

**2019年春季学期  
计算机学院《软件构造》课程**

**Lab 6实验报告**

|  |  |
| --- | --- |
| 姓名 | 范天祥 |
| 学号 | 1170300815 |
| 班号 | 1703008 |
| 电子邮件 | 2698791816@qq.com |
| 手机号码 | 18846818843 |

**目录**

[1 实验目标概述 1](#_Toc1393927)

[2 实验环境配置 1](#_Toc1393928)

[3 实验过程 1](#_Toc1393929)

[3.1 ADT设计方案 1](#_Toc1393930)

[3.2 Monkey线程的run()的执行流程图 4](#_Toc1393931)

[3.3 至少两种“梯子选择”策略的设计与实现方案 5](#_Toc1393932)

[3.3.1 策略1 5](#_Toc1393933)

[3.3.2 策略2 5](#_Toc1393934)

[3.3.3 策略3（可选） 5](#_Toc1393935)

[3.4 “猴子生成器”MonkeyGenerator 6](#_Toc1393936)

[3.5 如何确保threadsafe？ 6](#_Toc1393937)

[3.6 系统吞吐率和公平性的度量方案 7](#_Toc1393938)

[3.7 输出方案设计 8](#_Toc1393939)

[3.8 猴子过河模拟器v1 13](#_Toc1393940)

[3.8.1 参数如何初始化 13](#_Toc1393941)

[3.8.2 使用Strategy模式为每只猴子随机选择决策策略 13](#_Toc1393942)

[3.9 猴子过河模拟器v2 14](#_Toc1393943)

[3.9.1 对比分析：固定其他参数，选择不同的决策策略 14](#_Toc1393944)

[3.9.2 对比分析：变化某个参数，固定其他参数 16](#_Toc1393945)

[3.9.3 分析：吞吐率是否与各参数/决策策略有相关性？ 20](#_Toc1393946)

[3.9.4 压力测试结果与分析 20](#_Toc1393947)

[4 实验进度记录 22](#_Toc1393948)

[5 实验过程中遇到的困难与解决途径 27](#_Toc1393949)

[6 实验过程中收获的经验、教训、感想 27](#_Toc1393950)

[6.1 实验过程中收获的经验和教训 27](#_Toc1393951)

[6.2 针对以下方面的感受 27](#_Toc1393952)

