

**2019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 | 李震宇 |
| 学号 | 1170300110 |
| 班号 | 1703001 |
| 电子邮件 | [1137859144@qq.com](mailto:1137859144@qq.com) |
| 手机号码 | 18800419432 |

**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 和性能之间寻求较优的折中，为此计算吞 吐率和公平性等性能指标，并做仿真实验。

 Java 多线程编程

 面向线程安全的 ADT 设计策略选择、文档化

 模拟仿真实验与对比分析

# 实验环境配置

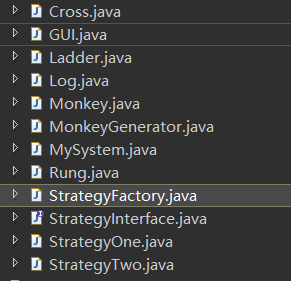
java + eclipse

<https://github.com/ComputerScienceHIT/Lab6-1170300110.git>

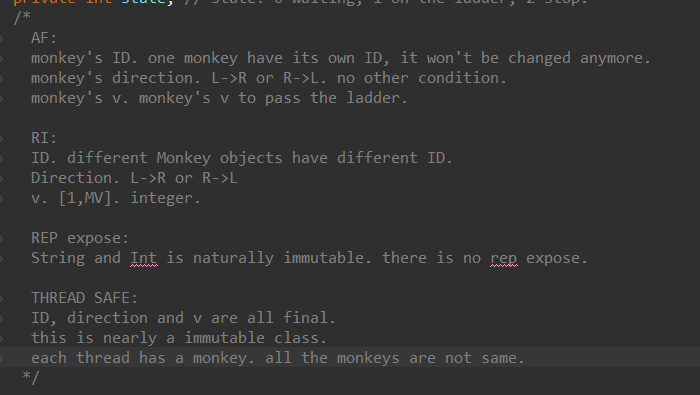
# 实验过程

## ADT设计方案

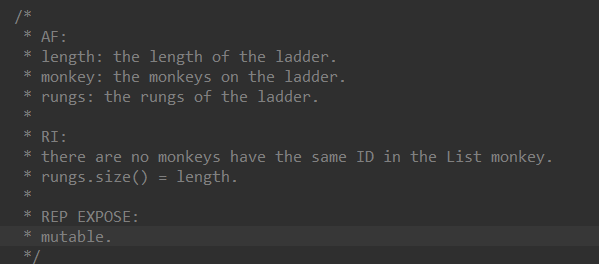
ADT：猴子、梯子、梯子上的木板、猴子决策、猴子移动、系统、GUI。



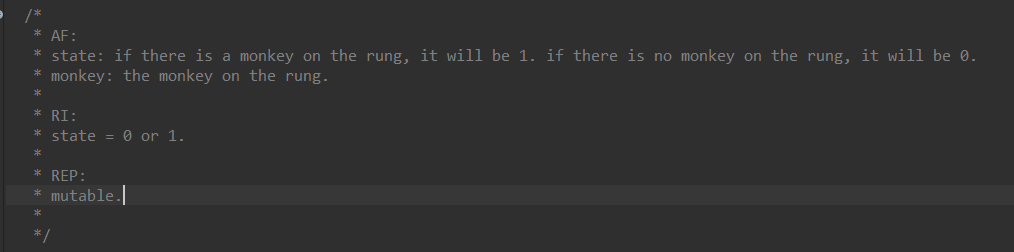
猴子：



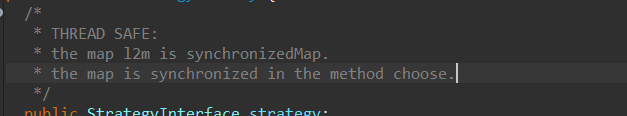
梯子：



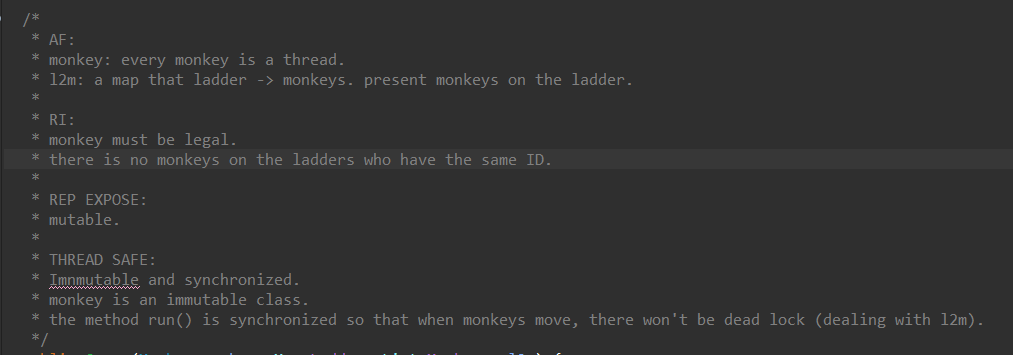
木板：



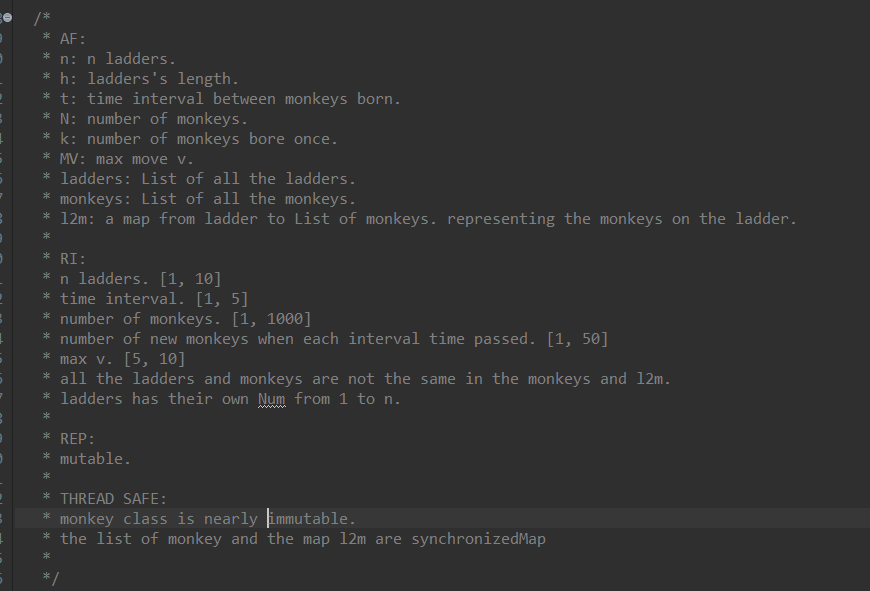
猴子决策：



猴子移动：



系统：



每个猴子是一个线程，在cross类中实现猴子的决策，使用strategy设计模式选择上桥的方式1或2，如果没上桥，则等待，如果上了桥，开始移动。

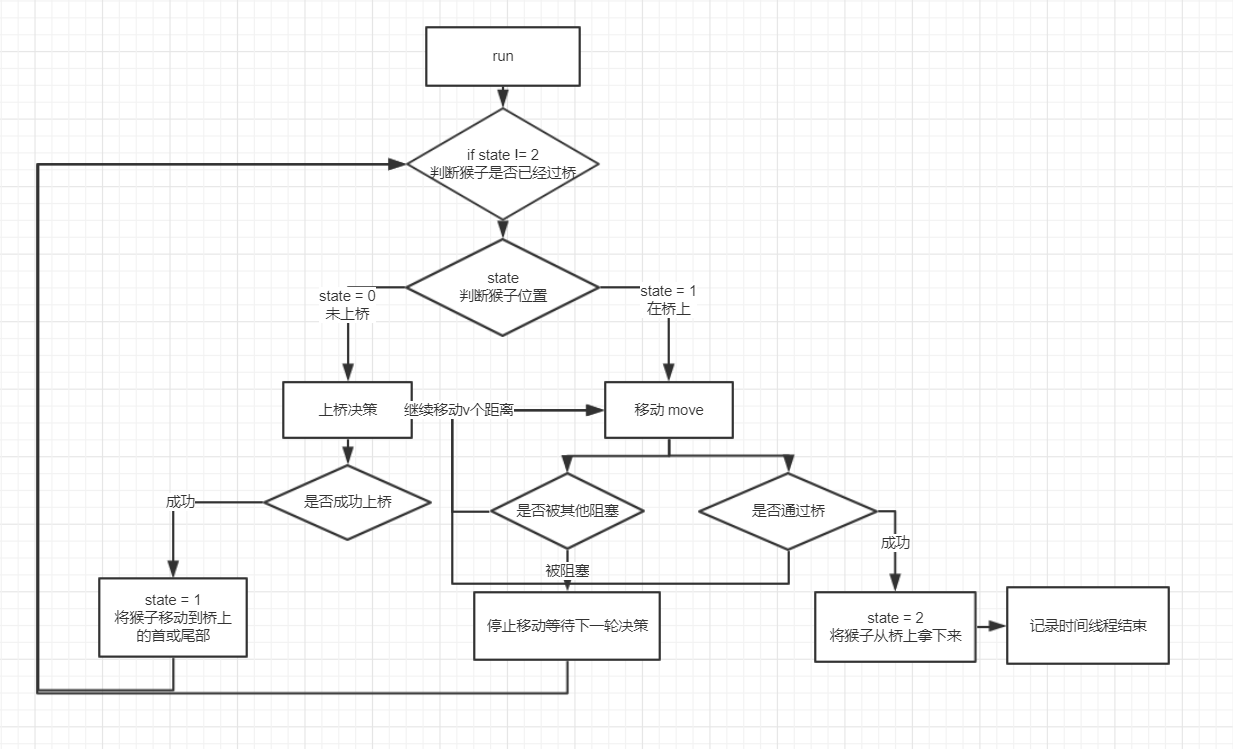
