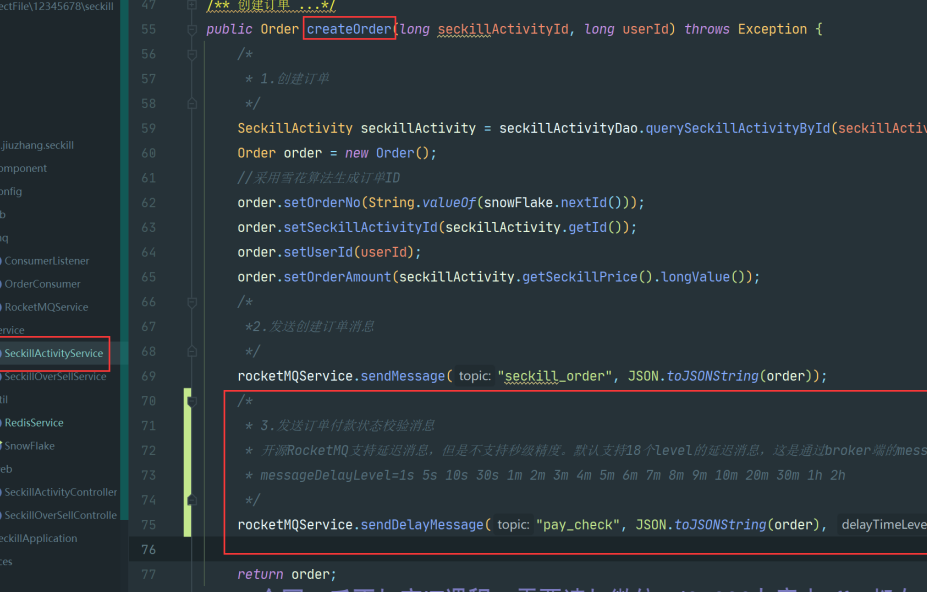


1. 在创建订单 createOrder 方法中添加“发送订单付款状态校验消息”功能



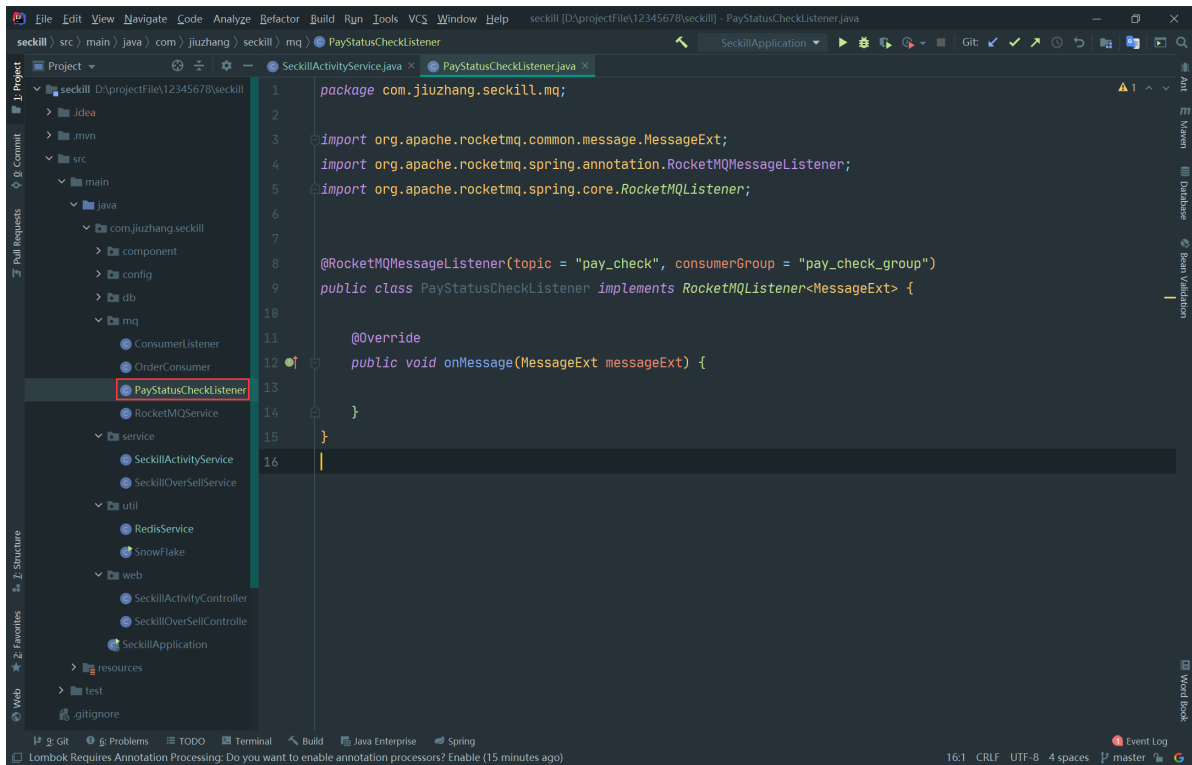
```
/** 创建订单... */
public Order createOrder(long seckillActivityId, long userId) throws Exception {
    /*
     * 1. 创建订单
     */
    SeckillActivity seckillActivity = seckillActivityDao.querySeckillActivityById(seckillActivityId);
    Order order = new Order();
    // 采用雪花算法生成订单ID
    order.setOrderNo(String.valueOf(snowFlake.nextId()));
    order.setSeckillActivityId(seckillActivity.getId());
    order.setUserId(userId);
    order.setOrderAmount(seckillActivity.getSeckillPrice().longValue());
    /*
     * 2. 发送创建订单消息
     */
    rocketMQService.sendMessage(topic: "seckill_order", JSON.toJSONString(order));
