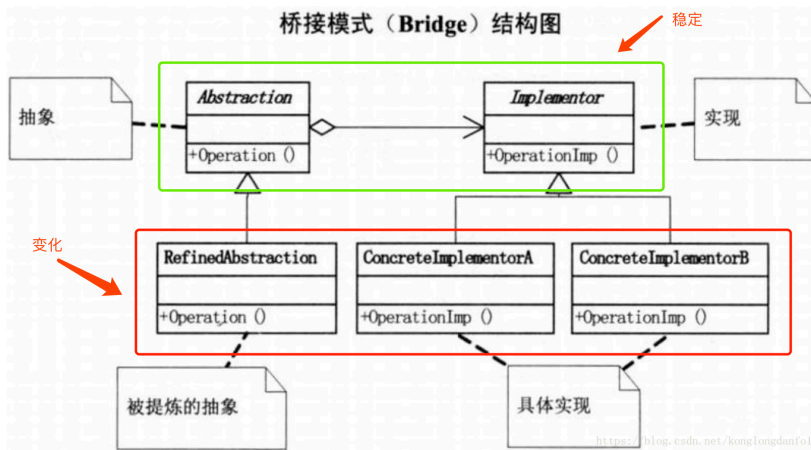
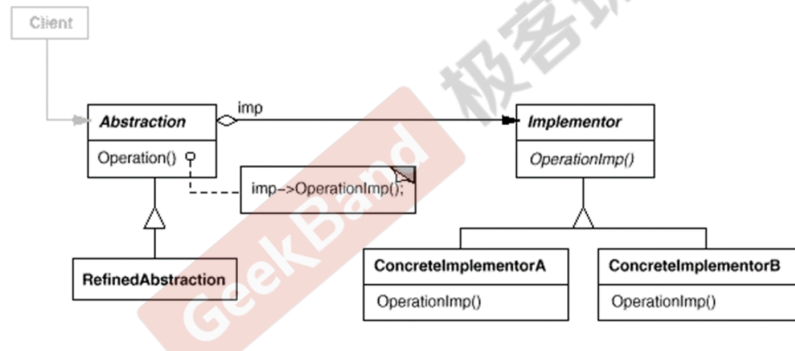


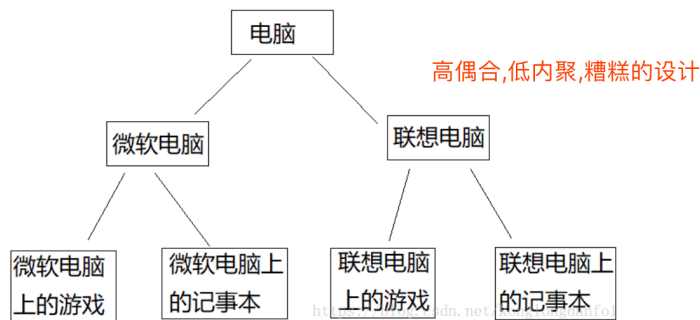
定义:将抽象部分(业务功能)与实现部分(平台实现)分离,使它们都可以独立地变化。——《设计模式》GoF

动机:由于某些类型的固有的实现逻辑,使得它们具有两个变化的维度,乃至多个纬度的变化。如何应对这种“多维度的变化”?如何利用面向对象技术来使得

类型可以轻松地沿着两个乃至多个方向变化,而不引入额外的复杂度?

