

**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 | 穆添愉 |
| 学号 | 1160301008 |
| 班号 | 1603010 |
| 电子邮件 | [1417553133@qq.com](mailto:1417553133@qq.com) |
| 手机号码 | 15636094072 |

**目录**

[1 实验目标概述 1](#_Toc63402142)

[2 实验环境配置 1](#_Toc211132483)

[3 实验过程 1](#_Toc1445967408)

[3.1 ADT设计方案 1](#_Toc866364897)

[3.2 Monkey线程的run()的执行流程图 8](#_Toc2113051707)

[3.3 至少两种“梯子选择”策略的设计与实现方案 8](#_Toc1301090173)

[3.3.1 策略1 8](#_Toc947392901)

[3.3.2 策略2 9](#_Toc1797361649)

[3.3.3 策略3（可选） 10](#_Toc1645988492)

[3.4 “猴子生成器”MonkeyGenerator 10](#_Toc224209154)

[3.5 如何确保threadsafe？ 11](#_Toc2020186856)

[3.6 系统吞吐率和公平性的度量方案 12](#_Toc174688195)

[3.7 输出方案设计 13](#_Toc248595915)

[3.8 猴子过河模拟器v1 14](#_Toc822357977)

[3.8.1 参数如何初始化 14](#_Toc732019782)

[3.8.2 使用Strategy模式为每只猴子随机选择决策策略 14](#_Toc986367178)

[3.9 猴子过河模拟器v2 15](#_Toc1886254050)

[3.9.1 对比分析：固定其他参数，选择不同的决策策略 16](#_Toc220674150)

[3.9.2 对比分析：变化某个参数，固定其他参数 16](#_Toc1927683196)

[3.9.3 分析：吞吐率是否与各参数/决策策略有相关性？ 17](#_Toc1328265345)

[3.9.4 压力测试结果与分析 17](#_Toc1873469574)

[4 实验进度记录 19](#_Toc1275801904)

[5 实验过程中遇到的困难与解决途径 19](#_Toc987115925)

[6 实验过程中收获的经验、教训、感想 19](#_Toc996336371)

# 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

(2)面向线程安全的 ADT 设计策略选择、文档化

(3)模拟仿真实验与对比分析

(4)基本的 GUI 编程

# 实验环境配置

仓库链接如下:

https://github.com/ComputerScienceHIT/Lab6-1160301008

# 实验过程

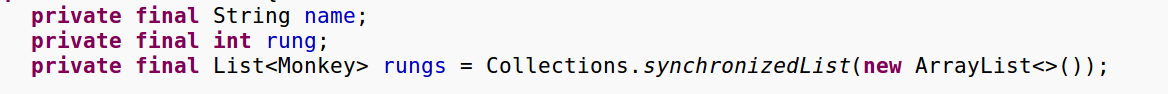
## ADT设计方案

ADT设计如下:

1. Ladder类

作用: 表示一个梯子

属性:



方法:

**public** Ladder(String name, **int** rung);

**public** **int** getLength();

**public** String getName();

**public** List<Monkey> getRungs();

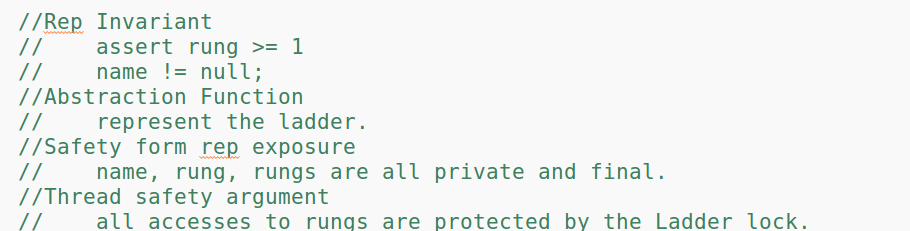
**public** **synchronized** **boolean** empty();

**public** **synchronized** **boolean** setMonkey(**int** location, Monkey m);

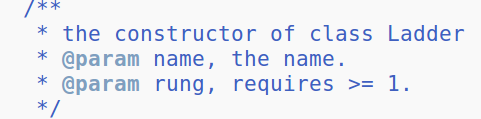
**public** **synchronized** **int** getTotalNumber();

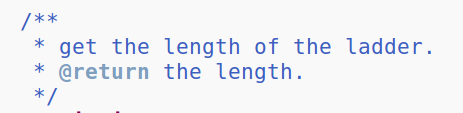
specification:

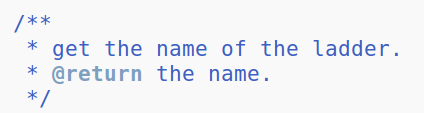
首先有如下的AF, RI等

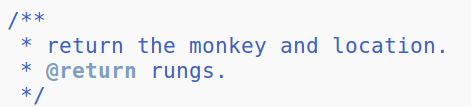


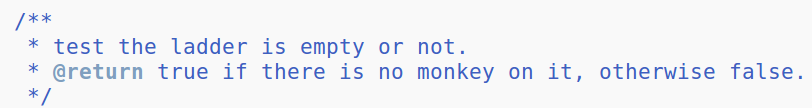
各个方法的spec如下:

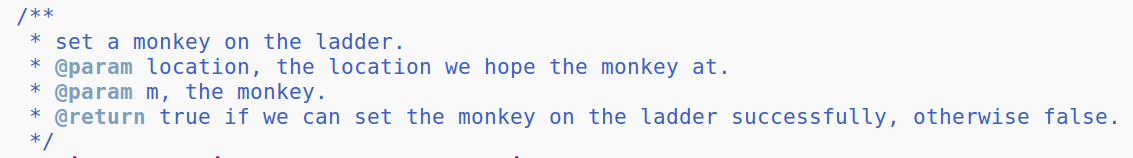


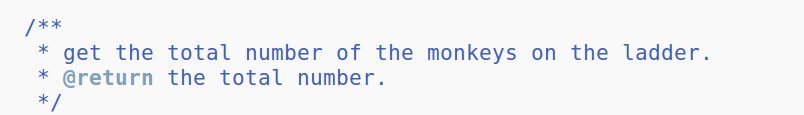








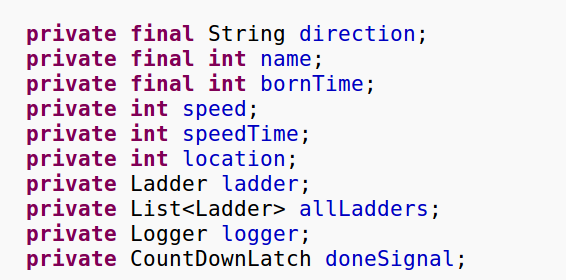




1. Monkey类

作用: 表示一只猴子

属性:



