

**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 | 张景润 |
| 学号 | 1172510217 |
| 班号 | 1703002 |
| 电子邮件 | 2584363094@qq.com |
| 手机号码 | 18530272728 |

**目录**

[1 实验目标概述 1](#_Toc11610860)

[2 实验环境配置 1](#_Toc11610861)

[3 实验过程 1](#_Toc11610862)

[3.1 ADT设计方案 1](#_Toc11610863)

[3.1.1 Monkey implements Runnable 1](#_Toc11610864)

[3.1.2 Ladder 2](#_Toc11610865)

[3.2 其他类的简单介绍 3](#_Toc11610866)

[3.3 Monkey线程的run()的执行流程图 3](#_Toc11610867)

[3.4 至少两种“梯子选择”策略的设计与实现方案 3](#_Toc11610868)

[3.4.1 策略1：NoMonkeyFirst 3](#_Toc11610869)

[3.4.2 策略2：NoOppositeFirst 4](#_Toc11610870)

[3.4.3 策略3（可选）：MyStrategy 5](#_Toc11610871)

[3.5 “猴子生成器”MonkeyGenerator 6](#_Toc11610872)

[3.6 如何确保threadsafe？ 7](#_Toc11610873)

[3.7 系统吞吐率和公平性的度量方案 7](#_Toc11610874)

[3.8 输出方案设计（实现了动态化模拟过程） 8](#_Toc11610875)

[3.8.1 日志-java自带的log 8](#_Toc11610876)

[3.8.2 GUI界面展示 9](#_Toc11610877)

[3.8.3 动态显示整个过程 10](#_Toc11610878)

[3.9 猴子过河模拟器v1 11](#_Toc11610879)

[3.9.1 参数如何初始化 11](#_Toc11610880)

[3.9.2 使用Strategy模式为每只猴子随机选择决策策略 11](#_Toc11610881)

[3.10 猴子过河模拟器v2 12](#_Toc11610882)

[3.10.1 对比分析：固其他参数，不同的决策策略和某个参数之间的关系对比 12](#_Toc11610883)

[3.10.1.1 对参数梯子数目进行变化(截图分别为n=1,3,5,10) 12](#_Toc11610884)

[3.10.1.2 对参数最大速度进行变化(截图分别为MV=5,6,8,10) 14](#_Toc11610885)

[3.10.1.3 对参数k进行变化(截图分别为k=1,5,10,50) 15](#_Toc11610886)

[3.10.1.4 对参数N进行变化(截图分别为N=10,100,200,1000) 17](#_Toc11610887)

[3.10.1.5 对参数t进行变化(截图分别为t=1,2,3,5) 18](#_Toc11610888)

[3.10.2 分析：吞吐率是否与各参数/决策策略有相关性？ 20](#_Toc11610889)

[3.10.3 压力测试结果与分析 20](#_Toc11610890)

[3.10.3.1 竞争情况极其严重：猴子多+产生快+梯子少 20](#_Toc11610891)

[3.10.3.2 猴子速度差异很大 21](#_Toc11610892)

[3.11 猴子过河模拟器v3 22](#_Toc11610893)

[3.11.1 实现读取文件建立系统 22](#_Toc11610894)

[3.11.2 测试第一个文件（吞吐率极限为3）（MyStrategy策略） 22](#_Toc11610895)

[3.11.3 测试第二个文件（吞吐率极限为10）（MyStrategy策略） 23](#_Toc11610896)

[3.11.4 测试第三个文件（吞吐率极限为3）（MyStrategy策略） 24](#_Toc11610897)

[4 实验进度记录 24](#_Toc11610898)

[5 实验过程中遇到的困难与解决途径 25](#_Toc11610899)

[6 实验过程中收获的经验、教训、感想 25](#_Toc11610900)

# 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

* 建立本地仓库并于远程仓库关联

1，README文件：echo “Lab6-1172510217” >> README.md

2，初始化本地仓库：git init

3，添加到暂存区：git add README.md

4，提交到本地仓库git commit -m “first commit”

5，建立与远程仓库的联系：git remote add origin <https://github.com/ComputerScienceHIT/Lab6-1172510217.git>

6，推送到远程仓库：git push -u origin master

* Lab6的URL地址（我的学号是：1172510217）

<https://github.com/ComputerScienceHIT/Lab6-1172510217>

# 实验过程

## ADT设计方案

### Monkey implements Runnable

* 他的spec如截图



* 属性

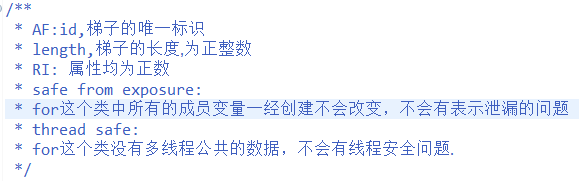
|  |  |
| --- | --- |
| StringBuilder *logBuilder* | 便于日志构造 |
| Integer id | 猴子唯一标识 |
| Integer direction | 猴子的方向 |
| Integer speed | 猴子的速度 |
| Integer bornTime | 猴子的出生时间 |
| Integer liveTime | 猴子的存活时间 |
| LadderStrategy ladderStrategy | 选择的过河策略 |

* 方法

|  |  |
| --- | --- |
| toString() | 重写toString |
| run() | 线程的run方法 |
| getter() | 所有的属性获取方法 |
| directionString() | 将0或1的方向表示为L->R和R->L |
| Monkey() | 构造器 |
| getLogString() | 获取猴子的经历信息 |

### Ladder

* 他的spec如截图



* 属性

|  |  |
| --- | --- |
| Integer length | 梯子踏板数 |
| Integer id | 梯子唯一标识 |

* 方法

|  |  |
| --- | --- |
| toString() | 重写toString |
| ladder() | 梯子构造器 |
| getter() | 所有的属性获取方法 |

## 其他类的简单介绍

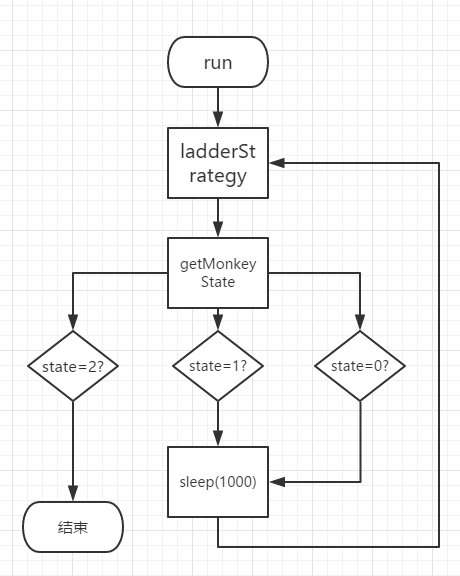
* SystemConstant.java

为简单化参数构造的一个类，其中设置了两个参数：系统加速常数Integer *speedUp*以及设计的策略的数目Integer *strategyNum*

我可以通过简单的修改这个类中的参数，从而实现加速猴子过河过程的模拟。

* LadderFactory.java：用于管理所有的梯子以及被占用的情况
* MonkeyFactory.java：用来管理所有的猴子以及其过桥状态
* MonkeyCrossRiverSystem.java：用来读取配置文件建立系统
* Gui.java：用于实现GUI
* 四个策略类：用于实现具体的猴子过河策略