# 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

简要陈述你配置本次实验所需环境的过程，必要时可以给出屏幕截图。

特别是要记录配置过程中遇到的问题和困难，以及如何解决的。

环境配置同实验五

在这里给出你的GitHub Lab6仓库的URL地址（Lab6-学号）。

https://github.com/ComputerScienceHIT/Lab6-1170300815

# 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；

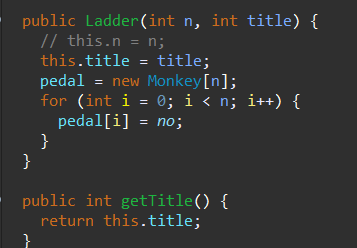
给出每个ADT的specification；

1. **首先是Ladder类**

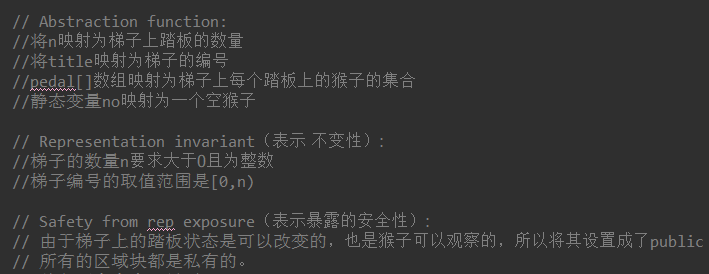
作用：猴子过桥的组件

属性：梯子的编号title，每个梯子上的踏板数量n

方法：得到梯子的属性。



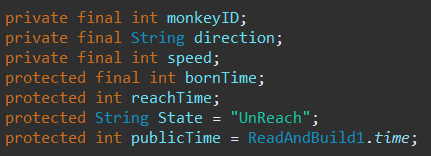
**Specification：**



1. **其次是Monkey类**

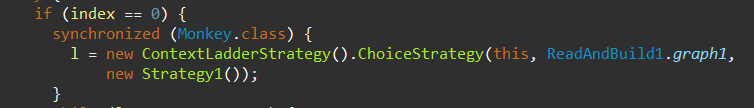
作用：过桥的对象

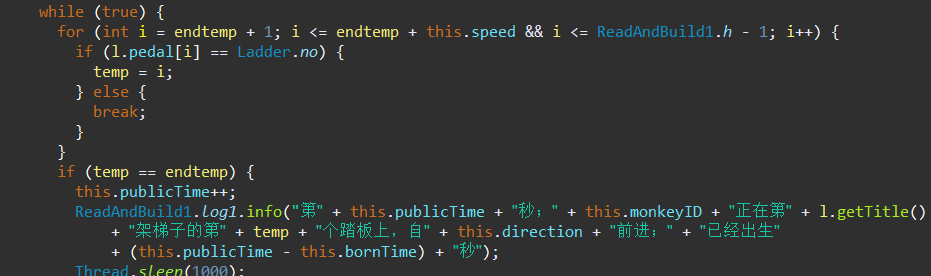
属性：猴子编号、猴子方向、猴子速度、猴子出生时间等。



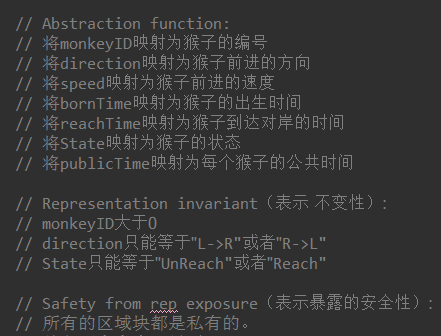
方法： 锁住猴子的类，让同一时间只能有一只猴子选择策略；

让猴子跑起来。





**Specification：**

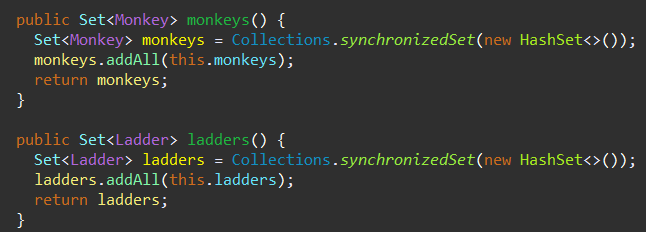


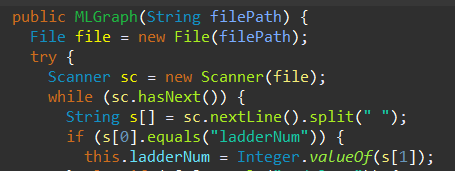
1. **Relation类构建monkey和ladder的关系：**

作用：搭建过桥系统

