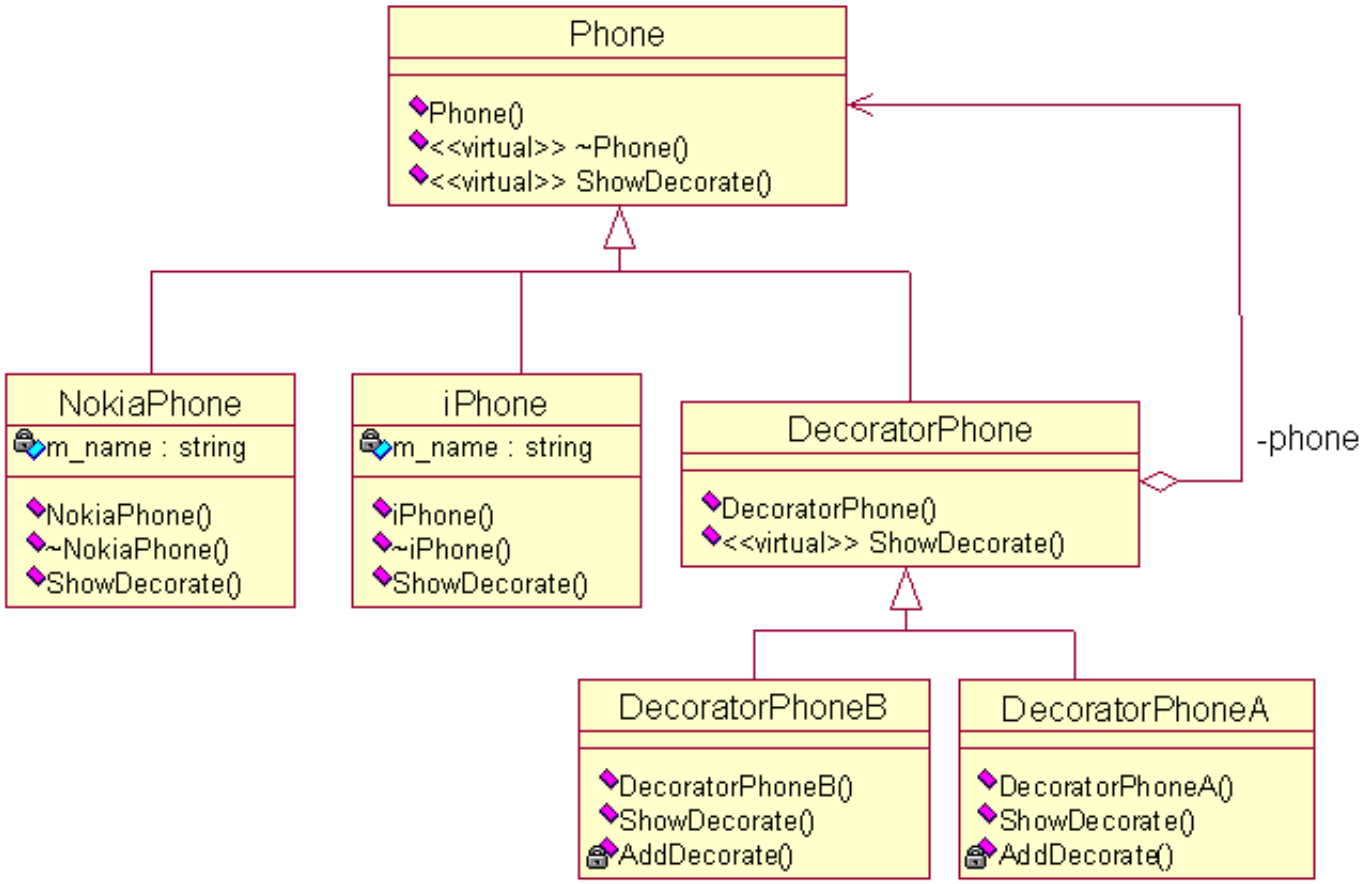
設計模式C++實現（11）——裝飾模式

星期六, 2013 12月 14, 1:08 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。本文介紹裝飾模式的實現。