方法:

**public** Monkey(String direction, **int** name, **int** bornTime, **int** speed, List<Ladder> allLadders);

**public** **int** getName();

**public** **int** getSpeed();

**public** String getDirection();

**public** **int** getBornTime();

**public** **int** getTime();

**public** **int** getTotalTime();

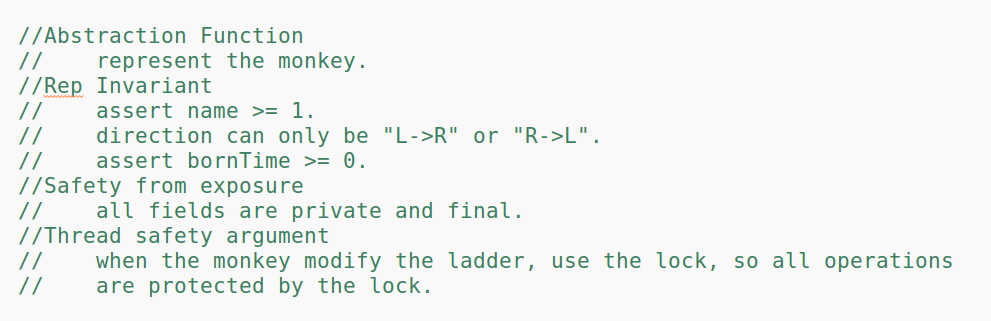
**public** **void** setLog(Logger logger);

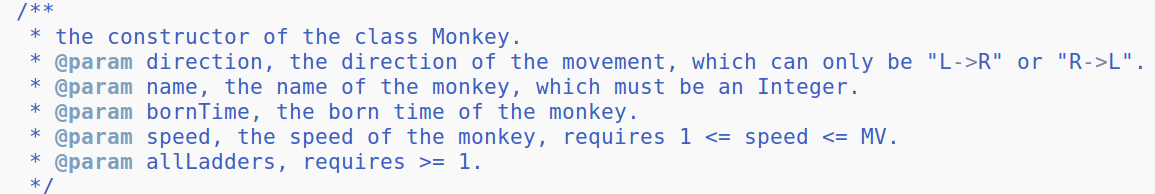
**public** **void** setCountDownLaunch(CountDownLatch doneSignal);

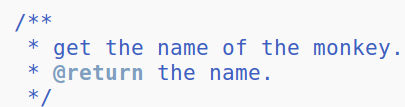
**public** **void** run();

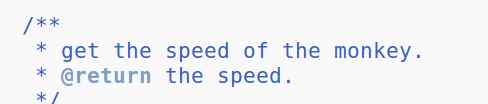
spec:

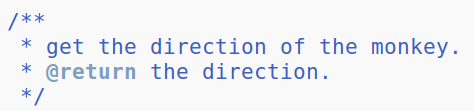
首先有如下的AF RI

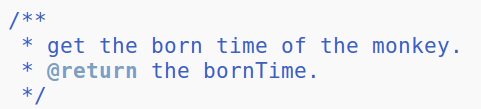


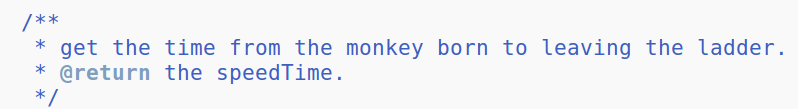


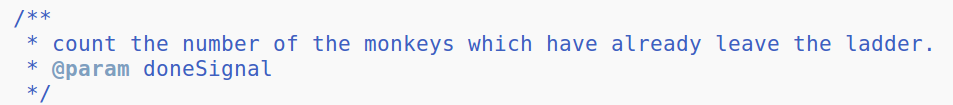








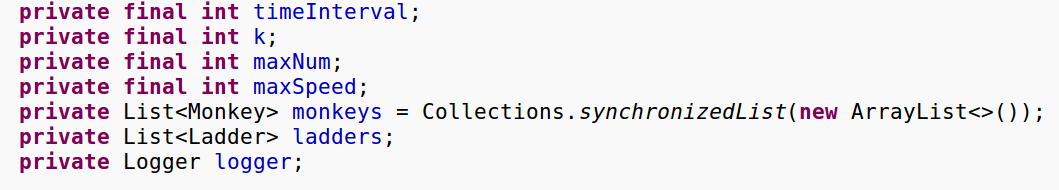




1. MonkeyGenerator 类

作用: 生成猴子

属性:



方法:

**public** MonkeyGenerator(**int** timeInterval, **int** k, **int** maxNum, **int** maxSpeed);

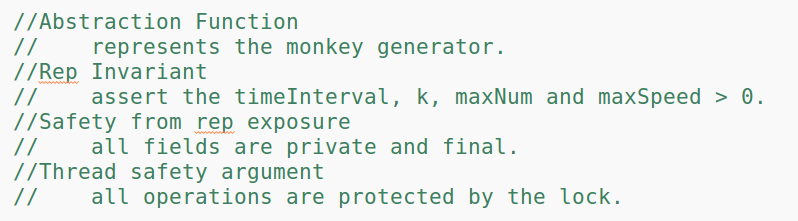
**public** **void** setLadders(List<Ladder> ladders);

