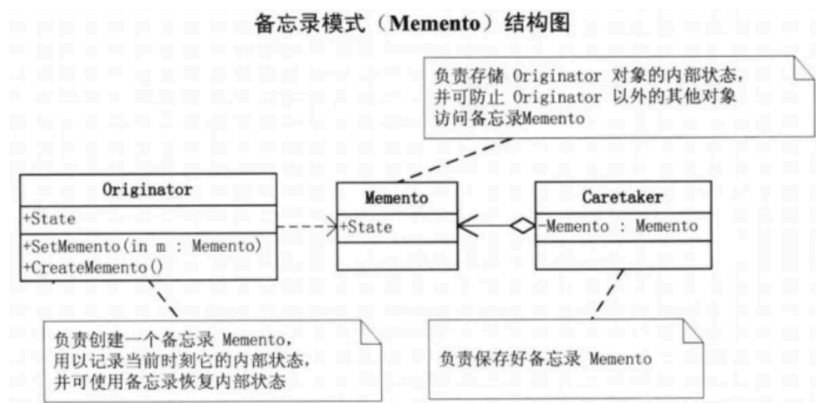


定义: 在不破坏封装性的前提下,捕获一个对象的内部状态,并在该对象之外保存这个状态. 这样以后就可以将该对象恢复到原先保存的状态. --<GOF>



备忘录模式的结构

- 发起人(Originator): 记录当前时刻的内部状态, 负责定义哪些属于备份范围的状态, 负责创建和恢复备忘录数据。
- 备忘录(Memento): 负责存储发起人对象的内部状态, 在需要的时候提供发起人需要的内部状态。
- 管理角色(Caretaker): 对备忘录进行管理, 保存和提供备忘录。

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 class Memento{ //备忘录
5 private:
6     string m_strState;
7 public:
8     Memento(string s){
9         m_strState = s;
10    }
11    void SetState(string s){
12        m_strState = s;
13    }
14    string GetState(){
15        return m_strState;
16    }
17 };
18 class Caretaker{ //管理者
19 private:
20     Memento *m_pMemento;
21 public:
22     ~Caretaker(){
23         delete m_pMemento;
24     }
25     void SetMemento(Memento *pM){
26         m_pMemento = pM;
27     }
28     Memento* GetMemento(){
29         return m_pMemento;
30     }
31 };
32 class Originator{ //发起人
33 private:
```

```

34     string m_strState;
35 public:
36     void SetState(string s){
37         m_strState = s;
38     }
39     string GetState(){
40         return m_strState;
41     }
42     Memento* CreateMemento(){ //创建备忘录对象，保存发起人的信息
43         return new Memento(m_strState);
44     }
45     void SetMemento(Memento* m){ //恢复备忘录
46         m_strState = m->GetState();
47     }
48     void ShowState(){
49         cout << "state: " << m_strState << endl;
50     }
51 };
52 int main(){
53     Originator *o = new Originator();
54     o->SetState("on");
55     o->ShowState();
56
57     Caretaker *c = new Caretaker();
58     c->SetMemento(o->CreateMemento()); //保存o的状态
59
60     o->SetState("off");
61     o->ShowState();
62
63     o->SetMemento(c->GetMemento()); //恢复状态
64     o->ShowState();
65
66     delete o; delete c;
67     return 0;
68 }

```

```

192:DesignPattnsStudy weishichun$ g++ -o Memento.out Memento_1.cpp
192:DesignPattnsStudy weishichun$ ./Memento.out
state: on
state: off
state: on
192:DesignPattnsStudy weishichun$

```