```
public class OrderSettlement extends Entity<OrderSettlementId> {
   private Integer payNumber;
   private Money payAmount;
   private List<Payment> payments;

   public Money totalAmount() {
      if (payNumber == payments.size()) {
         if (!payAmount.equals(totalPayAmount())) {
            throw new OrderSettlementException("Error with calculating total price
for Order Settlement.");
         }
      }
      return payAmount;
   }

   private Money totalPayAmount() {
      Money totalAmount = new Money(0);
      for (Payment payment : payments) {
         totalAmount = totalAmount.add(payment.getPayAmount());
      }
      return totalAmount;
   }
}
```

该领域行为并不复杂，但充分体现了行为的自给自足。整个方法仅操作了订单结算实体自己拥有的属性，包括payNumber、payAmount和payments。

3. 互为协作的领域行为

实体不可能都做到自给自足，有时也需要调用者提供必要的信息。这些信息往往通过方法参数传入，这就形成了领域对象之间互为协作的领域行为。例如，要计算贸易订单实际应缴的税额，首先应该获得该贸易订单的纳税额度。这个纳税额度等于订单所属的纳税调节额度汇总值减去手动调节纳税额度值。获得的纳税额度再乘以贸易订单的总金额，就是贸易订单实际应缴的税额。贸易订单的纳税调节为另一个实体对象TaxAdjustment。一个贸易订单存在多个纳税调节，因此可引入一个容器对象TaxAdjustments。该对象本质上是一个领域服务，提供了计算纳税调节额度汇总值和手动调节纳税额度值的方法：

```
public class TaxAdjustments {
   private List<TaxAdjustment> taxAdjustments;
   private BigDecimal zero = BigDecimal.ZERO.setScale(taxDecimals, taxRounding);

   public BigDecimal totalTaxAdjustments() {
      return taxAdjustments
                  .stream
                  .reduce(zero, (ta, agg) -> agg.add(ta.getAmount()));
   }

   public BigDecimal manuallyAddedTaxAdjustments() {
      return taxAdjustments
                  .stream
                  .filter(ta -> ta.isManual())
```