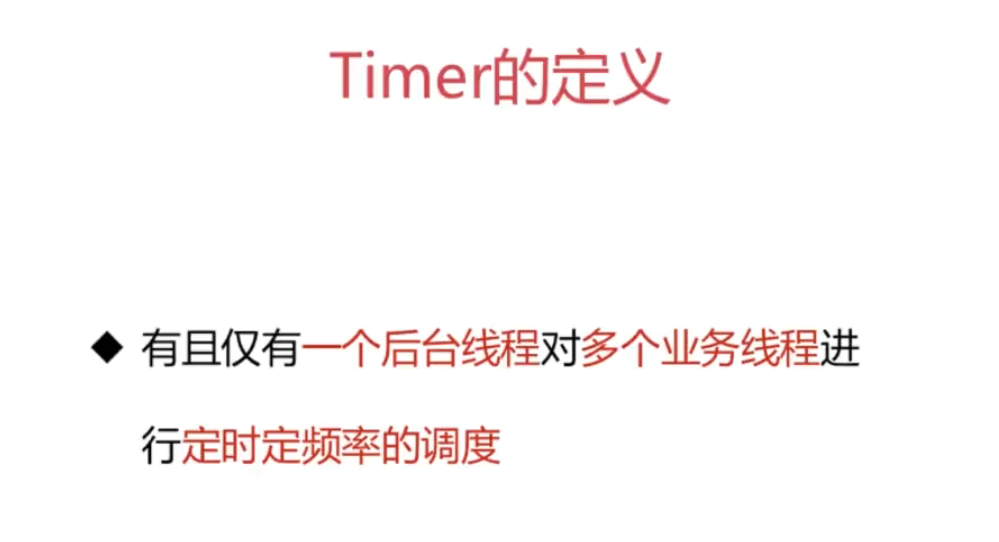
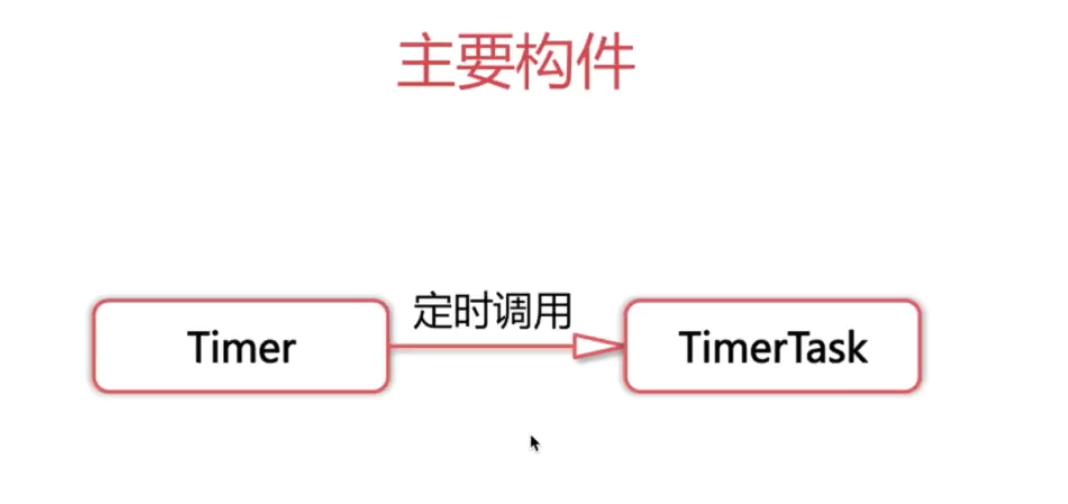
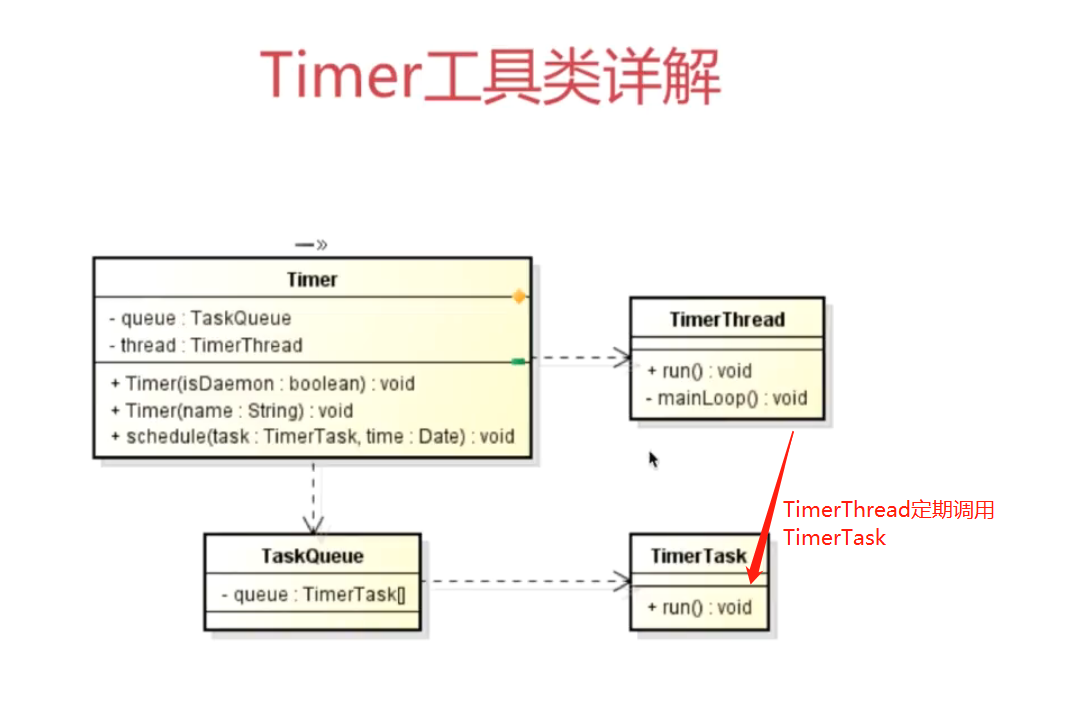
# Timer学习

