软件复用课程设计

Client-Server 应用程序

复用文档

License构件

小组： Team10

成员： 谢志杰 1352975

计鹏玥 1352914

王思尧 1352896

谢明玥 1352937

徐锦程 1353012

[1． 构件简介 1](#_Toc4804)

[2． 接口说明 1](#_Toc20246)

[2.1 License.java 1](#_Toc23069)

[(1) License(int max\_msg\_in\_second, int max\_num\_message, int firstTime, int period) 1](#_Toc26942)

[(2) setTime(int firstTime, int period) 2](#_Toc6829)

[(3) commence() 2](#_Toc23452)

[(4) cancel() 2](#_Toc1031)

[(5) setMax(int a, int b) 3](#_Toc18516)

[(6) increaseMsg() 3](#_Toc1331)

[(7) reset() 4](#_Toc10738)

[(8) checkMsgInSecond() 4](#_Toc32656)

[(9) checkTotalMsg() 4](#_Toc29017)

[2.1 MessageCount.java 5](#_Toc18821)

[(1) MessageCount() 5](#_Toc26650)

[(2) getMsgInSecond() 5](#_Toc18436)

[(3) getTotalMsg() 6](#_Toc8823)

[(4) increaseMsg() 6](#_Toc23797)

[(5) reset() 7](#_Toc26282)

[(6) run() 7](#_Toc6639)

[3．Demo 7](#_Toc23328)

# 构件简介

|  |  |  |
| --- | --- | --- |
|  | **许可证 － License** | **消息计数 － MessageCount** |
| 构件介绍 | 这一构件的主要功能是对消息的收发设置限制条件。 | 这一构件的主要功能是对于消息的计数功能，包括对于每分钟消息的计数以及对于消息总数的计数。 |
| 功能说明 | 用户可以使用该构件来设置每分钟消息收发最大数目、登录后收发的消息总数、在计数器规定范围内收发的消息数目等。 | 用户可以使用该构件实现对于消息的计数功能，计数包括每分钟消息的计数以及消息总数的计数。 |

# 接口说明

## 2.1 License.java

注意：应先导入License包。

### (1) License(int max\_msg\_in\_second, int max\_num\_message, int firstTime, int period)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| max\_msg\_in\_second | int | 每分钟收发消息的最大数 |
| max\_num\_message | int | 总收发消息的最大数 |
| firstTime | int | 计时器开启的时间 |
| period | int | 计时器的周期 |

**作用描述：**

这个方法标识了许可证中所包含的信息内容，其中有每分钟消息最大值、总消息最大值、计时器开启时间、计时器周期。

**使用方法：**

对于一个实例化的License对象，可以直接调用该方法，该方法输入四个int类型的参数，没有返回值，形如：

License license = new License(5, 100, 0, 60000);

**注意事项：**

在实例化License对象的时候，应传入四个整型的参数。

### (2) setTime(int firstTime, int period)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| firstTime | int | 计时器开启时间 |
| period | int | 计时器的周期 |

**作用描述：**

该方法用于设置消息计数的开始时间和周期长度。

**使用方法：**

对于一个实例化的License对象，可以直接调用该方法，该方法有两个int类型的参数，分别代表messageCount开始执行的时间和其执行周期长度，没有返回值，形如：

license.setTime(0, 60000);

### (3) commence()

**作用描述：**

该方法用于开启消息计数的计时器。

**使用方法：**

对于一个实例化的License对象，可以直接调用该方法，该方法没有输入参数，返回值为空，形如：

license.commence();

**注意事项：**

必须先对firstTime和period进行初始化，并对messageCount进行实例化才可调用。

### (4) cancel()

**作用描述：**

关闭计时器。

**使用方法：**

对于一个实例化的License对象，可以直接调用该方法，该方法既没有输入参数，也没有返回值，形如：

license.cancel();

**注意事项：**

只有在实例化计时器之后才可以将计时器关闭。

### (5) setMax(int a, int b)

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 含义 |
| a | int | 每分钟收发消息最大数 |
| b | int | 总收发消息最大数 |

**作用描述：**

该方法用于设置每分钟收发消息最大数和总收发消息的最大数。

**使用方法：**

对于一个实例化的License对象，可以直接调用该方法，该方法的输入两个int类型的参数，即可赋值，没有返回值，形如：

license.setMax(5, 100);

**注意事项：**

该方法的两个输入参数不惜为非负整数。

### (6) increaseMsg()

**作用描述：**

该方法将消息数进行加一。

**使用方法：**

对于一个实例化的License对象，可以直接调用该方法，该方法的无输入参数，也没有返回值，形如：

license.increaseMsg();

**注意事项：**

必须先对MAX\_MSG\_IN\_SECOND和MAX\_NUM\_MESSAGE进行初始化。

### (7) reset()

**作用描述：**

该方法将消息数重置为0。

**使用方法：**

对于一个实例化的License对象，可以直接调用该方法，该方法的没有输入参数和返回值，形如：

license.reset();