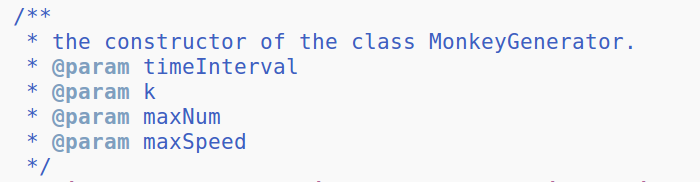
**public** **void** setLog(Logger logger);

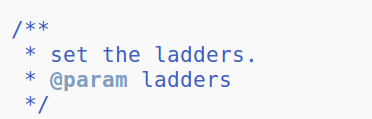
**public** **void** run();

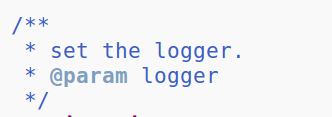
spec:

首先有如下的AF RI

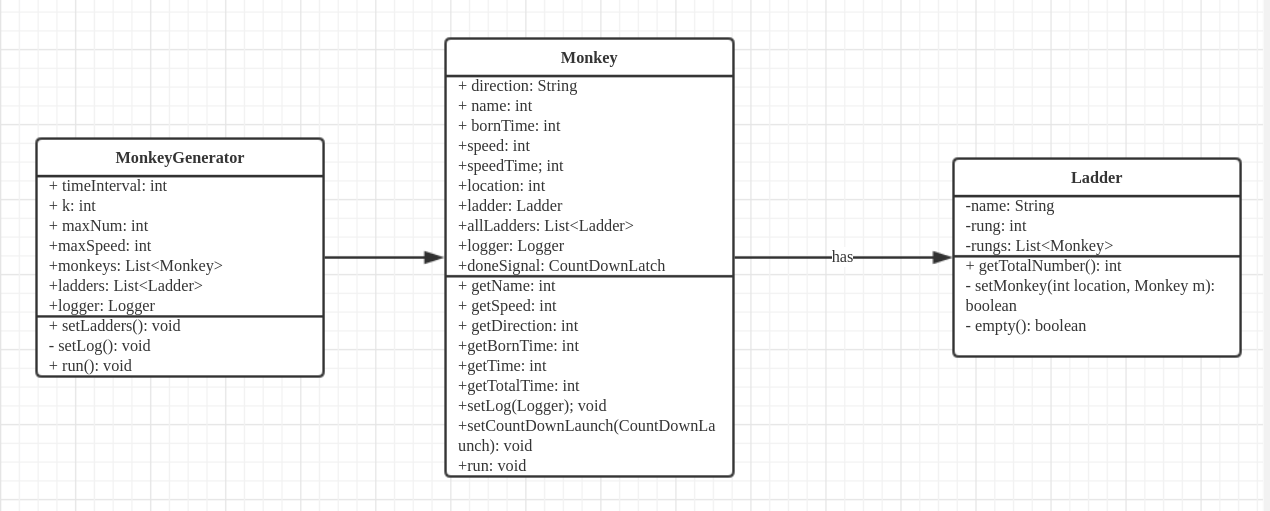


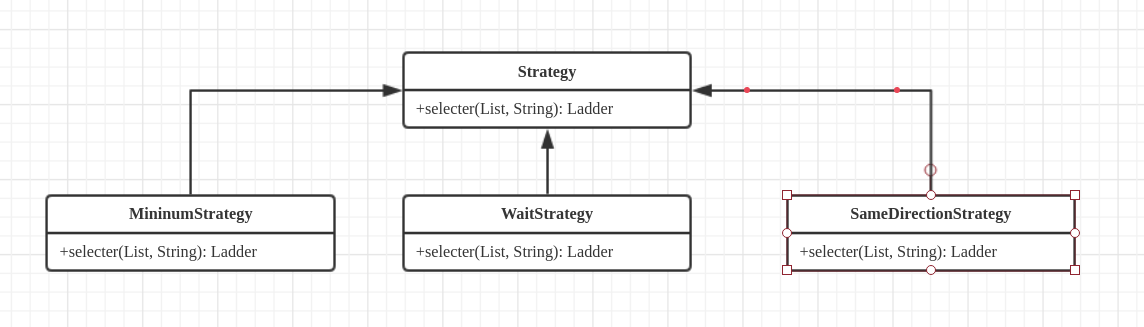




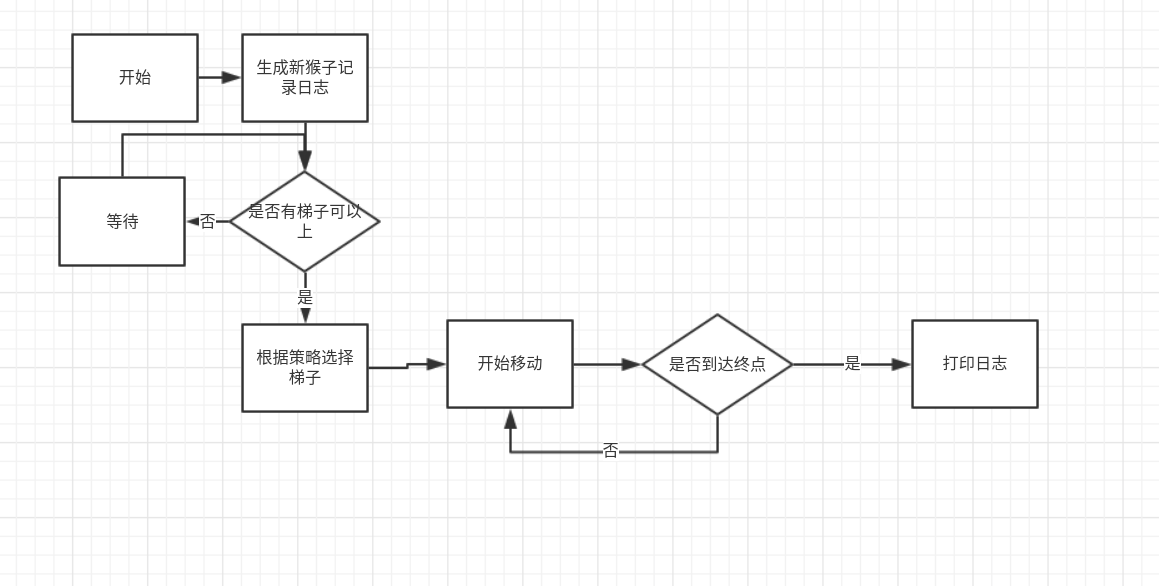


（可选）以类图形式给出多个类之间的关系。





## Monkey线程的run()的执行流程图



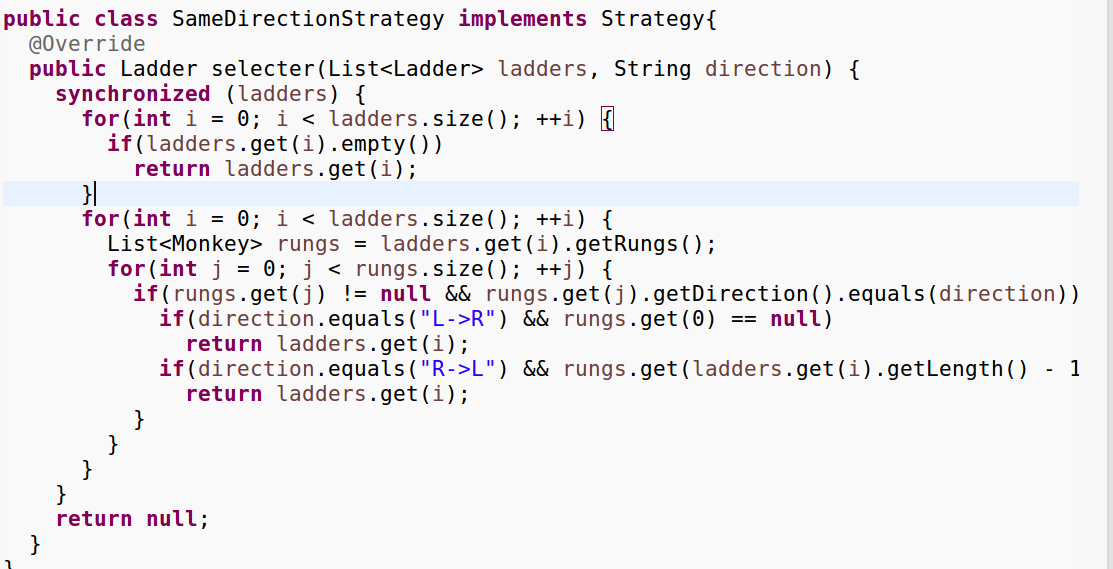