    /*
     * 3. 发送订单付款状态校验消息
     * 开源RocketMQ支持延迟消息，但是不支持秒级精度。默认支持18个level的延迟消息，这是通过broker端的messageDelayLevel配;
     * messageDelayLevel=1s 5s 10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m 30m 1h 2h
     */
    rocketMQService.sendDelayMessage(topic: "pay_check", JSON.toJSONString(order), delayTimeLevel: 3);
    return order;
}
```

```

/*
 * 3. 发送订单付款状态校验消息
 *  开源RocketMQ支持延迟消息，但是不支持秒级精度。默认支持18个level的延迟消息，这是通
过broker端的messageDelayLevel配置项确定的，如下：
 *  messageDelayLevel=1s 5s 10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m
30m 1h 2h
 */
rocketMQService.sendDelayMessage("pay_check", JSON.toJSONString(order),
3);

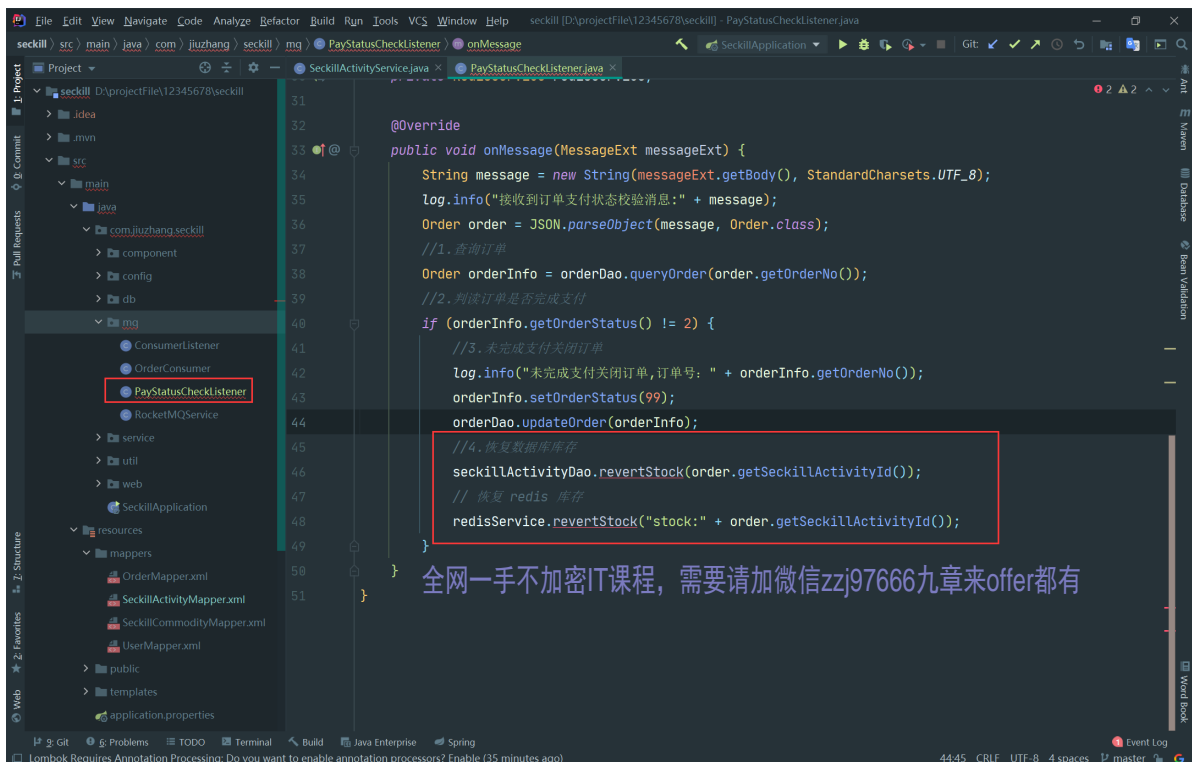
```

