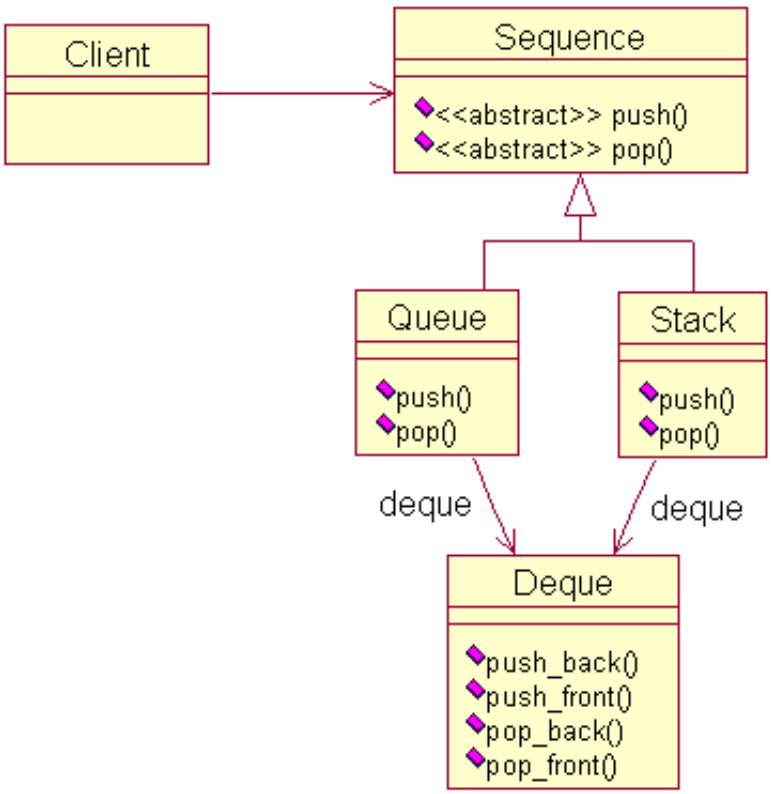
設計模式C++實現（３）——適配器模式

星期六, 2013 12月 14, 1:00 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》（DP）兩本書。本文介紹適配器模式的實現。

DP上的定義：適配器模式將一個類的接口轉換成客戶希望的另外一個接口，使得原本由於接口不兼容而不能一起工作的那些類可以一起工作。它包括類適配器和對象適配器，本文針對的是對象適配器。舉個例子，在STL中就用到了適配器模式。STL實現了一種數據結構，稱為雙端隊列（deque），支持前後兩段的插入與刪除。STL實現棧和隊列時，沒有從頭開始定義它們，而是直接使用雙端隊列實現的。這裡雙端隊列就扮演了適配器的角色。隊列用到了它的後端插入，前端刪除。而棧用到了它的後端插入，後端刪除。假設棧和隊列都是一種順序容器，有兩種操作：壓入和彈出。下面給出相應的UML圖，與DP上的圖差不多。



根據上面的UML圖，很容易給出實現。

```
1. //雙端隊列
2. class Deque
3. {
4. public:
5.     void push_back(int x) { cout<<"Deque push_back"<<endl; }
6.     void push_front(int x) { cout<<"Deque push_front"<<endl; }
7.     void pop_back() { cout<<"Deque pop_back"<<endl; }
8.     void pop_front() { cout<<"Deque pop_front"<<endl; }
9. };
```

```
10. //順序容器
11. class Sequence
12. {
13. public:
14.     virtual void push(int x) = 0;
15.     virtual void pop() = 0;
16. };
17. //棧
18. class Stack: public Sequence
19. {
20. public:
21.     void push(int x) { deque.push_back(x); }
22.     void pop() { deque.pop_back(); }
23. private:
24.     Deque deque; //雙端隊列
25. };
26. //隊列
27. class Queue: public Sequence
28. {
29. public:
30.     void push(int x) { deque.push_back(x); }
31.     void pop() { deque.pop_front(); }
32. private:
33.     Deque deque; //雙端隊列
34. };
```

使用方式如下：

```
1. int main()
2. {
3.     Sequence *s1 = new Stack();
4.     Sequence *s2 = new Queue();
5.     s1->push(1); s1->pop();
6.     s2->push(1); s2->pop();
7.     delete s1; delete s2;
8.     return 0;
9. }
```