**小弟：**Timer底层实现只有一个后台线程；

**大哥：**Quartz底层实现有线程池的支持；

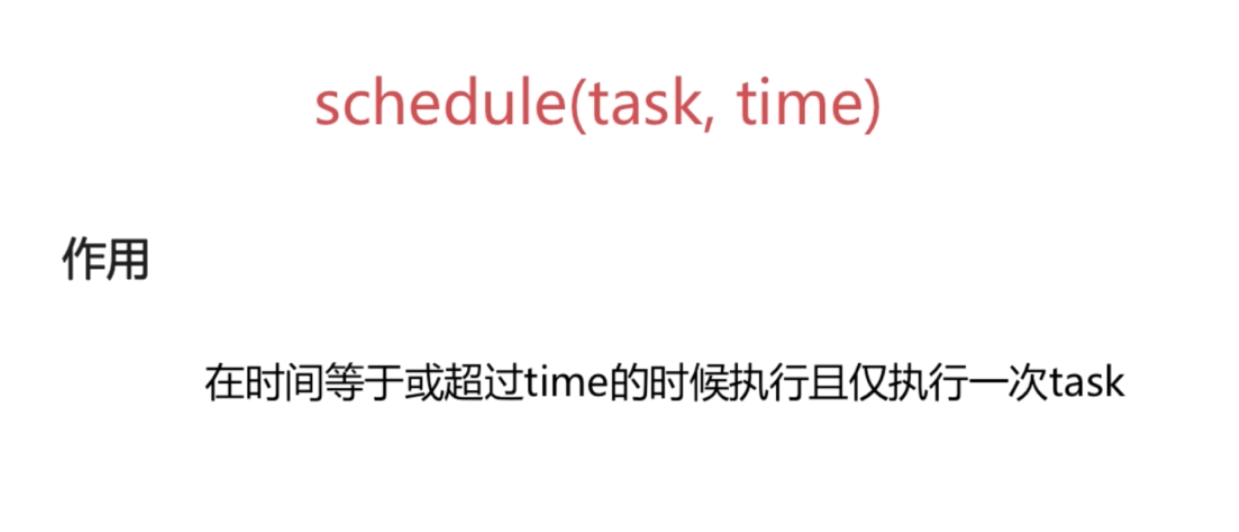


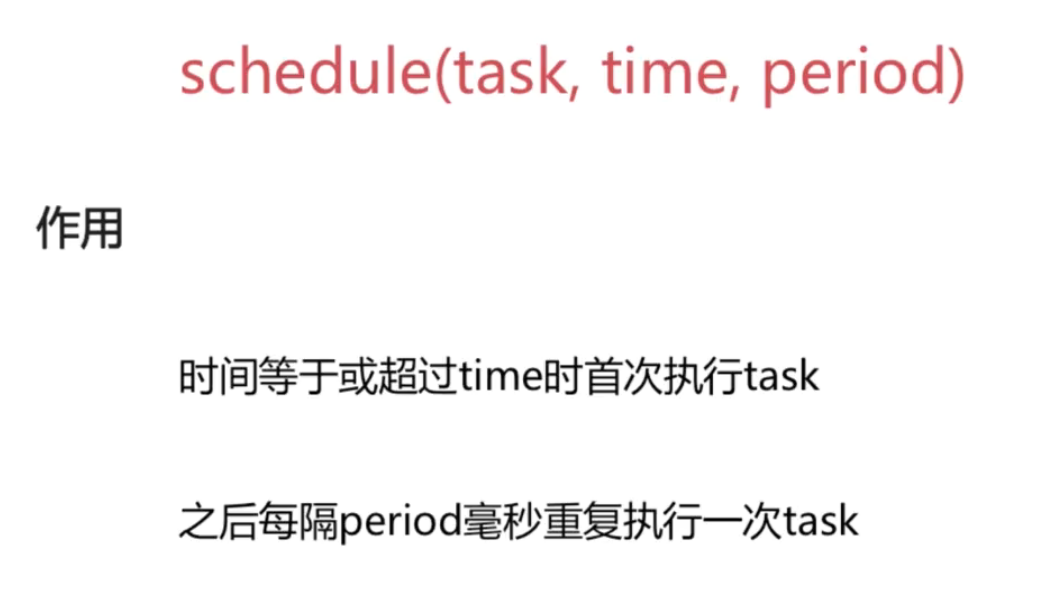