## 至少两种“梯子选择”策略的设计与实现方案

### 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;

只需要遍历所有梯子, 如果有梯子为空, 这直接选择这个梯子, 否则就去选方向一致的梯子.

代码如下:



### 策略2

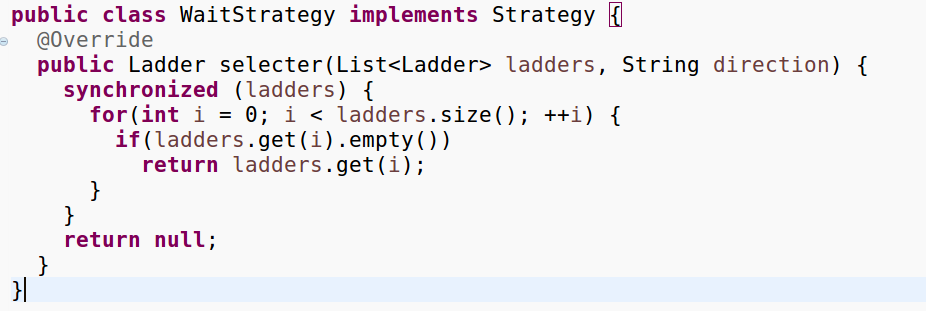
优先选择整体推进速度最快的梯子(没有与我对向而行的猴子、其上的猴子数量最少、梯子离我最近的猴子的真实行进速度最快);

代码如下:



### 策略3（可选）

优先选择没有猴子的梯子, 也就是空梯子, 如果没有空梯子, 再选择之前的两种



## “猴子生成器”MonkeyGenerator

首先看构造器, 接受四个参数, timeInterval, k, maxNum, maxSpeed

timeInterval为生成每一批猴子的时间间隔, k为每一次生成猴子的个数, maxNum为一共的猴子个数, maxSpeed为每只猴子的最大速度.

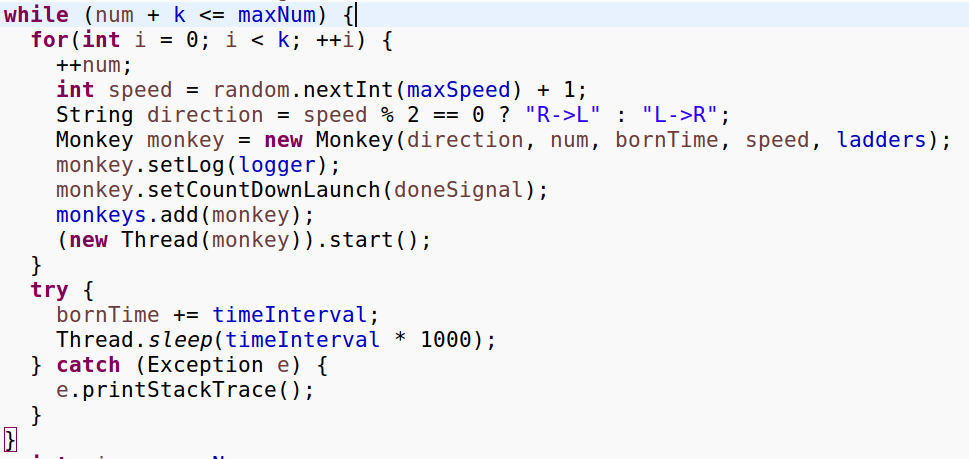
有如下两个方法:



分别用来进行接收所有的梯子和设置logger用来进行记录日志.

