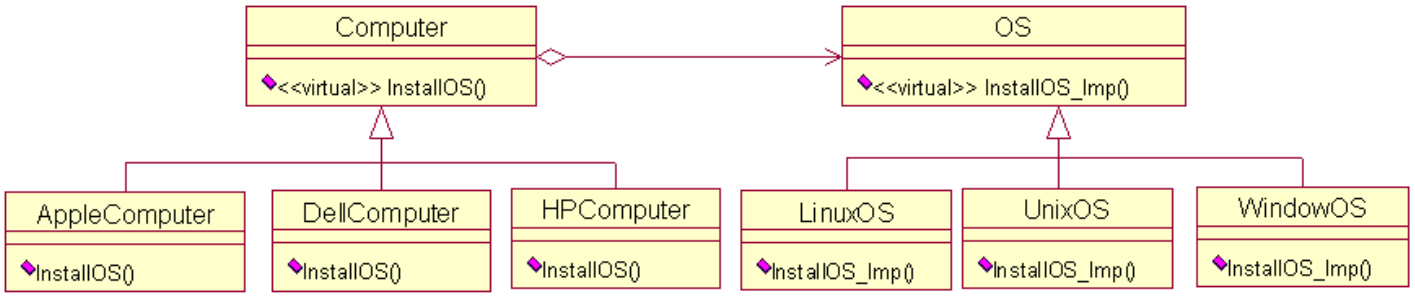
設計模式C++實現（10）——橋接模式

星期六, 2013 12月 14, 1:07 上午

軟件領域中的設計模式為開發人員提供了一種使用專家設計經驗的有效途徑。設計模式中運用了面向對象編程語言的重要特性：封裝、繼承、多態，真正領悟設計模式的精髓是可能一個漫長的過程，需要大量實踐經驗的積累。最近看設計模式的書，對於每個模式，用C++寫了個小例子，加深一下理解。主要參考《大話設計模式》和《設計模式:可復用面向對象軟件的基礎》兩本書。本文介紹橋接模式的實現。

[DP]書上定義：將抽象部分與它的實現部分分離，使它們都可以獨立地變化。考慮裝操作系統，有多種配置的計算機，同樣也有多款操作系統。如何運用橋接模式呢？可以將操作系統和計算機分別抽象出來，讓它們各自發展，減少它們的耦合度。當然了，兩者之間有標準的接口。這樣設計，不論是對於計算機，還是操作系統都是非常有利的。下面給出這種設計的UML圖，其實就是橋接模式的UML圖。



給出C++的一種實現：

```
//操作系統
class OS
{
public:
    virtual void InstallOS_Imp() {}
};
class WindowOS: public OS
{
public:
    void InstallOS_Imp() { cout<<"安裝Window操作系統"<<endl; }
};
class LinuxOS: public OS
{
public:
    void InstallOS_Imp() { cout<<"安裝Linux操作系統"<<endl; }
};
class UnixOS: public OS
{
public:
    void InstallOS_Imp() { cout<<"安裝Unix操作系統"<<endl; }
};
//計算機
class Computer
{
public:
    virtual void InstallOS(OS *os) {}
};
class DellComputer: public Computer
{
public:
    void InstallOS(OS *os) { os->InstallOS_Imp(); }
};
class AppleComputer: public Computer
{
public:
    void InstallOS(OS *os) { os->InstallOS_Imp(); }
};
class HPComputer: public Computer
{
public:
    void InstallOS(OS *os) { os->InstallOS_Imp(); }
};
```

客戶使用方式：

```
1. int main()
2. {
3.     OS *os1 = new WindowOS();
4.     OS *os2 = new LinuxOS();
5.     Computer *computer1 = new AppleComputer();
6.     computer1->InstallOS(os1);
7.     computer1->InstallOS(os2);
8. }
```