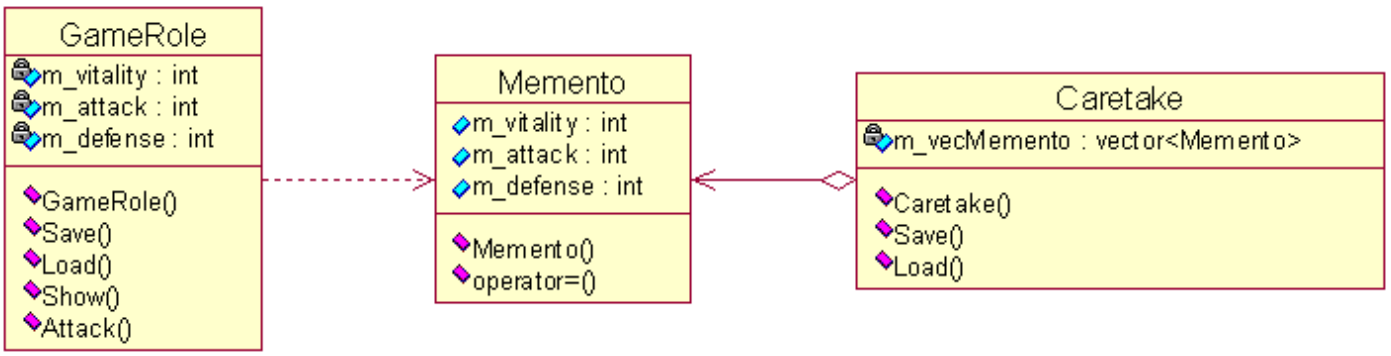
設計模式C++實現（12）——備忘錄模式

星期六, 2013 12月 14, 1:09 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。本文介紹備忘錄模式的實現。

備忘錄模式：在不破壞封裝性的前提下，捕獲一個對象的內部狀態，並在該對象之外保存這個狀態。這樣以後就可將該對象恢復到原先保存的狀態[DP]。舉個簡單的例子，我們玩遊戲時都會保存進度，所保存的進度以文件的形式存在。這樣下次就可以繼續玩，而不用從頭開始。這裡的進度其實就是遊戲的內部狀態，而這裡的文件相當於是在遊戲之外保存狀態。這樣，下次就可以從文件中讀入保存的進度，從而恢復到原來的狀態。這就是備忘錄模式。

給出備忘錄模式的UML圖，以保存遊戲的進度為例。



Memento類定義了內部的狀態，而Caretake類是一個保存進度的管理者，GameRole類是遊戲角色類。可以看到GameRole的對象依賴於Memento對象，而與Caretake對象無關。下面給出一個簡單的是實現。

```
1. //需保存的信息
2. class Memento
3. {
4. public:
5.     int m_vitality; //生命值
6.     int m_attack;   //進攻值
7.     int m_defense;  //防守值
8. public:
9.     Memento(int vitality, int attack, int defense):
10.      m_vitality(vitality),m_attack(attack),m_defense(defense){}
11.    Memento& operator=(const Memento &memento)
12.    {
13.        m_vitality = memento.m_vitality;
14.        m_attack = memento.m_attack;
15.        m_defense = memento.m_defense;
16.        return *this;
17.    }
18. };
19. //遊戲角色
20. class GameRole
21. {
22. private:
23.    int m_vitality;
24.    int m_attack;
```

```
    int m_defense;
public:
    GameRole(): m_vitality(100),m_attack(100),m_defense(100) {}
    Memento Save()  //保存進度，只與Memento對象交互，並不牽涉到Caretake
    {
        Memento memento(m_vitality, m_attack, m_defense);
        return memento;
    }
    void Load(Memento memento)  //載入進度，只與Memento對象交互，並不牽涉到Caretake
    {
        m_vitality = memento.m_vitality;
        m_attack = memento.m_attack;
        m_defense = memento.m_defense;
    }
    void Show() { cout<<"vitality : "<< m_vitality<<", attack : "<< m_attack<<", defense : "<< m_defense<<endl; }
    void Attack() { m_vitality -= 10; m_attack -= 10;  m_defense -= 10; }
};
//保存的進度庫
class Caretake
{
public:
    Caretake() {}
    void Save(Memento menento) { m_vecMemento.push_back(menento); }
    Memento Load(int state) { return m_vecMemento[state]; }
private:
    vector<Memento> m_vecMemento;
};
```

客戶使用方式：

```
//測試案例
int main()
{
    Caretake caretake;
    GameRole role;
    role.Show();   //初始值
    caretake.Save(role.Save()); //保存狀態
    role.Attack();
    role.Show();  //進攻後
    role.Load(caretake.Load(0)); //載入狀態
    role.Show();  //恢復到狀態0
    return 0;
}
```