在移动时，过河的瞬时情况存储在map l2m中，其中每一个键值对ladder -> List<Monkey>表示某一条桥上的猴子有哪些，在猴子移动时，通过这个map就可以对猴子自己的下一步异动情况进行决策与判定。每一个cross中都有着一个当前状态的l2m，以帮助进行决策。

而具体猴子在桥的哪个木板上则由每个ladder类中的list rungs决定，如果该块木板上有猴子则该rung的state属性为1，如果没有猴子则为0，通过遍历ladder类中的rungs列表即可确定猴子的具体位置。

MySystem就是一个系统，负责处理参数，开始线程，处理数据等。

GUI则是可视化参数读入和猴子过河过程模拟。

## Monkey线程的run()的执行流程图



省略了一些细节，但大体上是这个流程。

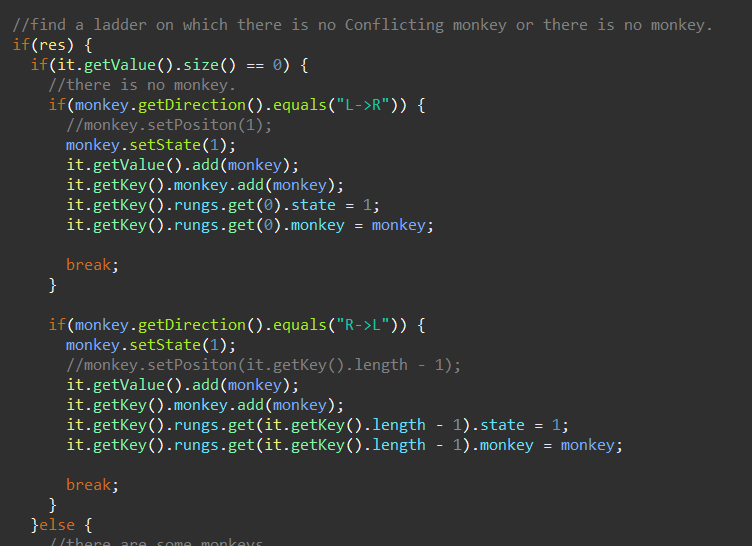
## 至少两种“梯子选择”策略的设计与实现方案

### 策略1



优先选择没有猴子的梯子，若所有梯子上都有猴子，则优先选择没有与其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

