上机实践 6 功能扩展实现文档

一、 构件选择：

此次上机实践我们选择的还是使用构件进行功能实现，最终选择使用的构件是我们组自己编写的 PerformanceManager 构件：

https://github.com/Wheellllll/PerformanceManager

二、 构件选择理由

在完成此次上机实践时，最开始我们想的是继续使用第五组的 PM 模块进行实现，他们的 PM 有定时追加写入文件和写入不同文件的两种功能可以选择，而且可以通过直接设置写入文件的格式为.zip 进行压缩。

但是在具体使用时发现因为没有实时写入的功能，所以在保存消息这一功能点上实现起来比较别扭，只能进行较短时间间隔的写入。此外，其压缩也只是将所有的内容写到同一文件里，无法将多个文件进行归档。因此，最后改而使用我们组自己的 PM 模块。

另外，在最开始使用第五组的 PM 构件时，发现了其存在 bug：存在多个同类 Reporter 时文件内容会相互写入，对此我们也向第五组反映了该 bug，他们也及时地进行了修复和更新。

三、 实现细节

通过 IntervalLogger 记录 PM 指标统计结果，通过 RealtimeLogger 保存消息，通过 ArchiveManager 实现每天结束后对当天生成的文件归档压缩。

服务端初始化 PM 代码：

```
protected void initPerformance() {
    //初始化intervalLogger
    intervalLogger.setLogDir("./serverlog");
    intervalLogger.setLogPrefix("server");
    intervalLogger.setLogSuffix("log");
    intervalLogger.setDateFormat("yyyy-MM-dd HH_mm");
    intervalLogger.setInitialDelay(1);
    intervalLogger.setInterval(1, TimeUnit.MINUTES);

    intervalLogger.addIndex("Valid Login Number");
    intervalLogger.addIndex("Invalid Login Number");
    intervalLogger.addIndex("Receive Message Number");
    intervalLogger.addIndex("Ignore Message Number");
    intervalLogger.addIndex("Forward Message Number");
    intervalLogger.setFormatPattern(
            "Valid Login Number : ${Valid Login Number}\n" +
            "Invalid Login Number : ${Invalid Login Number}\n" +
            "Receive Message Number : ${Receive Message Number}\n" +
            "Ignore Message Number : ${Ignore Message Number}\n" +
            "Forward Message Number : ${Forward Message Number}\n\n");
    intervalLogger.start();

    //初始化realtimeLogger
    realtimeLogger.setLogDir("./serverlog");
    realtimeLogger.setLogPrefix("server");
    realtimeLogger.setLogSuffix("mlog");
    realtimeLogger.setFormatPattern(
            "Username : ${username}\n" +
            "Time : ${time}\n" +
            "Message : ${message}\n\n");

    //初始化archiveManager
    archiveManager.setArchiveDir("./serverarchive");
    archiveManager.setDatePattern("yyyy-MM-dd");
    archiveManager.addLogger(intervalLogger);
    archiveManager.addLogger(realtimeLogger);
    archiveManager.setInitialDelay(1);
    archiveManager.setInterval(1, TimeUnit.DAYS);
    archiveManager.start();
}
```

客户端初始化 PM 代码与服务端类似，只是 IntervalLogger 所记录的指标不同以及生成文件的路径不同：

```
protected void initPerformance() {
    //初始化intervalLogger
    intervalLogger.setLogDir("./clientlog");
    intervalLogger.setLogPrefix("client");
    intervalLogger.setLogSuffix("log");
    intervalLogger.setDateFormat("yyyy-MM-dd HH_mm");
    intervalLogger.setInitialDelay(1);
    intervalLogger.setInterval(1, TimeUnit.MINUTES);

    intervalLogger.addIndex("Login successfully number");
    intervalLogger.addIndex("Login failed number");
    intervalLogger.addIndex("Send message number");
    intervalLogger.addIndex("Receive message number");
    intervalLogger.setFormatPattern(
            "Login successfully number : ${Login successfully number}\n" +
                    "Login failed number : ${Login failed number}\n" +
                    "Send message number : ${Send message number}\n" +
                    "Receive message number : ${Receive message number}\n\n");
    intervalLogger.start();

    //初始化realtimeLogger
    realtimeLogger.setLogDir("./clientlog");
    realtimeLogger.setLogPrefix("client");
    realtimeLogger.setLogSuffix("mlog");
    realtimeLogger.setFormatPattern(
            "Username : ${username}\n" +
                    "Time : ${time}\n" +
                    "Message : ${message}\n\n");

    //初始化archiveManager
    archiveManager.setArchiveDir("./clientarchive");
    archiveManager.setDatePattern("yyyy-MM-dd");
    archiveManager.addLogger(intervalLogger);
    archiveManager.addLogger(realtimeLogger);
    archiveManager.setInitialDelay(1);
    archiveManager.setInterval(1, TimeUnit.DAYS);
    archiveManager.start();
}
```