

课程设计报告

（ 18 / 19 学年 第 一 学期）

题 目：**利用Socket通信实现停机/复话处理**

**专 业 物联网工程**

**学 生 姓 名 黄黎明**

**班 级 学 号 B16070706**

**指 导 教 师 吴振宇**

**指 导 单 位 南京邮电大学**

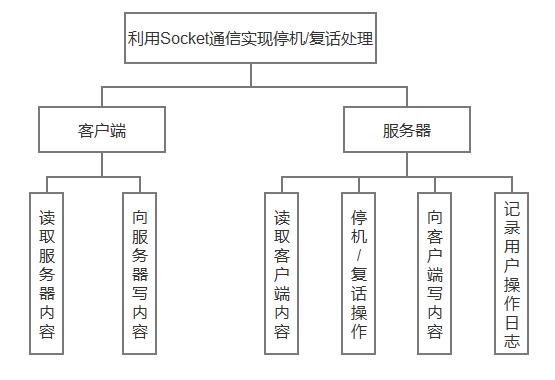
**日 期 2018-12-7**

**利用Socket通信实现停机/复话处理**

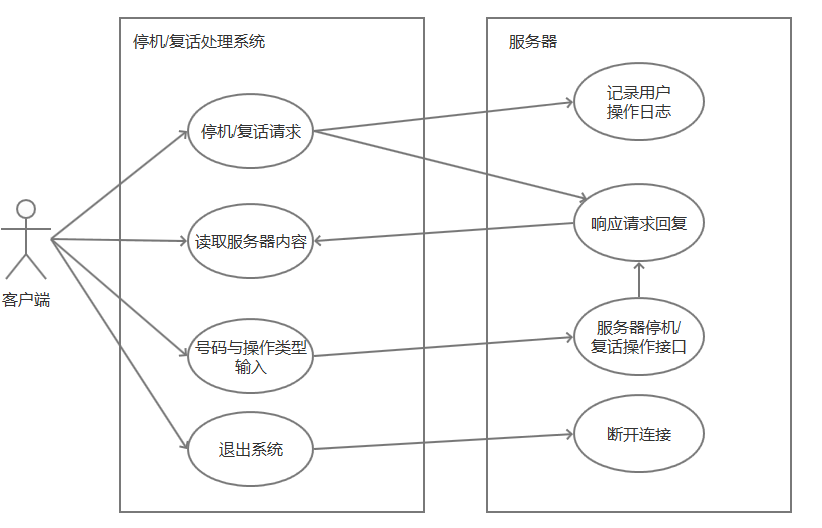
1. **概述**
2. 课题目标与任务

首先生成业务的订单，然后调用停机/复话接口执行相应操作，在施工结果后，进行业务完工处理。本课题要求依据上述需求设计并实现停机/复话业务处理功能。

1. 课题内容描述
2. 停机/复话请求生成。输入需要操作的用户号码和所做的操作，然后生成停机/复话请求，并将请求发送给服务器；
3. 停机/复话服务器接口的设计和实现。要求应用Socket进程通信技术编写一个服务器程序，以模拟停机/复话操作。当服务器接收到客户端发送的操作请求后，在客户端屏幕上显示请求内容：号码和操作类型，收到号码和操作类型并处理结束后，服务器返回0给客户端表示操作成功，否则返回1表示操作失败；
4. 设计关键字驱动程序，采用关键字驱动测试方法测试停机/复话服务接口。要求开发关键字驱动代码，对停机、复话服务2个接口进行测试。
5. 采用Spooling实现停机/复话服务器：允许客户端申请对一批号码执行停机/复话的操作，并要求服务器在硬盘上保存对每个号码的操作日志。
6. 分析服务器是否可能会出现死锁。
7. 课题内容分析
8. 客户端所需完成的功能：通过socket与服务器建立连接；生成停机/复话请求并发送；单一或批量输入需操作的用户号码和所做的操作并发送给服务器；生成与服务器断开连接请求。
9. 服务器所需完成的功能：通过socket与客户端建立连接；响应来自客户端的停机/复话请求；设计和实现停机/复话服务器接口，模拟停机/复话操作；保存用户对每个号码的操作日志。
10. **需求分析**
11. 利用socket通信技术实现客户端和服务器的连接与通信
12. 服务器：设计服务器socket通信中读和写操作于一体的线程，在进程中使用accept方法接受来自客户端的连接请求，并启动读写线程。
13. 客户端：设计客户端socket通信的读和写操作线程，在客户端进程中使用套接字（IP地址和端口号）建立和服务器的连接，并启动读线程和写线程。
14. 服务器端的停机/复话操作以号码和操作类型作为操作对象，将号码和操作类型封装为一个类，再将其实例化为对象，继而对实例化对象进行停复机操作。
15. 服务器需要对客户端发来的内容进行识别，再执行停机/复话操作及确定最终回复内容。当用户发来停机/复话请求，服务器回复“号码和操作类型”；当用户发来号码和操作类型，执行停机/复话操作并回复“0成功”或“1失败”；当用户发来未知请求时，服务器回复“请求有误，重新输入”；当用户发来断开连接请求，服务器和客户端断开连接。
16. 引入java的logback日志框架以实现服务器在硬盘上保存对每个号码的操作日志功能。
17. 通过java性能监控工具VisualVM来观察线程运行情况，由此分析服务器是否可能会出现死锁。
18. **设计与实现**
19. 需求分析和功能设计：