2. 创建 "订单付款状态信息的消费者" 类: PayStatusChangeListener



3. 完善 onMessage 方法，处理超时订单

接下来我们一一实现这二个方法。



```
@Slf4j
@Component
@RocketMQMessageListener(topic = "pay_check", consumerGroup = "pay_check_group")
public class PayStatusCheckListener implements RocketMQListener<MessageExt> {

    @Autowired
    private OrderDao orderDao;

    @Autowired
    private SeckillActivityDao seckillActivityDao;
```

```

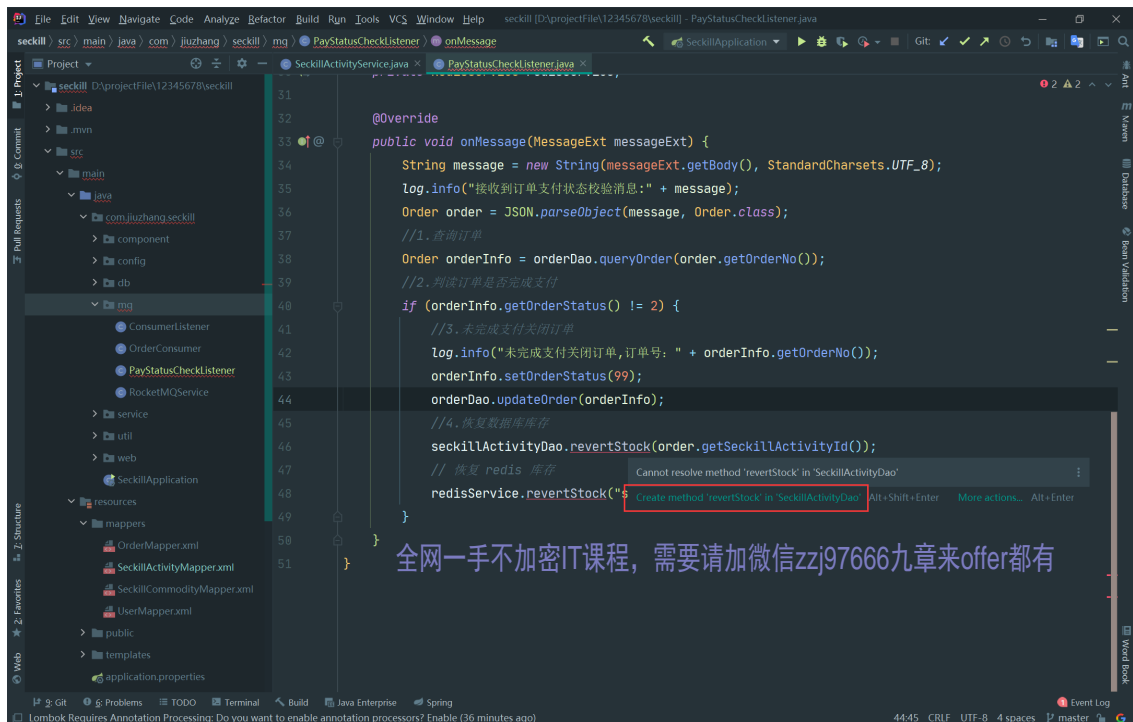
@Resource
private RedisService redisService;

@Override
@Transactional
public void onMessage(MessageExt messageExt) {
    String message = new String(messageExt.getBody(),
StandardCharsets.UTF_8);
    log.info("接收到订单支付状态校验消息:" + message);
    Order order = JSON.parseObject(message, Order.class);
    //1. 查询订单
    Order orderInfo = orderDao.queryOrder(order.getOrderNo());
    //2. 判断订单是否完成支付
    if (orderInfo.getOrderStatus() != 2) {
        //3. 未完成支付关闭订单
        log.info("未完成支付关闭订单, 订单号: " + orderInfo.getOrderNo());
        orderInfo.setOrderStatus(99);
        orderDao.updateOrder(orderInfo);
        //4. 恢复数据库库存
        seckillActivityDao.revertStock(order.getSeckillActivityId());
        // 恢复 redis 库存
        redisService.revertStock("stock:" + order.getSeckillActivityId());
    }
}
}
}

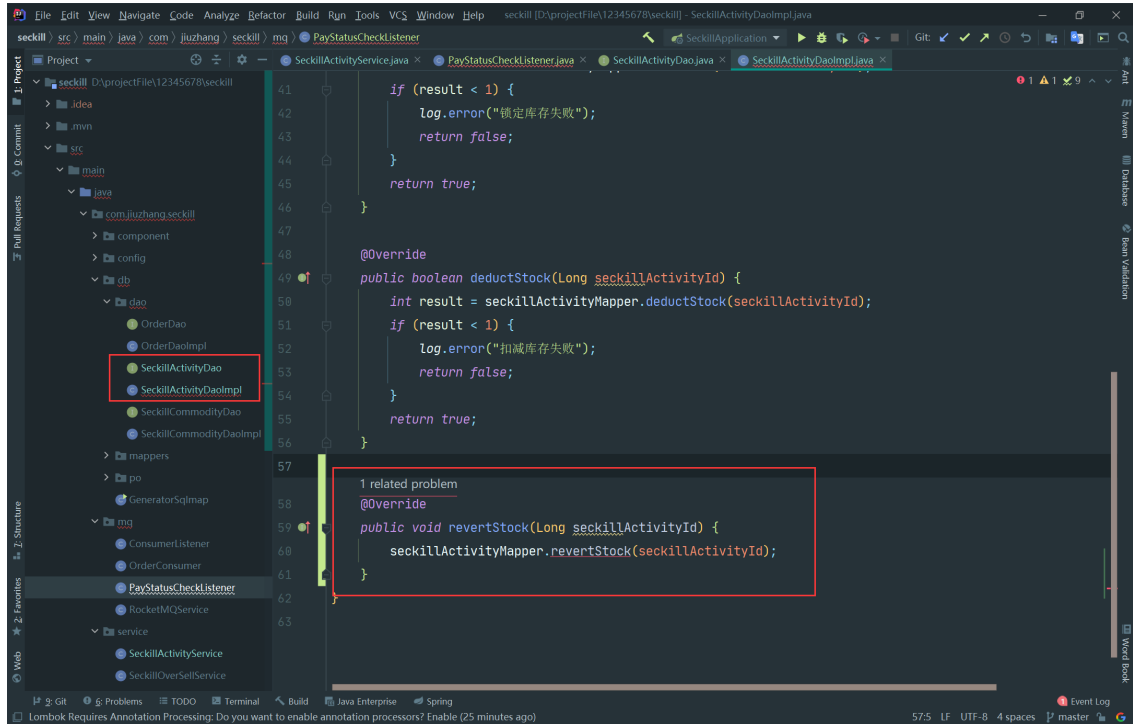
```

