7.备忘录模式

备忘录（Memento）模式的定义：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，以便以后当需要时能将该对象恢复到原先保存的状态。该模式又叫快照模式。

游戏角色状态恢复问题

游戏角色有攻击力和防御力，在大战 Boss 前保存自身的状态(攻击力和防御力)，当大战 Boss 后攻击力和防御力下降，从备忘录对象恢复到大战前的状态

传统方案解决游戏角色恢复

传统的方式的问题分析

1） 一个对象，就对应一个保存对象状态的对象， 这样当我们游戏的对象很多时，不利于管理，开销也很大.

2） 传统的方式是简单地做备份，new 出另外一个对象出来，再把需要备份的数据放到这个新对象，但这就暴露了对象内部的细节

3） 解决方案： => 备忘录模式

备忘录模式基本介绍

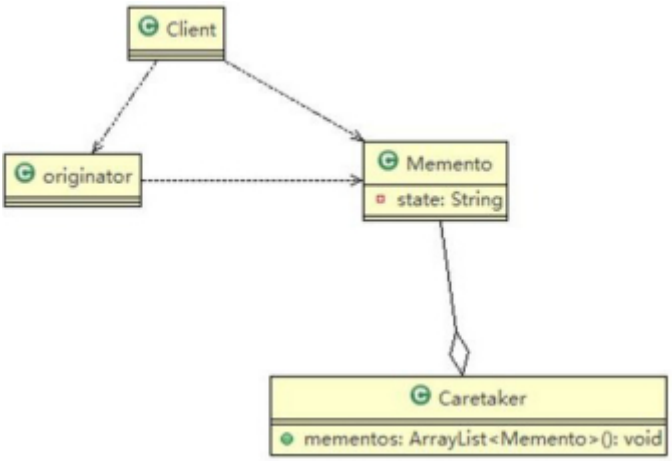
1） 备忘录模式（Memento Pattern）在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，以便以后当需要时能将该对象恢复到原先保存的状态。该模式又叫快照模式。

2） 可以这里理解备忘录模式：现实生活中的备忘录是用来记录某些要去做的事情，或者是记录已经达成的共同意见的事情，以防忘记了。而在软件层面，

备忘录模式有着相同的含义，备忘录对象主要用来记录一个对象的某种状态，或者某些数据，当要做回退时，可以从备忘录对象里获取原来的数据进行恢复操作

3) 备忘录模式属于行为型模式

备忘录模式的原理类图



对原理类图的说明-即(备忘录模式的角色及职责)

1) originator : 发起人对象(需要保存状态的对象)

2) Memento : 备忘录对象,负责保存好记录，即 Originator 内部状态

3) Caretaker: 守护者对象,负责保存多个备忘录对象， 使用集合管理，提高效率

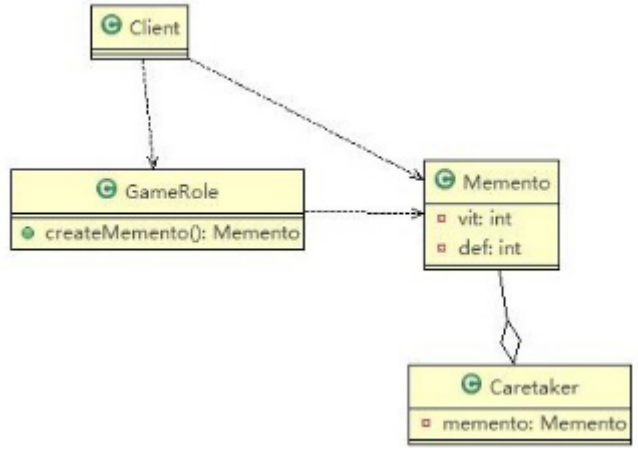
4) 说明：如果希望保存多个 originator 对象的不同时间的状态，也可以，只需要要 HashMap <String, 集合>

游戏角色恢复状态实例

1) 应用实例要求

游戏角色有攻击力和防御力，在大战 Boss 前保存自身的状态(攻击力和防御力)，当大战 Boss 后攻击力和防御力下降，从备忘录对象恢复到大战前的状态

2) 思路分析和图解(类图)




```
public class GameRole {
   private int vit;
   private int def;

   //创建Memento ,即根据当前的状态得到Memento
   public Memento createMemento() {
      return new Memento(vit, def);
   }

   //从备忘录对象，恢复GameRole的状态
   public void recoverGameRoleFromMemento(Memento memento) {
      this.vit = memento.getVit();
      this.def = memento.getDef();
   }
   .....省略一些set,get方法以及没用方法
}


public class Memento {
   //攻击力
   private int vit;
   //防御力
   private int def;
   public Memento(int vit, int def) {
      super();
      this.vit = vit;
      this.def = def;
   }
   ....省略set,get方法
}
```

备忘录模式的注意事项和细节

1） 给用户提供了一种可以恢复状态的机制，可以使用户能够比较方便地回到某个历史的状态

2) 实现了信息的封装，使得用户不需要关心状态的保存细节

3) 如果类的成员变量过多，势必会占用比较大的资源，而且每一次保存都会消耗一定的内存，这个需要注意

4) 适用的应用场景：1、后悔药。 2、打游戏时的存档。 3、Windows 里的 ctri + z。 4、IE 中的后退。 4、数据库的事务管理

5) 为了节约内存，备忘录模式可以和原型模式配合使用

6)备忘录模式可以和命令模式配合使用