state模式

场景

（1）销售出库单，状态需要从新建、待审批、已审批、已完成，四个状态之间流转，流转到每个状态的时候，都需要执行那个状态对应的一个逻辑

public **interface State** {

   void handle(String parameter);

}

public class **StateA implements State** {

   public void handle(String parameter) {

       System.out.println("状态A的处理逻辑");

  }

}

public class **StateB implements State** {

   public void handle(String parameter) {

       System.out.println("状态B的处理逻辑");

  }

}

【重点在 Context，根据入参不同，决定调用不同的状态流转逻辑】

public class **Context** {

   private State state;

   public void setState(State state) {

       this.state = state;

  }

   public void request(String parameter1, String parameter2) {

       state.handle(parameter1);

       if(parameter2.equals("B")) {

           this.state = new StateB();

      }

  }

}

其实状态模式，就是维护多种不同的状态，每种状态下可以去执行一些特殊的逻辑。然后由一个Context类负责根据请求参数调用，来维护这些状态之间的切换，形成一个状态机的概念。

状态模式里，**非常重要的一点就是将状态之间流转的逻辑，封装在Context类里面**。本来可能需要调用方自己维护复杂的状态流转逻辑，流转到不同的状态之后，执行状态对应的代码逻辑。

这个状态模式，在电商系统中有天然的运用场景，我们的系统中很多数据都有大量的状态变更的逻辑，我们可以将状态变更的逻辑用状态模式来实现，将状态流转封装在一个地方，每次状态变更就通知那个组件，让状态流转，同时执行状态对应的代码逻辑

853769620