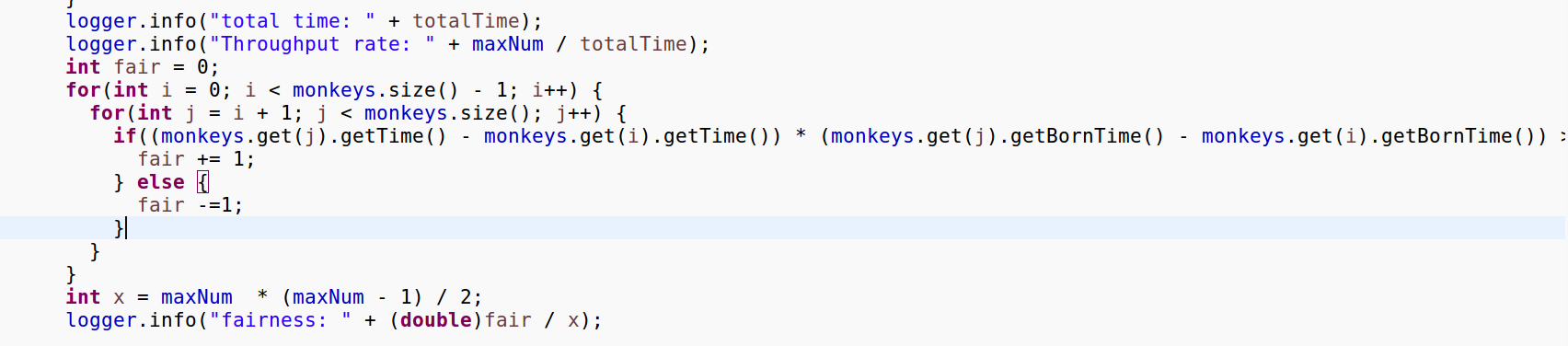
接下来就是重写的run()方法

有一个内部计数器, 用来判断生成的猴子数是不是已经达到了最大的数字, 并且可以根据这个计数器来为猴子进行命名, 满足按照时间顺序递增的要求, 每生成一只猴子就进行记录日志



接下来是进行日志记录的部分, 计算吞吐率, 公平性

根据给出的公式进行计算即可, 代码如下:

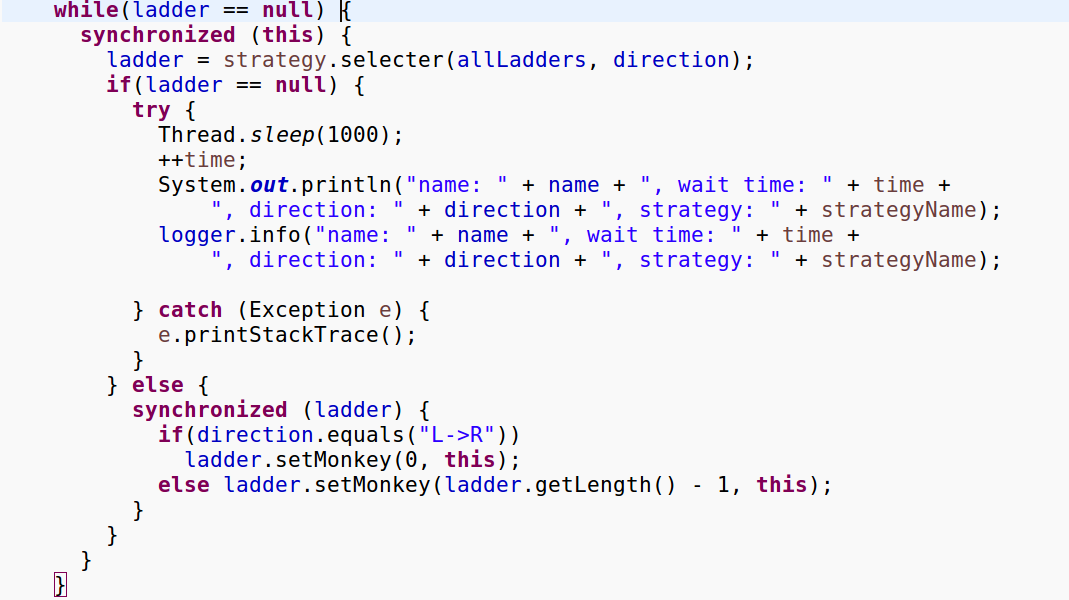


## 如何确保threadsafe？

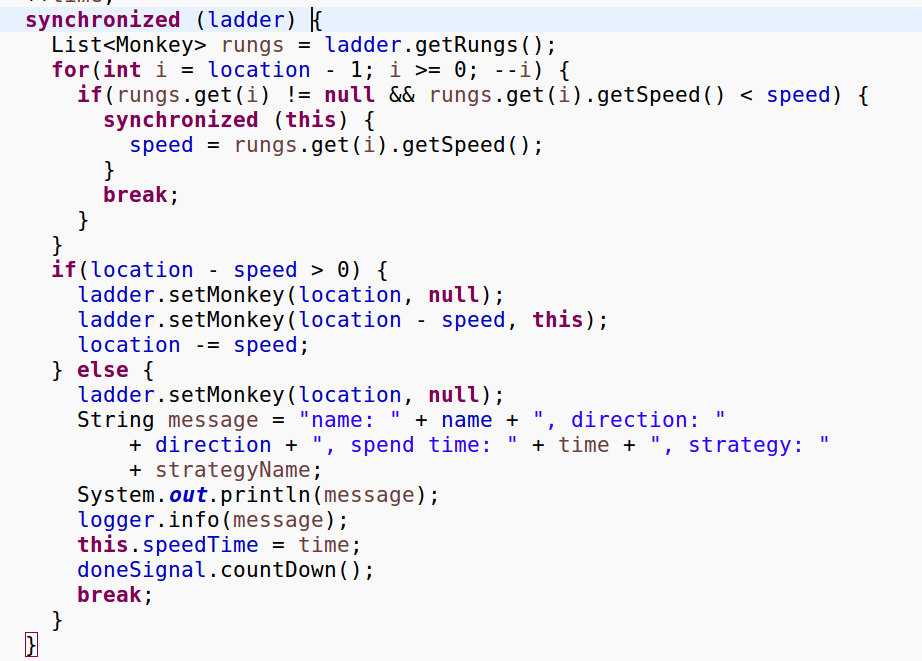
首先确认竞争的来源, 就是Monkey之间对于Ladders的竞争, 所以只需要考虑对于梯子进行操作的时候加上锁就好.