裝飾模式：動態地給一個對象添加一些額外的職責。就增加功能來說，裝飾模式相比生成子類更為靈活。有時我們希望給某個對象而不是整個類添加一些功能。比如有一個手機，允許你為手機添加特性，比如增加掛件、屏幕貼膜等。一種靈活的設計方式是，將手機嵌入到另一對象中，由這個對象完成特性的添加，我們稱這個嵌入的對象為裝飾。這個裝飾與它所裝飾的組件接口一致，因此它對使用該組件的客戶透明。下面給出裝飾模式的UML圖。



在這種設計中，手機的裝飾功能被獨立出來，可以單獨發展，進而簡化了具體手機類的設計。下面給出Phone類的實現：

```
1. //公共抽象類
2. class Phone
3. {
4. public:
5.     Phone() {}
6.     virtual ~Phone() {}
7.     virtual void ShowDecorate() {}
```

```
};
```

具體的手機類的定義：

```
//具體的手機類
class iPhone : public Phone
{
private:
    string m_name; //手機名稱
public:
    iPhone(string name): m_name(name){}
    ~iPhone() {}
    void ShowDecorate() { cout<<m_name<<"的裝飾"<<endl;}
};
//具體的手機類
class NokiaPhone : public Phone
{
private:
    string m_name;
public:
    NokiaPhone(string name): m_name(name){}
    ~NokiaPhone() {}
    void ShowDecorate() { cout<<m_name<<"的裝飾"<<endl;}
};
```

裝飾類的實現：

```
//裝飾類
class DecoratorPhone : public Phone
{
private:
    Phone *m_phone;  //要裝飾的手機
public:
    DecoratorPhone(Phone *phone): m_phone(phone) {}
    virtual void ShowDecorate() { m_phone->ShowDecorate(); }
};
//具體的裝飾類
class DecoratorPhoneA : public DecoratorPhone
{
public:
    DecoratorPhoneA(Phone *phone) : DecoratorPhone(phone) {}
    void ShowDecorate() { DecoratorPhone::ShowDecorate(); AddDecorate(); }
private:
    void AddDecorate() { cout<<"增加掛件"<<endl; } //增加的裝飾
};
//具體的裝飾類
class DecoratorPhoneB : public DecoratorPhone
{
public:
    DecoratorPhoneB(Phone *phone) : DecoratorPhone(phone) {}
    void ShowDecorate() { DecoratorPhone::ShowDecorate(); AddDecorate(); }
private:
```

```
26.     void AddDecorate() { cout<<"屏幕貼膜"<<endl; } //增加的装飾
27. };
```

客戶使用方式：

```
1. int main()
2. {
3.     Phone *iphone = new NokiaPhone("6300");
4.     Phone *dpa = new DecoratorPhoneA(iphone); //裝飾，增加掛件
5.     Phone *dpb = new DecoratorPhoneB(dpa);    //裝飾，屏幕貼膜
6.     dpb->ShowDecorate();
7.     delete dpa;
8.     delete dpb;
9.     delete iphone;
10.    return 0;
11. }
```

裝飾模式提供了更加靈活的向對象添加職責的方式。可以用添加和分離的方法，用裝飾在運行時刻增加和刪除職責。裝飾模式提供了一種「即用即付」的方
法來添加職責。它並不試圖在一個複雜的可定製的類中支持所有可預見的特徵，相反，你可以定義一個簡單的類，並且用裝飾類給它逐漸地添加功能。可以從簡單的部件組合出複雜的功能。[DP]

在本文的例子中，我們定義了兩個具體的手機類，iPhone類和NokiaPhone類，通過單獨的裝飾類為它們添加特性，從而組合出複雜的功能。