## Monkey线程的run()的执行流程图



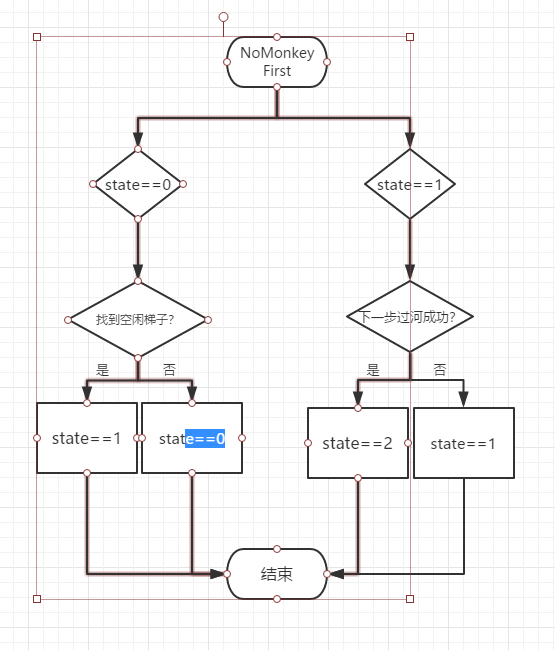
## 至少两种“梯子选择”策略的设计与实现方案

说明1：我实现了三种策略：两种pdf上给出的策略+一种自己设计的策略

说明2：这三种策略均实现接口LadderStrategy，采用了strategy设计模式

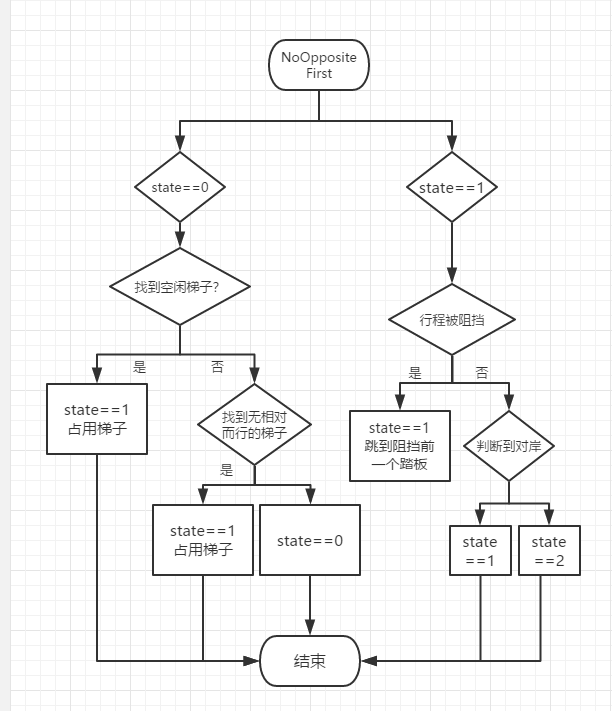
### 策略1：NoMonkeyFirst

猴子只选择没有被占用的梯子，若均被占用，则只是简单的原地等待。



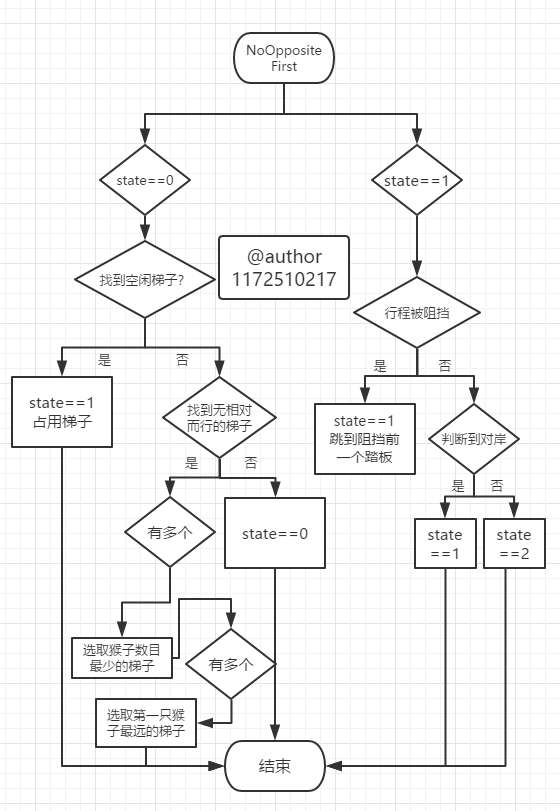
### 策略2：NoOppositeFirst

猴子优先选择未被占用的梯子，若均被占用，则选取无相对而行猴子的梯子，若这样的选择有很多，进行随机选取。



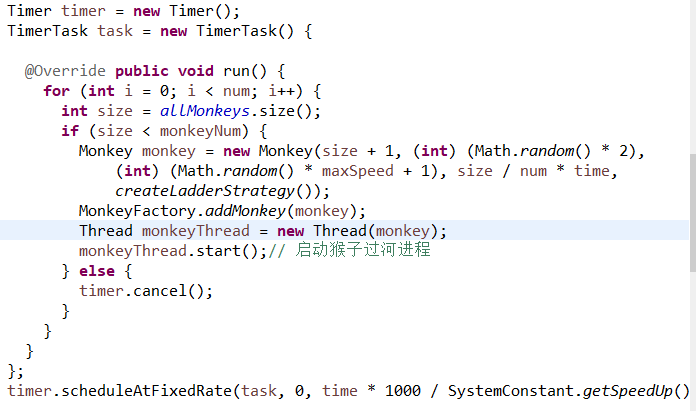
### 策略3（可选）：MyStrategy

猴子优先选择未被占用的梯子，若均被占用，则选取无相对而行猴子的梯子，若这样的选择有很多，则优先选择梯子上猴子数目最少的一个梯子；若数目仍然一致，则选取梯子上的第一只猴子距离自己最远的梯子。



## “猴子生成器”MonkeyGenerator

* 该方法在读取配置文件并创建系统的时候被调用。目的是隔一段时间产生一定数目的猴子，并启动他们的线程。
* 该方法内部有一个Timer对象和一个TimeTask对象。目的是可以做到隔一段时间执行一次run方法。
* 具体的run方法内部代码如下。不断生成猴子，当生成的猴子数目等于要生成的猴子数目时，调用Timer对象的cancel方法结束此线程。生成猴子的同时将猴子线程启动。

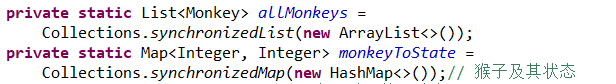


* 同时注意到该Timer对象调用了scheduleAtFixedRate方法，且参数中右SystemConstant.getSpeedUp方法。目的是可以做到加速此模拟猴子过河系统的进行。