图一：功能框图



图二：用例图

如图一功能框图及图二用例图，其中客户端实现读取服务器发送来的内容及通过键盘写入请求并发送给服务器的功能，服务器实现读取客户端发送来的内容、执行停机/复话操作、对客户端请求进行回复和记录用户操作日志，响应用户退出请求断开连接的功能。

客户端的读写功能分别设计为两个线程，并在与服务器建立连接后保持启动状态。在客户端读线程中，客户端将所读取的消息打印至屏幕；在客户端写线程中，线程读取客户键盘输入的信息并写入服务器。

服务器的读写功能设计为一个线程，并在服务器建立后保持启动状态。读写线程中，服务器通过判断客户端发送来的内容执行相应操作或回复相应内容。服务器将客户端发来的内容分为四种情况：停机/复话请求、号码与操作类型、退出请求和未知请求。当收到停机/复话请求时，服务器回复号码和操作类型请求；当收到号码和操作类型时，服务器调用停机/复话接口方法执行停机/复话操作；当收到退出请求时，服务器断开与客户端之间的socket通信；当收到未知请求时，服务器回复重新输入的请求。读写线程中，服务器将收到的客户操作内容全部记录于日志文件中，记录格式中包含记录日期、时间、线程名、级别和内容。

1. 软件结构设计：

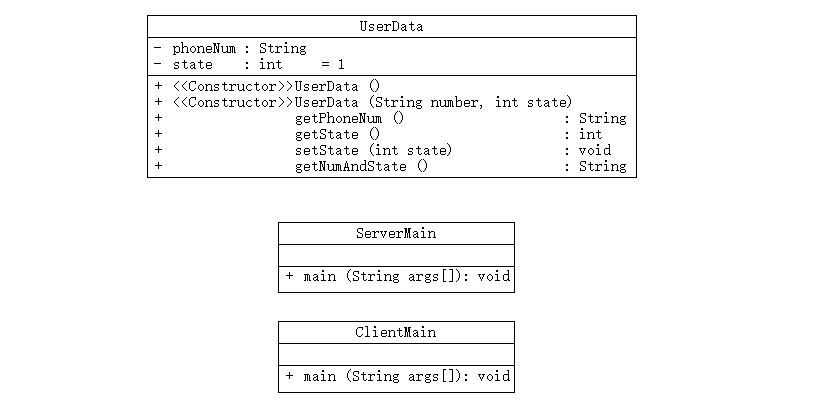
利用socket通信实现停机/复话的功能需要两个主程序，其中一个作为客户端，另一个作为服务器。客户端由客户进行操作，服务器通过语言处理自动执行相应操作并向客户端回复内容。

两个主程序通过socket通信方式进行连接，进而实现服务器与客户端互相通话的功能。服务器程序完成建立端口、等待客户端连接并启动一个读写于一体的线程的功能，客户端程序完成连接服务器并启动两个分别为读与写的线程的功能。

