

**2018年春季学期  
计算机学院大二软件构造课程**

**Lab 6实验报告**

|  |  |
| --- | --- |
| 姓名 | 冯运 |
| 学号 | 1160300524 |
| 班号 | 1603005 |
| 电子邮件 | [18846188605@163.com](mailto:18846188605@163.com) |
| 手机号码 | 18846188605 |

**目录**

[1 实验目标概述 1](#_Toc514162176)

[2 实验环境配置 1](#_Toc514162177)

[3 实验过程 1](#_Toc514162178)

[3.1 ADT设计方案 1](#_Toc514162179)

[3.2 Monkey线程的run()的执行流程图 1](#_Toc514162180)

[3.3 至少两种“梯子选择”策略的设计与实现方案 2](#_Toc514162181)

[3.3.1 策略1 2](#_Toc514162182)

[3.3.2 策略2 2](#_Toc514162183)

[3.3.3 策略3（可选） 2](#_Toc514162184)

[3.4 “猴子生成器”MonkeyGenerator 2](#_Toc514162185)

[3.5 如何确保threadsafe？ 2](#_Toc514162186)

[3.6 系统吞吐率和公平性的度量方案 2](#_Toc514162187)

[3.7 输出方案设计 2](#_Toc514162188)

[3.8 猴子过河模拟器v1 2](#_Toc514162189)

[3.8.1 参数如何初始化 2](#_Toc514162190)

[3.8.2 使用Strategy模式为每只猴子随机选择决策策略 2](#_Toc514162191)

[3.9 猴子过河模拟器v2 2](#_Toc514162192)

[3.9.1 对比分析：固定其他参数，选择不同的决策策略 3](#_Toc514162193)

[3.9.2 对比分析：变化某个参数，固定其他参数 3](#_Toc514162194)

[3.9.3 分析：吞吐率是否与各参数/决策策略有相关性？ 3](#_Toc514162195)

[3.9.4 压力测试结果与分析 3](#_Toc514162196)

[4 实验进度记录 3](#_Toc514162197)

[5 实验过程中遇到的困难与解决途径 3](#_Toc514162198)

[6 实验过程中收获的经验、教训、感想 3](#_Toc514162199)

# 实验目标概述

本次实验训练学生的并行编程的基本能力，特别是Java多线程编程的能力。根据一个具体需求，开发两个版本的模拟器，仔细选择保证线程安全（threadsafe）的构造策略并在代码中加以实现，通过实际数据模拟，测试程序是否是线程安全的。另外，训练学生如何在threadsafe和运行性能之间寻求较优的折中，为此计算吞吐率等性能指标，并做仿真实验。1.Java多线程编程面向线程安全的ADT设计策略选择、文档化2.模拟仿真实验与对比分析基本的GUI编程

# 实验环境配置

实验需要配置elclipse,但是我之前已经习惯于使用jetbrain系列的开发ide,所以我这次使用了IDEA，同时下载安装了jdk，版本号是jdk1.8.0\_101，将jdk的安装路径加入了系统的环境变量。在IDEA里面下载安装了junit插件,配置好了java的测试环境。

Lab6的仓库url:

https://github.com/ComputerScienceHIT/Lab6-1160300524.git

# 实验过程

请仔细对照实验手册，针对三个问题中的每一项任务，在下面各节中记录你的实验过程、阐述你的设计思路和问题求解思路，可辅之以示意图或关键源代码加以说明（但千万不要把你的源代码全部粘贴过来！）。

## ADT设计方案

设计了哪些ADT、各自的作用、属性、方法；

ADT有三个: Ladder，Monkey,Rung

Ladder类：

这个类是对一个梯子的抽象，用于表示一个梯子，这个梯子内有rung对象，即梯子由台阶组成，封装了外部对梯子台阶的操作。

ID->梯子的唯一标识

size-> 当前梯子上爬着的猴子数量（便于在选择过河策略中的实现） direction->当前梯子上所有猴子的过河方向（null->梯子上没有猴子 "R->L"所有猴子的方向都是从右到左 "L->R"所有的猴子方向都是从左到右）

rungs-> 一个列表，存储的台阶对象

rungs的数量始终等于传入的rungNumber数量 direction的值只能是null,"L->R" ,"R->L"三个值 size>=0 梯子上所有猴子的方向必须是一致的