## 如何确保threadsafe？

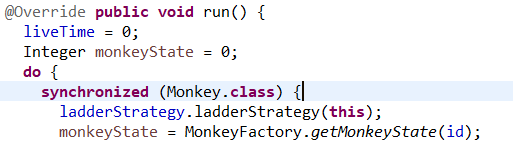
* 使用了线程安全的Map和List保证线程安全。

如截图，我的集合结构的声明确保是线程安全的。



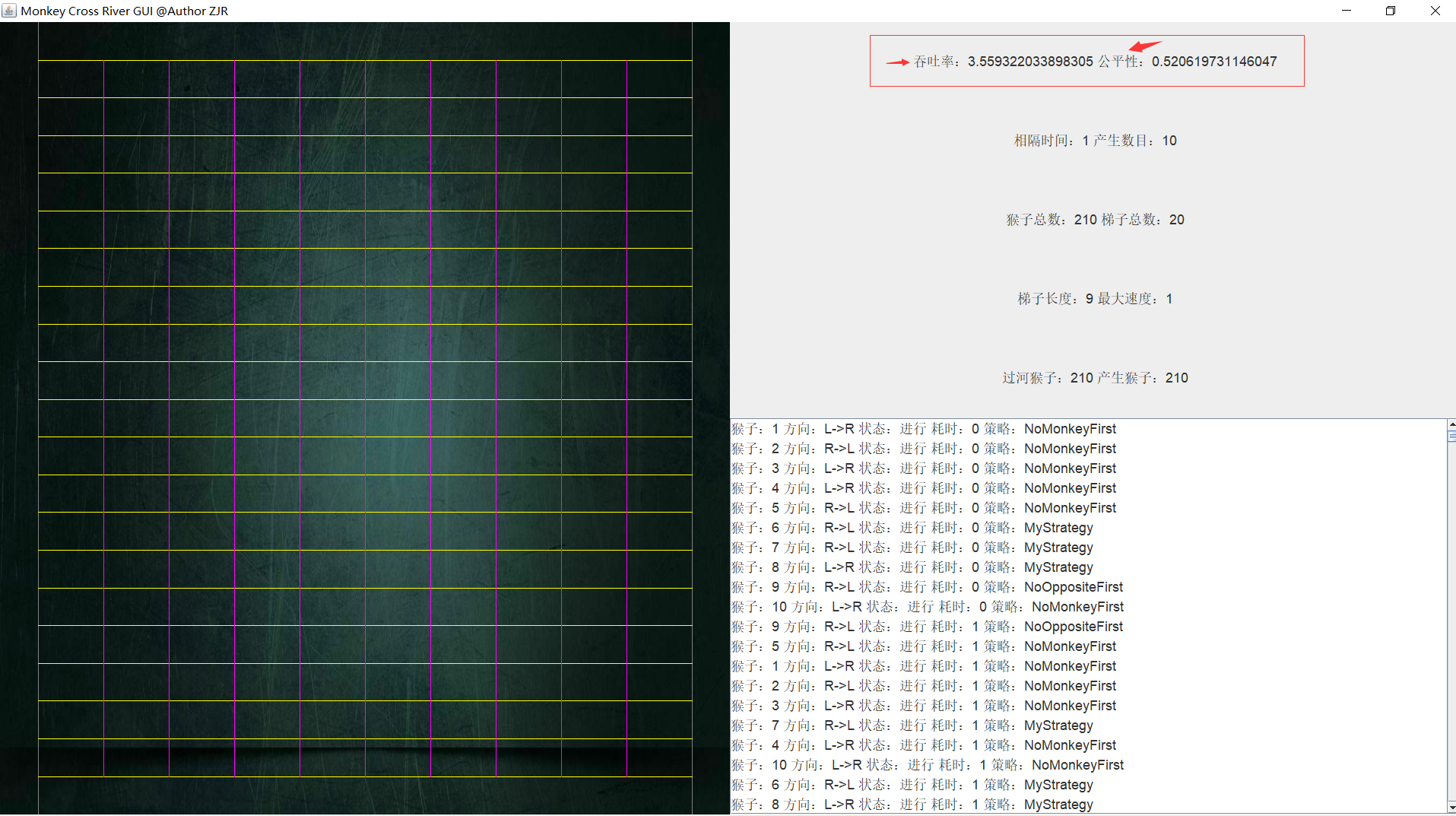
* 使用线程锁Synchronized关键字。

确保每一只猴子在进行自己的策略选择梯子时独占线程且不被中断。



## 系统吞吐率和公平性的度量方案

* 注：吞吐率和公平性的度量只在所有猴子均已过河时，才会启动。
* 这些度量可以在GUI上显示，如截图。



* 遍历所有的猴子，得到最后一只过河猴子的存活时间和出生时间即可确定所有猴子的过河的最长时间，即可确定吞吐率
* 两层for循环遍历所有的猴子，判断两只猴子是否过河公平，若公平，则公平数++，否则公平数--。最终即可确定公平性度量常量。

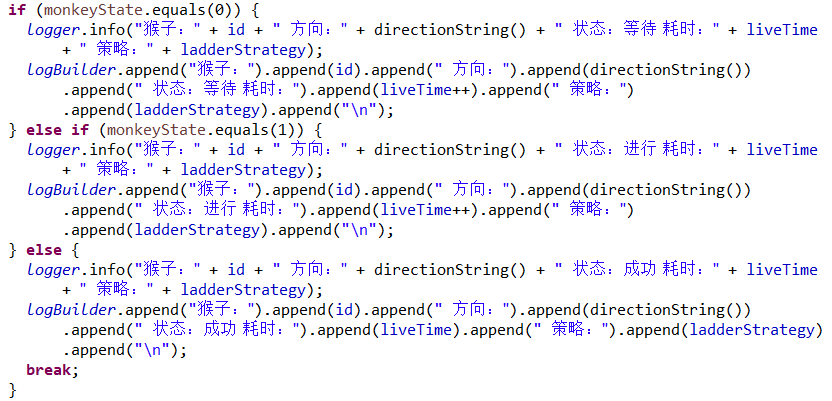
## 输出方案设计（实现了动态化模拟过程）

### 日志-java自带的log

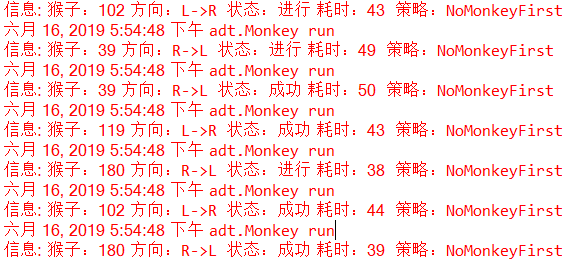
* 在Monkey类中设置了一个静态属性Logger。确保所有的猴子在进行自己的线程时，均会在Logger中留下记录。