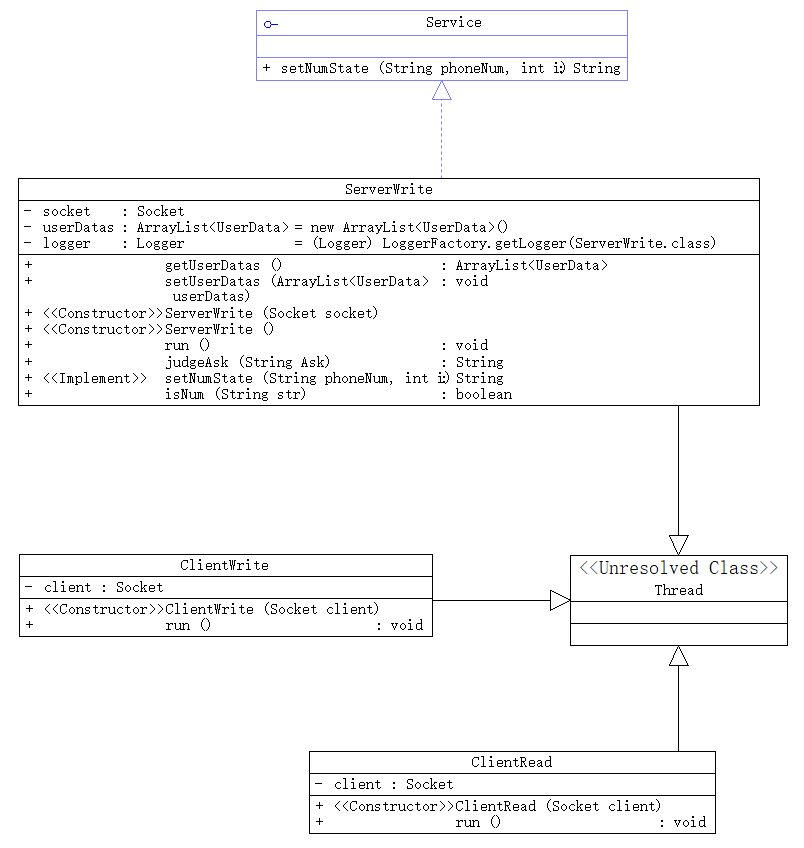
服务器的停机/复话操作被设计为一个接口方法，并在服务器读写线程中得到具体实现。用户所输入的号码和号码的停机、正常状态被封装为一个类，停机/复话的一系列操作将在这个类的对象上进行。

服务器的日志记录功能通过引入java的logback框架实现，最终的日志可在控制台或日志文件中打印输出，日志文件保留时间为三十天，日志文件最大容量为10MB，日志内容输出级别为INFO。

1. 核心算法及数据结构设计：
2. 核心数据结构设计



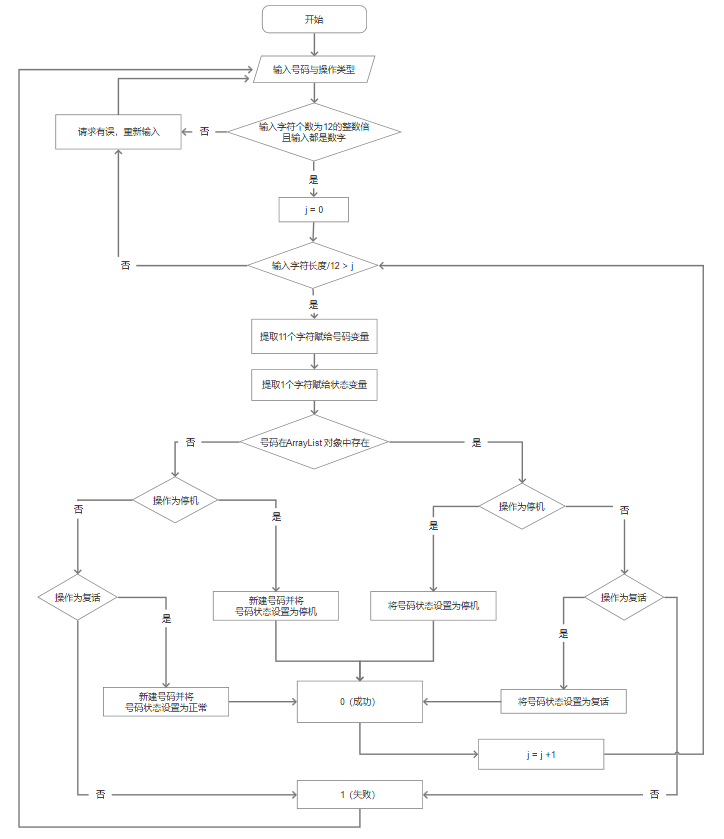
图三：程序类图（一）



图四：程序类图（二）

如图三和图四的程序类图所示，UserData类定义了String型变量、int型变量和一些访问器、修改器；ServerWrite类定义了Socket类型变量、ArrayList<UserData>动态数组、Logger类对象和部分读取内容判断及处理的方法；ClientWrite类、ClientRead类和ServerWrite类继承了Thread类；ServerWrite类实现Service接口的setNumState方法。