**注意事项：**

必须先对messageCount进行实例化。

### (8) checkMsgInSecond()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| true/false | boolean | 判断每分钟获取的消息数是否超过了max\_msg\_in\_second |

**作用描述：**

该方法判断消息计数器每分钟获取的消息数是否超过了MAX\_MSG\_IN\_SECOND。

**使用方法：**

对于一个实例化的License对象，可以直接调用该方法，该方法的没有输入参数，返回值为布尔值，形如：

boolean msgInSecond = license.checkMsgInSecond();

**注意事项：**

必须先对messageCount进行实例化。

### (9) checkTotalMsg()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| true/false | boolean | 判断获取的总消息数是否超过了max\_num\_message |

**作用描述：**

该方法断消息计数器获取的消息总数是否超过了MAX\_NUM\_MESSAGE。

**使用方法：**

对于一个实例化的License对象，可以直接调用该方法，该方法的没有输入参数，返回值为布尔值，形如：

boolean msgTotal = license.checkMsgInSecond();

**注意事项：**

必须先对messageCount进行实例化。

## 2.1 MessageCount.java

注意：应先导入License包。

### (1) MessageCount()

**作用描述：**

将每秒钟的消息数和总消息数都初始化为0。

**使用方法：**

对于一个实例化的DataSource对象，可以直接调用该方法，该方法没有参数和返回值，形如：

MessageCount messageCount = new MessageCount();

**注意事项：**

此方法在实例化MessageCount对象时被调用。

### (2) getMsgInSecond()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| msg\_in\_second | Int | 每分钟收发的消息数 |

**作用描述：**

该方法用于获取每分钟消息的计数数额。

**使用方法：**

对于一个实例化的DataSource对象，可以直接调用该方法，该方法没有参数，返回值为int类型，代表每分钟消息的总数，形如：

messageCount.getMsgInSecond();

**注意事项：**

该方法返回一个int类型的参数，代表每分钟消息的计数数值。

### (3) getTotalMsg()

|  |  |  |
| --- | --- | --- |
| 返回值 | 类型 | 含义 |
| num\_message | Int | 总收发的消息数 |

**作用描述：**

该方法用于获取消息的计数数额。

**使用方法：**

对于一个实例化的DataSource对象，可以直接调用该方法，该方法没有参数，返回值为int类型，代表消息的计数总额，形如：

messageCount.getTotalMsg();

**注意事项：**

该方法返回一个int类型的参数，代表消息的计数数值。

### (4) increaseMsg()

**作用描述：**

该方法用于将消息计数加1，包括每分钟消息计数与总消息计数。

**使用方法：**

对于一个实例化的DataSource对象，可以直接调用该方法，该方法没有参数，也没有返回值，形如：

messageCount.increaseMsg();

**注意事项：**

该方法会对每分钟消息计数与总消息计数均加1。

### (5) reset()

**作用描述：**

该方法用于将消息计数归零，包括每分钟消息计数与总消息计数。

**使用方法：**

对于一个实例化的DataSource对象，可以直接调用该方法，该方法没有参数，也没有返回值，形如：

messageCount.reset();

**注意事项：**

该方法会对每分钟消息计数与总消息计数均归零。

### (6) run()

**作用描述：**

该方法继承于TimerTask，可以用在定时器中，主要功能为将每分钟消息计数归零。

**使用方法：**

该方法一般用于定时器中，启用定时器后，可以实现每单位时间每分钟消息计数归零，形如：

timer.schedule(messageCount, 0, 1000);

**注意事项：**

该方法一般用于定时器，如上所示，也可以单独使用，单独使用时该方法没有参数也没有返回值。

# 3．Demo

class ServerThread {  
 private int MAX\_MESSAGE\_PER\_SECOND = 5;  
 private int MAX\_MESSAGE\_FOR\_TOTAL = 10;  
 private License license;  
  
 public void work() {  
 license = new License(1,1,1,1);  
 try {  
 // 设置指标。设置第一次执行延迟时间，以及以后每次执行间隔。  
 license.setMax(MAX\_MESSAGE\_PER\_SECOND, MAX\_MESSAGE\_FOR\_TOTAL);  
 license.setTime(0, 60000);  
 // 使用前重置License。  
 license.reset();  
 // 启动内部计时器。  
 license.commence();  
  
 String line = in.readLine();  
 while(true) {  
 // 检查每秒信息量是否超标。  
 if (license.checkMsgInSecond()) {  
 // 未超标，计数加一。  
 license.increaseMsg();  
 ...  
 } else {  
 ...  
 }  
 // 检查总信息量是否超标。注意取反。  
 if (!license.checkTotalMsg()) {  
 ...  
 // 完成工作后，重置计数。  
 license.reset();  
 }  
 }  
 line = in.readLine();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 // 停止内部计时器。  
 license.cancel();  
 }  
}  
}