4. 实现 seckillActivityDao.revertStock() 方法

1. 创建 seckillActivityDao.revertStock() 方法

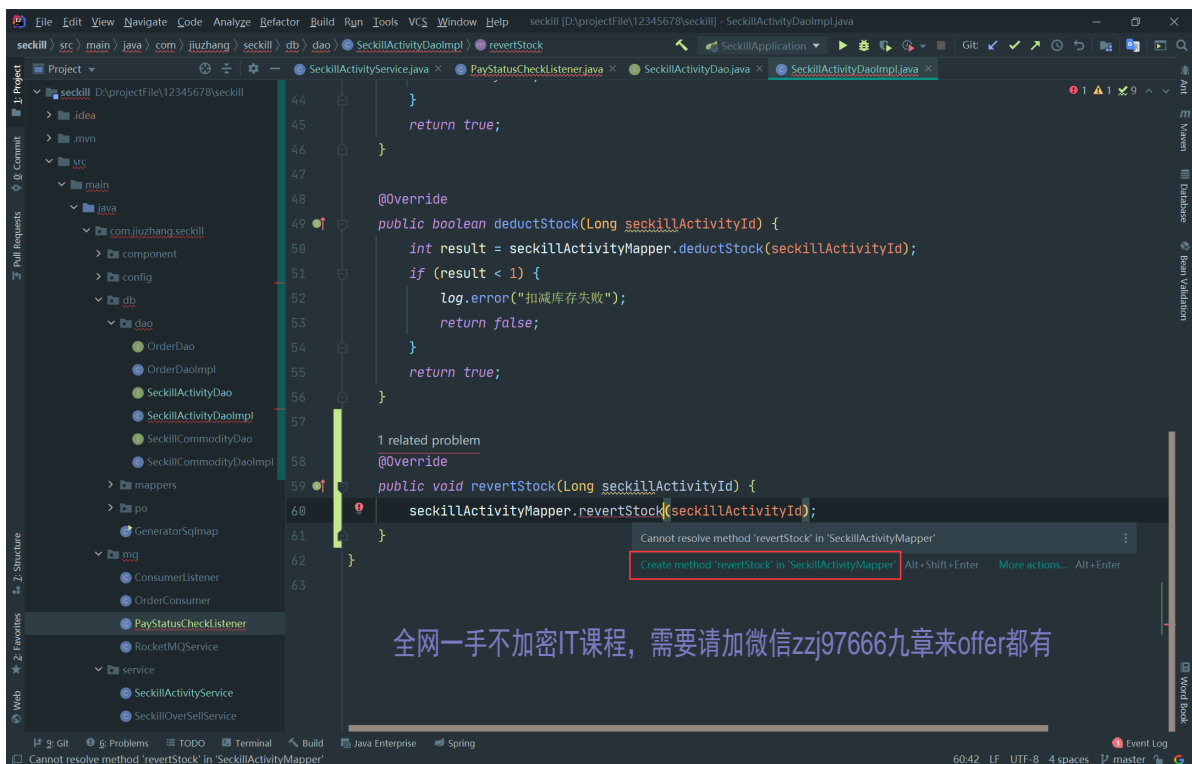


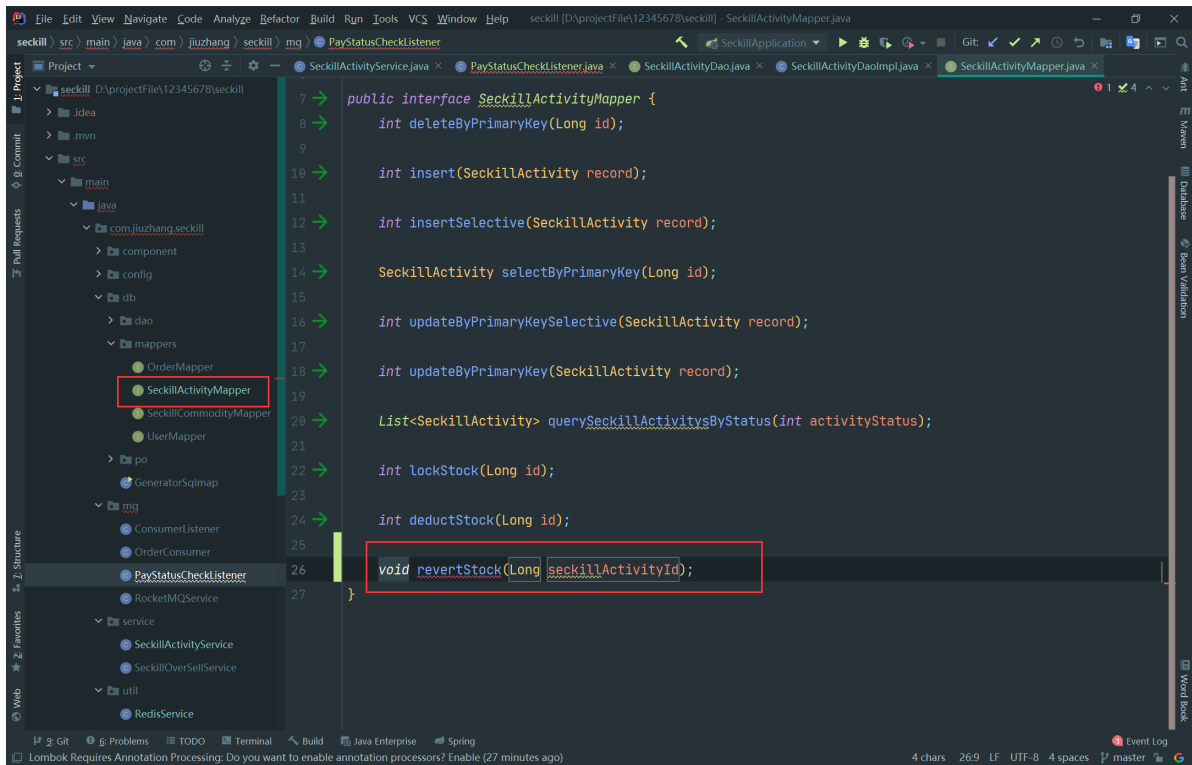
2. 实现 SeckillActivityDaoImpl.revertStock() 方法