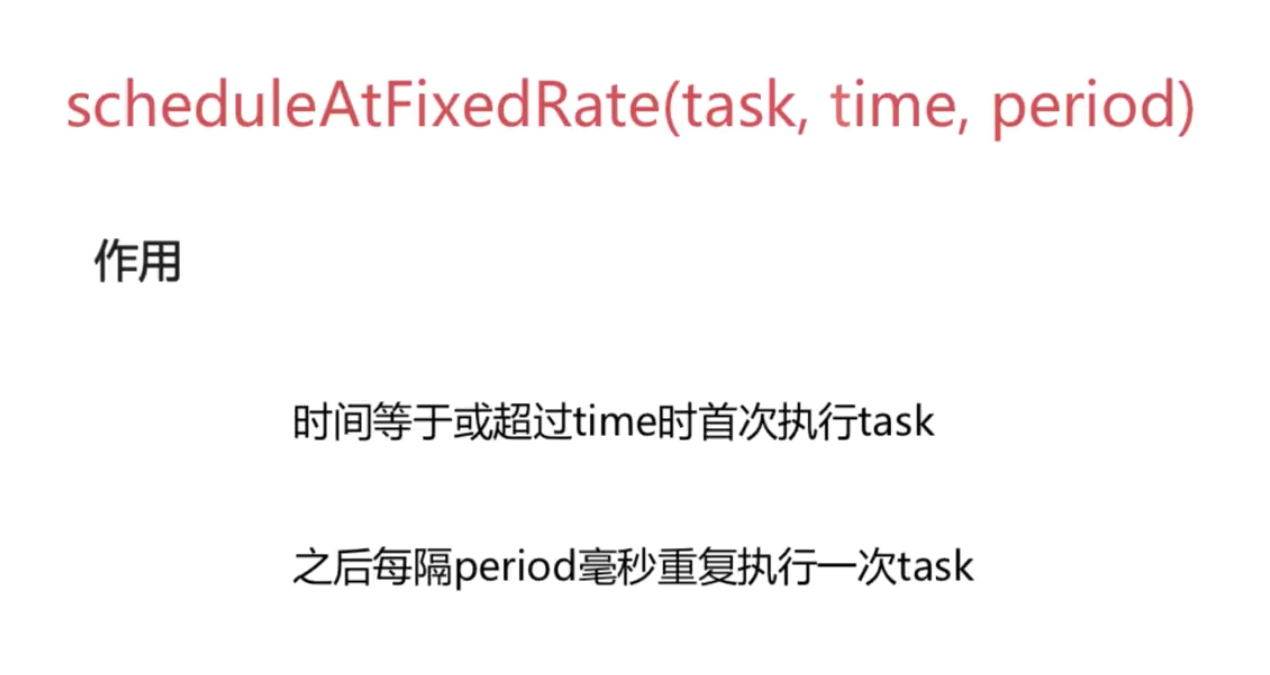


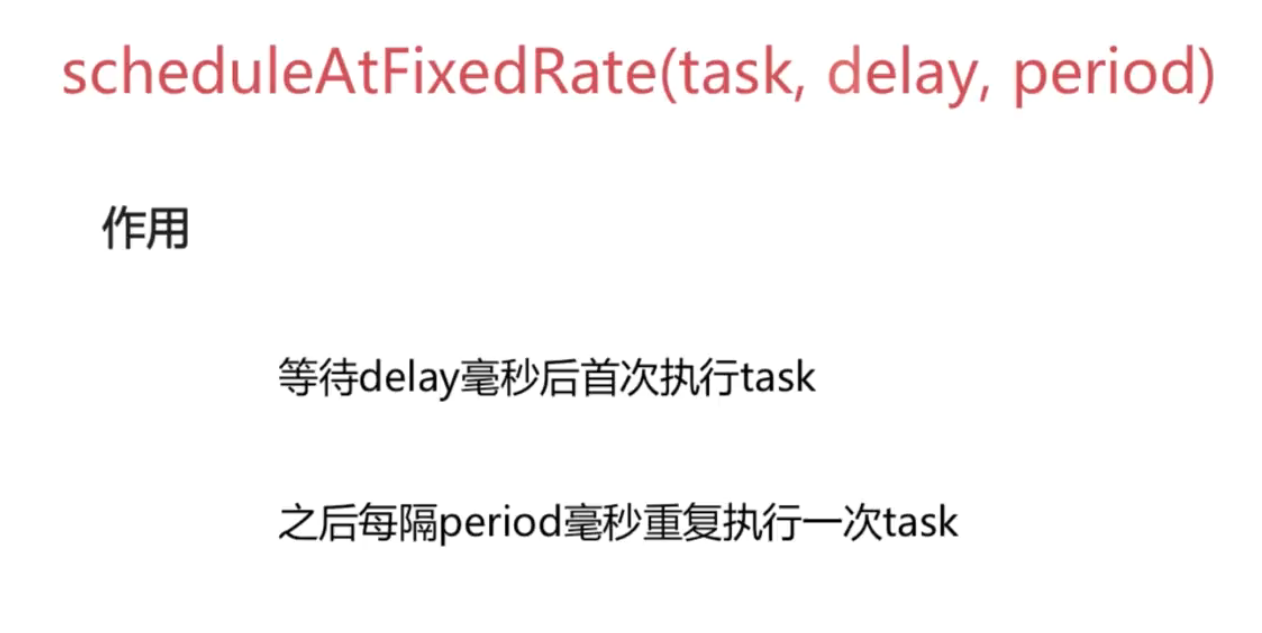
**一个TimerTask只能交由一个Timer来调度；一个Timer可以调度多个TimerTask。**