属性：猴子、桥、关系和设置系统参数

方法：读取配置文件构建过桥系统

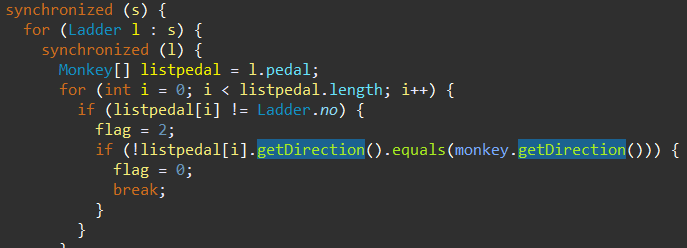




**4. Strategy类**

作用：提供策略

方法：具体选择方式（在此只列举一个）



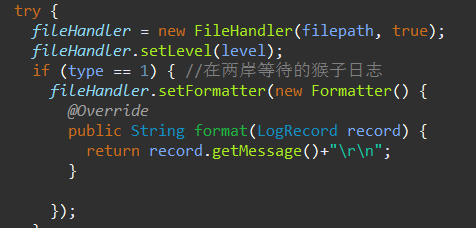
**Specification：**



**5.factory**

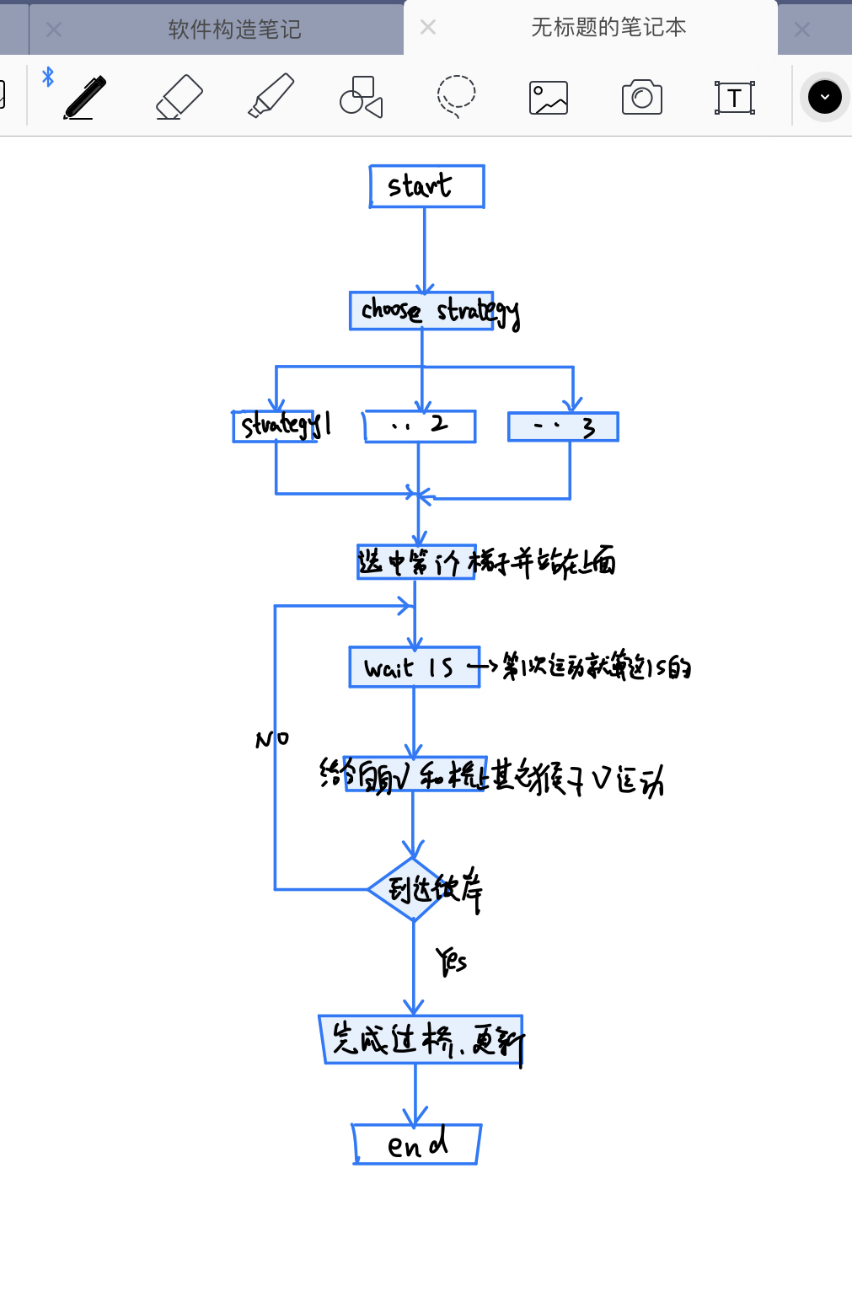
作用：记录操作及猴子过桥的过程

方法：就是搭建log的基本格式



## 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，**

A.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；

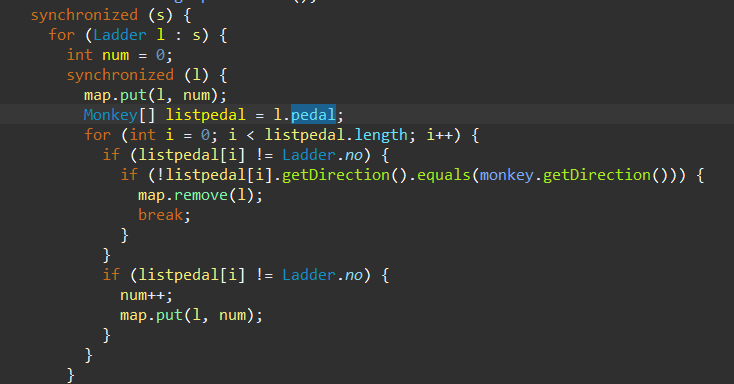
B.若满足该条件的梯子有很多，则随机选择



### 策略2

**优先选择整体推进速度最快的梯子**

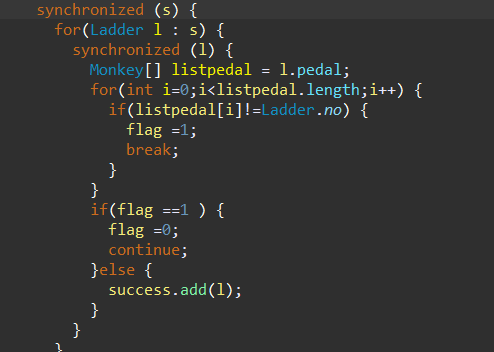
（没有与我对向而行的猴子、其上的猴子数量最少）；



### 策略3（可选）

**优先选择没有猴子的梯子，**

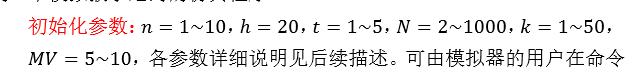
若所有梯子上都有猴子，则在岸边等待，直到某个梯子空闲出来；



## “猴子生成器”MonkeyGenerator

“猴子生成器”主要是在规定的时间片段内产生设定数量的猴子。

设计方式为：按照指导书给的参数范围



**1．先写一个规范文件**，然后遵守这个规范文件生成猴子，生成的猴子直接会开启线程。

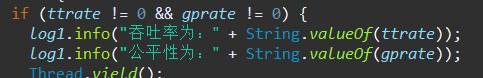
**2.调用配置文件**的参数来初始化过桥系统，具体的生成猴子的逻辑在MonkeyGenerator()方法中。；



