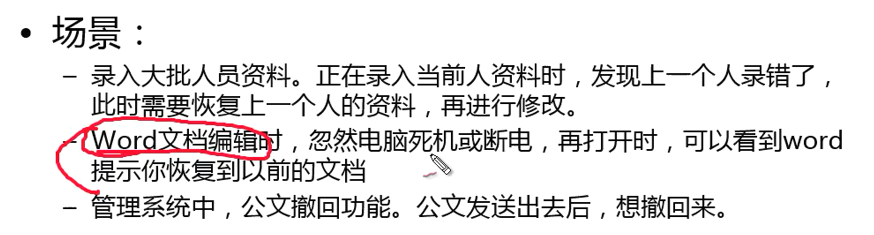
备忘录模式



Memento是纪念章的意思

\*\*可以保存对象内部的状态

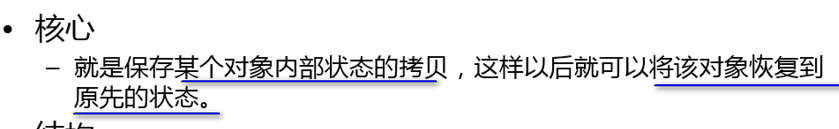


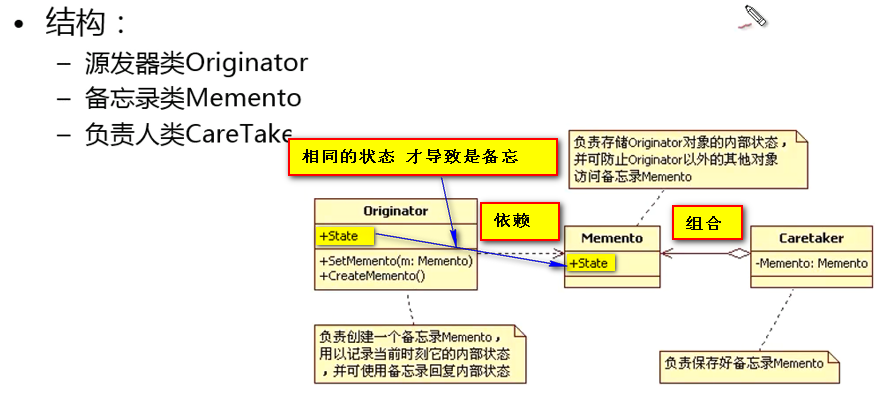
提供保存对象状态的功能

用过ps的话 修改某一个图片 会提供历史记录 点击对应的历史记录 会恢复到历史记录的样子 这也是备忘录模式实现的

备忘录模式 如何恢复到以前的状态

也就是 备忘录模式可以实现回滚





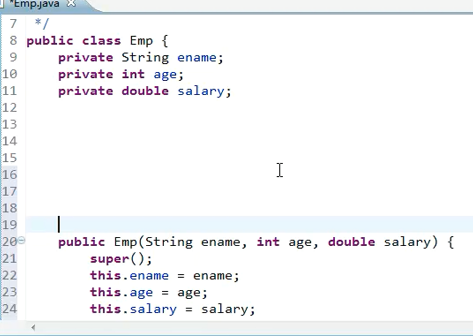
结构就有 源发器类 Originator 保存对象内部的状态

备忘录类 Memento 负责存储拷贝

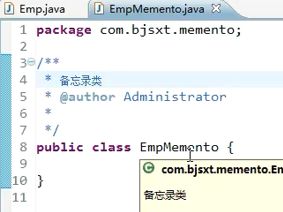
负责人类 CareTaker 用来存储备忘录 ----- 可以提供一堆备忘录

Ctrl + z 一直回退 ---- 负责人管理备忘录List

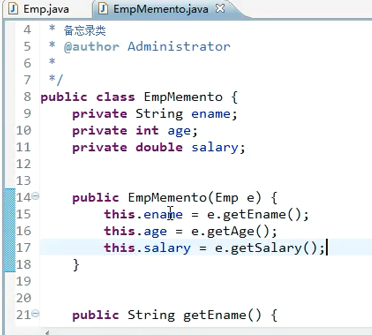
Emp.memento()就生成了一个备忘录对象【备忘录都是针对指定的对象做备份 不能的对象 备份的信息不同 这样 具体对象持有真实数据 所以 可以根据这个生成该对象的备忘录对象 ---- 所以 拥有数据的对象才有权利提供方法 所以 他可以提供备忘录对象的实例化方法 所以 依赖这个备忘录对象 ------ 一个单独的管理备忘录的对象CareTaker】