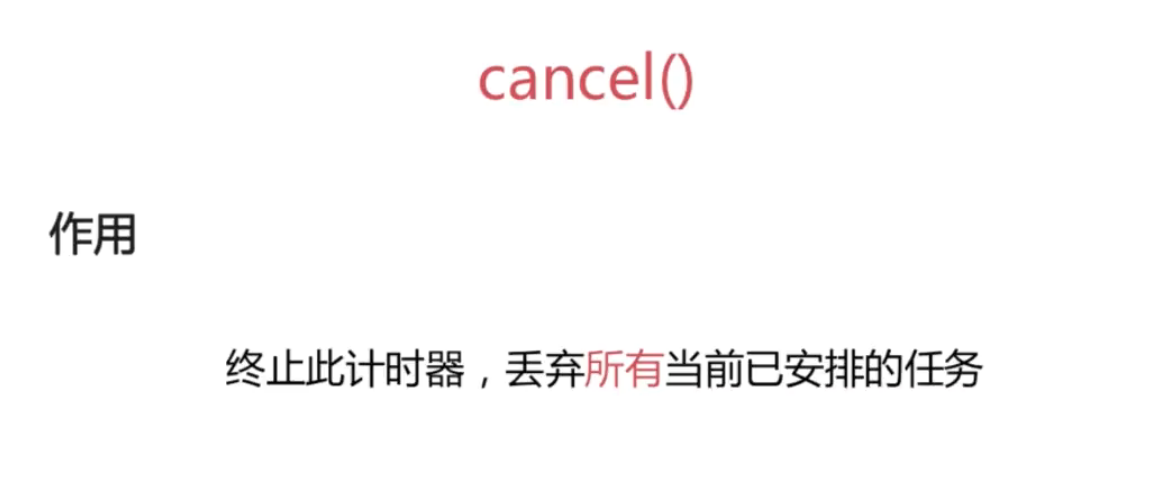
## Timer中比较重要的函数介绍

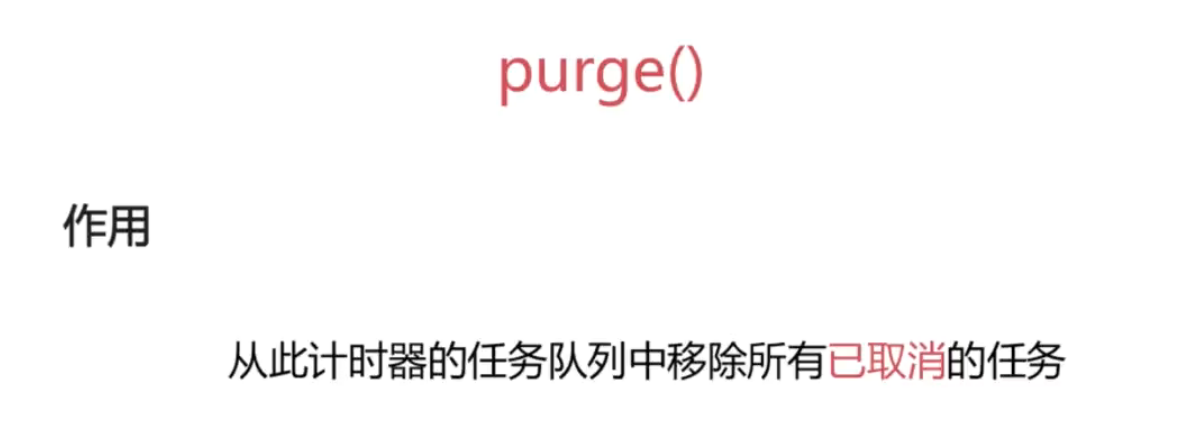




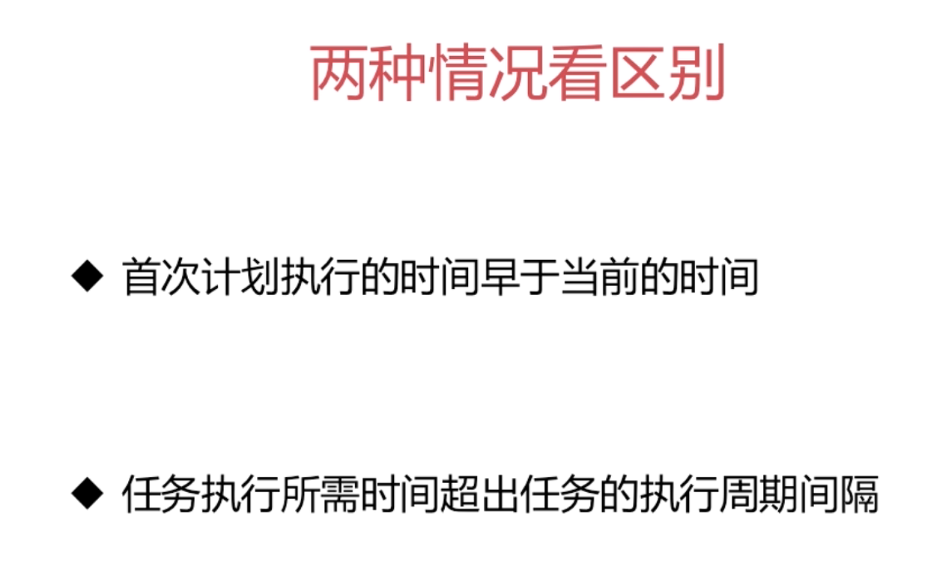




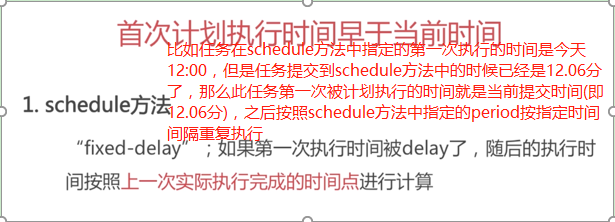




### Timer中Schedule与ScheduleAtFixedRate的区别



#### 首次计划时间早于当前时间：





#### 任务执行时间超出执行周期间隔：

**1、schedule方法实现中每次任务执行完之后才会更新任务的scheduledExecutionTime的值；**

例1：任务预计每隔3s执行一次，每次任务执行实际花费的时间是1s；scheduledExecutionTime开始为12:00:00,任务执行过后时间为12:00:01，预计下次的scheduledExecutionTime为12:00:03，当前时间为12:00:01，并没有超过预计的下次计划时间，所以将下次计划执行时间正常设置为12:00:03等待下次执行。

例2：任务预计每隔3s执行一次，每次任务执行实际花费的时间是5s；scheduledExecutionTime开始为12:00:00,任务执行过后时间为12:00:05，预计下次的scheduledExecutionTime为12:00:03，当前时间为12:00:05，已经超过了预计的下次计划时间，所以将下次计划执行时间设置为12:00:05并直接开始执行。

**2、scheduleAtFixedRate方法实现中每次任务执行之前就会更新任务的scheduledExecutionTime的值；**

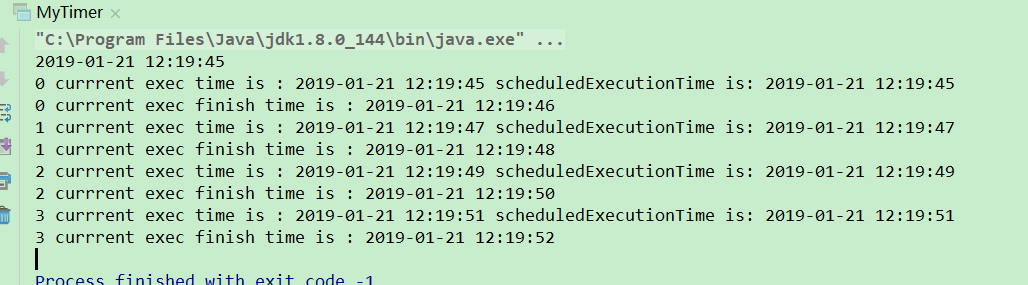
例1：任务预计每隔3s执行一次，每次任务执行实际花费的时间是1s；scheduledExecutionTime开始为12:00:00, **任务执行之前更新**下次的scheduledExecutionTime为12:00:03，任务第一次执行过后时间为12:00:01，并没有超过预计的下次计划时间，所以正常等待下次执行。