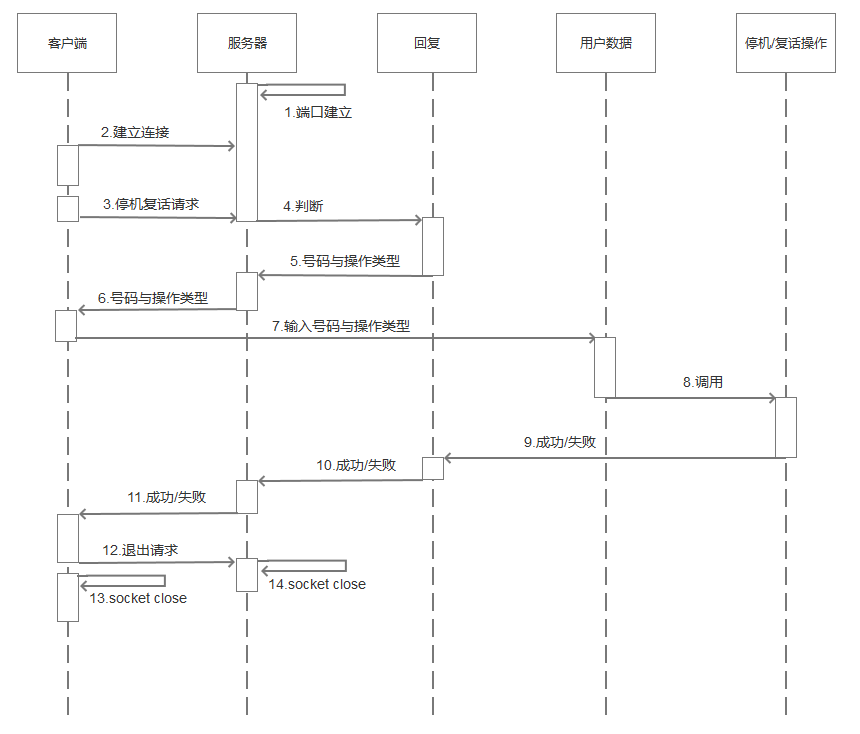
1. 核心算法设计



图五：停机/复话操作流程图

如图五，该流程图为用户输入号码和操作类型后，服务器执行停机/复话操作的过程，使用简单的if-else判断最终的回复内容。

1. 核心组件设计



图六：程序时序图

核心组件：服务器的请求处理响应、用户数据记录和停机/复话操作接口。其中日志功能在客户写入服务器时同步调用，不再添加于其中。

1. **测试与分析**
2. 测试方法：传统黑盒测试
3. 测试用例：
4. 功能测试

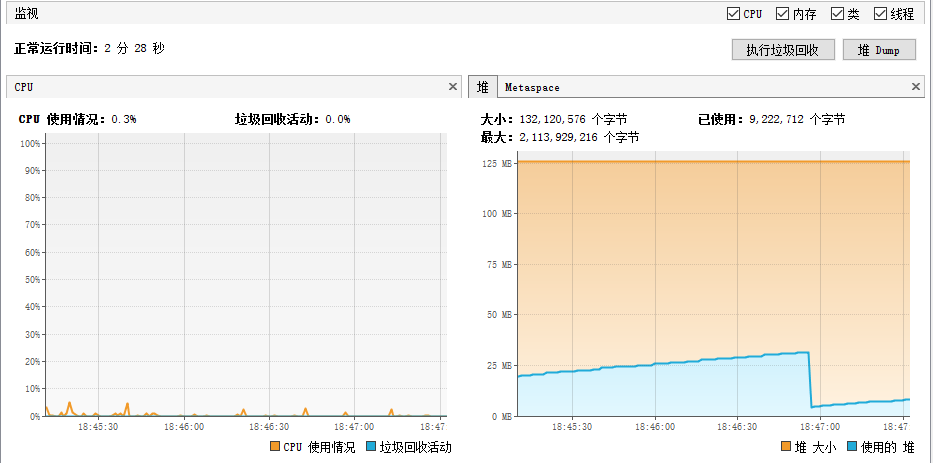
表1 关键字

|  |  |
| --- | --- |
| **关键字** | **关键字对应的动作** |
| readline | 读入用户号码和操作类型 |
| judgeAsk | 调用停机/复话操作接口方法并回复 |