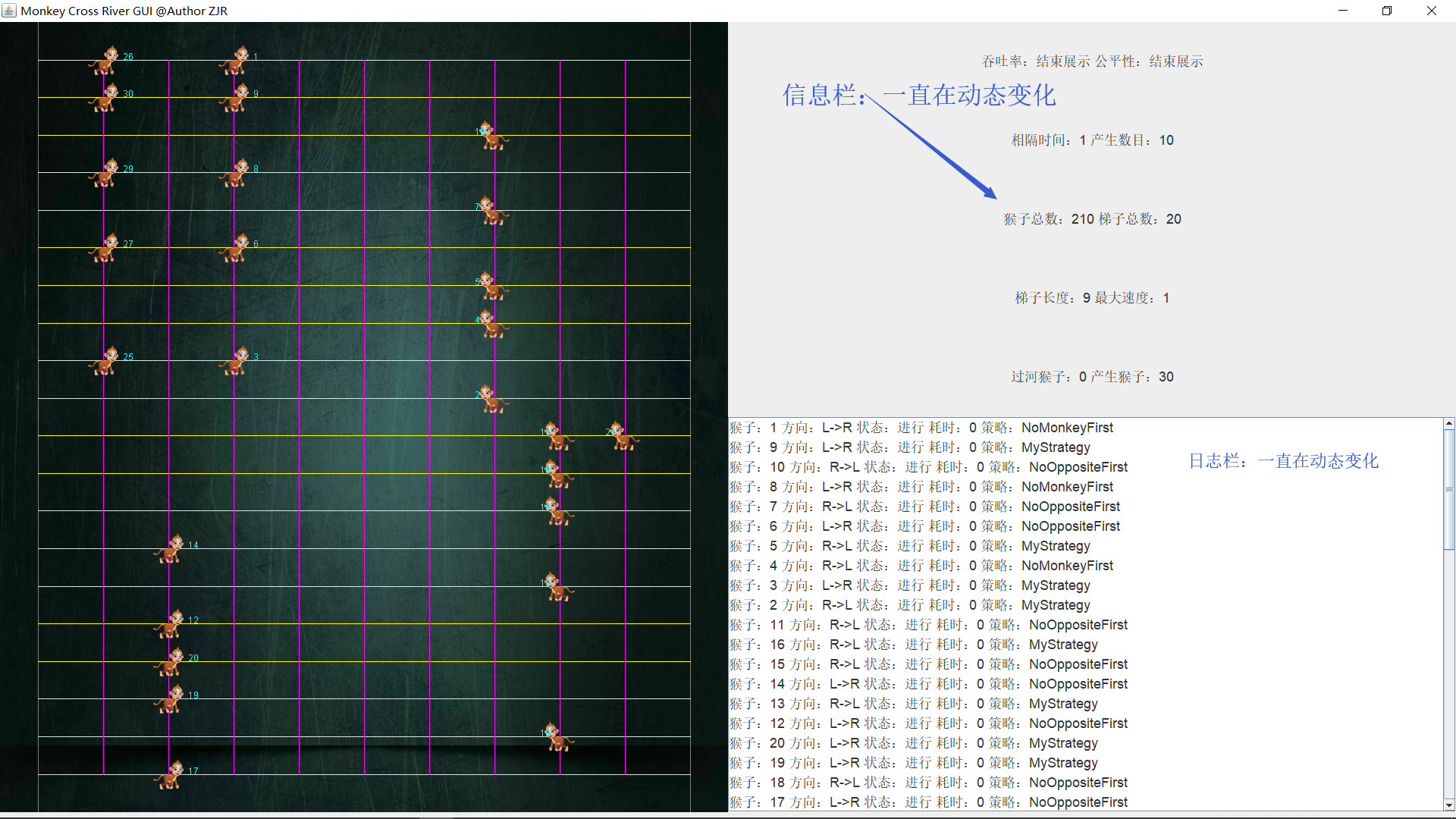
* 在Monkey的run方法中执行对Logger的记录。分三种情况：等待，进行和成功，分别进行单独记录。