例2：任务预计每隔3s执行一次，每次任务执行实际花费的时间是5s；scheduledExecutionTime开始为12:00:00, **任务执行之前更新**下次的scheduledExecutionTime为12:00:03，任务第一次执行过后时间为12:00:05，已经超过预计的下次计划时间，所以不再等待，先更新下次执行时间为12:00:06,并开始执行任务。

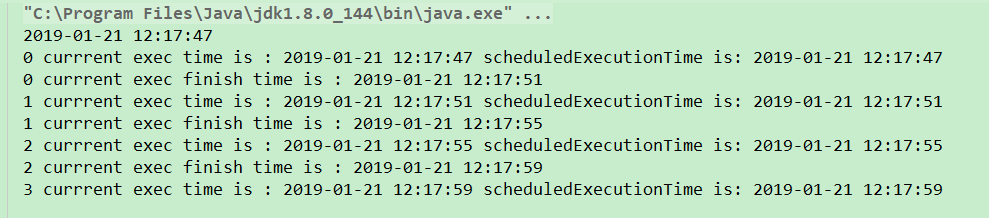
**需要注意的是，不论在单线程还是多线程的情况下，被schedule方法调度的Task都仅会被单线程执行(Timer默认情况下内部使用的是单个后台线程)；但在使用多个后台线程调度任务的情况下，如果出现了上面介绍的两种情况，被scheduleAtFixedRate方法调度的Task是可能被多线程并发执行的。**

**Schedule单后台线程调度实例：**

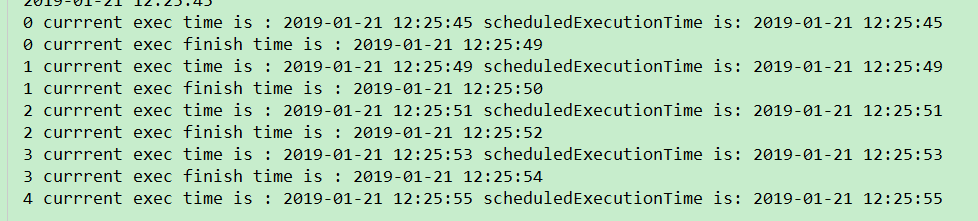
每隔2s执行一次，函数执行时间1s；首次计划执行时间：12.19.45首次实际执行时间：12.19.45



每隔2s执行一次，函数执行时间4s；首次计划执行时间：12.17.47首次实际执行时间：12.17.47

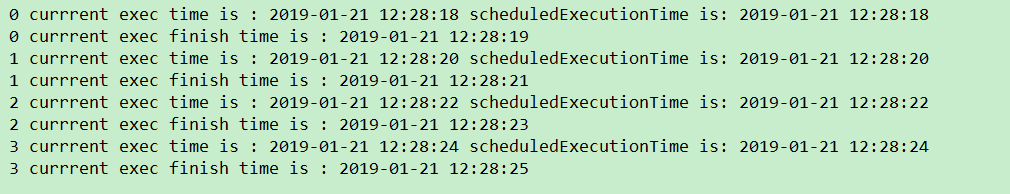


每隔2s执行一次，函数第一次执行时间为4s，之后执行时间为1s：首次计划执行时间：12.25.45首次实际执行时间：12.25.45

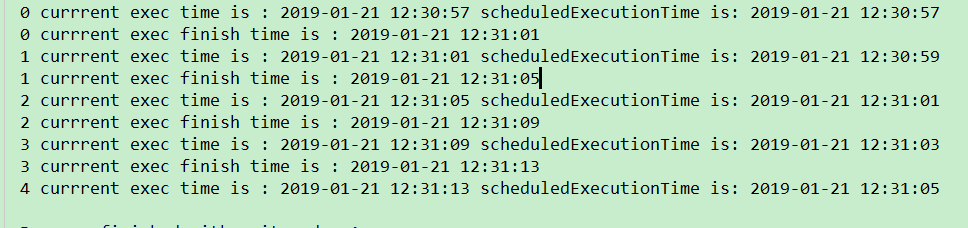


**scheduleAtFixedRate单后台线程调度实例：**

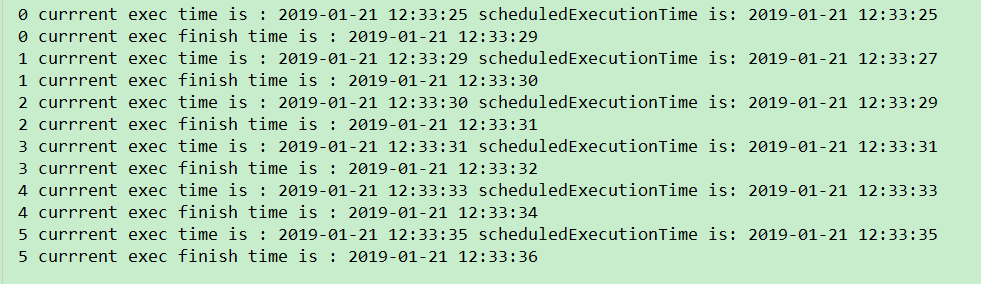
每隔2s执行一次，函数执行时间1s；首次计划执行时间：12.28.18首次实际执行时间：12.28.18



