APP 抽奖活动

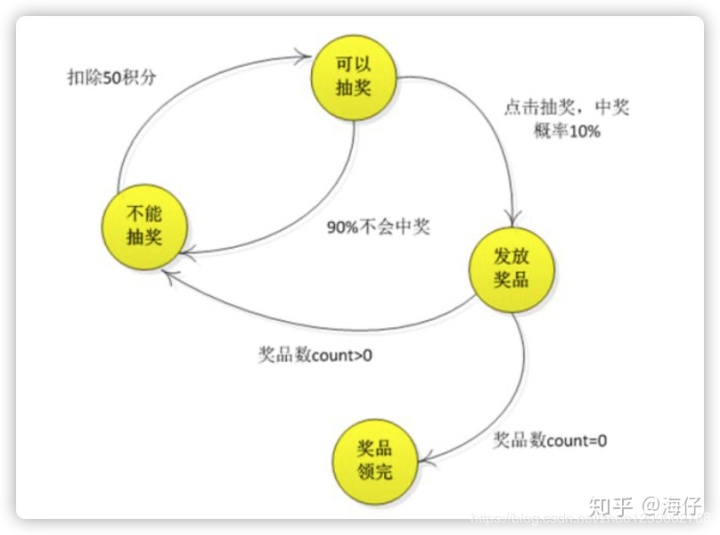
1.请编写程序完成APP抽象活动，具体要求如下 ：

1）加入每参加一个这个活动要扣除用户50积分，中奖概率是10%。

2）奖品数量固定，抽完就不能抽奖。

3）活动有四个状态 ：可以抽象、不能抽象、发放奖品和奖品领完。

4）活动的四个状态转换关系图（右图）

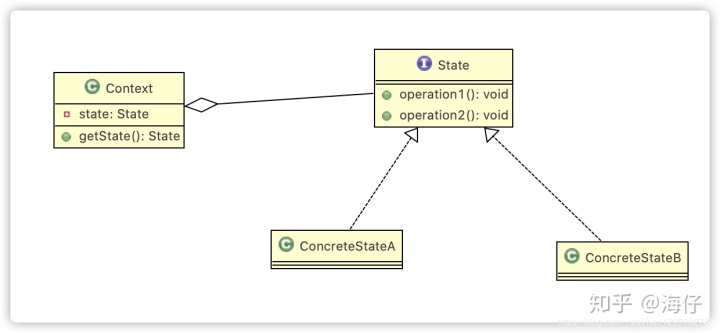


2. 状态模式基本介绍

1）状态模式（State Pattern）：它主要用来解决对象在多种状态转换时，需要对外输出不同的行为的问题。状态和行为是一一对应的，状态之间可以相互转换。

2）当一个对象的内在状态改变时，允许改变其行为，这个对象看起来像是改变类。

状态模式原理图：



对原理类图的说明 ：

1）Context 类为环境角色，用于维护State 实例，这个实例定义当前状态。

2）State 是抽象状态角色，定义一个接口封装与Context 的一个特点接口相关行为。

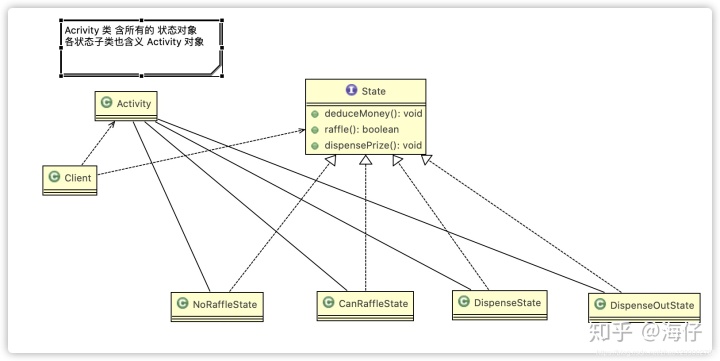
3）ConcreteState 具体的状态角色，每个子类实现一个与Context 的一个状态相关行为。

3.状态模式解决APP抽奖问题

1）应用实例要求完成APP抽象活动项目，使用状态模式。

2）类图定义出一个接口叫状态接口，每个状态都实现它。

接口有扣除积分方法、抽象方法、发放奖品方法



状态模式的注意事项和细节

1）代码有很强的可读性。状态模式将每个状态的行为封装到对应的一个类中。  
2）方便维护。将容易产生问题的 if - else 语句删除了，如果把每个状态的行为都放到一个类中，每次调用方法时都要判断当前是什么状态，不但会产出很多if - else语句，而且容易出错。  
3）符合 “开闭原则”。容易增删状态。  
4）会产生很多类。每个状态都要一个对应的类，当状态过多时会产生很多类，加大维护难度。  
5）应用场景 ：当一个事件或者对象很很多种状态，状态之间会相互依赖，对不同的状态要求有不同的行为的时候，可以考虑使用状态模式。