方法：

getID

private int getID()

返回:

返回梯子的id值

getSize

private int getSize()

返回:

当前梯子上的猴子的数量

getDirection

public String getDirection()

返回:

当前梯子上猴子的行进方向

removeMonkey

public boolean removeMonkey(int index)

删除指定踏板上的某个猴子

参数:

index - 踏板的下标

返回:

boolean值，表示是否删除成功

addMonkey

public boolean addMonkey(int index, Monkey monkey)

在梯子的指定台阶上放置猴子 需要设置猴子的台阶必须原来是空的，否则无法添加成功，会返回false

参数:

index - 指定梯子上的台阶位置

monkey - 需要设置的猴子

返回:

是否设置成功的boolean值

给出每个ADT的specification；

Rung类：

作用:这个类是对一个台阶的抽象，内部存储了当前阶梯上的猴子，如果是空，就表示这个台阶是空的。

monkey->存储当前台阶上的猴子对象

getMonkey

Monkey getMonkey()

得到当前台阶上的猴子对象

返回:

返回当前台阶上的猴子对象

setMonkey

boolean setMonkey(Monkey newMonkey)

给当前阶梯设置一个新的猴子对象 添加成功的条件： 规定如果阶梯是空的才可以放新的猴子 或者放进的猴子对象为空（将台阶置空） 其他情况都会添加失败 这个方法是线程安全的方法

参数:

newMonkey - 添加进来的新的猴子对象

返回:一个布尔值

表示这个台阶的猴子是否添加成功

Monkey对象

ID->猴子的唯一标识 direction->true="L->R",false="R->L" speed -> 猴子在梯子上的爬行速度

speed>=1，且为整数

方法：

getID

public int getID()

返回:

猴子的ID

getDirection

public java.lang.String getDirection()

返回:

猴子的方向 ”L->R" 表示从左到右 "R->L"表示从右到左

getSpeed

public int getSpeed()

返回:

猴子的移动速度

getBornTime

public long getBornTime()

得到这只猴子的出生时间

返回:

猴子的出生时间，单位 ms

getArrivedTime

public long getArrivedTime()

得到这只猴子到达河对岸的时间

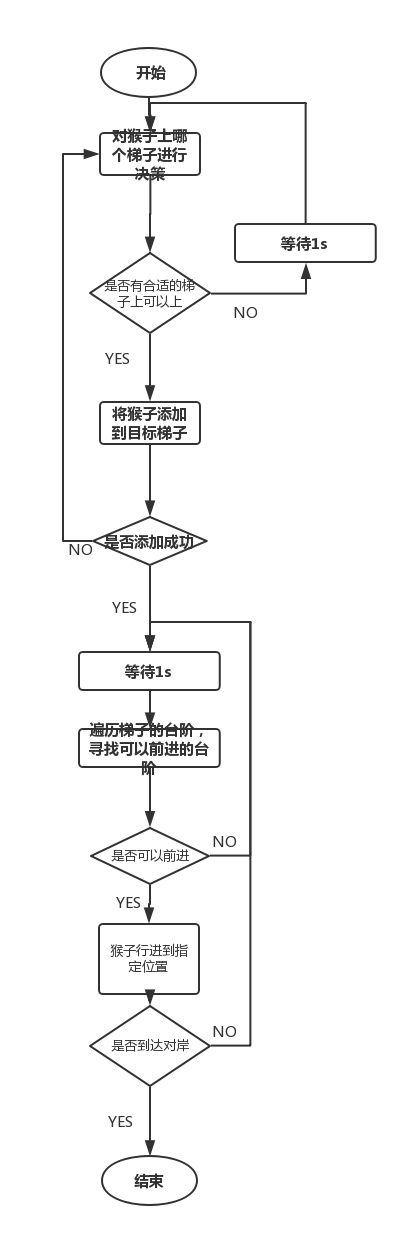
返回:

猴子到达对岸的时间，单位 ms

（可选）以类图形式给出多个类之间的关系。

## Monkey线程的run()的执行流程图

这里无需考虑具体采用的梯子选择策略。



## 至少两种“梯子选择”策略的设计与实现方案

### 策略1

优先选择没有猴子的梯子，若所有梯子上都有猴子，则优先

选择没有与我对向而行的猴子的梯子；若满足该条件的梯子有很多，

则随机选择

### 策略2

优先选择整体推进速度最快的梯子（

没有与我对向而行的猴

子、其上的猴子数量最少、梯子离我最近的猴子的真实行进速度最

快）；

### 策略3（可选）

优先选择没有猴子的梯子，若所有梯子上都有猴子，则在岸

边等待，直到某个梯子空闲出来

## “猴子生成器”MonkeyGenerator

我使用了java.util中的Timer类，用于计时，设置的间隔时间为1000ms

每次创建k个新的monkey对象，直到已经创建好的猴子数量达到了N个，结束计数器的运行。每次创建好一个新的猴子对象，就立即启动猴子的过河线程，让猴子开始过河，直到猴子过河后，过河线程结束。

## 如何确保threadsafe？

在猴子过河的多线程任务中，公共的数据对象有Ladder,Rung这两个，所以对线程安全的讨论就针对Ladder类和Rung类，因为Ladder类中有大量对Rung类的应用和操作，所以Ladder类的线程安全很大程度上依靠Rung类的线程安全操作保障

对Rung类：

Monkey成员变量的访问属性是private，只能通过方法来对其进行修改，避免了外部的直接修改。

getMonkey()方法自身就是原子操作，不会出现线程危险

setMonkey()方法加了锁，即在方法前面加了synchronized修饰词，它只能被一个线程同时调用，是线程安全的

对于Ladder类：

ID是private和不可变的，是线程安全的 getSize(),getID(),getDirection()方法内部都是原子操作，不会有线程安全问题 getMonkeys()对访问公共数据的部分，加了线程锁，即对整个梯子对象加了synchronized修饰词，保证了线程安全 setMonkey()方法在修改线程公共数据rung时，使用了线程安全的方法get()和setMonkey()，同时对size和direction的修改不会因为多线程而改变结果 removeMonkey()方法 使用了线程安全的方法get()和getMonkey()，同时对size和direction的修改加了线程锁 ，所以这个类是线程安全类