**3.猴子生成器，猴子生成之后立即启动线程：**



**4.等待猴子全部过桥后，立即计算吞吐率和公平性：**



## 如何确保threadsafe？

1. **将线程不安全的集合转化为线程安全的集合。**





**理由如下：**

A.在多线程程序中，应当**尽量使用线程安全的集合**。

B.在集合的修改和查询过程中往往涉及到很多复杂的操作。

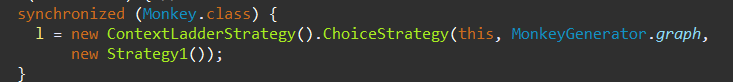
C.比如在本实验中，**monkeys和ladders是使用set数据结构的，**在set集合中，在添加或删除元素时，需要对其中的树结构进行调整，一般需要在log(n)时间内才能完成，这样如果两个线程同时对同一个集合进行修改，就很可能造成这个集合的崩溃。可以使用读写锁来对集合的修改加以控制，但是这种控制往往是复杂的，并且低效。因此java提供了一些线程安全的集合类，在多线程程序中可以使用这些线程安全的集合以避免可能的不一致和崩溃现象。

**D. HashSet不同步**，为了同步这些不同步的集合，使用

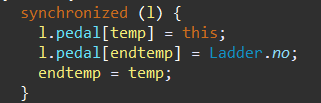
Set s=Collections.synchronizedSet(new HashSet());即可。

1. **在选择策略时**

**锁住猴子的类，让同一时间只能有一只猴子选择策略**



1. **在猴子移动的时候锁住梯子（l即为梯子）**

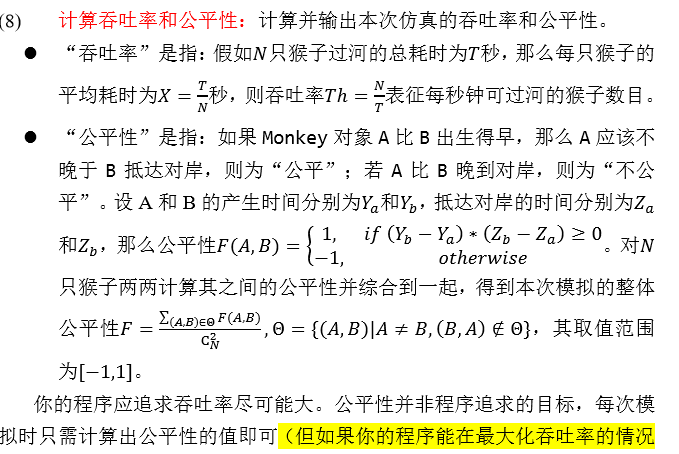


1. **最后在生成猴子时，立即启动线程**

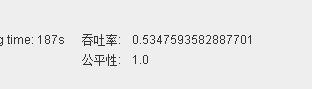


## 系统吞吐率和公平性的度量方案

**首先看指导书给的方案：**



**GUI效果：**



1. 计算吞吐率：

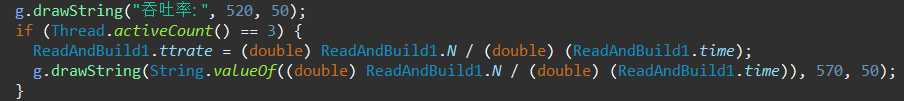
“吞吐率”是指：

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，

吞吐率Th=N/T表示每秒钟可以过河的猴子的数目。

根据线程池中的线程全部执行结束时距离第一只猴子开始过河的时间

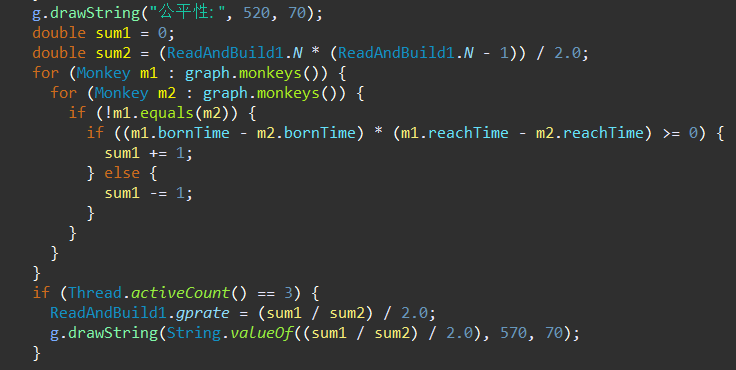
以及猴子的总数目可以求得。



(2)计算公平性：

公平性是指：

1. 如果A猴子比B猴子出生的早，但是到达河对岸的时间比B晚，那么对于A和B而言就不公平，相反就是公平。
2. 这只适用于不同批次产生的猴子之间的差别，比如第1批产生的猴子比第3批产生的猴子出生时间要早；但是如果两只猴子属于同一个批次，即为在相同的秒数出生，那么我们认为无论谁先到达对岸都是公平的。

