

**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 | 朱明彦 |
| 学号 | 1160300314 |
| 班号 | 1603003 |
| 电子邮件 | [1160300314@stu.hit.edu.cn](mailto:1160300314@stu.hit.edu.cn) |
| 手机号码 | 18846082306 |

**目录**

[1 实验目标概述 1](#_Toc515980279)

[2 实验环境配置 1](#_Toc515980280)

[3 实验过程 1](#_Toc515980281)

[3.1 ADT设计方案 1](#_Toc515980282)

[3.1.1 basic包 1](#_Toc515980283)

[3.1.2 display包 2](#_Toc515980284)

[3.1.3 profiler包 3](#_Toc515980285)

[3.1.4 strategy包 3](#_Toc515980286)

[3.1.5 ladder包 4](#_Toc515980287)

[3.1.6 monkey包 5](#_Toc515980288)

[3.1.7 包之间的依赖关系 8](#_Toc515980289)

[3.2 Monkey线程的run()的执行流程图 9](#_Toc515980290)

[3.3 至少两种“梯子选择”策略的设计与实现方案 10](#_Toc515980291)

[3.3.1 策略1 10](#_Toc515980292)

[3.3.1.1 WaitUtilEmptyStrategy的设计 10](#_Toc515980293)

[3.3.1.2 WaitUtilEmptyStrategy的实现方案 10](#_Toc515980294)

[3.3.2 策略2 10](#_Toc515980295)

[3.3.2.1 WaitFirstPedalStrategy的设计 10](#_Toc515980296)

[3.3.2.2 WaitFirstPedalStrategy的实现方案 11](#_Toc515980297)

[3.3.3 策略3 11](#_Toc515980298)

[3.3.3.1 ChooseCrossingFastestStrategy的设计 11](#_Toc515980299)

[3.3.3.2 ChooseCrossingFastestStrategy的实现方案 11](#_Toc515980300)

[3.4 “猴子生成器”MonkeyGenerator 11](#_Toc515980301)

[3.5 如何确保threadsafe？ 12](#_Toc515980302)

[3.6 系统吞吐率和公平性的度量方案 13](#_Toc515980303)

[3.6.1 吞吐率 13](#_Toc515980304)

[3.6.2 公平性 13](#_Toc515980305)

[3.7 输出方案设计 14](#_Toc515980306)

[3.7.1 日志 14](#_Toc515980307)

[3.7.1.1 等待的记录 14](#_Toc515980308)

[3.7.1.2 行进的记录 14](#_Toc515980309)

[3.7.1.3 到达的记录 15](#_Toc515980310)

[3.7.2 GUI 15](#_Toc515980311)

[3.8 猴子过河模拟器v1 16](#_Toc515980312)

[3.8.1 参数如何初始化 16](#_Toc515980313)

[3.8.2 使用Strategy模式为每只猴子随机选择决策策略 16](#_Toc515980314)

[3.9 猴子过河模拟器v2 17](#_Toc515980315)

[3.9.1 对比分析：固定其他参数，选择不同的决策策略 17](#_Toc515980316)

[3.9.2 对比分析：变化某个参数，固定其他参数 18](#_Toc515980317)

[3.9.3 分析：吞吐率是否与各参数/决策策略有相关性？ 20](#_Toc515980318)

[3.9.3.1 吞吐率与决策策略的相关性 20](#_Toc515980319)

[3.9.3.2 吞吐率与各参数之间的相关性 21](#_Toc515980320)

[3.9.4 压力测试结果与分析 23](#_Toc515980321)

[3.9.4.1 使用多猴子少梯子进行测试 23](#_Toc515980322)

[3.9.4.2 使用相差较大的速度进行压力测试 24](#_Toc515980323)

[4 实验进度记录 24](#_Toc515980324)

[5 实验过程中遇到的困难与解决途径 24](#_Toc515980325)

[6 实验过程中收获的经验、教训、感想 25](#_Toc515980326)