1. Monkey对于梯子的竞争

Monkey在进行选择梯子的时候, 要对梯子进行加锁, 这样子就可以防止多线程带来的一系列弊端.



1. 多只猴子在梯子上过河时候, 由于速度不同, 对于踏板的竞争



只要猴子在梯子上, 就将其即将要进行走的踏板加上锁.

## 系统吞吐率和公平性的度量方案

可以根据实验手册中给出的公式来进行计算

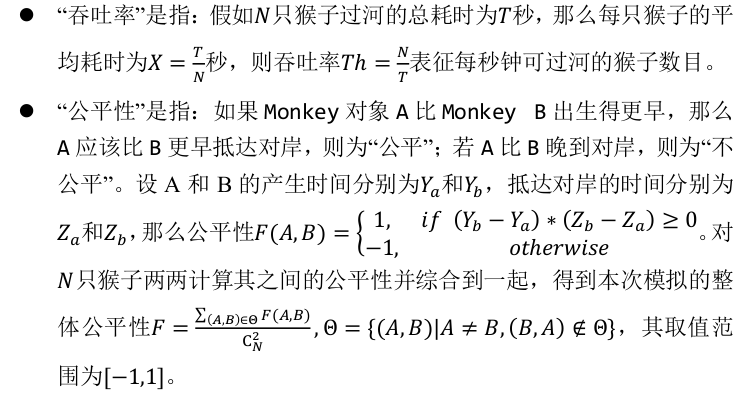
吞吐率表示每秒过河的猴子的数目,用公式N/T表示

公平性𝐹 = ∑(𝐴,𝐵)∈Θ 𝐹(𝐴,𝐵)/n/(n-1)\*2 , Θ = {(𝐴, 𝐵)|𝐴 ≠ 𝐵, (𝐵, 𝐴) ∉ Θ}，

计算方法:

吞吐率:求出找出所有猴子的最大到达时间longestTime,这个时间就是猴子们过桥的总时间,然后用N/(double)longestTime即可求得吞吐率

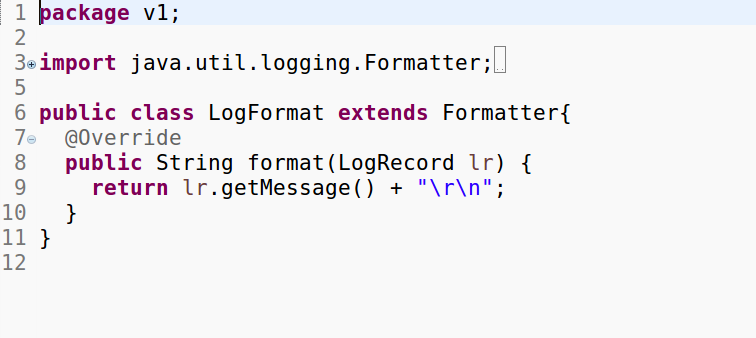
公平性:对于每一个猴子,让其跟id大于它的猴子比较时间,如果这两个猴子的BirthTime和ArriveTime不满足公平性的要求,那么adjustment减一,否则加一,最后除以即可