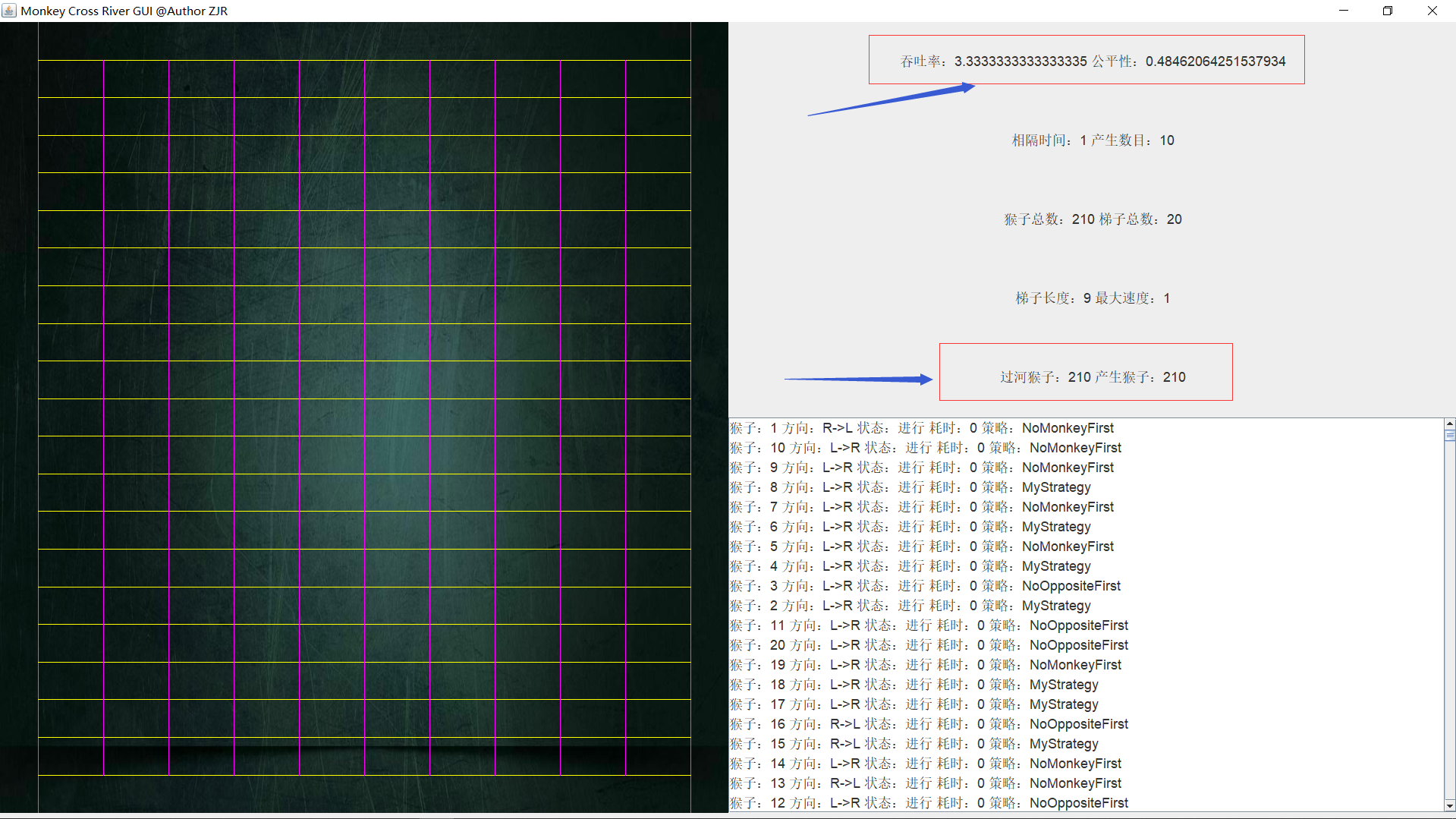
* 终端显示如截图，做到了信息清晰易读。



### GUI界面展示



* GUI展示划分：动态化猴子过河区；右上为配置文件信息展示区以及过河猴子数目、产生猴子数目；右下为日志栏
* 这三部分均是动态变化的：日志不断刷新、猴子数目不断刷新、猴子过河动画不断刷新。
* 过河结束GUI展示



### 动态显示整个过程

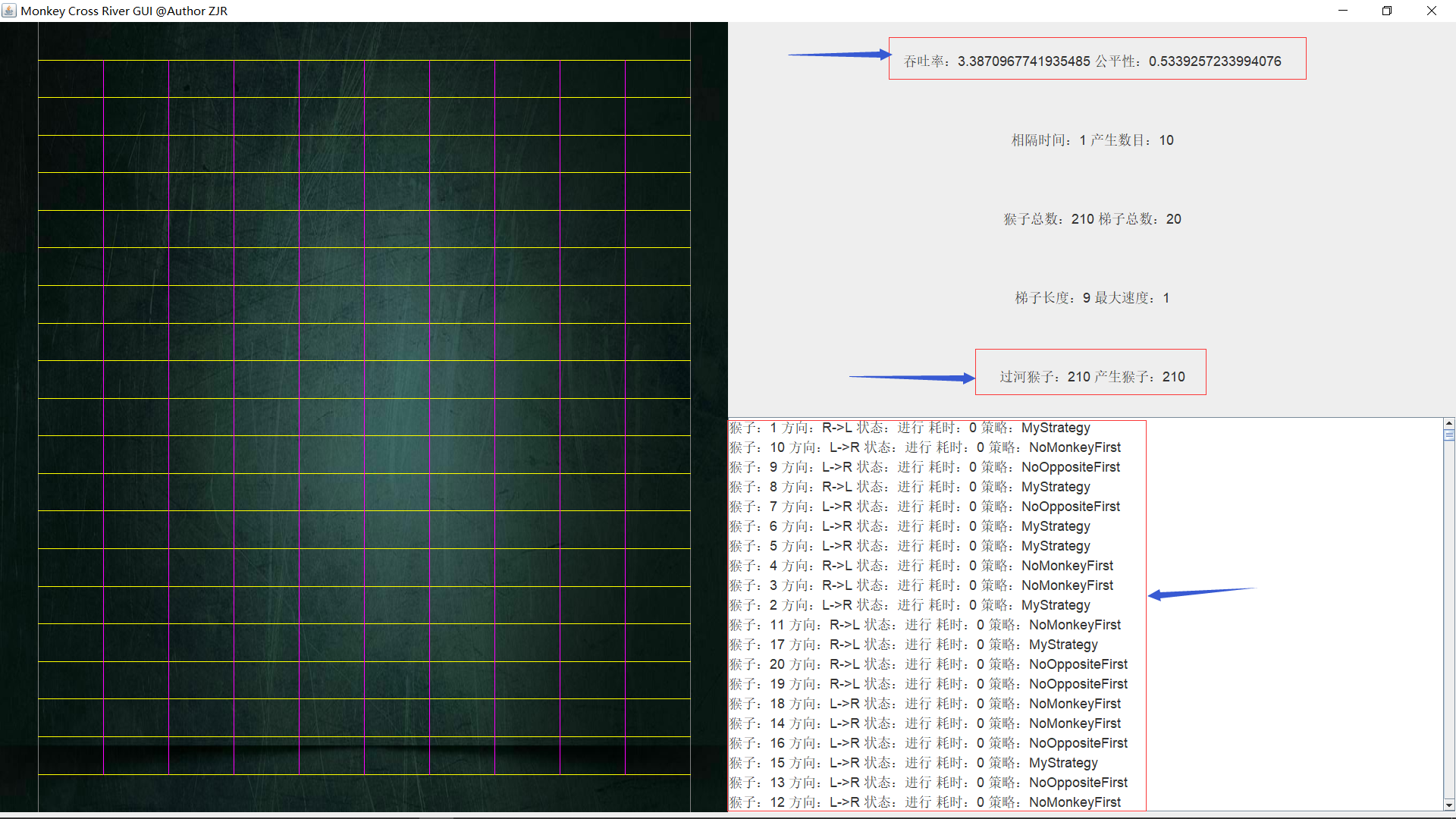
一点说明

为了更好的观看理解整个过程，我特意选取了一张猴子的图片，同时将其进行对称镜像处理，从而可以很好的看到猴子运行的方向；

同时我还运用PS技术对选取的图片进行了抠图处理，目的是扣去图片上的白色背景板。如截图。

分隔符

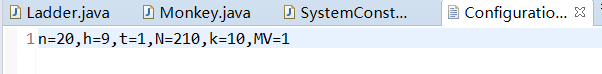
* 整个过程是动态显示的：猴子可移动、右上信息栏不断刷新、右下日志栏不断刷新、吞吐率以及公平性会自动在过河结束自动显示。
* 在截图上用箭头标识出所有的可以动态化的组件



## 猴子过河模拟器v1

### 参数如何初始化

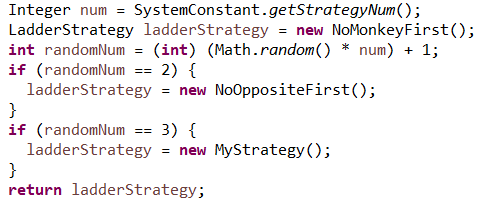
* 在src文件夹下新建了一个配置文件，只需要修改配置文件，即可完成对参数的初始化。文件目录src/ConfigurationFile/Configuration1.txt



* 通过调用MonkeyCrossRiverSystem.readFileAndCreateSystem的方法。实现参数初始化，并建立系统。

### 使用Strategy模式为每只猴子随机选择决策策略

* 在MonkeyFactory类内部调用createLadderStrategy方法实现为猴子随机分配一种策略。注意需要在SystemConstant类中手动修改一个参数strategyNum，从而确保随机数可以产生数字1，2，3，从而进行随机策略分配。
* 分析createLadderStrategy方法。如截图