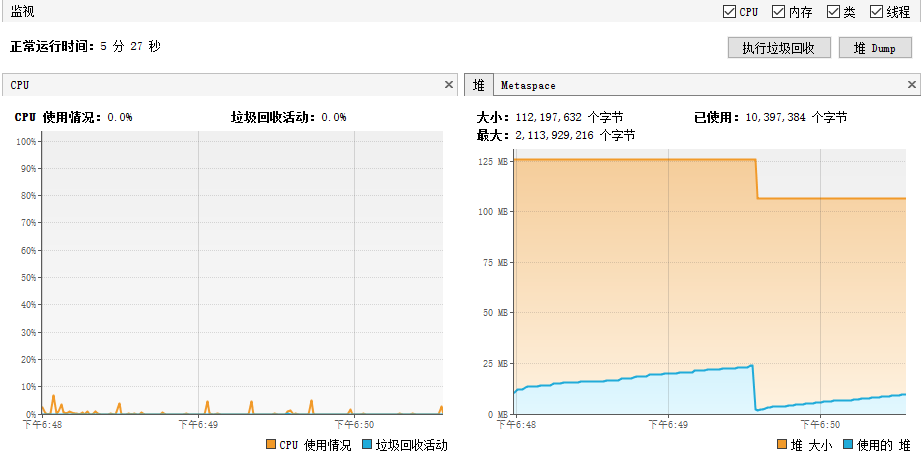
表2 关键字测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 关键字 | 参数 | 说明 |
| 1 | readline | 15295516852 1 | 输入号码和操作类型 |
| judgeAsk | 0（成功） | 检查是否为预期结果 |
| 2 | readline | 15295524852 1 15211484845 2 | 输入号码和操作类型 |
| judgeAsk | 0（成功） | 检查是否为预期结果 |
| 3 | readline | 15111 | 输入号码和操作类型 |
| judgeAsk | 请求错误，重新输入 | 检查是否为预期结果 |
| 4 | readline | exit | 输入退出请求 |
| judgeAsk | 无 | 检查是否断开连接 |