每隔2s执行一次，函数执行时间4s，首次计划执行时间：12.30.57首次实际执行时间：12.30.57

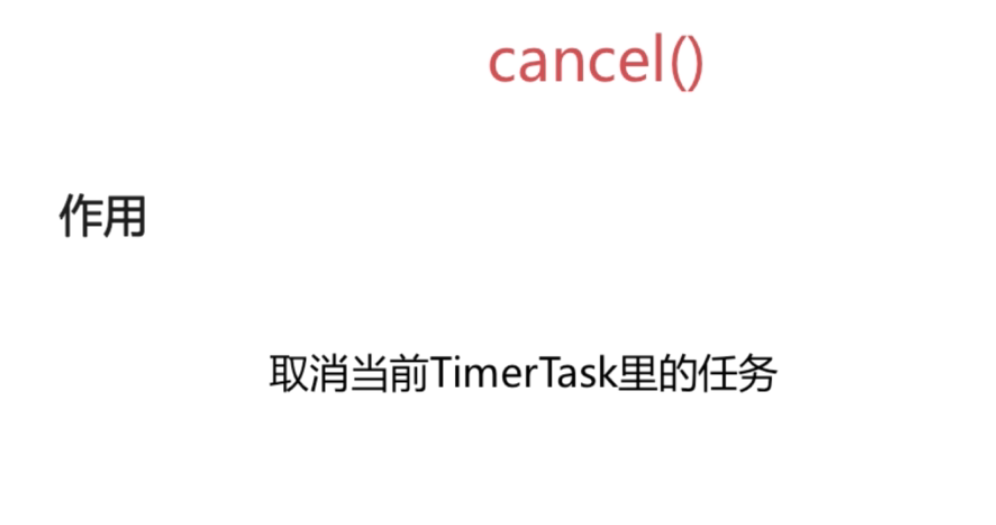


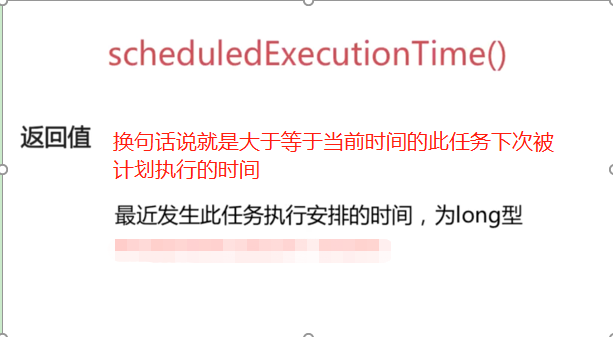
每隔2s执行一次，函数第一次执行时间为4s，之后执行时间为1s，首次计划执行时间：12.33.25首次实际执行时间：12.33.25