猴子生成器的目的是每隔一定时间生成几个猴子。这里我使用的方式是在系统中计时，每隔固定的时间调用猴子生成器，生成ID不重复按时间排序的x个猴子。



传入参数：

k：每次产生猴子的个数。

MV：猴子的最大速度，使用random方法随机生成猴子的速度。

num：已有猴子数，方便产生ID不同的猴子

start：调用本次猴子生成器的时间，此时生成的猴子含有相同的出生时间start

return：返回一个含有本次生成的所有猴子的列表。

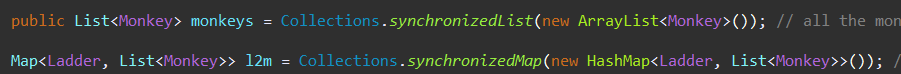
## 如何确保threadsafe？

1.在调用move()方法时将其上锁



这样确保移动的时候不会被其他猴子干扰。

2.System中的monkey列表和l2m图使用线程安全的类型



3.决策方法中的图也使用线程安全的类型



4.在决策时使用部分上锁，保证线程安全并防止死锁的发生：



## 系统吞吐率和公平性的度量方案

“吞吐率”是指：假如𝑁只猴子过河的总耗时为𝑇秒，那么每只猴子的平均耗时为𝑋 = 𝑇 𝑁 秒，则吞吐率𝑇ℎ = 𝑁 𝑇 表征每秒钟可过河的猴子数目。

“公平性”是指：如果 Monkey 对象A比B出生得早，那么 A 应该不晚于 B 抵达对岸，则为“公平”；若 A 比 B 晚到对岸，则为“不公平”。设 A 和 B 的产生时间分别为𝑌𝑎和𝑌𝑏，抵达对岸的时间分别为𝑍𝑎和𝑍𝑏，那么公平性：

𝐹(𝐴,𝐵) = 1, 𝑖𝑓 (𝑌𝑏 − 𝑌𝑎) ∗ (𝑍𝑏 − 𝑍𝑎) ≥ 0

−1, 𝑜𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒

对 𝑁 只猴子两两计算其之间的公平性并综合到一起，得到本次模拟的整体，公平性

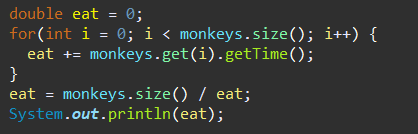
𝐹 =∑ 𝐹(𝐴,𝐵) (𝐴,𝐵)∈Θ C𝑁 2 ,Θ = {(𝐴,𝐵)|𝐴 ≠ 𝐵,(𝐵,𝐴) ∉ Θ}，其取值范围为[−1,1]。