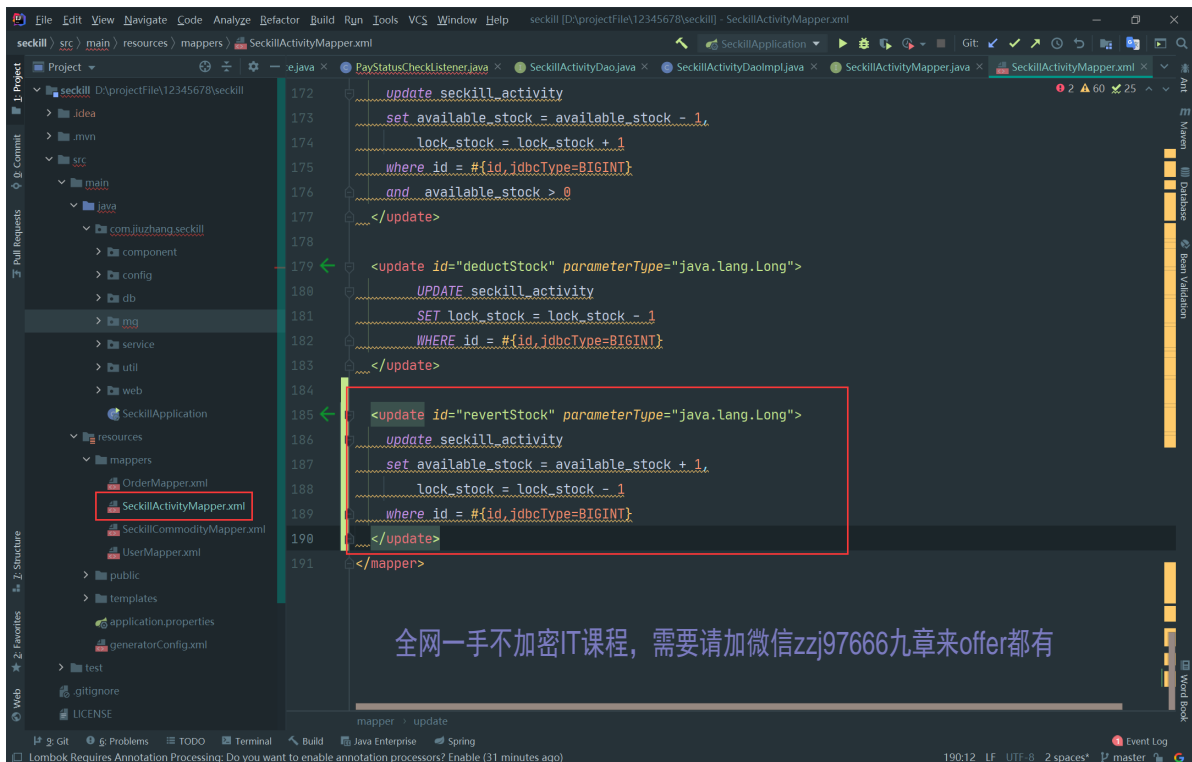
```
@Override
public void revertStock(Long seckillActivityId) {
    seckillActivityMapper.revertStock(seckillActivityId);
}
```

3. 编写 SeckillActivityMapper.revertStock() 方法



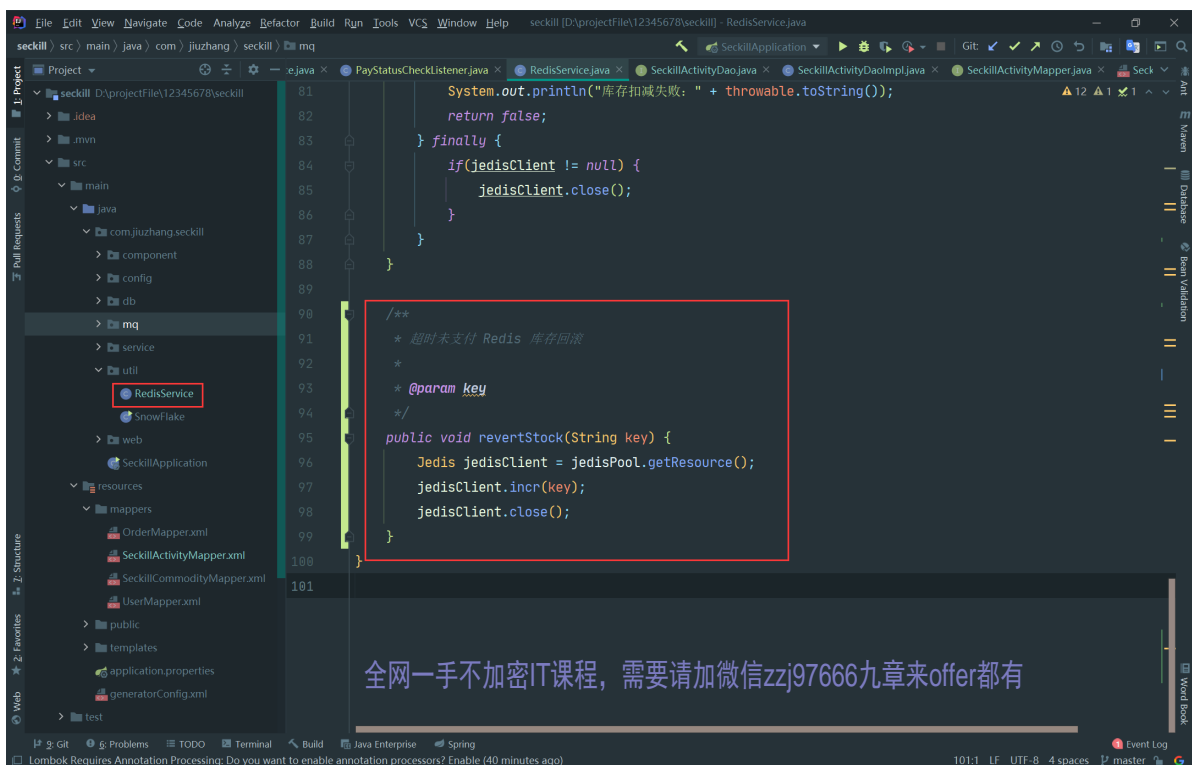
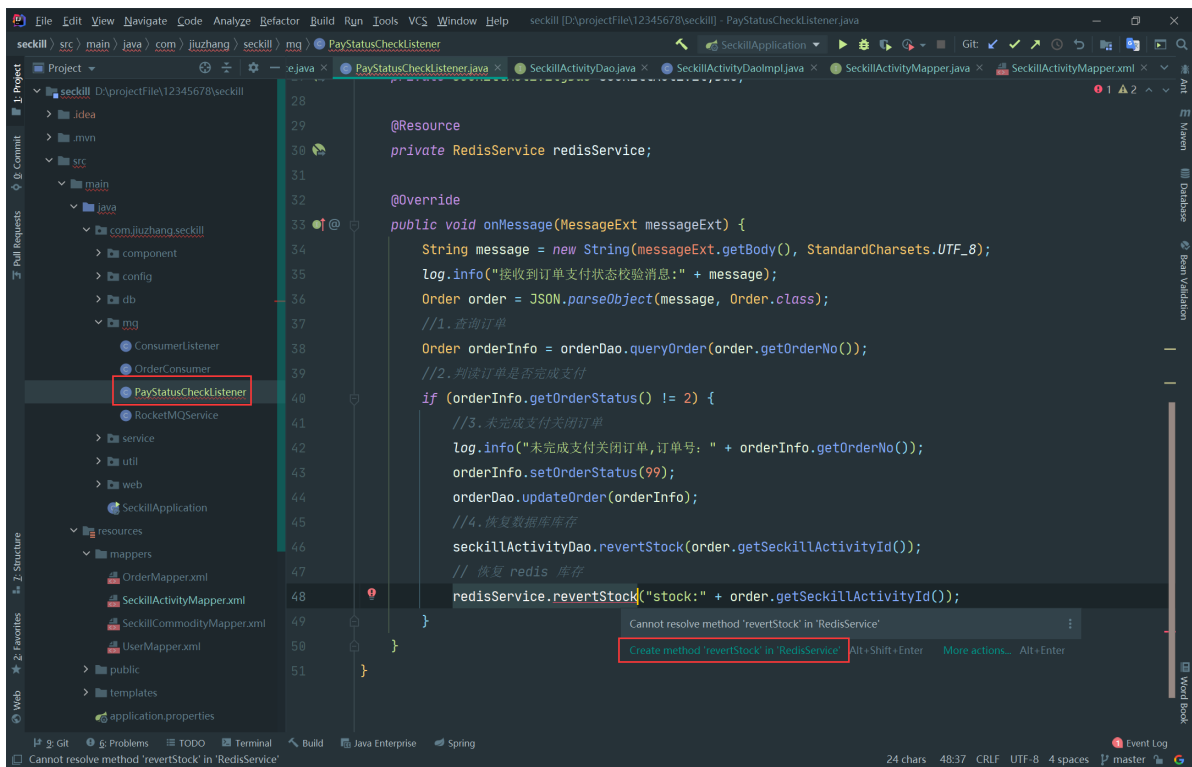