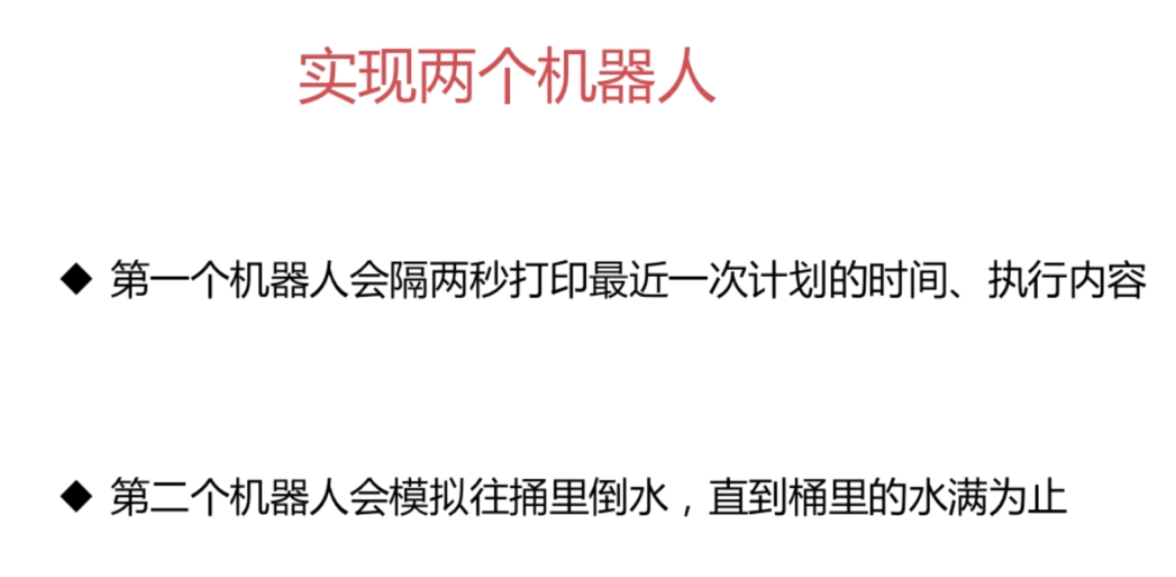
## TimerTask中比较重要的函数介绍

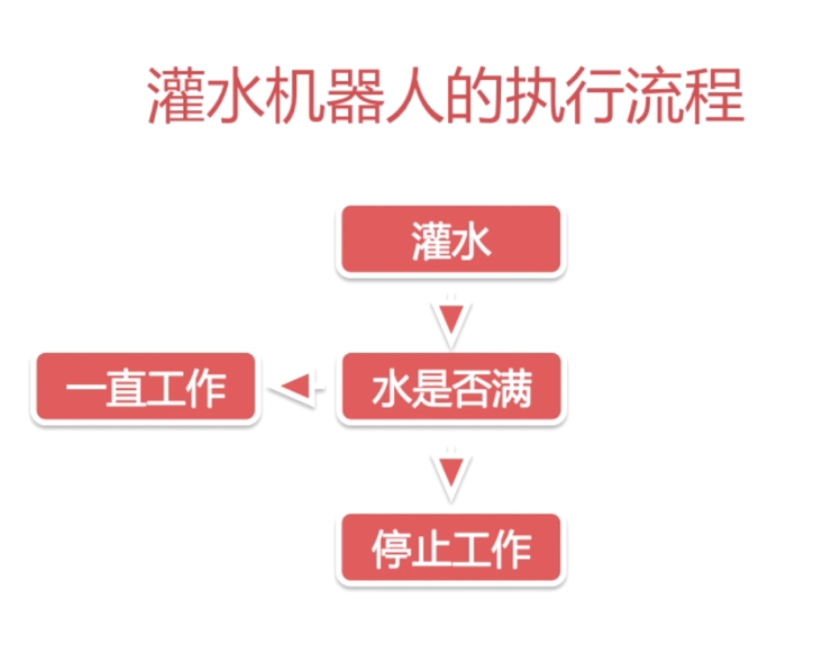


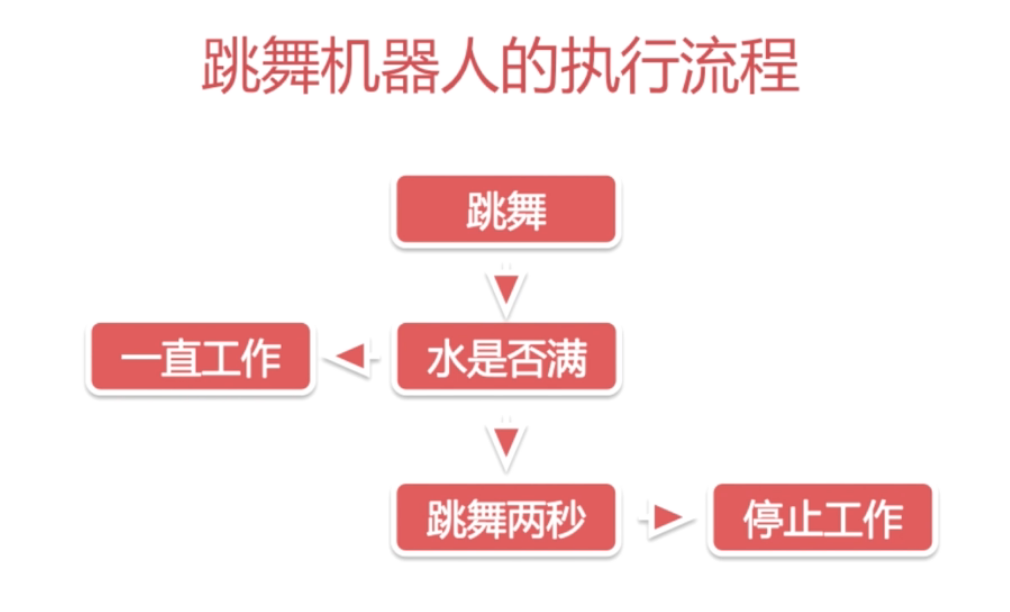




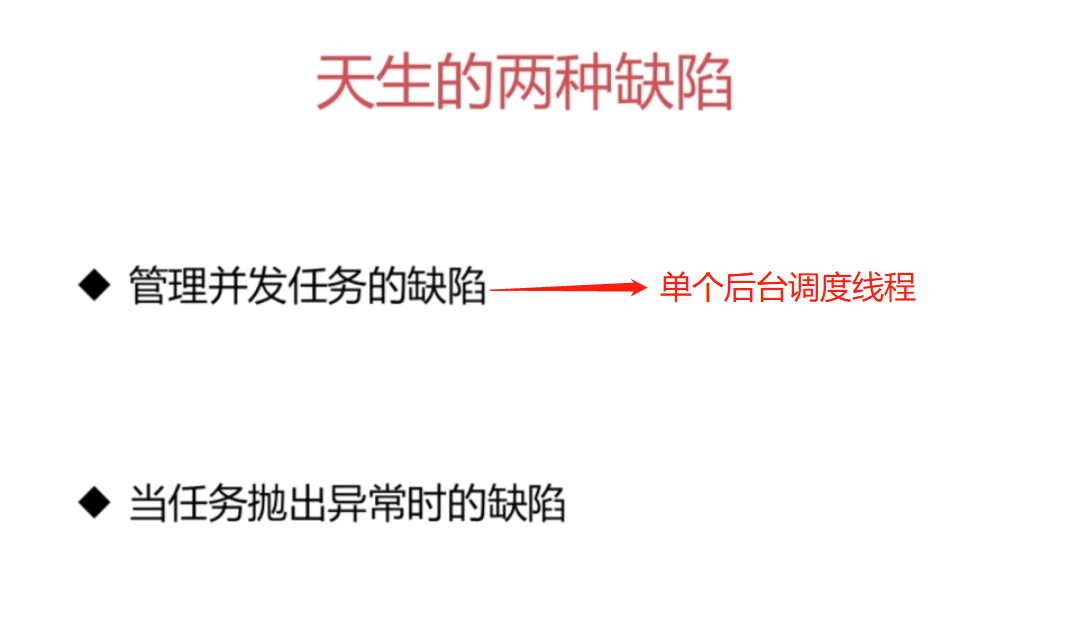
## Timer函数的综合应用

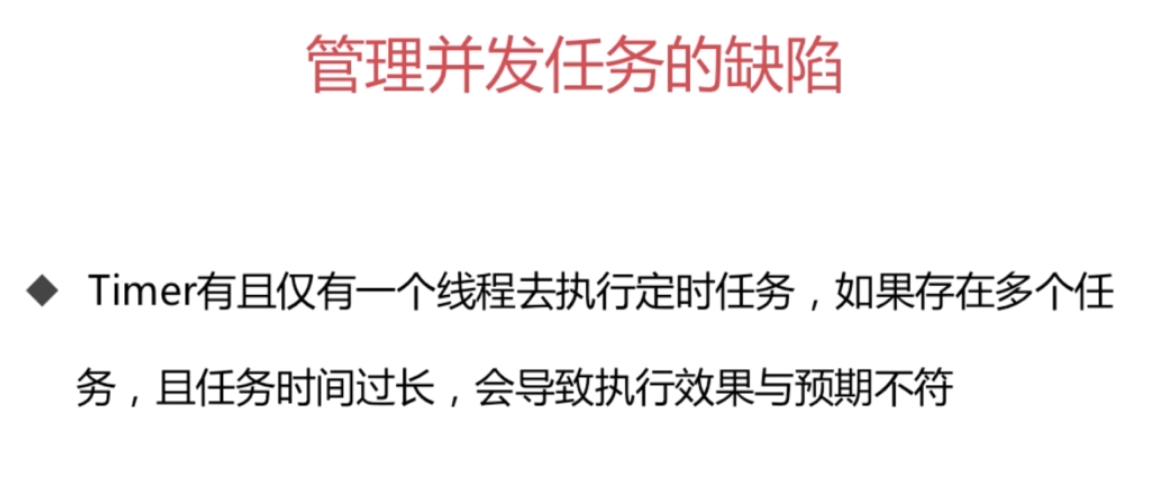


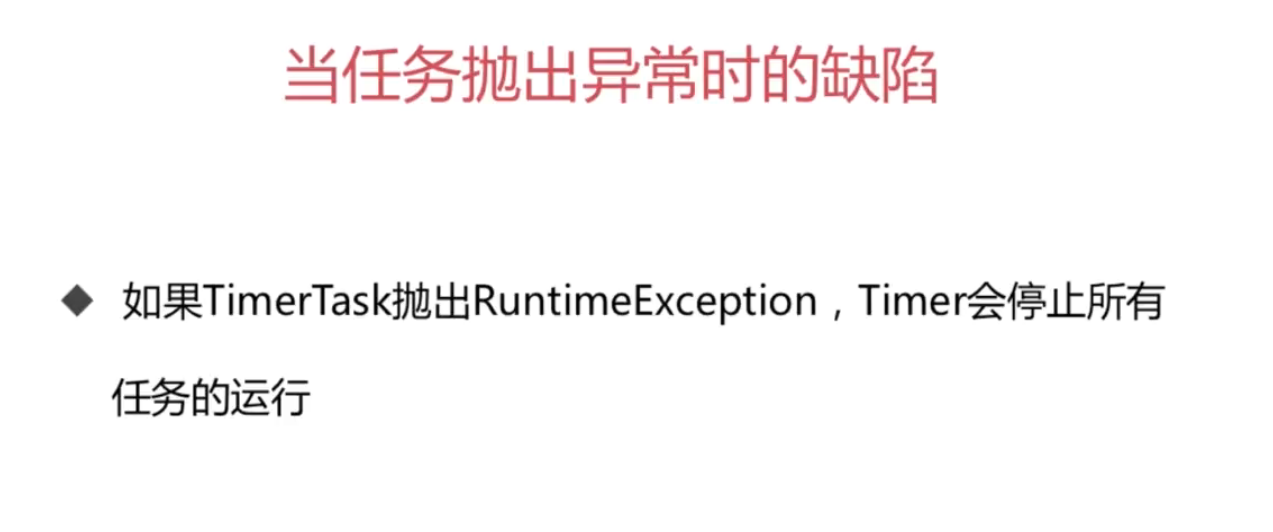