实现方法：

首先等待所有线程结束再开始计算：



计算吞吐率：

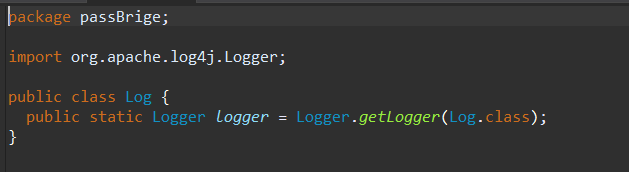


计算公平性：

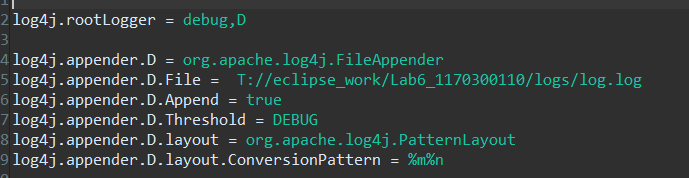


## 输出方案设计

日志：

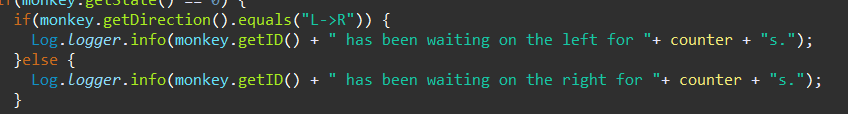


日志配置：

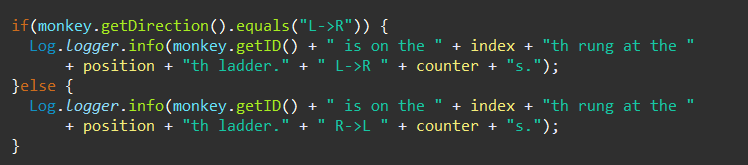


写入日志位置：

判断在河边等待后写入状态信息。



发生移动后写入状态信息。



过河后写入过河计时信息。

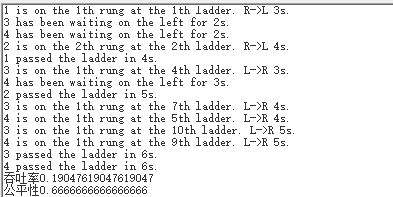


GUI：



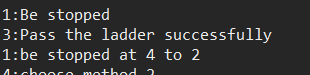
输入参数循行后：

打印出每次的执行情况，猴子的位置，以及吞吐率和公平性。



在控制台也有更详细的过程分析，额外有在哪里猴子被阻塞了：

比如：

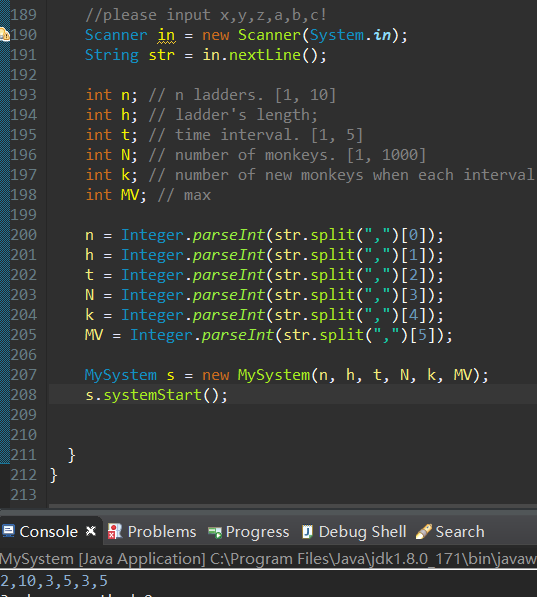


从这里我们可以知道1在移动的时候被前边的猴子阻塞了。从4号板移动到了2号板子。

## 猴子过河模拟器v1

### 参数如何初始化

可以直接从命令行中读入：

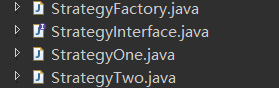


也可以GUI中读入参数：

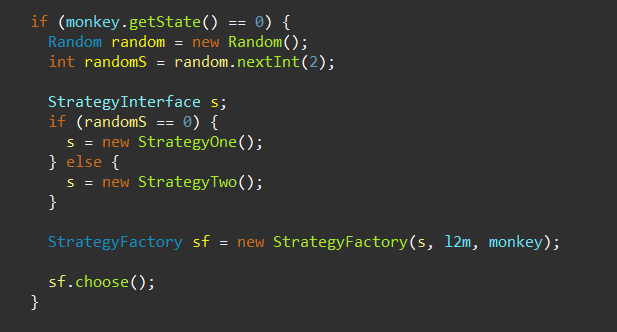


### 使用Strategy模式为每只猴子随机选择决策策略

Strategy设计模式，包含一个工厂类、一个方法接口和两个具体方法实现。



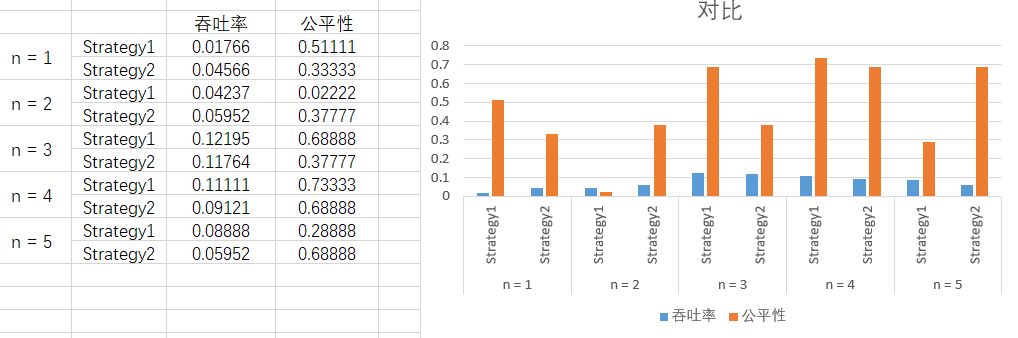
随选测策略：



生成一个随机数0或者1，如果是0就向工厂传入StrategyOne调用方法1，如果是1就向工厂传入StrategyTwo调用方法2，这样就实现了随机选择策略。

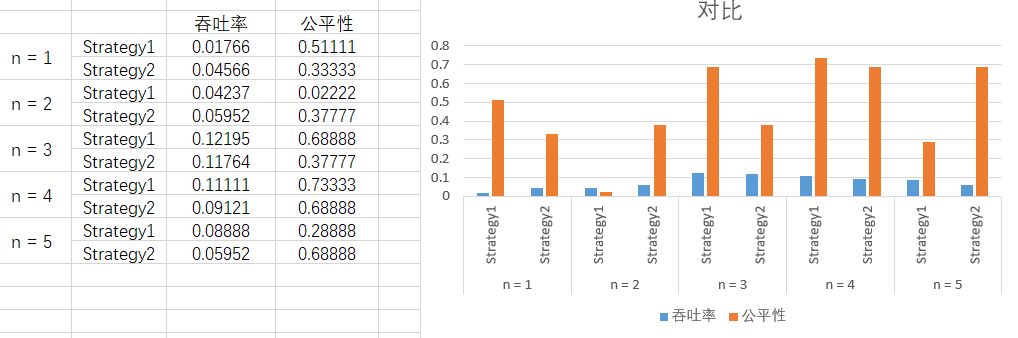