由于这两个类都是线程安全的，而在猴子过河线程中要想对公共数据梯子进行操作，都必须经过这两个类，得以保证线程安全。

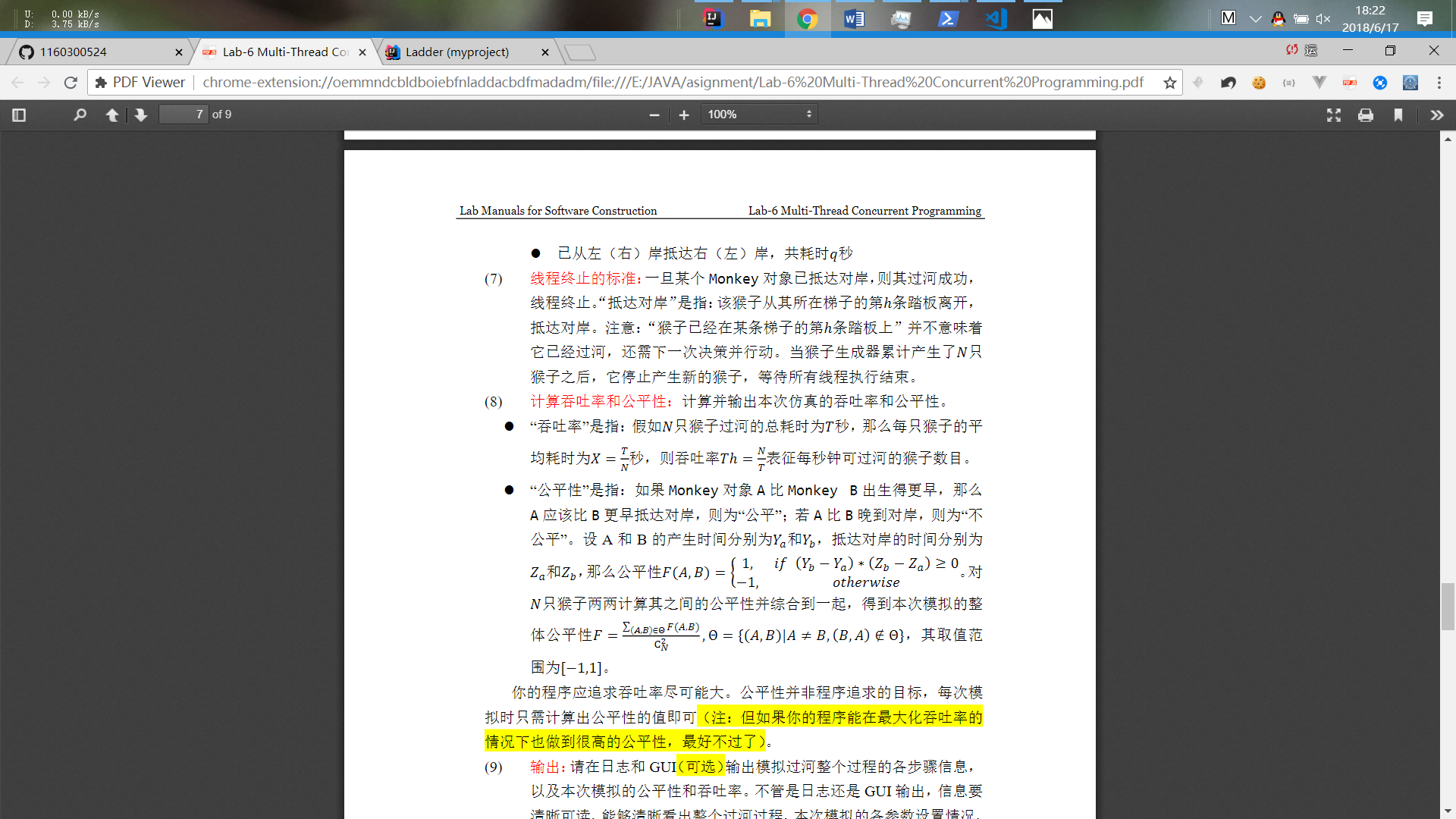
## 系统吞吐率和公平性的度量方案

吞吐率N/T表示每秒钟可过河的猴子数目，实现方案是：

在第一个猴子产生时，记录一个开始时间，在创建猴子对象时，都将创建进程的Tread对象存进一个列表中，在主程序的结尾，遍历整个列表，调用Tread.join()方法，保证所有的线程都结束后，记录一个猴子过河结束的时间，两个时间相减就是猴子过河需要的总时间T,然后用N除以T,就可以得到吞吐率了。

公平性的计算：

每个猴子对象内部都有一个产生时间字段bornTime和过河结束时间的字段arrivedTime，在所有猴子都过河结束后，用一个二重循环，分别计算任意两只猴子的 (bornTime2-bornTime1)^(arrivedTime2-arrivedTime1),^表示按位异或，如果结果大于等于0，则公平性累加变量fairness 加1，如果小于0，则-1。

