Java程序设计实验报告

计算机04 邓硕灵 10055088

实验要求

1. 用Java实现GUI计算器。
2. 计算器实现中利用socket与服务器进行交互，将每次计算的结果传入服务器，同时能从服务器上获取以前的计算。

实验内容

项目概述

本项目分为两个子项目。其一为SimpleCaculator项目，该项目是整个GUI计算器的实现；第二个子项目是CaculatorServer项目，该项目实现了GUI计算器的服务器，通过网络与SimpleCaculator进行交互。

1. **SimpleCaculator:**

   该子项目是GUI计算器的实现，由以下三个类组成程序整体结构：

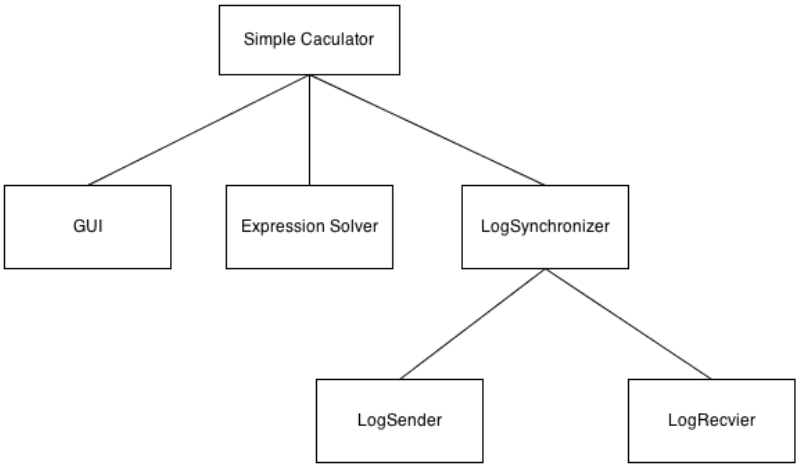
   1. 在GUI上，采用了Java AWT实现了Layout的绘制，Component的摆放，及其用Inerclass完成了Listenner的设置。
   2. 在Logic上，采用了Binary Tree数据结构解析整个算术表达式，最后用Tree对表达式进行求值。
   3. 在Network 上，采用了多线程模型的方式在后台接受服务器的日志数据，同时发送计算表达式。

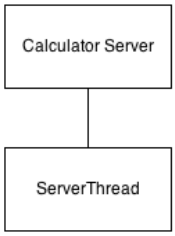
2. **CaculatorServer:**

   该子项目是GUI计算器的服务器的实现，主要由以下的类组成：

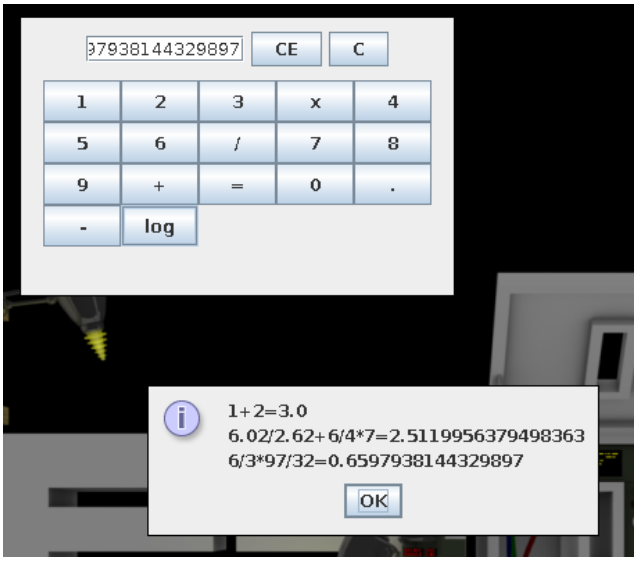
   1. 简单的Echo服务器，多线程模型的方式实现了对客户访问的并发请求的响应。

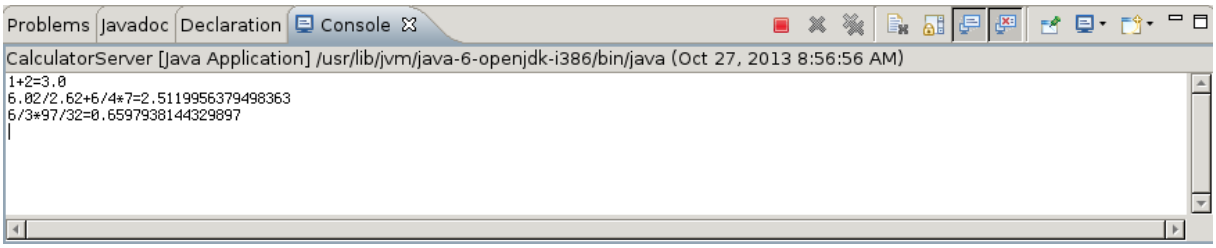
项目架构


Simple Caculator
GUI
Expression Solver
LogSynchronizer
LogSender
LogRecvier


Calculator Server
ServerThread

实验截图展示


97938144329897
CE
C
1
2
3
x
4
5
6
/
7
8
9
+
=
0
.
-
log
1+2=3.0
6.02/2.62+6/4*7=2.5119956379498363
6/3*97/32=0.6597938144329897
OK



实验难点总结

- 由于对Java IO 的理解不到位，以至于在写Socket通信模块中出现了没有调用flush函数，导致数据一直在客户端对缓冲区没有发出。如果此时用户调用从服务器上获取信息的函数，就会直接导致程序被socket的read方法一直阻塞，造成问题。如下代码所示，必须调用flush函数，才能实现真正地发送，否则数据一直在缓冲区内无法送出。

```
public void sendLog(String exp, String res) {
    try {
        String equation = exp + "=" + res;
        PrintWriter sockWriter = new PrintWriter(sock.getOutputStream());
        sockWriter.println(equation);
        sockWriter.flush();
    } catch (Exception e) {
    e.printStackTrace();
    }
}
```

- 在编写客户端的RecvLog模块的时候，必须调用read方法，但是read方法只有在服务器端写入了EOF之后，才会返回NULL值，直接导致无法用单线程程序去运行，否则一调用read方法，只要服务器端不写入EOF，就会一直阻塞在read方法上面。因而，采用了多线程，让另外一个线程去读socket, 把服务器上所有的数据写入String中，主线程每次仅需读该String即可。如下代码所示，新增加一个线程对Sock进行异步地读写，防止因为服务器端没有写入EOF而导致阻塞。

```
public class LogRecvThread extends Thread {
private String expLog = "";
private BufferedReader sockReader;

    public LogRecvThread(InputStream in) {
        sockReader = new BufferedReader(new InputStreamReader(in));
    }

    public void run() {
        String s;
        try {
                while ((s = sockReader.readLine()) != null) {
                expLog += s + "\n";
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
}

    public String getExpLog() {
        return expLog;
    }
}
```

项目源代码地址

- Java_Exp_Calculator_Client
- Java_Exp_Calculator_Server