## 猴子过河模拟器v2

### 对比分析：固定其他参数，选择不同的决策策略



### 对比分析：变化某个参数，固定其他参数

变化梯子个数：



梯子个数不同，不同决策方式导致的吞吐率和公平性的相对大小会不同，并没有明显的趋势表明某个决策方式的吞吐率与公平性明显好于另一种。

### 分析：吞吐率是否与各参数/决策策略有相关性？

吞吐率应该是和各参数、决策策略有相关性的。

如果梯子数很多、梯子长度较短、猴子的移动速度很快，从直觉来讲都会增大这个系统

的吞吐率，因为这个梯子可以容纳更多的猴子且让梯子上的猴子迅速的完成任务。

对于决策策略，如果选择一个优秀的决策策略，应该也会让吞吐率升高，因为可以同时用一个梯子处理多个猴子线程的过河工作。

在实验中因为MV会随机变化很大，所以尽管进行了多次实验数据的采集，也没有很明显的规律显示出来。可能需要更多的数据堆积。

### 压力测试结果与分析

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。

使用参数：



梯子数2，梯子长度20，生成猴子间隔1，猴子数量50，每次生成10，最大速度5

结果：



吞吐率：0.0296，公平性：-0.0138

显然，当猴子特别多的情况，它们只有很少一部分能在桥上运动，因为很密集的产生猴子，大量猴子将堆积在桥两侧等待，将导致系统的吞吐率下降，当某个猴子通过桥时，其他的猴子会竞争桥的使用权，那么就有可能一些后出生的猴子从先出生的猴子手中抢到了使用权，这样就导致公平性下降了。

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。

使用参数：



梯子数3，梯子长度20，生成猴子间隔3，猴子数量20，每次生成5，最大速度10

结果：



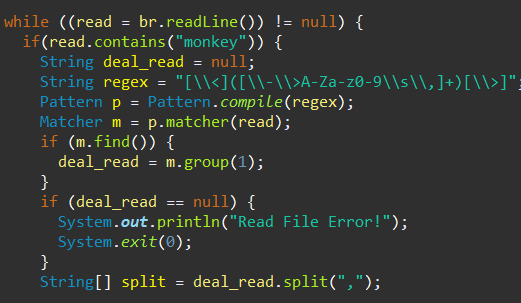
吞吐率：0.0896，公平性：0.5894

当这种速度差异非常大时，吞吐率表现一般，但是公平性表现的比较好，因为猴子以一上桥，就有可能以很快的速度，用很短的时间过桥，保证在他之后生成的猴子很难比他还快，这样就保障了一些公平性。