在结算结束后，计算 fairness/(N\*(N-1)/2) ，其中N\*(N-1)/2是 的计算表达式。

这样就得到了公平性的计算结果。

## 输出方案设计

日志

日志的部分我依然采用了实验4和实验5的日志系统部分，实现的原理和之前的实验一致。

我在每个猴子产生的时候输出一次猴子产生的编号和产生的时间。

在登上梯子的时候，输出一次猴子的信息，登上的梯子的编号，以及时间

在每次猴子移动位置时，都日志输出一次猴子的的编号，梯子的编号，以及移动到哪个阶梯。

在猴子到达对岸时，就输出一次猴子的编号，以及从 哪个梯子上下来的信息。

GUI

可视化（可选）

## 猴子过河模拟器v1

### 参数如何初始化

通过从一个配置文件中读入参数

然后使用正则表达式去匹配相应的格式，得到参数值

格式示例如下：

n = 4

h = 10

t = 1

N = 30

k = 2

MV = 5

### 使用Strategy模式为每只猴子随机选择决策策略

首先我定义了一个接口:ladderChoice，在这个接口里，有一个函数getLadder(),传入做决策的猴子和所有的梯子，提供做决策时使用。

然后在实现每一种策略时，都需要实例化这个接口，即根据提供的梯子对象，帮助传入的猴子做出决策，确定上哪一个梯子.这样在猴子线程中可以很轻松的就替换决策的策略，不需要繁杂的改动过程。

## 猴子过河模拟器v2

在不同参数设置和不同“梯子选择”模式下的“吞吐率”和“公平性”实验结果及其对比分析。

### 对比分析：固定其他参数，选择不同的决策策略

固定参数如下:

n = 4

h = 10

t = 1

N = 30

k = 2

MV = 5

策略1:吞吐率 1.1534025374855823 公平性: 0.7471264367816092

策略2:吞吐率 1.199616122840691 公平性 : 0.7103448275862069

策略3: 吞吐率 0.6974797730865805 公平性: 0.38850574712643676

由以上数据可以看出策略1和策略2的吞吐率和公平性方面差距不大,十分接近,策略3 和其他的策略有不小的性能差距.

### 对比分析：变化某个参数，固定其他参数

N1 = 4 n2=30

h = 10

t = 1

N = 30

k = 2

MV = 5

策略1:

吞吐率 1.1534025374855823 | 1.249843769528809

公平性: 0.7471264367816092 | 0.7655172413793103

策略2:

吞吐率 1.199616122840691 | 1.2497396375755052

公平性 : 0.7103448275862069 | 0.8298850574712644

策略3:

吞吐率 0.6974797730865805 | 1.3041210224308815

公平性: 0.38850574712643676 | 0.6551724137931034

n = 4

h1= 10 h2 = 30

t = 1

N = 30

k = 2

MV = 5

策略1:

吞吐率 1.1534025374855823 | 0.6247787242018451

公平性: 0.7471264367816092 | 0.6735632183908046

策略2:

吞吐率 1.199616122840691 | 0.5880162292479273

公平性 : 0.7103448275862069 | 0.5448275862068965

策略3:

吞吐率 0.6974797730865805 | 0.2726033621081327

公平性: 0.38850574712643676 | 0.15862068965517243

n = 4

h = 10

t1 = 1 t2 = 4

N = 30

k = 2

MV = 5

策略1:

吞吐率 1.1534025374855823 | 0.4838085408334408

公平性: 0.7471264367816092 | 0.9218390804597701

策略2:

吞吐率 1.199616122840691 | 0.4999750012499375

公平性 : 0.7103448275862069 | 0.9586206896551724

策略3:

吞吐率 0.6974797730865805 | 0.4545179080055754

公平性: 0.38850574712643676 | 0.9264367816091954

n = 4

h = 10

t = 1

N1 = 30 N2 = 100

k = 2

MV = 5

策略1:

吞吐率 1.1534025374855823 | 1.562255897516013

公平性: 0.7471264367816092 | 0.8614141414141414

策略2:

吞吐率 1.199616122840691 | 1.2655824843384167

公平性 : 0.7103448275862069 | 0.6662626262626262

策略3:

吞吐率 0.6974797730865805 | 0.6754383594953125

公平性: 0.38850574712643676 | 0.3325252525252525

n = 4

h = 10

t = 1

N = 30

K1 = 2 k2 = 10

MV = 5

策略1:

吞吐率 1.1534025374855823 | 1.4992503748125936

公平性: 0.7471264367816092 | 0.20919540229885059

策略2:

吞吐率 1.199616122840691 | 1.4988758431176616

公平性 : 0.7103448275862069 | 0.24597701149425288

策略3:

吞吐率 0.6974797730865805 | 0.6814929238318075

公平性: 0.38850574712643676 | 0.2689655172413793

n = 4

h = 10

t = 1

N = 30

k = 2

MV1= 5 MV2 = 1

策略1:

吞吐率 1.1534025374855823 | 1.199568155464033

公平性: 0.7471264367816092 | 0.9080459770114943

策略2:

吞吐率 1.199616122840691 | 1.1534468837710024

公平性 : 0.7103448275862069 | 0.8850574712643678

策略3:

吞吐率 0.6974797730865805 | 0.3527876100991333

公平性: 0.38850574712643676 | -0.016091954022988506

### 分析：吞吐率是否与各参数/决策策略有相关性？

吞吐率与各参数的设定有很大的关系和决策策略都有关系

从实验结果来看,策略1和策略2在吞吐率和公平性方面有较好的变化

在相同策略下,梯子的个数增多,梯子的长度减小,产生猴子间隔的时间缩小,每秒产生的猴子数量增多,以及提高速度最大值的上限,都可以较明显的提高猴子的吞吐率.

但是增大减小猴子的个数对于吞吐率没有影响,是一个无关变量

### 压力测试结果与分析

这次压力 测试我采用了以下参数

n = 1000

h = 10

t = 1

N = 2000

k = 1000

MV = 5

实验的结果是

吞吐率: 126.44622874122778

公平性: 0.21162981490745372

这次压力测试的吞吐率较高,是因为我提高了梯子的个数,增大了每秒产生猴子的数量,根据上一节中实验得出的结论,这次压力测试得到的实验结果符合上面得出的结论,说明这次压力测试的结果符合预期,与实验结论相符.

# 实验进度记录

请尽可能详细的记录你的进度情况。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 6.10 | 全天 | 构建ADT | 完成 |
| 6.13 | 全天 | 实现多线程的猴子过河模拟器 | 完成 |
| 6.17 | 全天 | 完成实验报告 | 完成 |

# 实验过程中遇到的困难与解决途径

在实验中遇到了死锁的情况(程序不停止,却没有猴子过河)

原因是两只方向相反的猴子跳上了同一个梯子,在行进过程中互相冲突,由此产生了死锁.

解决办法:认真分析猴子上梯子的决策过程,同时又重新温习了多线程对公共数据结构操作的同步方法,对向梯子的指定台阶放置猴子的方法加了线程锁,解决了死锁问题.

# 实验过程中收获的经验、教训、感想

本节除了总结你在实验过程中收获的经验和教训，也可就以下方面谈谈你的感受（非必须）：

1. 多线程程序比单线程程序复杂在哪里？你是否能体验到多线程程序在性能方面的改善？
2. 你采用了什么设计决策来保证threadsafe？如何做到在threadsafe和性能之间很好的折中？
3. 你在完成本实验过程中是否遇到过线程不安全的情况？你是如何改进的？
4. 关于本实验的工作量、难度、deadline。
5. 到此为止你对《软件构造》课程的意见和建议。

通过这次实验,我第一次熟悉了多线程编程的方法,原理以及如果保证线程安全的基本知识,同时体验到了多线程编程的优势和长处, 不过也感受到了线程安全的重要性,以至于程序在调试过程中出现了太多的奇异现象,很难分析结果,以后还需多练习多线程编程,对项目中性能的优化有非常大的帮助.