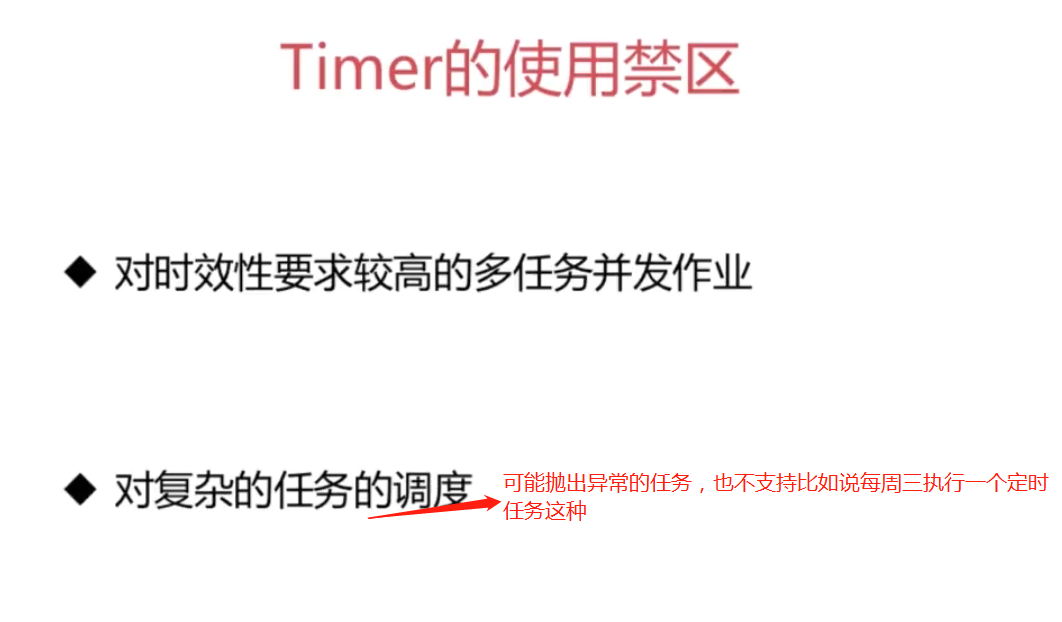


## Timer的缺陷









Timer开启的后台线程在在对Timer对象的最后一次实时引用消失并且所有未完成的任务都已完成执行之后，计时器的任务执行线程正常终止（并且变为垃圾回收）。 但是，这可能会发生在任意长的时间之后，默认情况下，任务执行线程不作为守护程序线程运行，因此它能够阻止应用程序终止。 如果调用者想要快速终止计时器的任务执行线程，则调用者应该调用计时器的cancel方法。

# Quartz学习