4. 编写 SeckillActivityMapper.revertStock() 方法对应的 SQL 语句



好了，到这里我们“恢复数据库库存”的方法 revertStock 就写完了

5. 实现 redisService.revertStock() 方法



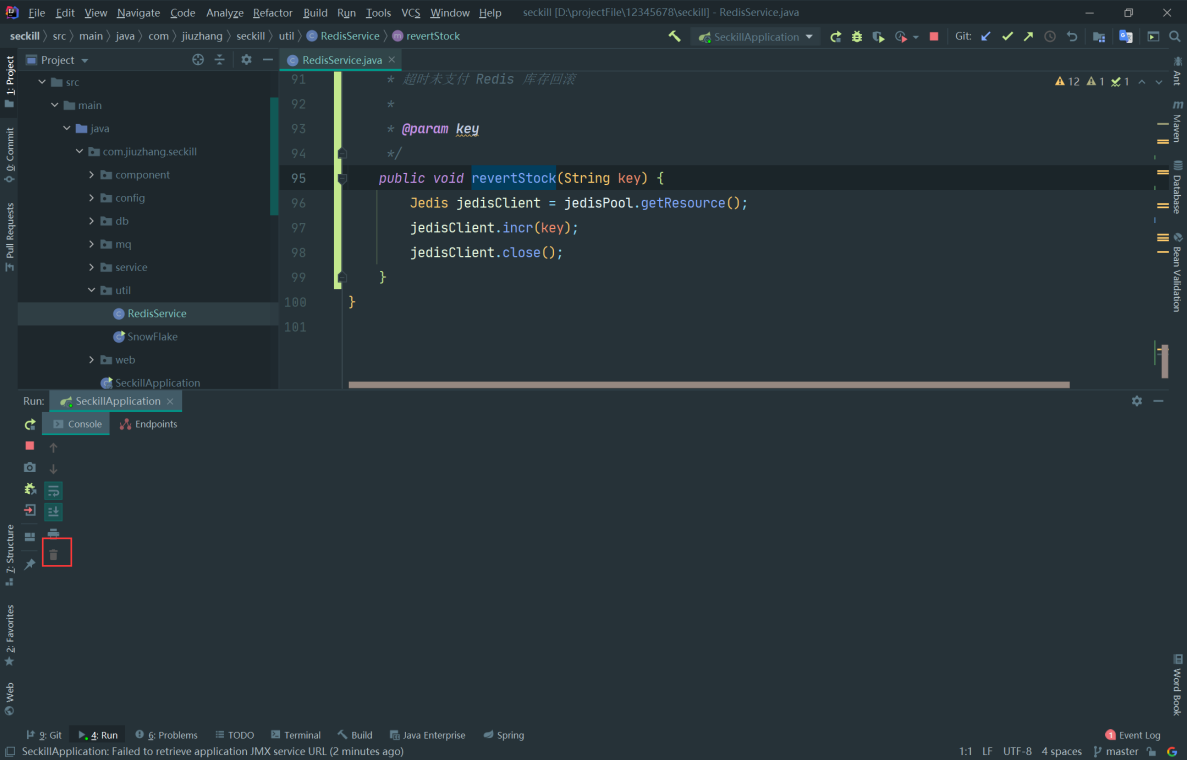
全网一手不加密IT课程，需要请加微信zzj97666九章来offer都有

```
/**
 * 超时未支付 Redis 库存回滚
 *
 * @param key
 */
public void revertStock(String key) {
    Jedis jedisClient = jedisPool.getResource();
    jedisClient.incr(key);
    jedisClient.close();
}
```