## 猴子过河模拟器v2

五点说明：

1，实现所有猴子选择同一种策略的方法是：

2，实现更改猴子的初始化参数的方法是：

3，由于原来实验中的3.9.1和3.9.2可以进行通过图表进行合并处理，因此我将其改为现在的3.9.1加以处理分析；

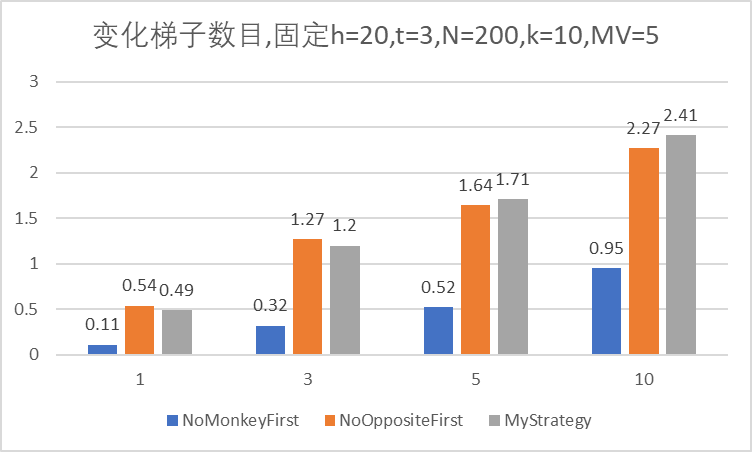
4，为了统计以及作图的方便，做了两个操作：吞吐率和公平性数据统一取两位有效数字，设置了过河进行速率比例常数为100表示加速这个过程100倍而不失正确性；

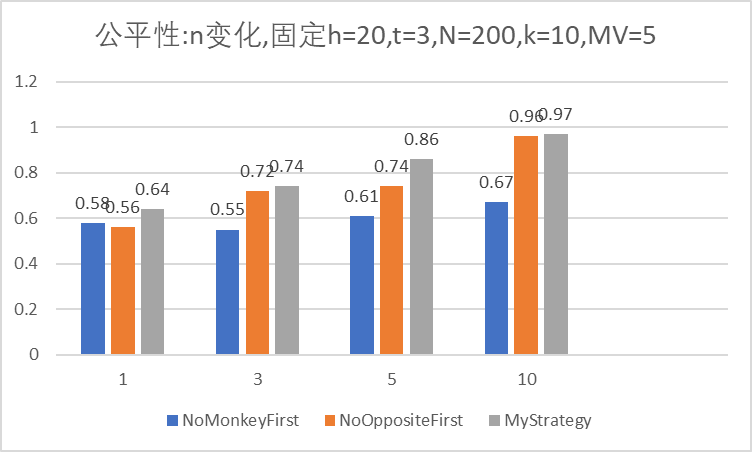
5，截图为GUI的原始数据，作图使用的是取两位小数的真实数据。

### 对比分析：固其他参数，不同的决策策略和某个参数之间的关系对比

#### 对参数梯子数目进行变化(截图分别为n=1,3,5,10)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | h | t | N | k | MV |
| 1-10 | 20 | 3 | 200 | 10 | 5 |





1，对于NoMonkeyFirst策略









2，对于NoOppositeFirst策略









3，对于MyStrategy策略



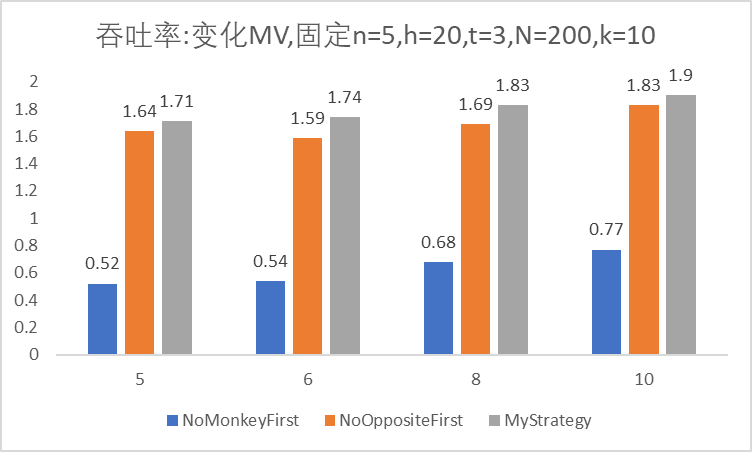


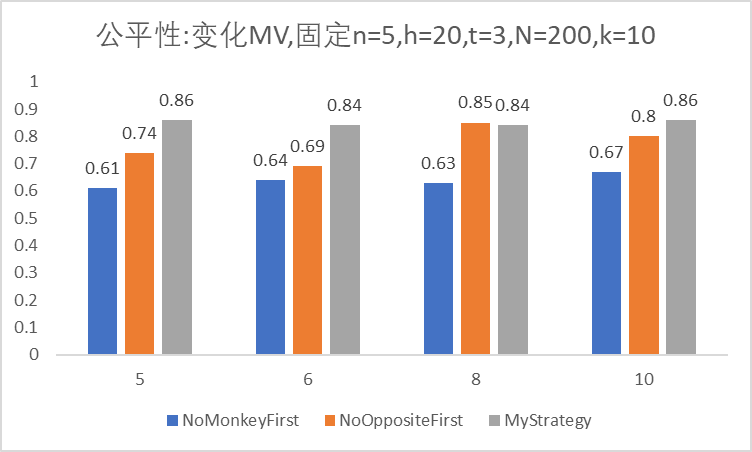




#### 对参数最大速度进行变化(截图分别为MV=5,6,8,10)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | h | t | N | k | MV |
| 5 | 20 | 3 | 200 | 10 | 5-10 |