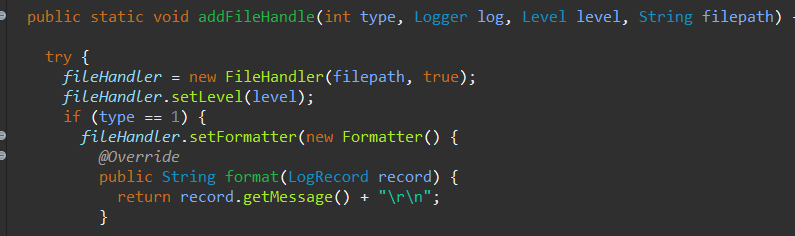


## 输出方案设计

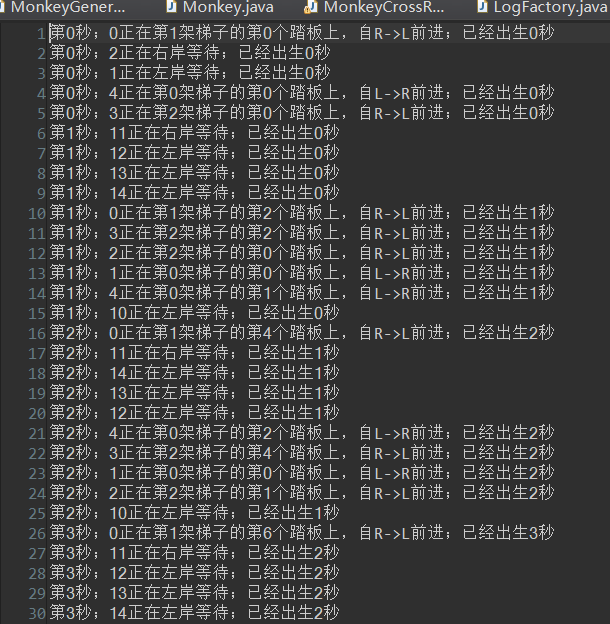
**日志**：

写了一个日志文件的配置方案，内部是设置日志的等级和读写文件等基本操作，设置好时间格式，记录猴子出生、等待过河的时间、过河花费的时间和运行总时间等。

将所有关于 Monkey 对象过河的全过程均记录在 monkey.generator的txt文件中。

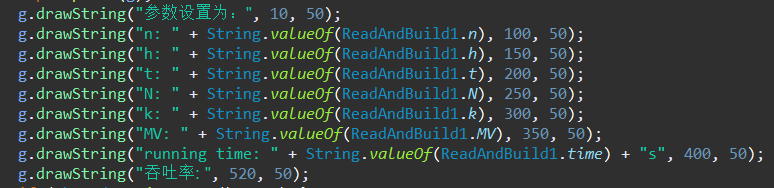


**日志效果**（记录猴子出生、等待过河的时间、过河花费的时间和运行总时间等）：

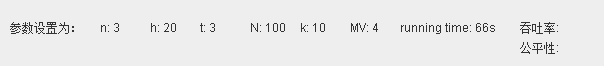


GUI：

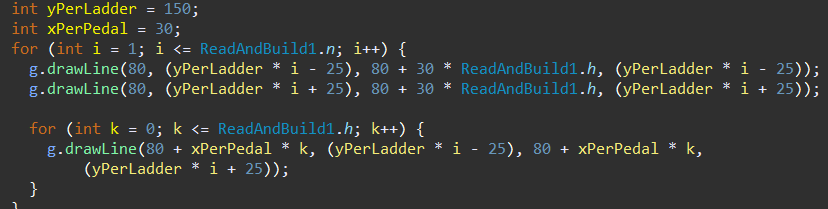
**展示参数：**



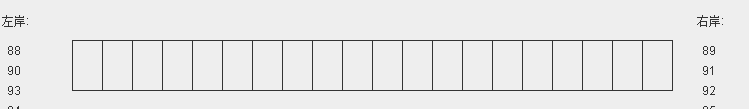
**效果：**



**画梯子：**

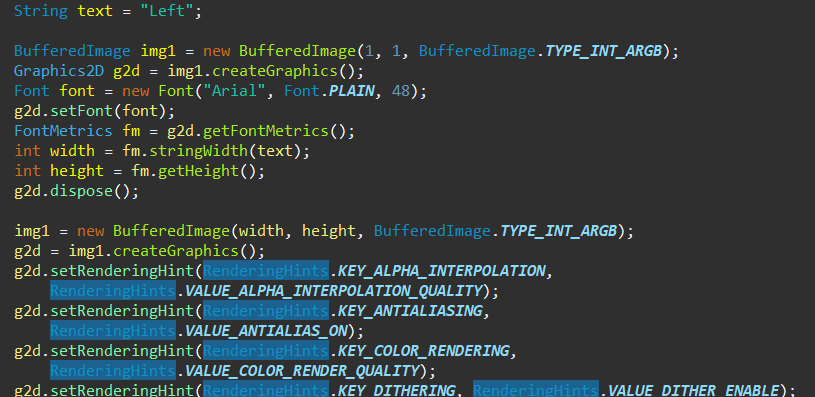


**效果：**



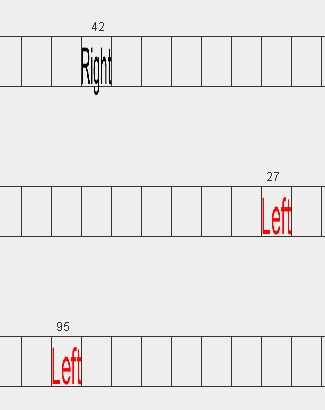
**画猴子：**

我这里是用BufferedImage和Graphics2D，实现把字符串转化为图片代替猴子，并添加到梯子上：



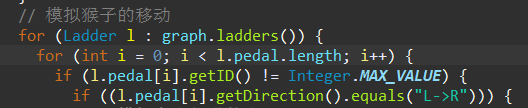


**效果：（Right为右边产生的猴子，Left为左边产生的猴子）**

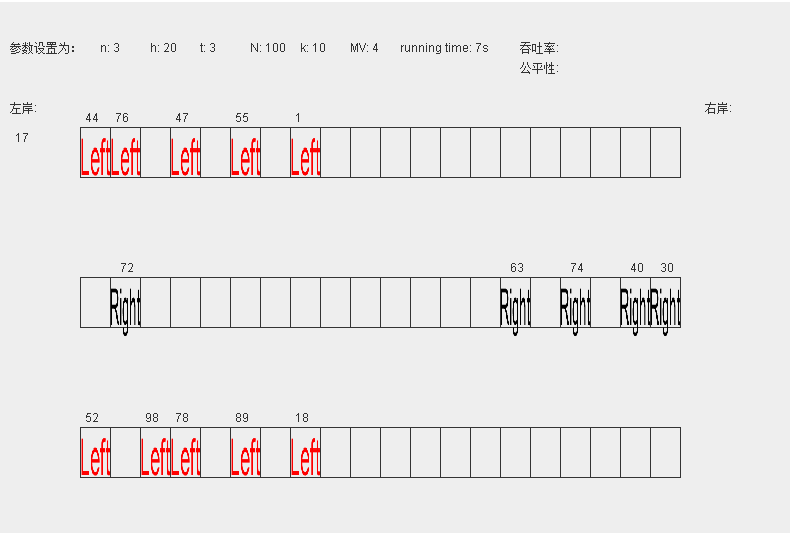


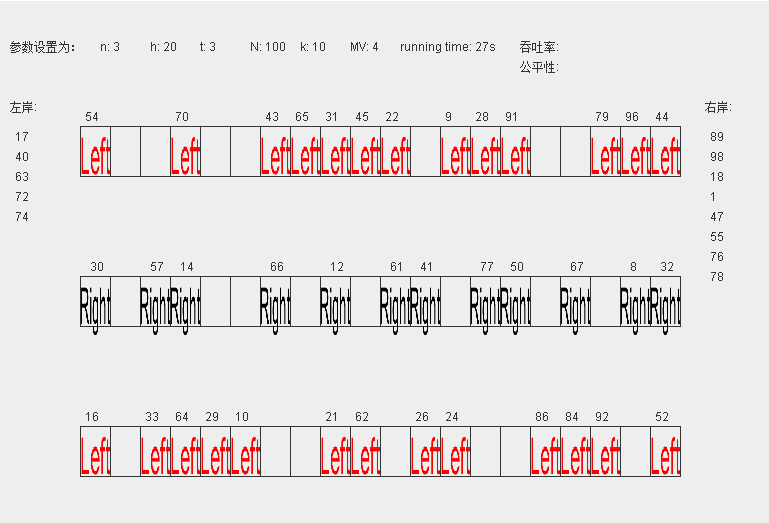
**可视化（可选）：**

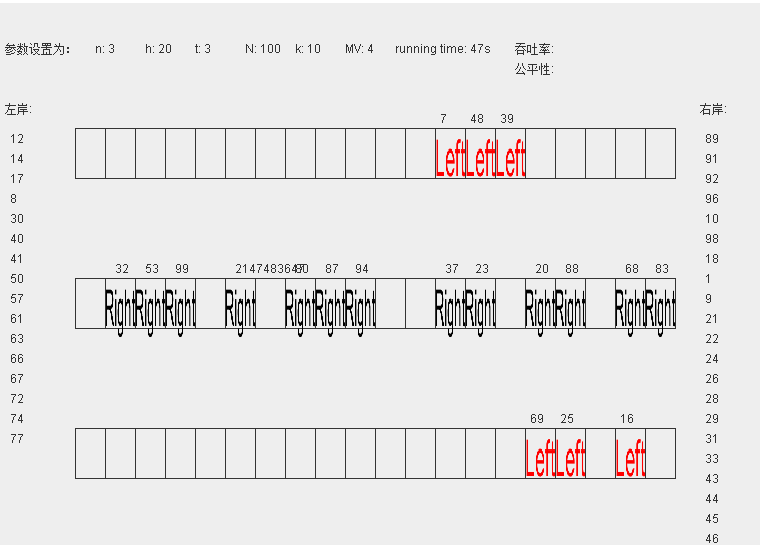
**模拟猴子运动:**



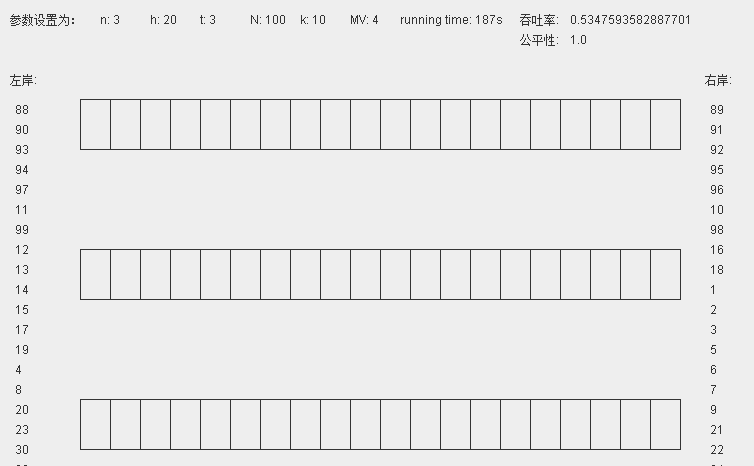
