```
    private Long version;

    private OrderState state;
    private Long consumerId;
    private Long restaurantId;

    @Embedded
    private OrderLineItems orderLineItems;

    @Embedded
    private DeliveryInformation deliveryInformation;

    @Embedded
    private PaymentInformation paymentInformation;

    @Embedded
    private Money orderMinimum;
```

此类使用JPA持久化，并映射到ORDERS表。id字段是主键。version字段用于乐观锁。Order的状态由枚举类型OrderState表示。DeliveryInformation和PaymentInformation字段使用@Embedded注解进行映射，并存储到ORDERS表对应的列。orderLineItems字段是包含订单行项的嵌入式对象。Order聚合包含的不仅仅是字段。它还实现了业务逻辑，后者可以由状态机描述。我们来看看这个状态机。

Order聚合状态机

为了创建或更新订单，Order Service必须使用Saga与其他服务协作。OrderService或Saga的第一步调用Order方法，该方法验证操作可以被执行并将Order的状态更改为Pending状态。如第4章所述，Pending状态是一个语义锁对策，它有助于确保Saga彼此隔离。最终，一旦Saga调用了参与方服务，它就会更新Order以便反映出调用的结果。例如，如第4章所述，Create Order Saga包含多个参与方服务，包括Consumer Service、Accounting Service和Kitchen Service。OrderService首先在APPROVAL_PENDING状态下创建一个Order，稍后将其状态更改为APPROVED或REJECTED。Order的行为可以建模为图5-14所示的状态机。

类似地，其他Order Service的操作（如revise()和cancel()）首先将Order更改为*Pending状态，并使用Saga验证是否可以执行操作。然后，一旦Saga成功验证该操作可以执行，它就会将Order转换为反映操作成功结果的其他状态。如果操作验证失败，则订单将恢复到之前的状态。例如，cancel()操作首先将Order转换为CANCEL_PENDING状态。如果订单可以取消，则Cancel Order Saga会将Order状态更改为CANCELED状态。否则，如果cancel()操作被拒绝，例如，取消订单为时已晚，则Order转换回APPROVED状态。