1. 性能测试

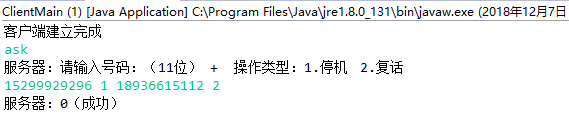


图七：服务器运行时性能检测

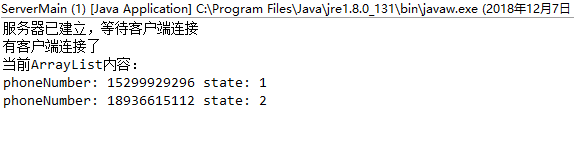


图八：客户端运行时性能检测

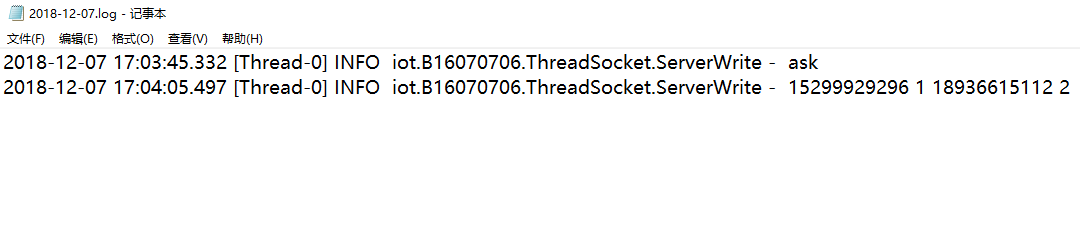
1. 测试结果



图九：客户端批量输入测试结果



图十：服务器相应测试结果



图十一：相应日志记录结果

在java visualVM检测中可以观察到线程在执行过程中没有产生死锁现象，经分析，服务器和客户端的线程不会同时申请资源，破坏了死锁产生的条件。

1. 时间复杂度和空间复杂度

服务器、客户端时间复杂度与用户输入相关，最大为O()，所以时间复杂度为O()。

服务器程序空间复杂度：O()，客户端程序空间复杂度：O(1)。

1. **总结**

在最开始成功完成了服务器和客户端之间的通信之后，我开始尝试添加新的功能，而在添加一点点功能之后，总是会报出Connection reset的错误信息，报错的位置还显示的是客户端的读线程。头一次面对这个报错时，把整个程序从最开始的创建到当时的进度都重演了一遍，意外得到解决，却想不通为什么。后来又碰上这个问题的时候，怎么肉眼看都看不出自己的问题到底在哪里，即使通过调试，找到了错误发生的位置，也不知道到底是哪里有问题，因为自己的经验非常浅。在看了Stack Overflow之中一些人的解答，我才反应过来这一报错的错误原因并不是服务器和客户端的通信问题，所以我决定将想添加的功能从socket通信中独立出来运行，最终编译排查完错误再放回socket通信服务器端，才得以解决。让人非常郁闷，因为Connection reset错误就像在玩诡术欺诈一样，将一些报错的信息全部笼统归结成socket通信出问题，让人难以找到自己真实的错误原因。

在添加记录日志的功能时，我慢慢了解到java的log4j、logback日志框架和Maven开源项目管理工具，仿佛开启了新的大门。java还有千千万万的包可以像logback一样被依赖，其成就者就是Maven，它提高了项目的构建效率，让我感觉eclipse这个集成开发环境终于更有亲和力了，不过当然，是前人栽树后人乘凉的感觉使我觉得它更有了亲和力。

在这一程序设计中我还有很多不足，比如数据应该用数据库的增删查改来处理；停机/复话操作也并没有完全通过接口方法实现；运行界面还停留在控制台而不是图形化界面，非常简陋；对应用场景可能发生的情况考虑得也并不全面等等。在移动终端课程的大作业中有用到一点数据库的东西，但是基本都是即拿即用，一知半解的，再想到先前一个报错浪费了大量的时间，心有余悸，能力有限，不敢多添功能了。

这一次程序设计让我对java编程和socket通信有了更深入的了解，回首过去学习程序设计总是眼高手低懒得实践，但在这两周的程序设计中我才发现，实践、实现功能、完成需求才是硬道理，看书上两行代码不如自己敲上一行，不自己敲上一行，还真以为自己能是什么牛人……

目前还有一件非常担心的事是在配置Maven环境后这个程序添加了log4j和logback的依赖，我不能保证将程序打包移植后它还可以正常运行。日志生成路径是绝对路径，我设定在E:/socketlog路径下。最终打包出来的jar包有客户端和服务器，还需解压获取，无法直接执行。

**参考文献**

[1]风泊月.Socket编程服务端和客户端互相通信,实现简单的聊天.[2018-12-7].

https://blog.csdn.net/hello\_word2/article/details/66975779

[2]mark4o.the answer.[2018-12-7].https://stackoverflow.com/questions/1434451

[3]yaoyy09.如何在eclipse中配置maven项目.[2018-12-7]. https://jingyan.

baidu.com/article/59703552cb9b988fc00740a4.html

[4]行走在云端的愚公.logback的使用和logback.xml详解.[2018-12-7].http://www.cnblogs.com/warking/p/5710303.html

|  |  |  |
| --- | --- | --- |
| **评分细则** | **评分项** | **成绩** |
| 学习态度：努力学习，遵守纪律，作风严谨务实，按期圆满完成规定的任务。（10分） |  |
| 设计质量：论证、分析、设计、计算、结构、建模、实验正确合理；验收：回答问题正确，实物演示顺利。（50分） |  |
| 撰写质量：结构严谨，文字通顺，用语符合技术规范，图表清楚，书写格式规范，符合规定字数要求。（40分） |  |
| 总评分 |  |
| **简短评语** | **教师签名：**  **年 月 日** | |
| **评分等级** |  | |
| **备注** | **评分等级有五种：优秀、良好、中等、及格、不及格** | |