**效果展示：**







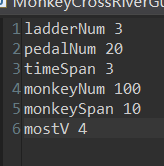
**过河完毕:**



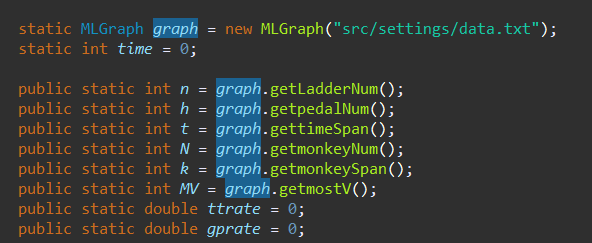
## 猴子过河模拟器v1

### 参数如何初始化

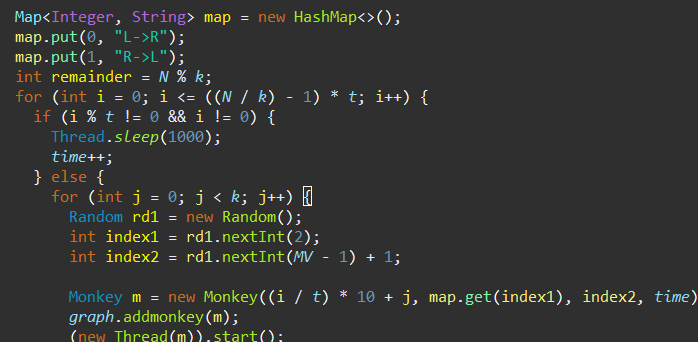
**写一个参数文件**：



**读文件的方式获取参数;**

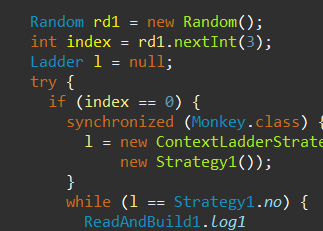


**然后在指定的参数范围内随机产生猴子**

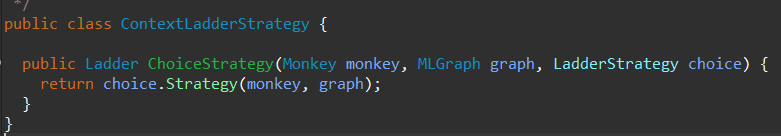


### 使用Strategy模式为每只猴子选择决策策略

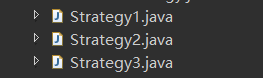
先随机生成数字作为选择的策略序号：



再用strategy设计模式返回选择的方案：



所有的三个策略类，打包在一个strategy的包内：

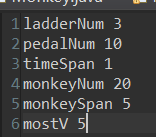


## 猴子过河模拟器v2

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。

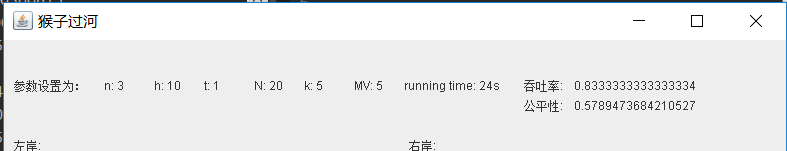
### 对比分析：固定其他参数，选择不同的决策策略

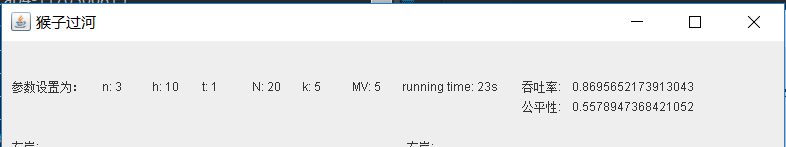
**固定参数如下：**



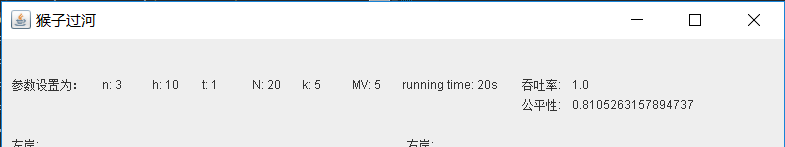
**所有的猴子选择策略一：**

实验结果：（测量4次）



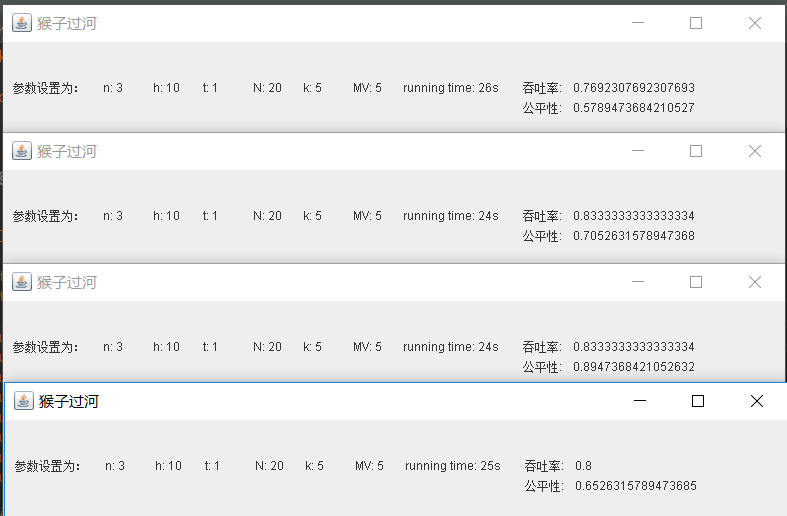






**所有的猴子选择策略二：**

实验结果：

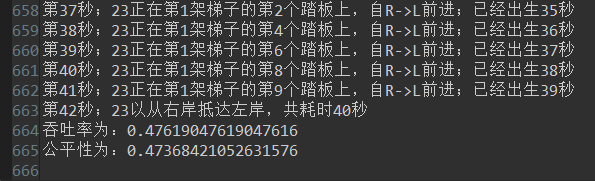


**所有的猴子选择策略三：**

实验结果：



具体数据在log文件里面有详细的记录:



**对比分析：**

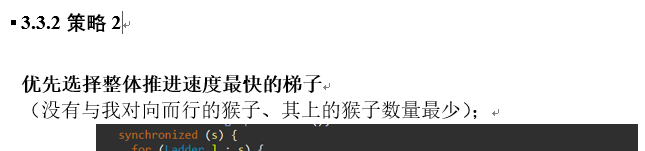
每次测试中每只猴子采用的策略相同。对比，很直观地可以看见，

策略一和策略二的吞吐率差不多，但是都比策略三强很多；

策略二公平性最高，策略一和策略三差不多；

最后是策略二最优，吞吐率和公平性都是最高的，策略三表现最差。

**事实证明：**

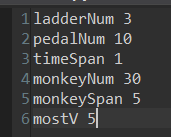


**这个策略最好。**

### 对比分析：变化某个参数，固定其他参数

**所有的测试都是基于策略二这个最优策略下进行测试的：**

**1. 参数如下：（变化梯子数量：LadderNum）**



**LadderNum = 2**



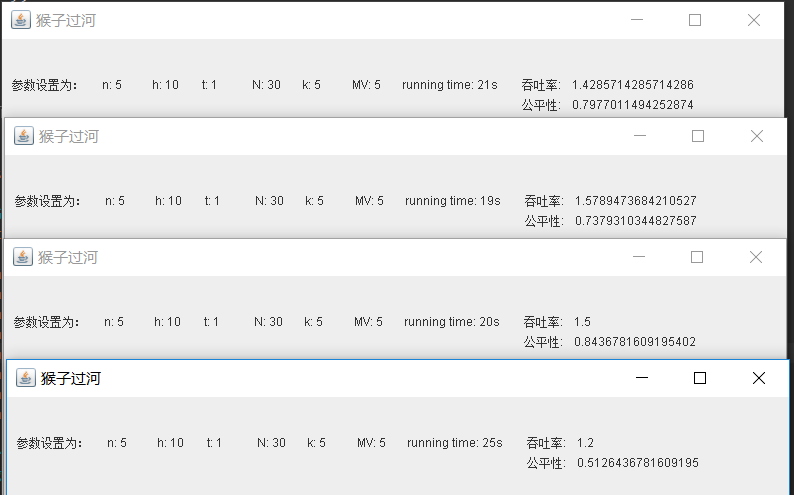
**LadderNum = 3**



**LadderNum = 4**



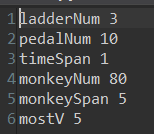
**LadderNum = 5**



**对比分析：**

**梯子数量LadderNum增加，显著增加的是吞吐率，当然公平性也稍微增加。**

**2. 参数如下：（变化每秒产生的猴子数量：monkeySpan）**



**monkeySpan = 3**



**monkeySpan = 5**



**monkeySpan = 10**



**对比分析：**

可以看到随着monkeySpan的增大，吞吐率逐渐提升，公平性逐渐下降。

### 分析：吞吐率是否与各参数/决策策略有相关性？

**由实验结果可以得到：**

吞吐率确实与一些参数和策略是有关的。

由3.9.1内容可以知道：**策略二最优**，吞吐率和公平性都是最高的，策略三表现最差。

由3.9.2内容可以知道：**梯子的数目LadderNum增加**，那么吞吐率也在上升，很明显，因为单位时间可以通过的猴子更多了。

**单次产生的猴子数目monkeySpan**：可以看到随着monkeySpan的增大，吞吐率逐渐提升，公平性逐渐下降。

**猴子的总数目N，**可以看到随着 N 的增大，吞吐率逐渐提升。

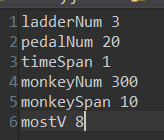
**产生猴子的时间间隔t**，可以看到随着t的增大，吞吐率逐渐下降。

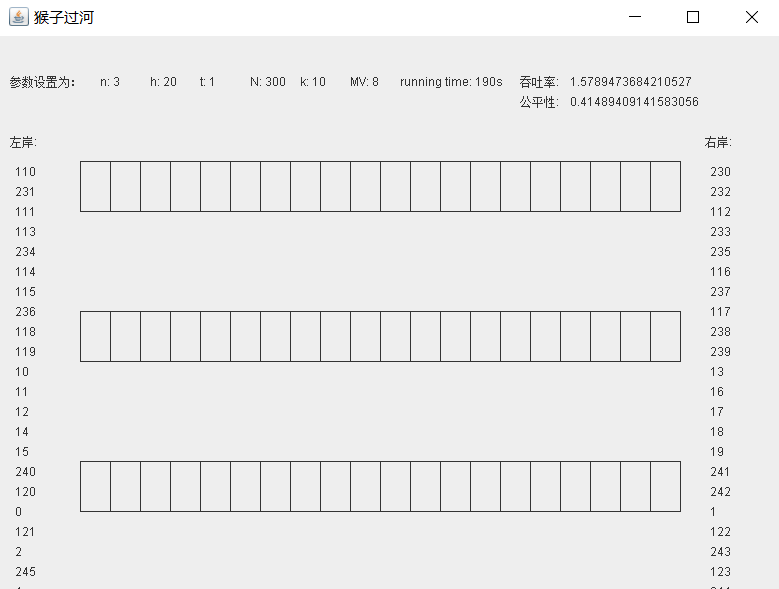
### 压力测试结果与分析

**压力测试 1：**

**全部为策略一**

参数配置：生成300个猴子



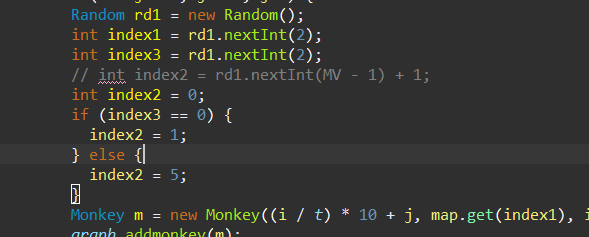


**吞吐率为：1.579**

**公平性为：0.415**

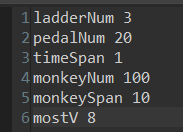
**表现还行。**

**压力测试 2：修改源码，只生成速度为1和5的猴子**



**全部为策略一**

参数配置：生成100个猴子





**吞吐率为：1.087**

**公平性为：0.419**

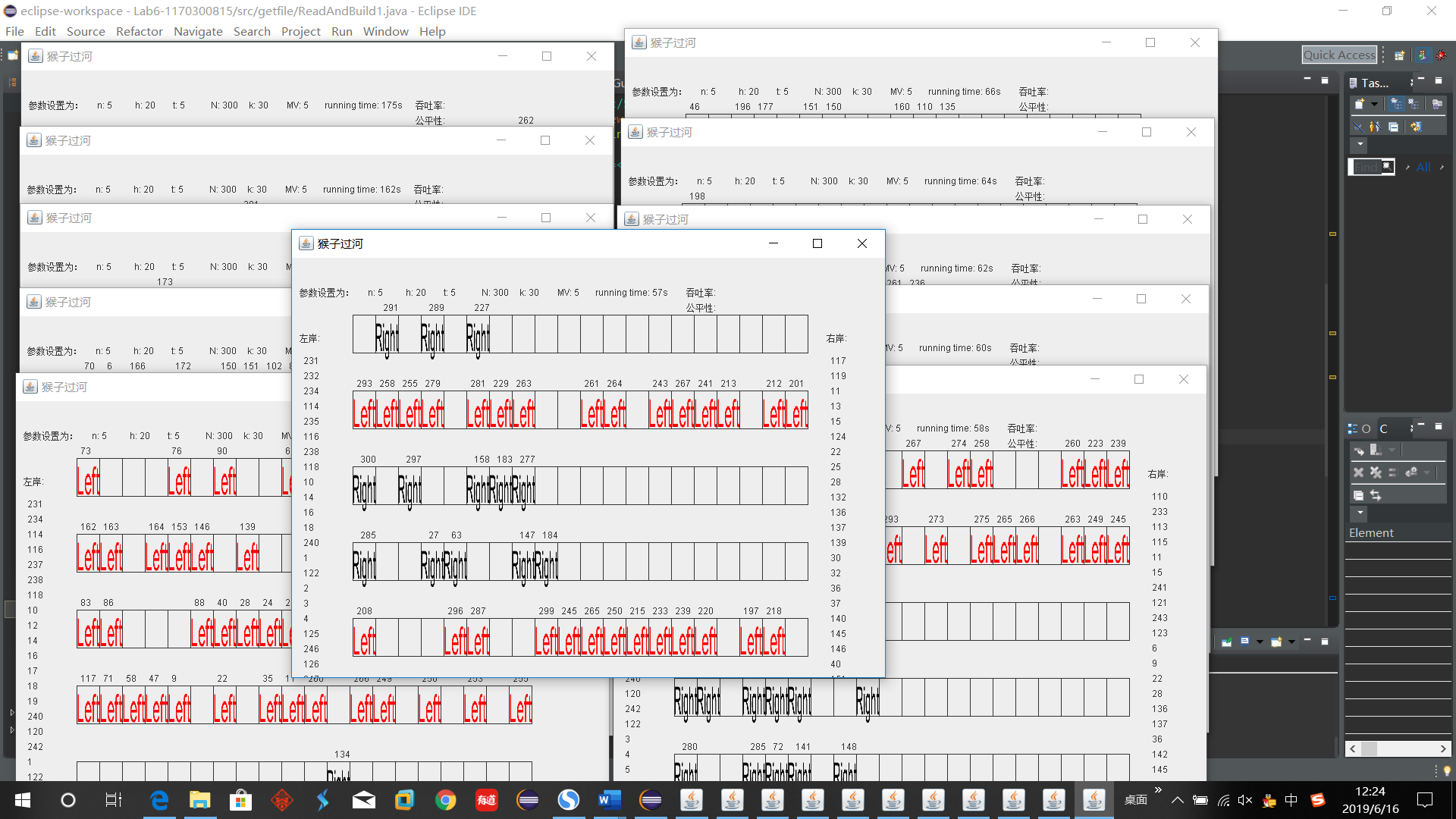
**表现不如随机产生速度的方案，很多速度快的猴子被速度慢的猴子拦截了下来，由于速度差异非常大，导致速度快的猴子严重停滞，最终表现为吞吐率显著下降。**

