1，发牌定时任务

从va\_ta\_jc\_allbussiness\_infosum视图中查询所有的业务数据，该视图包括所有的办件实例数据，所有的投诉信息，所有的咨询信息。

bussiness\_type=1：办件数据；

bussiness\_type=2：投诉数据；

bussiness\_type=3：咨询数据。

每个线程，往TA\_JC\_SP\_STATUS表里插入一条数据，记录执行成功或失败。

在ruleInit方法中：

获取TA\_JC\_RULE\_ENGINECFG数据，并执行init类型数据指向类的start方法。

在start方法里，计算办件类型数据（bussiness\_type=1）的已处理时间，并反写到办件实例表TA\_SP\_INSTANCE里。

具体算法：

1、先计算最后时间与受理时间差异的天数（去掉节假日）。

2、再计算受理当天耗时数加上最后时间耗时数。

3、计算特别程序所耗时间。

4、用1+2-3计算实际耗时，小数进1。

在ruleCaculate方法中：

删除所有业务数据的监察要素结果TA\_JC\_RULE\_ELEM\_RESULT。因为这张表的作用，相当于是一张临时表，用于存储业务数据在某个监察要素的已处理时间，在计算是否要发牌时，要根据这张表的已处理时间，关联规则的表达式，构造出一个完整表达式并计算，所以只要当次监察定时任务已经将发牌结果写入到监察信息表TA\_JC\_SUPERVISE\_INFO后，这张临时表的数据就没意义了，所以每次执行前先清空这张表，避免数据积累太多，在数据关联时产生很多垃圾数据。

查询所有的监察要素，循环监察要素，并执行监察要素里指向类的doCaculator方法。

1，咨询办理时间超时计算：

咨询时限监察 算法：

1、对于咨询事件，以系统当前时间减到期时间，计算超出时间

2、计算超出时间时，必须去掉节假日、特别程序时间

循环之前查询出来的所有业务数据，如果bussiness\_type=3，再根据bussiness\_id从ta\_sp\_consult表中查询出咨询详细数据，bussiness\_id=ta\_sp\_consult.consult\_id，将查询出来的数据，放入到List<ConsultInfo>中，循环List<ConsultInfo>，计算处理时间，然后批量写入到监察结果表TA\_JC\_RULE\_ELEM\_RESULT中。

2，投诉办理时间超时计算

循环之前查询出来的所有业务数据，如果bussiness\_type=2，再根据bussiness\_id从投诉信息表TA\_JC\_COMPLAIN\_BASEINFO中查询出state不等于“不予受理”，也不等于“待受理”，并且过期时间不为空的数据，bussiness\_id=TA\_JC\_COMPLAIN\_BASEINFO.Complain\_Id，放入List<TaJcComplainBaseinfo>中，循环List<TaJcComplainBaseinfo>，计算处理时间，然后批量写入到监察结果表TA\_JC\_RULE\_ELEM\_RESULT中。

3，业务办理时间按承诺时限超时计算

办理时限监察算法：

1、对于在办办件，以系统当前时间减受理时间，计算实际耗时

2、针对正常办结了的办件（一般办结、正常办结、上报办结、转报办结），以办 结时间减去受理时间，计算实际耗时

3、计算实际耗时时，必须去掉节假日、特别程序时间

循环之前查询出来的所有业务数据，如果bussiness\_type=1，将数据插入到监察结果表TA\_JC\_RULE\_ELEM\_RESULT中。因为bussiness\_type=1的办件数据，在之前的ruleInit方法中已经计算过处理时间，并反写到办件实例表TA\_SP\_INSTANCE里了，所以不用再计算一次，直接关联TA\_SP\_INSTANCE，插入到TA\_JC\_RULE\_ELEM\_RESULT即可。

4，业务办理时间按法定时限超时计算

同上3，业务办理时间按承诺时限超时计算。

5，特殊程序时限监察规则计算

算法：如果特别程序流程实际办理时长超过了时限，则计算结果为true。

6，网上办事时限监察（网上办事默认时限2天）

算法： 1、对于网上提交的办件，以受理时间减提交时间，计算超出时间 2、计算超出时间时，必须去掉节假日、特别程序时间

7，网厅时限监察（办件状态9网上提交）

算法：

\* 1、对于在办办件，以系统当前时间减受理时间，计算实际耗时

\* 2、针对正常办结了的办件（一般办结、正常办结、上报办结、转报办结），以办结时间减去受理时间，计算实际耗时

\* 3、计算实际耗时时，必须去掉节假日、特别程序时间

8，计算环节实际处理时长

\* 算法：

\* 1、先计算最后时间与受理时间差异的天数（去掉节假日）。

\* 2、再计算受理当天耗时数加上最后时间耗时数。

\* 3、计算特别程序所耗时间。

\* 4、用1+2-3计算实际耗时，小数进1

\* 5、将计算结果更新到环节表

9，电子证照时限监察（电子证照默认时限3天）

算法： 1、对于需要出证的办件，以出证时间减办结时间，计算超出时间 2、计算超出时间时，必须去掉节假日

在ruleRun方法中：

根据bussiness\_type，分别从不同的视图中查询对应bussiness\_id的监察规则信息，并且是ta\_jc\_rule\_baseinfo中有效（is\_valid = '1'）以及自动运行的（is\_auto\_run = '1'）的监察规则。

bussiness\_type=1：办件数据：从视图V\_RULEBASEINFOBEANALL中查询；

bussiness\_type=2：投诉数据：从视图V\_RULEBASEINFO\_COMPLAIN中查询；

bussiness\_type=3：咨询数据：从视图V\_RULEBASEINFOCONSULT中查询；

然后再封装监察规则的计算表达式，规则定义信息，发牌类型，已处理时间等数据（getRuleWithBusinessList方法）。

循环封装的规则ID，取出规则对应的业务数据ID，执行简单规则的监察方法doRun（目前系统里只处理了简单规则，简单规则就是把预警，黄牌，红牌分别定义一个规则，复杂规则就是把预警，黄牌，红牌放在同一个规则里，所以复杂规则也可转化成简单规则），在doRun方法里循环业务数据ID，从监察信息表TA\_JC\_SUPERVISE\_INFO中查询该业务数据在当前规则ID的当前规则版本下是否已经有监察数据，如果已经监察过，则不再监察（比如已经发了预警牌，就不再发预警牌）即判断superviseInfoId是否存在，如果不存在则再根据上面封装的数据（主要是ruleWithBusinessBean对象），构造表达式，并计算表达式（expreesionServises.execute方法），如果表达式计算的结果为true，表明已经违反了当前监察规则，则执行监察方法。在执行监察方法前依据本次发牌结果数据与对应类型数据表中数据发牌结果对比，暂存入监察临时表（ta\_jc\_supervise\_consult\_temp），需要发牌再执行监察方法。

在执行监察方法doSupervise里，插入监察信息数据TA\_JC\_SUPERVISE\_INFO（发牌），然后发送短信（规则中配置了才发）。