到这里我们的处理超时订单的功能就实现了，接下来我们就开始测试吧。

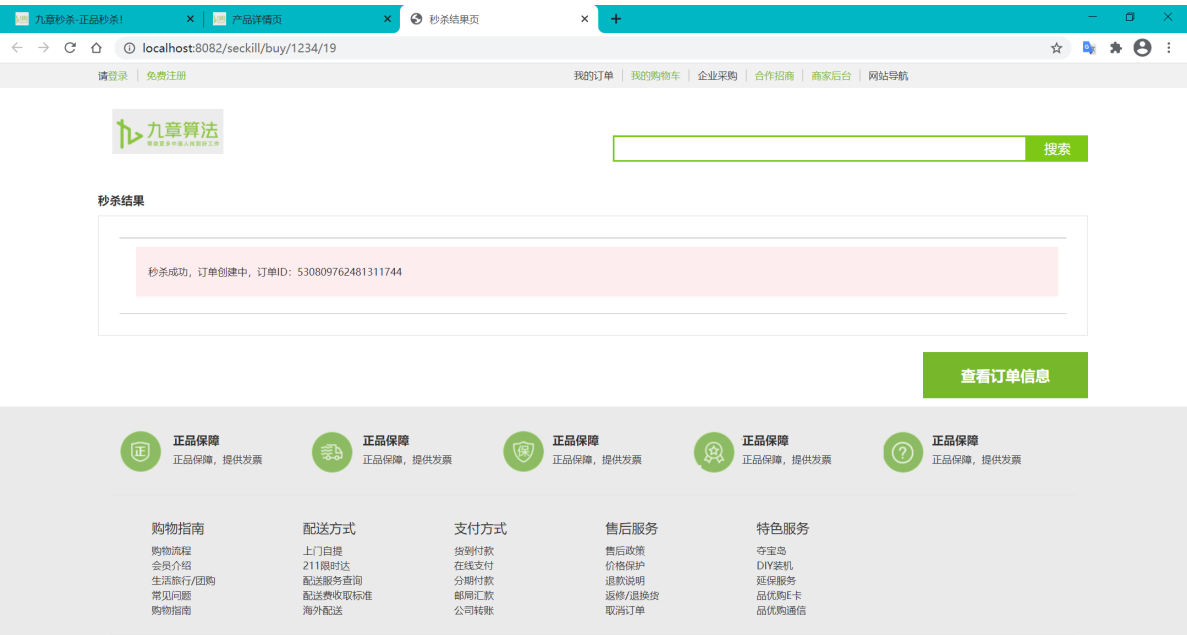
6. 测试 处理超时未支付订单 功能

1. 启动项目，清空控制台日志信息

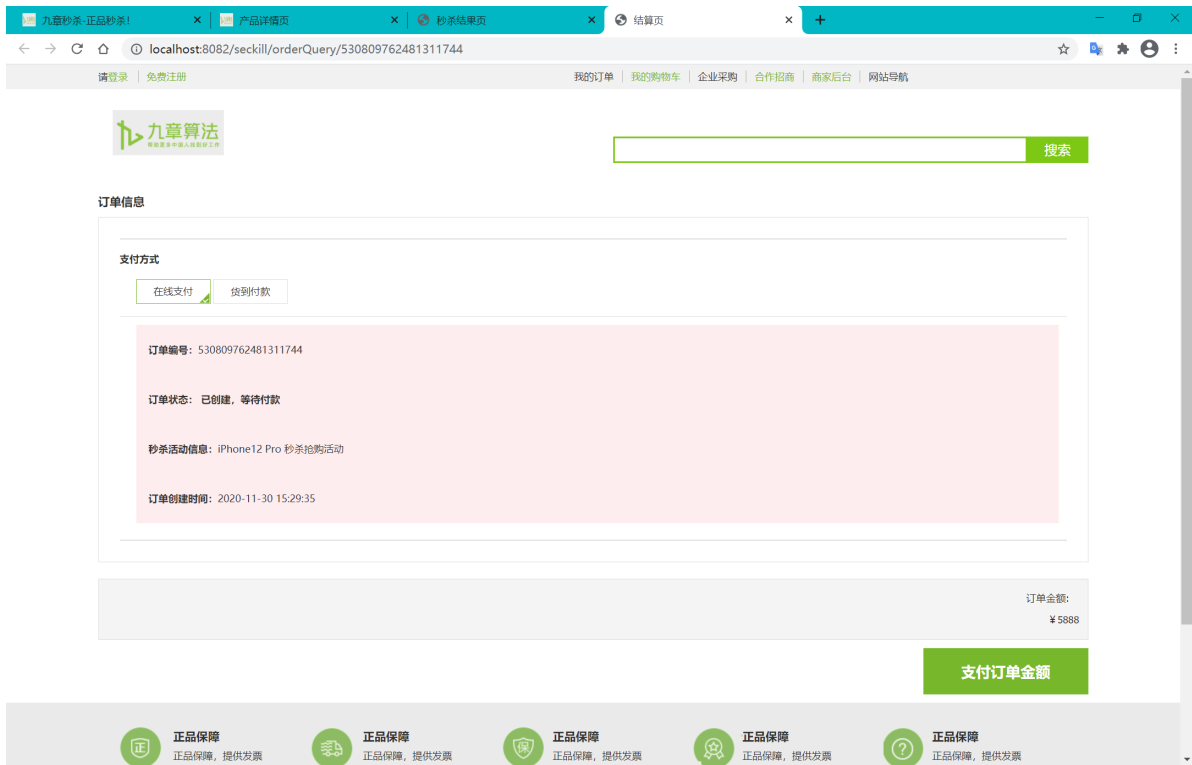


2. 抢购商品

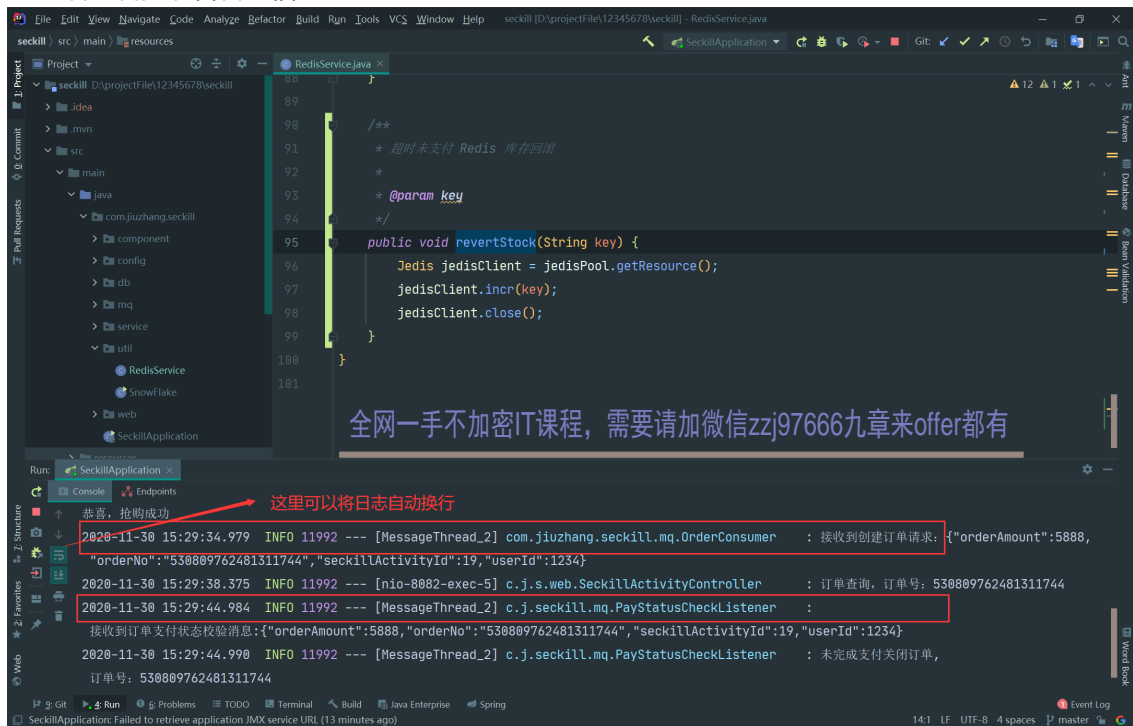