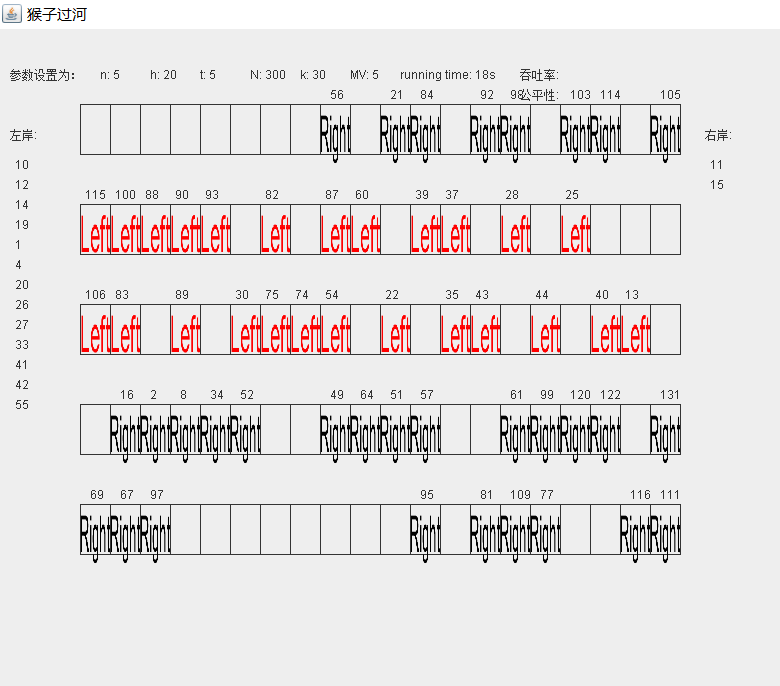
## 猴子过河模拟器v3

针对教师提供的三个文本文件，分别进行多次模拟，记录模拟结果。

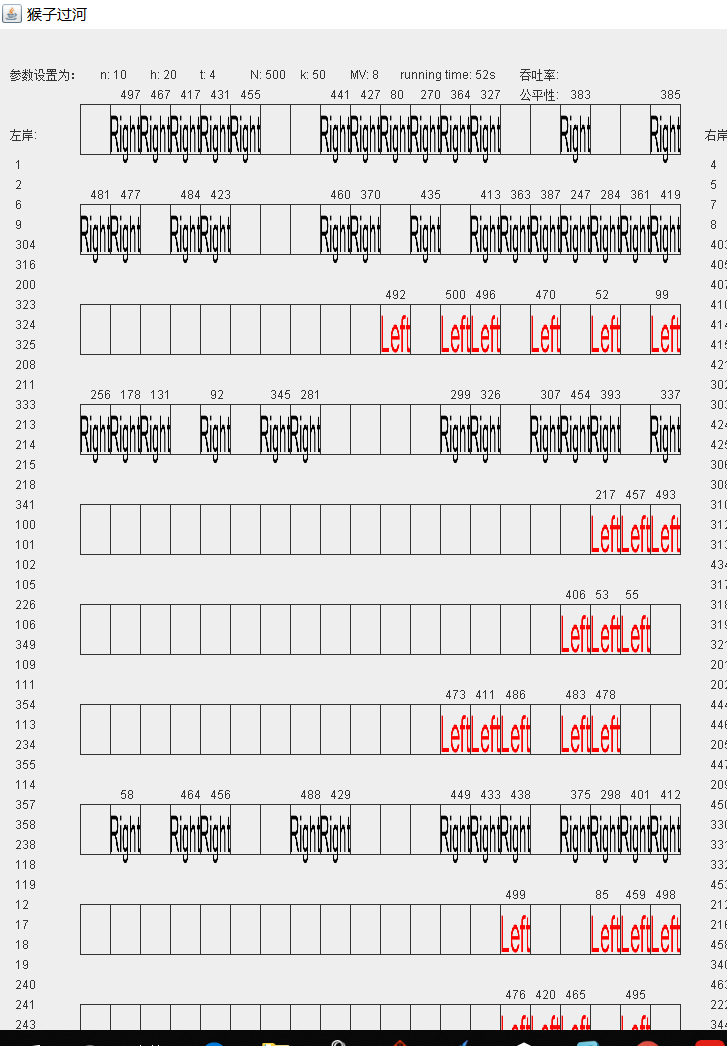
Competiton\_1.txt

**疯狂测试：**

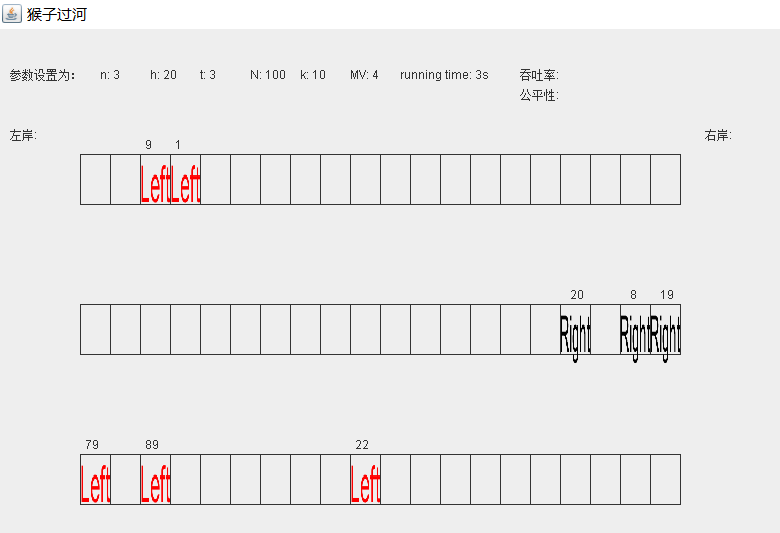




**Competiton\_2.txt**



**Competiton\_3.txt**



|  |  |  |
| --- | --- | --- |
|  | 吞吐率 | 公平性 |
| Competiton\_1.txt |  |  |
| 第1次模拟 | 0.92879 | 0.7863 |
| 第2次模拟 | 0.95238 | 0.6952 |
| 第3次模拟 | 1.16732 | 0.7051 |
| 第4次模拟 | 1.05634 | 0.7249 |
| 第5次模拟 | 1.09890 | 0.6745 |
| 第6次模拟 | 1.10294 | 0.6739 |
| 第7次模拟 | 1.04530 | 0.7148 |
| 第8次模拟 | 1.09890 | 0.6892 |
| 第9次模拟 | 1.07527 | 0.7371 |
| 第10次模拟 | 0.95541 | 0.7192 |
| 平均值 | 1.04816 | 0.7120 |
| Competiton\_2.txt |  |  |
| 第1次模拟 | 2.45098 | 0.4125 |
| 第2次模拟 | 2.41546 | 0.3956 |
| 第3次模拟 | 2.55102 | 0.4072 |
| 第4次模拟 | 2.34742 | 0.3568 |
| 第5次模拟 | 2.63158 | 0.3729 |
| 第6次模拟 | 2.56410 | 0.4237 |
| 第7次模拟 | 2.38095 | 0.3951 |
| 第8次模拟 | 2.39234 | 0.4215 |
| 第9次模拟 | 2.65957 | 0.4038 |
| 第10次模拟 | 2.55102 | 0.3967 |
| 平均值 | 2.49444 | 0.3986 |
| Competiton\_3.txt |  |  |
| 第1次模拟 | 0.45045 | 0.4596 |
| 第2次模拟 | 0.51020 | 0.5621 |
| 第3次模拟 | 0.49261 | 0.5347 |
| 第4次模拟 | 0.51813 | 0.5482 |
| 第5次模拟 | 0.51020 | 0.5173 |
| 第6次模拟 | 0.49261 | 0.4981 |
| 第7次模拟 | 0.50761 | 0.5217 |
| 第8次模拟 | 0.50251 | 0.5009 |
| 第9次模拟 | 0.50252 | 0.4625 |
| 第10次模拟 | 0.48309 | 0.5014 |
| 平均值 | 0.49699 | 0.5107 |

# 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

每次结束编程时，请向该表格中增加一行。不要事后胡乱填写。

不要嫌烦，该表格可帮助你汇总你在每个任务上付出的时间和精力，发现自己不擅长的任务，后续有意识的弥补。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 6.3 | 15:30-17:30 | 了解实验要求 | 完成 |
| 6.7 | All Day | 写V1 | 未完成 |
| 6.8 | All Day | 写V1 | 未完成 |
| 6.9 | All Day | 写V1 | 完成 |
| 6.14 | All Day | 写V2 | 完成 |
| 6.15 | All Day | 写V3，报告 | 完成 |
| 6.16 | 10:00-13:00 | 撰写实验报告 | 完成 |

# 实验过程中遇到的困难与解决途径

|  |  |
| --- | --- |
| 遇到的难点 | 解决途径 |
| 想把字符串转化为图片 | 百度 |
| 线程的安全性 | 看PPT和查看CSDN等平台的博客 |
| Log的配置 | 百度 |

# 实验过程中收获的经验、教训、感想

## 实验过程中收获的经验和教训

单线程程序的调试是容易出现复现错误的，而对于多线程来说，很多错误不会复现。甚至在其中增加一条涉及IO操作的语句就能够彻底改变执行的顺序。

多线程性能方面的改善主要还是来自于并行，这种性能的提升主要的来自于多个任务并行执行时的优势。

## 针对以下方面的感受

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. 多线程程序需要比单线程程序多考虑竞争条件的发生，需要保证数据线程安全。
3. 多线程程序在性能方面要比单线程好，并行比串行效率更高。
4. 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？

1.A.使用线程安全的数据结构和 synchronized 加锁来实现。

B.优先保证线程安全，然后再考虑性能问题。

2.尽量避免了不必要的同步，使同步块在保证安全的底线下尽可能小。

1. 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？
2. 在多个猴子同时访问 ladder 的时候，要求每节梯子同一时间只能有一只猴子；
3. 刚开始一只猴子上了梯子之后还没来得及设置状态，就是占用的信息没有及时的写回，就因为线程调度被挂起；
4. 下一只猴子开始运行，同样的上了同一架梯子，导致冲突。
5. 考虑原因是在数据结构选择上使用了线程安全的，但是没有考虑多个原子操作的组合情况。
6. 关于本实验的工作量、难度、deadline。

工作量不算很大，难度适中，deadline还行。

1. 到此为止你对《软件构造》课程的意见和建议。

课程设置很有挑战性，内容丰富，实验量不小，但是经历了这一学期的锻炼，我的编程能力有了不小的提升。

我从对java一无所知到能够用java解决一些基本的问题，初步了解并应用设计模式，学习了面向对象的一些基本概念和软件构造过程的各个部分，收获非常大。

最后感谢徐老师和助教们一学期来的工作和努力！