# 实验目标概述

本次实验主要针对的是并行编程的能力。根据一个具体的需求，实现两个线程安全的猴子过河模拟器。并通过数据模拟保证的线程安全。

主要训练的方面：

* Java多线程编程
* 面向线程安全的ADT设计策略的选择与文档化
* 模拟仿真实验和对比实验
* 基本的GUI编程

# 实验环境配置

在这次的实验中，主要需要的实验环境与以往的实验环境类似，并没有新增其余的实验环境。因此没有再配置多余的实验环境。

在这里给出你的GitHub Lab6仓库的URL地址（Lab6-学号）。

<https://github.com/ComputerScienceHIT/Lab6-1160300314>

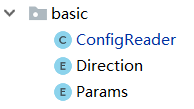
# 实验过程

## ADT设计方案

主要使用了basic、strategy等包，下面按照包结构进行解释设计

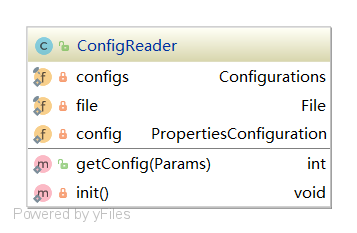
### basic包

包内的组织结构如下



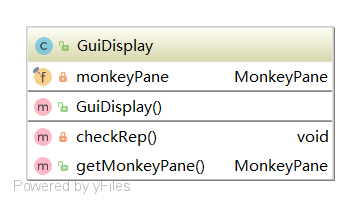
在basic包中有Direction，Params两个枚举类型，和ConfigReader一个类。其中Direction主要是用来规定猴子的方向只有两种R2L或者L2R；对于Params枚举类型，主要是所有的参数可能的名字，如Params.h就是指Config.properties中的h。

对于ConfigReader主要的功能就是实现在config.properties中读取配置信息。其详细的属性方法如下：



### display包

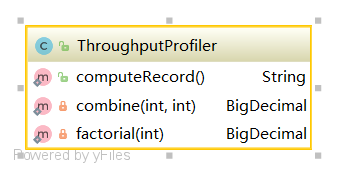
在这个包中只有一个GuiDisplay类，是用做使用GUI展示猴子过河的过程的类。其中的属性值与方法见下UML图



其中MonkeyPane是在monkey包中用做展示猴子位置的Pane类。

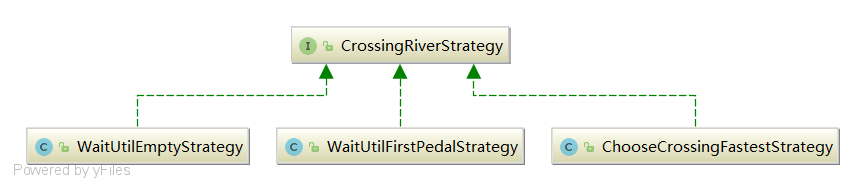
### profiler包

在profiler包中仅有ThroughputProfiler一个类，用做读取配置文件并分析其中的吞吐率指标和公平性指标。其中的属性与方法如下

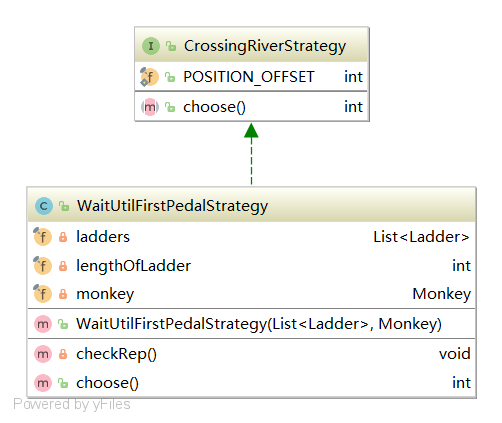


### strategy包

对于strategy包，主要是所有的策略类都在这个包中，其中的类之间的关系如下：



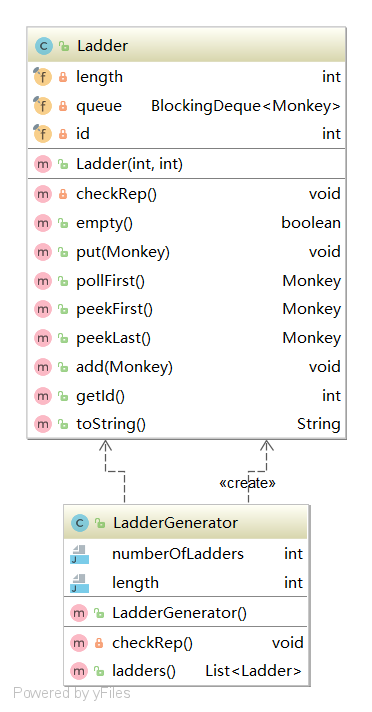
对于每一个实现CrossingRiverStrategy接口的类，都具有下面的结构