## 输出方案设计

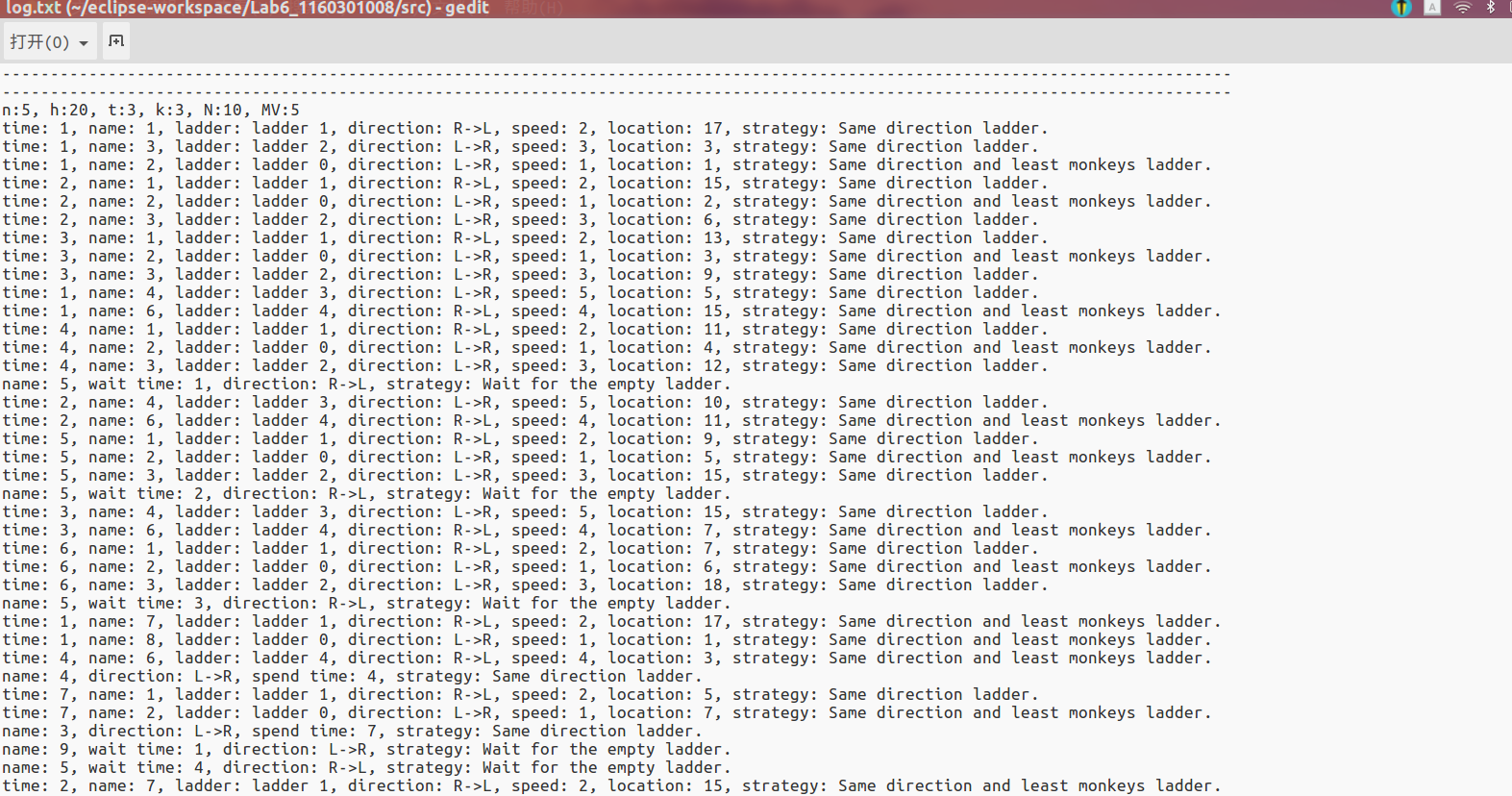
选择日志输出

代码如下:



每一步操作都进行日志记录

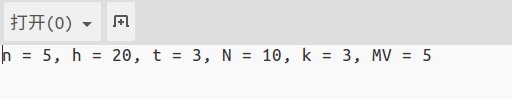
最终日志文件如下:



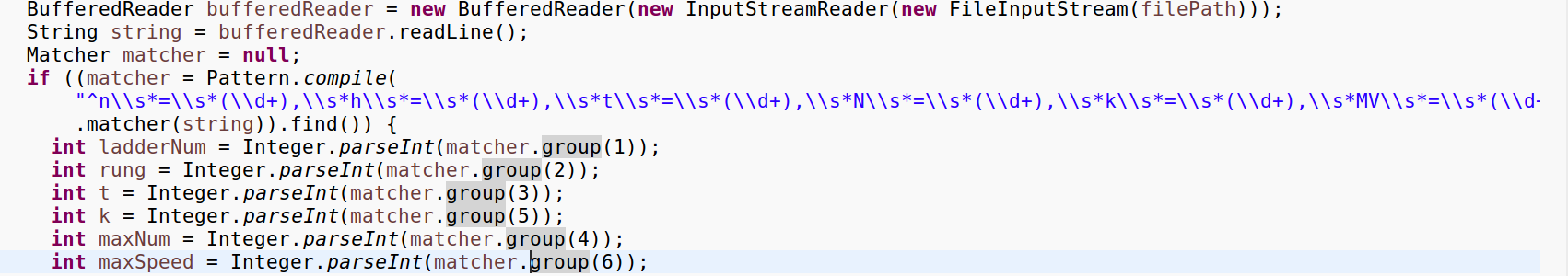
## 猴子过河模拟器v1

### 参数如何初始化

采用从文件中读取的形式来进行参数的初始化, 参数文件如下:



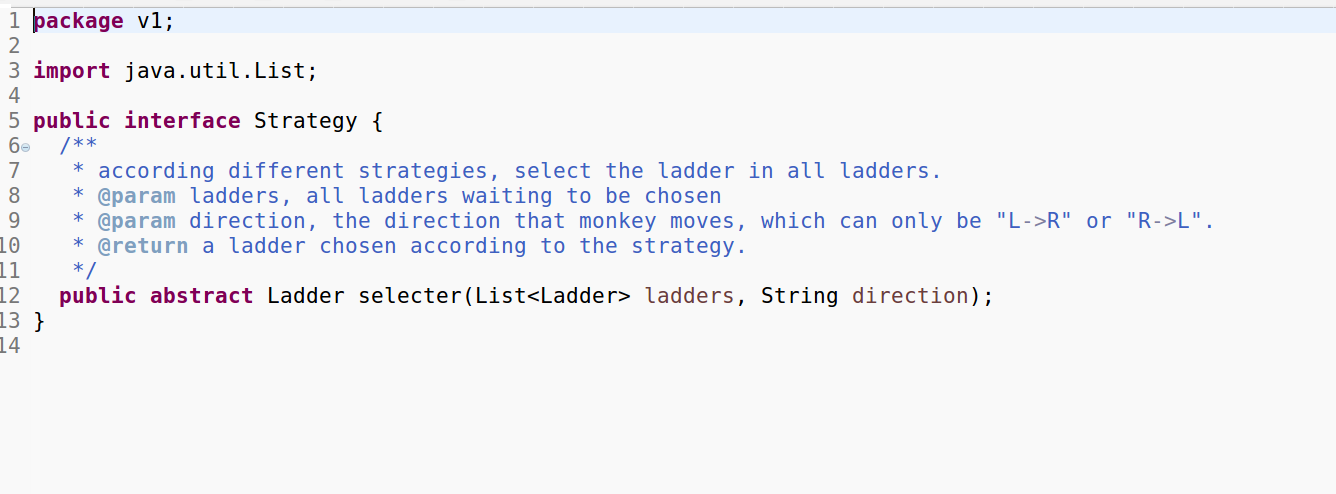
其中, 采取正则表达式进行字符串的匹配, 具体代码如下:



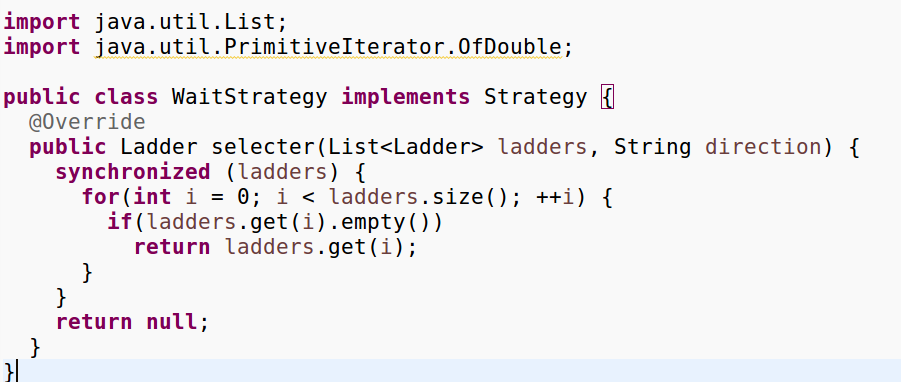
这样只需要修改文件中的参数, 就可以进行多次的实践.

