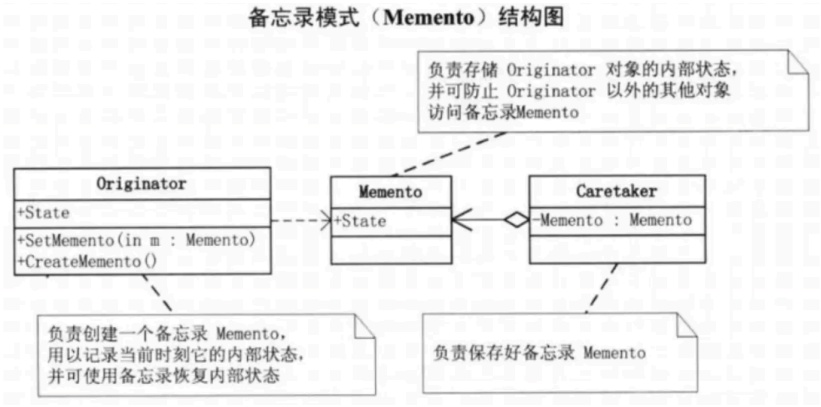
定义: 在不破坏封装性的前提下,捕获一个对象的内部状态,并在该对象之外保存这个状态. 这样以后就可以将该对象恢复到原先保存的状态. --<GOF>



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(Menento)**：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。

```
#include <iostream>
#include <string>
using namespace std;
class Memente{  //备忘录
private:
    string m_strState;
public:
    Memente(string s){
        m_strState = s;
    }
    void SetState(string s){
        m_strState = s;
    }
    string GetState(){
        return m_strState;
    }
};
class Caretaker{  //管理者
private:
    Memente *m_pMemente;
public:
    ~Caretaker(){
        delete m_pMemente;
    }
    void SetMemento(Memente *pM){
        m_pMemente = pM;
    }
    Memente* GetMemente(){
        return m_pMemente;
    }
};
class Originator{  //发起人
private:
```

```
    string m_strState;
public:
    void SetState(string s){
        m_strState = s;
    }
    string GetState(){
        return m_strState;
    }
    Memente* CreateMemente(){  //创建备忘录对象，保存发起人的信息
        return new Memente(m_strState);
    }
    void SetMemente(Memente* m){  //恢复备忘录
        m_strState = m->GetState();
    }
    void ShowState(){
        cout << "state: " << m_strState << endl;
    }
};
int main(){
    Originator *o = new Originator();
    o->SetState("on");
    o->ShowState();

    Caretaker *c = new Caretaker();
    c->SetMemento(o->CreateMemente()); //保存o的状态

    o->SetState("off");
    o->ShowState();

    o->SetMemente(c->GetMemente());//恢复状态
    o->ShowState();

    delete o; delete c;
    return 0;
}
```