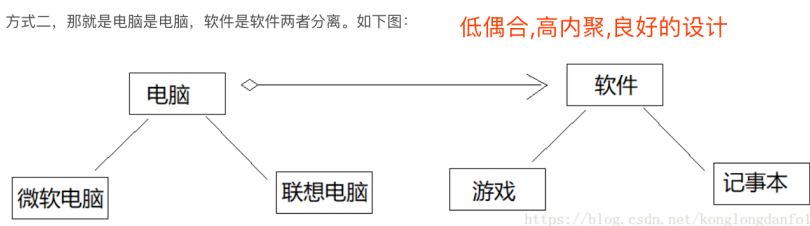


方式一：



这种方式耦合性太高，比如像添加一种软件，需要添加微软电脑的软件和联想电脑的软件。同样的像添加一种电脑，也要添加相应的软件。如果要修改一种软件，则要修改很多软件。

方式二，那就是电脑是电脑，软件是软件两者分离。如下图：



很明显方式二耦合性底，容易维护和扩展。这里电脑就相当于抽线，而软件则相当于实现。总的来说就是桥接模式主要是实现系统可能有多个角度分类，每一种分类都有可能变化。那么就把这种多角度分离出来让它们独立变化，减少它们之间的耦合。

常用的场景

1. 当一个对象有多个变化因素的时候，考虑依赖于抽象的实现，而不是具体的实现。
2. 当多个变化因素在多个对象间共享时，考虑将这部分变化的部分抽象出来再聚合/合成进来。
3. 当我们考虑一个对象的多个变化因素可以动态变化的时候，考虑使用桥接模式。

优点

1. 将实现抽离出来，再实现抽象，使得对象的具体实现依赖于抽象，满足了依赖倒转原则。
2. 将可以共享的变化部分，抽离出来，减少了代码的重复信息。
3. 对象的具体实现可以更加灵活，可以满足多个因素变化的要求。

缺点

1. 客户必须知道选择哪一种类型的实现。

要点总结: Bridge模式使用“对象间的组合关系”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。所谓抽象和实现沿着各自纬度的变化，即“子类化”它们。Bridge模式有时候类似于多继承方案，但是多继承方案往往违背单一职责原则（即一个类只有一个变化的原因），复用性比较差。Bridge模式是比多继承方案更好的解决方法。Bridge模式的应用一般在“两个非常强的变化维度”，有时一个类也有多于两个的变化维度，这时可以使用Bridge的扩展模式。

```
1 #include <iostream>
2 using namespace std;
3 class Software{
4 public:
5     virtual void Run() = 0;
6     virtual ~Software(){}
7 };
8 class Computer{
9 protected:
10     Software *m_pSoft;
11 public:
12     Computer() {
13         m_pSoft = NULL;
14     }
15     virtual void Run() = 0;
```

```

16     virtual ~Computer(){
17         if (NULL != m_pSoft)
18         {
19             delete m_pSoft;
20             m_pSoft = NULL;
21         }
22     }
23     void SetSoftware(Software *pSoft) {
24         m_pSoft = pSoft;
25     }
26 };
27 class Game : public Software{
28 public:
29     void Run() { cout << "运行游戏!" << endl; }
30 };
31 class Office : public Software{
32 public:
33     void Run() { cout << "运行办公软件!" << endl; }
34 };
35 class Lenovo : public Computer{
36 public:
37     void Run() {
38         cout << "Lenovo 电脑:";
39         m_pSoft->Run();
40     }
41 };
42 class Apple : public Computer{
43 public:
44     void Run() {
45         cout << "Apple Macbook Pro:";
46         m_pSoft->Run();
47     }
48 };
49 int main()
50 {
51     Computer *pC = new Lenovo();
52     pC->SetSoftware(new Game());
53     pC->Run();
54     delete pC;
55     pC = NULL;
56     pC = new Apple();
57     pC->SetSoftware(new Office());
58     pC->Run();
59     delete pC;
60     pC = NULL;
61     return 0;
62 }

```

```
192:DesignPattnsStudy weishichun$ ls Bridge
Bridge_1.cpp      Bridge桥模式.pdf
192:DesignPattnsStudy weishichun$ g++ -o Bridge.out Bridge_1.cpp
192:DesignPattnsStudy weishichun$ ./Bridge.out
Lenovo 电脑:运行游戏!
Apple Macbook Pro:运行办公软件!
192:DesignPattnsStudy weishichun$ █
```