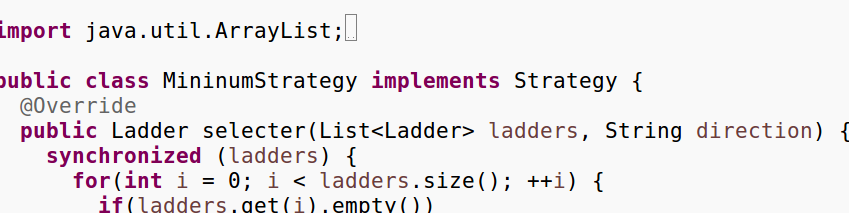
### 使用Strategy模式为每只猴子随机选择决策策略

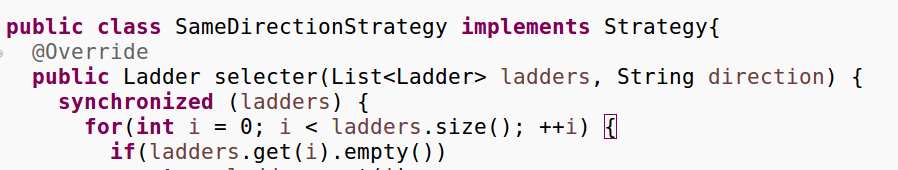
首先设计Strategy接口, 其中只有一个方法用来选择梯子, 代码如下:



然后有三种Strategy类分别用不同的实现策略去实现select方法, 代码如下:





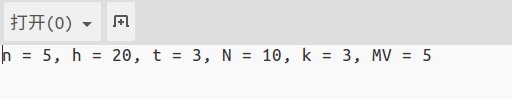


## 猴子过河模拟器v2

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

基础参数设置如下:



Strategy1:



Strategy2:



Strategy3:



### 对比分析：变化某个参数，固定其他参数

1. 首先固定其他参数, 只改变N

当N = 20 时:



当N = 40 时:



1. 固定其他参数, 只改变n

当n = 1 时:



当n = 2 时:



当n = 4 时:

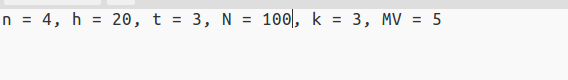


### 分析：吞吐率是否与各参数/决策策略有相关性？

根据上述结果, 可以看出来吞吐率和参数之间的依赖性还是比较大的, 比如说和n还有N都有依赖性, 而且影响较大, 但是对于策略来说, 有影响, 但是没有参数的影响大.

### 压力测试结果与分析

1. 压力测试1: 设计一种参数配置，使得产生的猴子数量非常多、非常密集，而梯子数量有限。观察此时你的程序的吞吐率和公平性表现如何。参数如下:



实验结果如下:



1. 压力测试2: 设计一种参数配置，使得各猴子的速度差异非常大。观察此时你的程序的吞吐率和公平性表现如何。这里调整最大速度特别大, 这样造成猴子之间差距特别大, 参数如下:

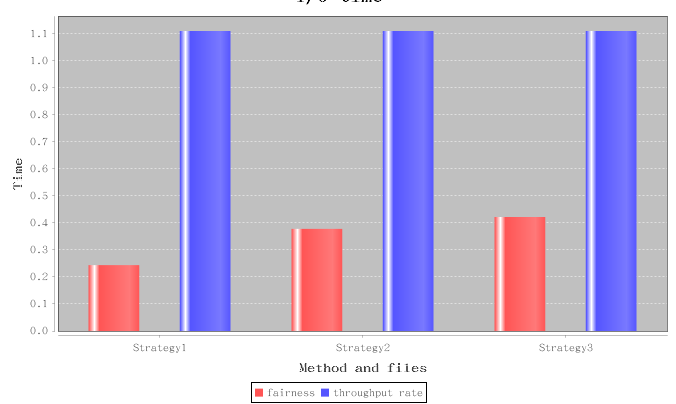


结果如下:

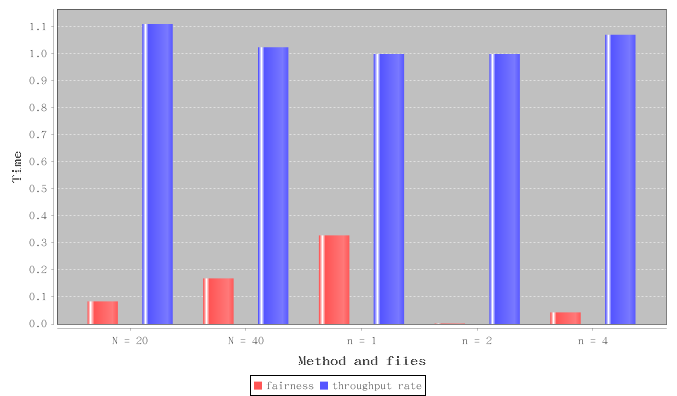


**采用JFree Chart进行对比**

(1)不同策略下对于公平性和吞吐率的影响



1. 不同参数下对于吞吐率和公平性的影响



# 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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 实验过程中遇到的困难与解决途径

对于多线程相关的知识一直都不太懂, 上课没太听懂, 所以首先看了<Java编程思想>, 自学了多线程, 然后才开始写实验.

# 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。