1，对于NoMonkeyFirst策略









2，对于NoOppositeFirst策略









3，对于MyStrategy策略



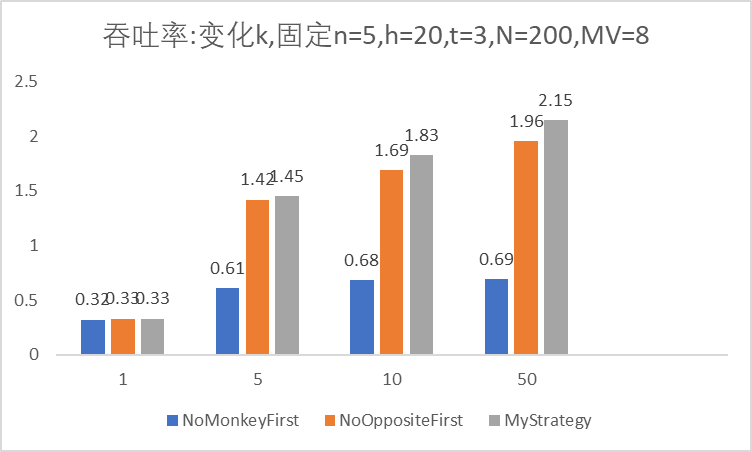


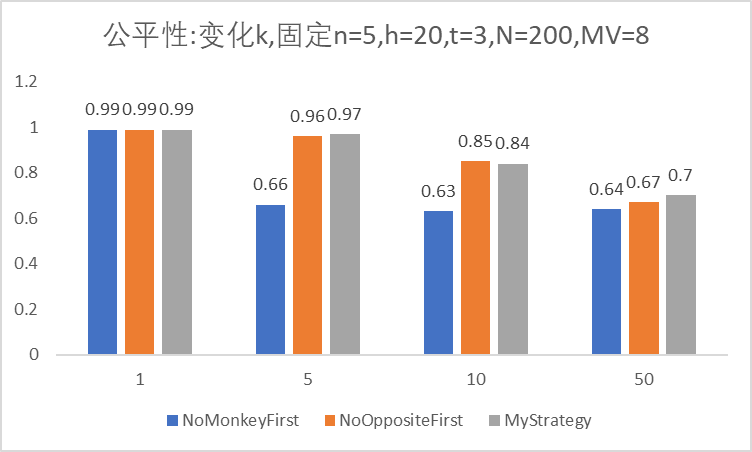




#### 对参数k进行变化(截图分别为k=1,5,10,50)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | h | t | N | k | MV |
| 5 | 20 | 3 | 200 | 1-50 | 8 |





1，对于NoMonkeyFirst策略









2，对于NoOppositeFirst策略









3，对于MyStrategy策略



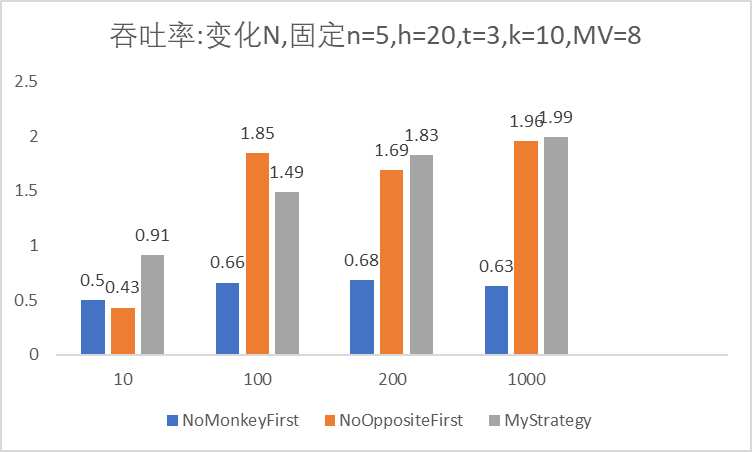


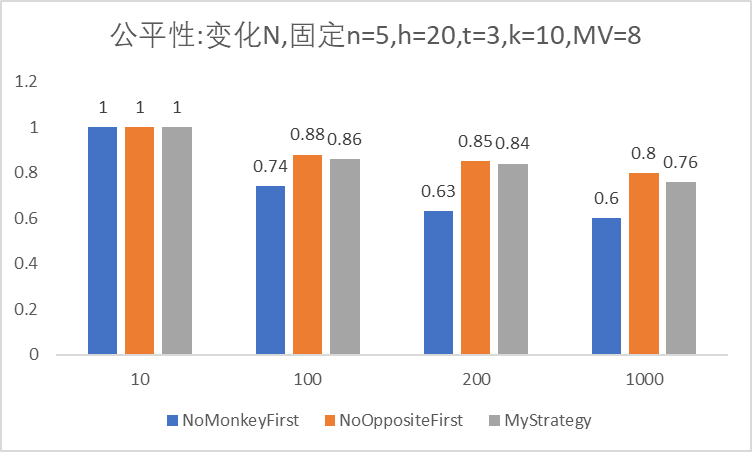




#### 对参数N进行变化(截图分别为N=10,100,200,1000)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | h | t | N | k | MV |
| 5 | 20 | 3 | 2-1000 | 10 | 8 |





1，对于NoMonkeyFirst策略









2，对于NoOppositeFirst策略









3，对于MyStrategy策略



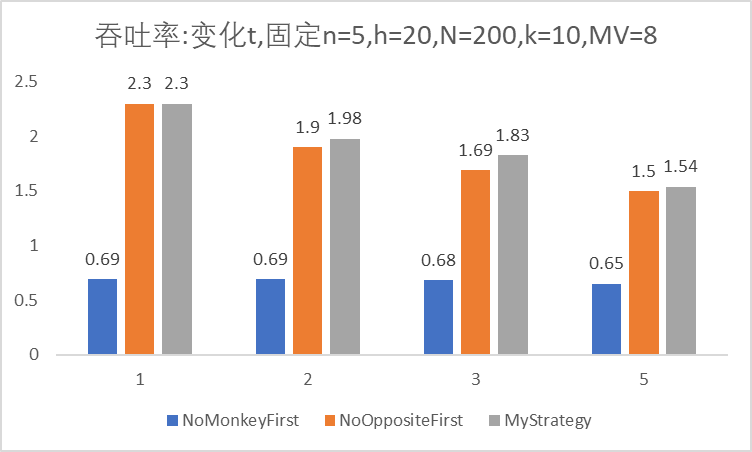


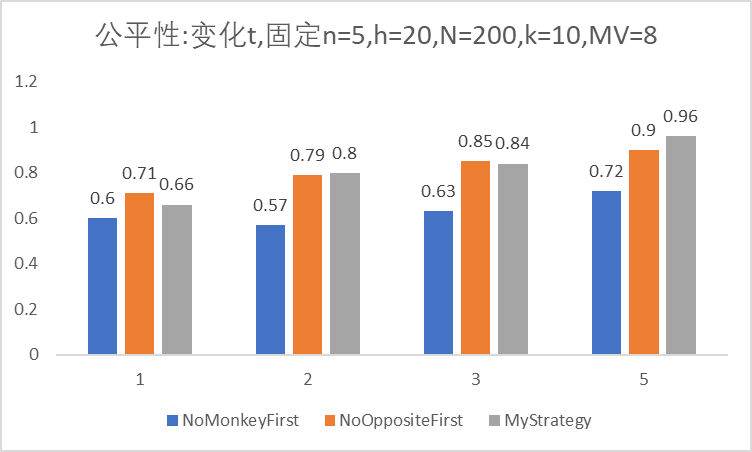




#### 对参数t进行变化(截图分别为t=1,2,3,5)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | h | t | N | k | MV |
| 5 | 20 | 1-5 | 200 | 10 | 8 |