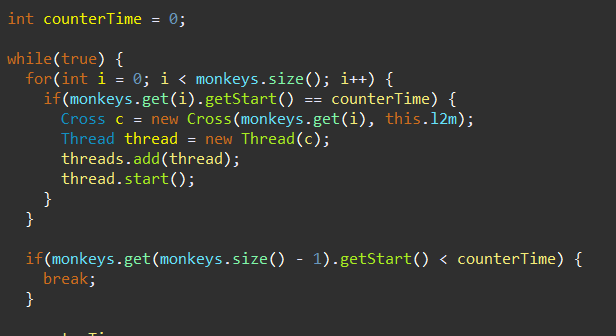
## 猴子过河模拟器v3

修改代码：

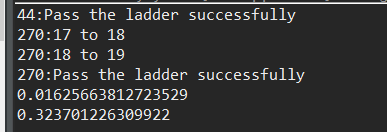
用正则表达式匹配的方法来构造猴子：



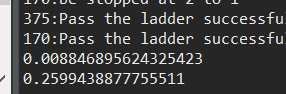
开启线程：



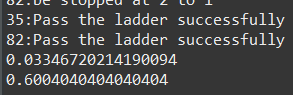
文件一某一次完成的截图:



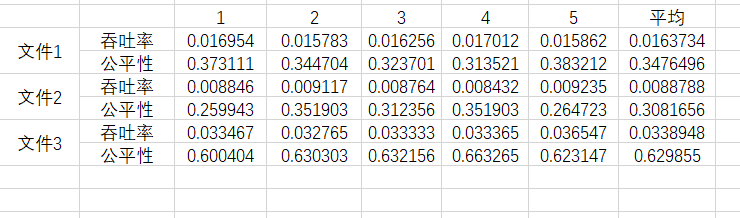
文件二某一次完成的截图：

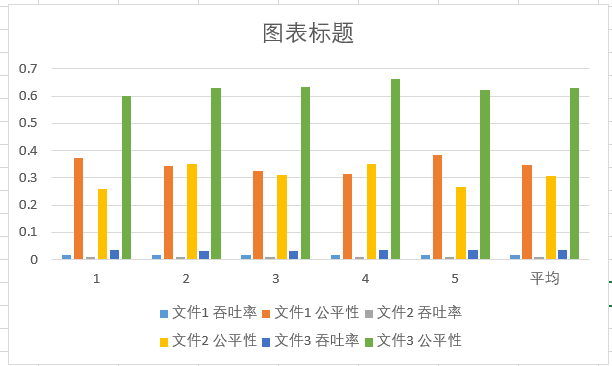


文件三某一次完成的截图：



最终结果：





# 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 |
| 6.1 | 晚上 | 设计v1 | 完成 |
| 6.2 | 晚上 | 实现v1 | 未完成 |
| 6.3 | 晚上 | 实现v1、实现v2 | 完成 |
| 6.4 | 晚上 | 对v2进行分析 | 完成 |
| 6.5 | 晚上 | 实现v3 | 完成 |
| 6.6 | 晚上 | 测试v3得出结论 | 完成 |

# 实验过程中遇到的困难与解决途径

出现了几次线程不安全的问题，也有死锁的情况产生，更换锁的大小来解决这些问题。

# 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。

1.容易出现好多死锁情况，也有线程不安全的情况发生，问题很麻烦，很难debug。但是处理问题的效率更高。

2.使用immutable并加上锁来保证threadsafe。没办法必须保证程序正确的前提下实现threadsafe舍弃性能没有办法。

3.遇到过不安全的情况，将锁变小来解决。

4.工作量不大，很难，deadline可以接受。

5.终于要结课了。