1. Timer定时器
2. 概述

JDK库中，Timer类负责计划任务，即确定在指定的时间开始执行某一个任务，而任务代码需要封装在TimeTask类的子类中。

1. Timer类的方法
2. schedule(TimeTask task, Date time)

在指定的日期执行一次任务，当指定的日期早于当前的时间时，它会从当前时间开始执行；当指定的日期晚于当前的时间时，它会从指定的日期开始执行。对于多个TimeTask任务，这些任务是以队列的方式按照顺序执行的，因此后面的任务执行的时间可能要随着每个任务的耗时来决定，可能有任务会超时导致后面的任务延时启动。

创建TimerTask类的子类：

package timer;  
  
import java.sql.Timestamp;  
import java.util.Date;  
import java.util.TimerTask;  
  
public class MyTaskA extends TimerTask {  
 @Override  
 public void run() {  
 System.*out*.println("A start " + new Date());  
 System.*out*.println("A end " + new Date());  
 }  
}

创建测试类：

package timer;  
  
import java.util.Calendar;  
import java.util.Date;  
import java.util.Timer;  
  
public class Test {  
 public static void main(String[] args) {  
 System.*out*.println("当前时间 " + new Date());  
 Calendar calendar = Calendar.*getInstance*();  
 Date time = calendar.getTime();  
 System.*out*.println("A计划时间 " + time);  
 MyTaskA myTaskA = new MyTaskA();  
 Timer timer = new Timer();  
 timer.schedule(myTaskA, time);  
 }  
}

1. schedule(TimerTask task, Date firstTime, long period)

在指定的日期之后按指定的间隔周期，无限循环地执行某一任务。如果计划时间早于当前时间，则立即执行任务，且循环执行；如果计划时间晚于当前时间，则在计划时间循环执行任务；若任务耗时超过了间隔时间，则下一个任务的开始时间以超时的任务结束时间为准。

1. cancel()

Timer类的cancel()方法将任务队列中的所有任务都清空，且进程被销毁。不过有时cancel()方法也抢不到queue锁，让任务正常执行，总会有这样的情况出现。

TimerTask类的cancel()方法将自身这个任务从任务队列中清除。

1. schedule(TimerTask task , long delay)

以执行schedule方法当前的时间为基准，在此基础上延迟指定的毫秒数后执行一次TimerTask任务。

1. schedule(TimerTask task . long delay, long period)

以执行schedule方法当前的时间为基准，在此基础上延迟指定的毫秒数后循环执行TimerTask任务。

1. scheduleAtFixedRate方法与 schedule方法的区别

scheduleAtFixedRate方法具有追赶性，当任务的计划时间在当前时间之前时，schedule直接从当前时间开始执行，忽略计划时间到当前时间之间的任务。而scheduleArFixedRate方法会将计划时间到当前时间之间的任务补充性地执行。