1，对于NoMonkeyFirst策略









2，对于NoOppositeFirst策略









3，对于MyStrategy策略









### 分析：吞吐率是否与各参数/决策策略有相关性？

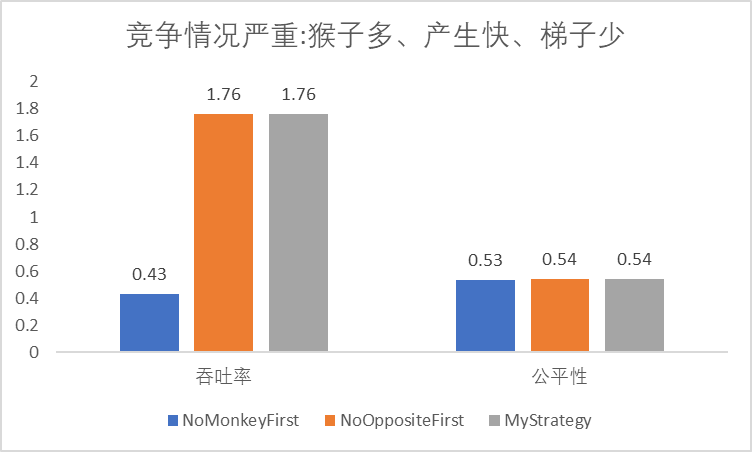
总结：与参数配置以及决策策略关系很大，下面对上一个部分的5组实验进行细致的分析。

* 第一组：梯子数目越多，显然吞吐率越高以及公平性越高；同时发现MyStrategy策略是最优的。
* 第二组：最大速度越大，说明平均一只猴子过河所需时间越短，那么岸上猴子需要等待时间越短，公平性越高，且MyStrategy是最优的设计。
* 第三组：每次产生的猴子数目越多，整体吞吐率和公平性均得到了提高。说明整体的性能未得到完全的挖掘，未达到阙值。但是公平性却逐渐下降，说明等待的猴子越来越多，随机性也越来越大。
* 第四组：猴子总数越多，整体的公平性越低，但是吞吐率在猴子数目较多时，变化不太明显。因为猴子总数使整个参数配置达到了阈值，在进行改变时，变化不会太大。
* 第五组：猴子产生间隔越短，整体的吞吐率和公平性越低，因为有大量的猴子进行等待，而且梯子数目不是特别多，只要第一波猴子的方向均为从左到右或者从右到左，那么就会有的大量的不同方向的猴子进行等待。而我们的策略选择是选择无对向行驶的猴子的梯子，这样会使所有的梯子被同一个方向的猴子占用，那么恶性循环，反方向的猴子不会得到梯子的使用权，直到这些一个方向的猴子全部过河。因此整体的公平性逐渐降低。而且说明系统的整体的吞吐率与第一波产生的猴子的方向的均匀性有极大的关系。

### 压力测试结果与分析

#### 竞争情况极其严重：猴子多+产生快+梯子少

* 参数配置为：n=3,h=20,t=1,N=1000,k=50,MV=8
* 分析：MyStrategy和NoOppositeStrategy策略模式的整体差别不大，因为梯子数目少，导致两种策略的梯子选择差别很小很小，但是公平性测度和吞吐率测度均好于NoMonkeyStrategy策略



1，对于NoMonkeyFirst策略



2，对于NoOppositeFirst策略

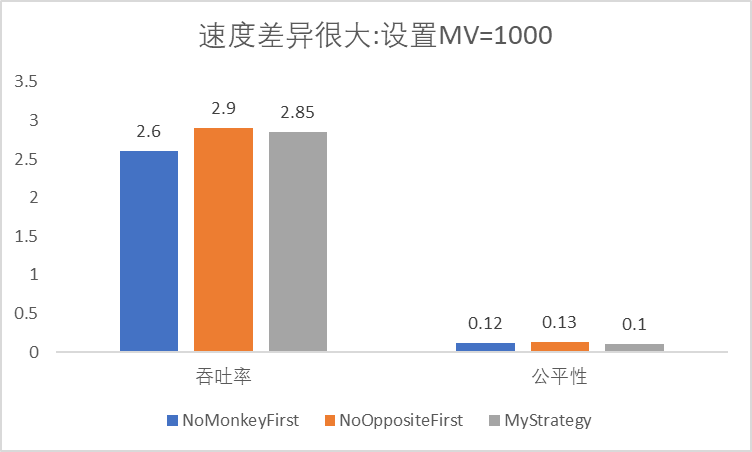


3，对于MyStrategy策略



#### 猴子速度差异很大

* 参数配置为：n=3,h=20,t=1,N=1000,k=50,MV=1000
* 分析：速度很大导致不公平现象越来越多，但同时由于整体推进速度很快，使总体吞吐率接近极限3（永远不可能超过3）。



1，对于NoMonkeyFirst策略



2，对于NoOppositeFirst策略



3，对于MyStrategy策略



## 猴子过河模拟器v3

### 实现读取文件建立系统

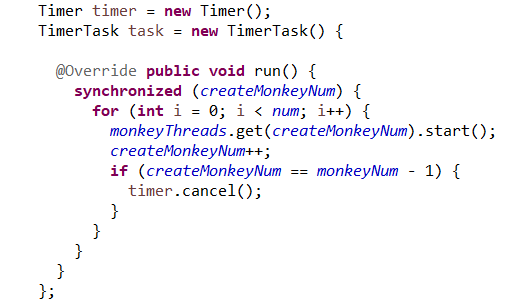
* 均采用我自己设计的策略MyStrategy
* 将原本的读取配置文件的输入文件改为下载的文件路径



* 从v2->v3思路改变

原来思路是猴子创建的同时，便调用Thread.start方法；现在改为先产生所有猴子及其线程，再根据文件参数分析得到相关的时间间隔，一次产生的猴子数目参数，最终对这些线程进行统一的管理，如截图。对线程进行管理的集合结构是线程安全的。





### 测试第一个文件（吞吐率极限为3）（MyStrategy策略）

平均吞吐率：2.308405  
平均公平性：0.340053

十次测试结果如下

















### 测试第二个文件（吞吐率极限为10）（MyStrategy策略）

平均吞吐率：4.897559  
平均公平性：0.639368

十次测试结果如下





















### 测试第三个文件（吞吐率极限为3）（MyStrategy策略）

平均吞吐率：1.124451  
平均公平性：0.4146

十次测试结果如下





















# 实验进度记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
| 06-04 | 晚上 | 读懂题意 | 部分完成 |
| 06-06 | 下午实验课 | 彻底读懂题意 | 绝大部分完成 |
| 06-10 | 下午+晚上 | adt设计完毕 | ok |
| 06-13 | 下午+晚上 | 策略完成 | 完成两个 |
| 06-14 | 上午+晚上 | 完成第三个策略 | ok+完成部分gui |
| 06-15 | 全天 | 完成gui | 完成gui+v2 |
| 06-16 | 全天 | 完成v3+实验报告 | 完成 |

# 实验过程中遇到的困难与解决途径

* 多线程编程基础差，运用能力差：不断尝试+网上搜索问题的产生与解决
* 多线程debug极其困难：对自己的代码整体进行细致线程安全检查
* adt设计的时候比较犹豫：纠结于如何进行更好的数据结构管理，最终设计factory类进行统一的管理

# 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。

* debug困难，不知道何处出现了bug；保证线程安全也很困难。
* 多线程编程性能上确实有了较大的提高
* 遇到过线程不安全的情况。通过使用线程安全的集合类以及加以线程锁保证线程安全。
* 本实验工作量适度，但是赶上了考试周，就有点顾此失彼，个人需要好好的平衡时间。