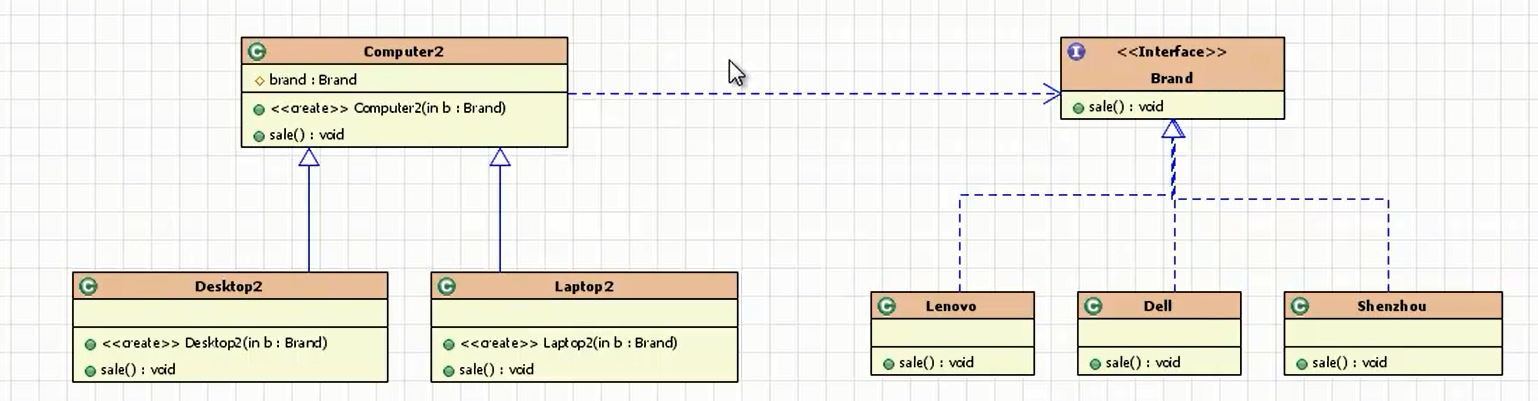
[](https://images0.cnblogs.com/blog/381060/201309/12212906-fb3f5780d2384acdbb60ec07fc3c71a1.png)

      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、 模式实现**

      模式场景我们就采用哪个画图的。其UML结构图如下



**package** bridge;  
  
*/\*\*  
 \* 品牌  
 \*/***public interface** Brand {  
 **void** sale();  
}

**package** bridge;  
  
**public class** AppleBrand **implements** Brand {  
  
 @Override  
 **public void** sale() {  
 System.***out***.print(**"苹果品牌"**);  
 }  
}

**package** bridge;  
  
**public class** ASUSBrand **implements** Brand {  
 @Override  
 **public void** sale() {  
 System.***out***.print(**"华硕品牌"**);  
 }  
}

**package** bridge;  
  
**public class** DellBrand **implements** Brand {  
 @Override  
 **public void** sale() {  
 System.***out***.print(**"戴尔品牌"**);  
 }  
}

**package** bridge;  
  
**public class** Computer {  
 **private** Brand **brand**;  
  
 **public** Computer(Brand brand) {  
 **this**.**brand** = brand;  
 }  
  
 **public void** sale() {  
 **brand**.sale();  
 }  
  
}

**package** bridge;  
  
**public class** BookComputer **extends** Computer {  
  
 **public** BookComputer(Brand brand) {  
 **super**(brand);  
 }  
  
 @Override  
 **public void** sale() {  
 **super**.sale();  
 System.***out***.println(**"笔记本在销售"**);  
 }  
}

**package** bridge;  
  
**public class** DeskComputer **extends** Computer {  
  
 **public** DeskComputer(Brand brand) {  
 **super**(brand);  
 }  
  
 @Override  
 **public void** sale() {  
 **super**.sale();  
 System.***out***.println(**"台式机在销售"**);  
  
 }  
}

**package** bridge;  
  
**public class** Test {  
 **public static void** main(String[] args) {  
 Computer AsusDeskComputer = **new** DeskComputer(**new** ASUSBrand());  
 AsusDeskComputer.sale();  
  
 Computer DellBookComputer = **new** BookComputer(**new** DellBrand());  
 DellBookComputer.sale();  
  
 }  
}

结果：

华硕品牌台式机在销售

戴尔品牌笔记本在销售