 这个是源发器类

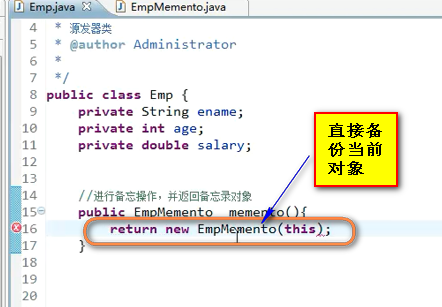
定义一个备忘录类



使用来保存Emp中的各个属性状态的



回到源发器类 创建一个备忘录

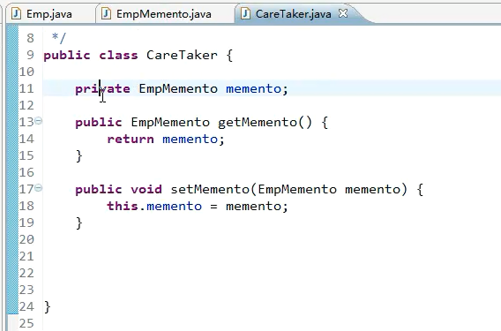


进行恢复 给出一个恢复的方法 恢复哪一个备忘录？传哪个备忘录 就恢复哪一个

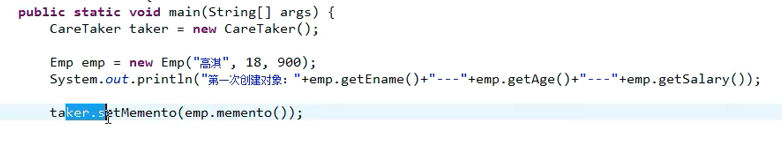


memento(创建备忘录的过程 就是一个备忘保存的过程)和recovery是一对互逆的过程

CareTaker ---- 负责备忘录的管理类



接下来 做个试验

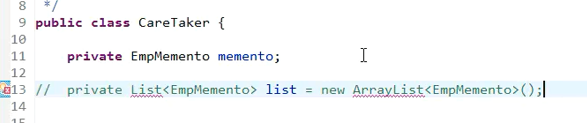




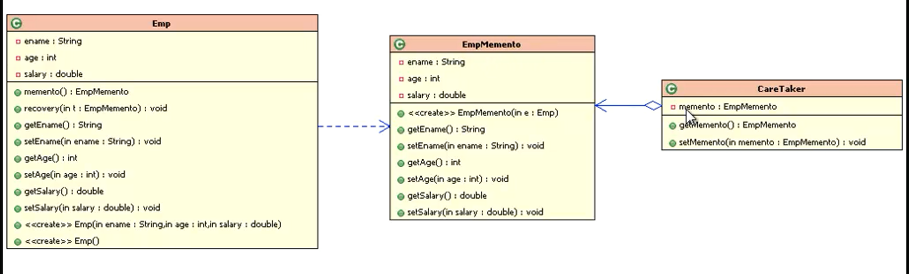
这样就恢复出来了

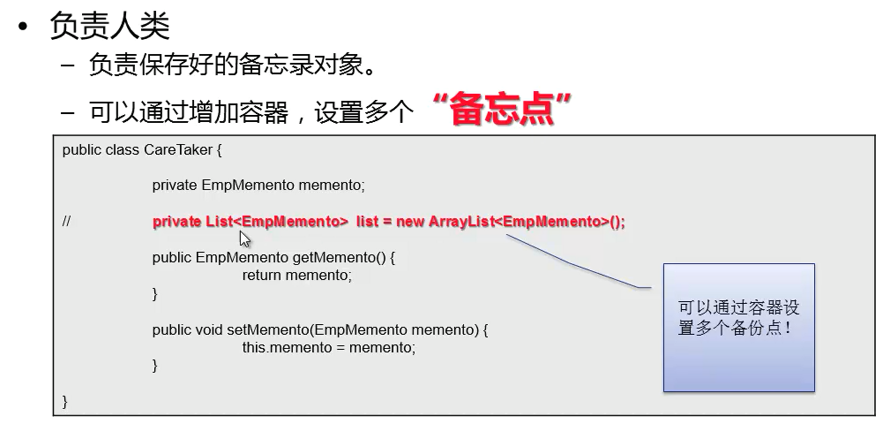
CareTaker 仅仅备份一次 一个成员变量 就可以了

如果想要保存很多次 通过一个容器来做



这样就存储了多个备忘点

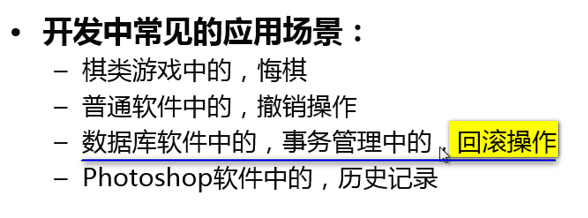






使用Stack非常适合恢复点 先进后出 后进先出

保存的点特别多 放在内存中不合适 所以 持久化 序列化到本地硬盘 不可能一直放在内存



事务管理的底层架构是命令模式 ----- 执行SQL之前 就要使用备忘录模式备忘一下

回滚到执行事务之前的状态

事务的底层机制实际上是 备忘录 + 命令模式