全网一手不加密IT课程，需要请加微信zzj97666九章来offer都有



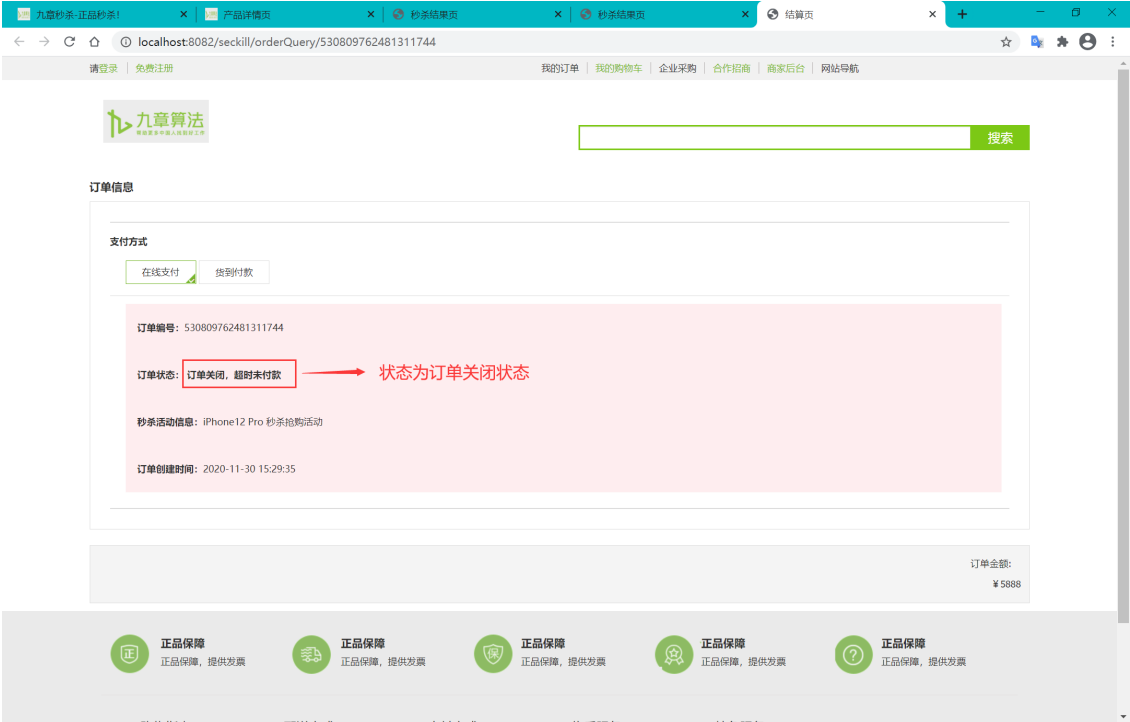
3. 查看订单信息



4. 10s 后查看控制台日志信息



5. 我们在刷新一下查询订单信息的页面



全网一手不加密IT课程, 需要请加微信zzj97666九章来offer都有

超时订单处理部分就到这里了。