都必须实现choose方法，可以在不同的梯子之间进行选择。

### ladder包

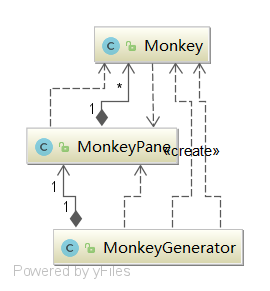
在ladder包中有两个类，Ladder和LadderGenerator两个类。具体的方法和属性如下：



在其中对于Ladder类来说，由于需要保证在不同的猴子线程访问时能够保证线程安全，所以所有涉及到对其中queue进行修改的操作都是使用synchronized进行修饰，以保证线程安全。

### monkey包

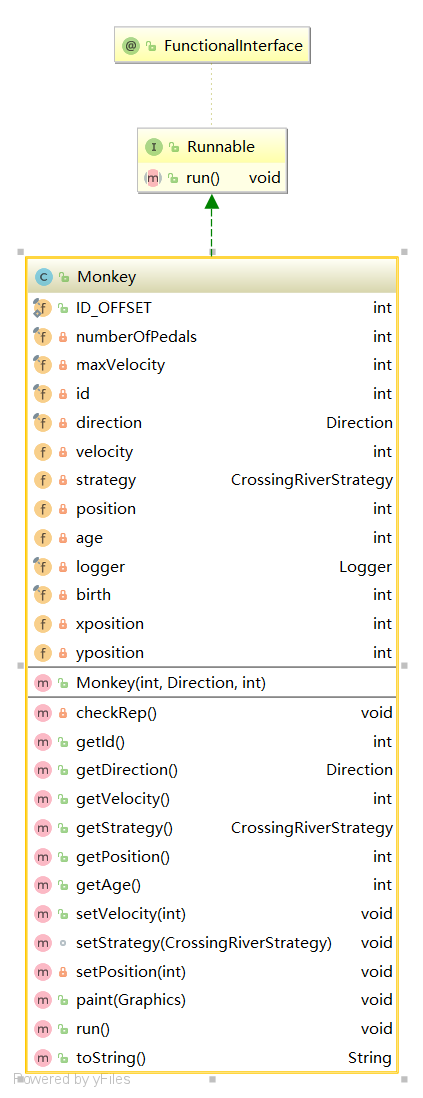
在monkey包中所有类的依赖关系如下：



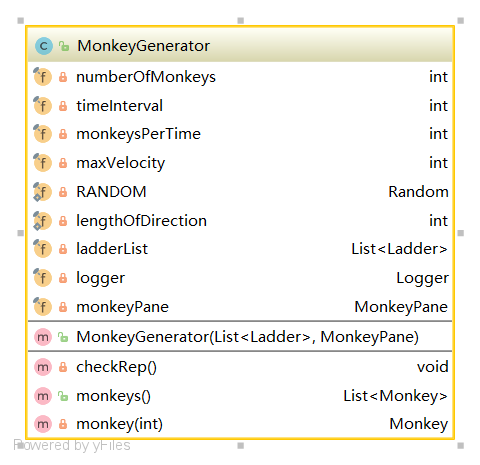
分别再看其余的类的设计如下：

Monkey类：

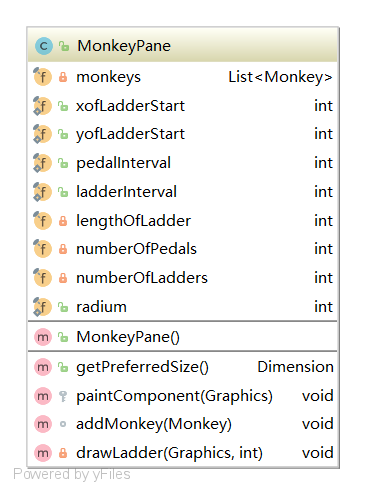
其作为主要的ADT，必须要记录应有的信息，比如ID、direction和速度等等。另外对于ladder来说其只保留在梯子上面的猴子的引用，具体对于选择梯子的一些操作还是在Monkey类中实现。



MonkeyGenerator类，类图如下：具体说明见[MonkeyGenerator](#_“猴子生成器”MonkeyGenerator)



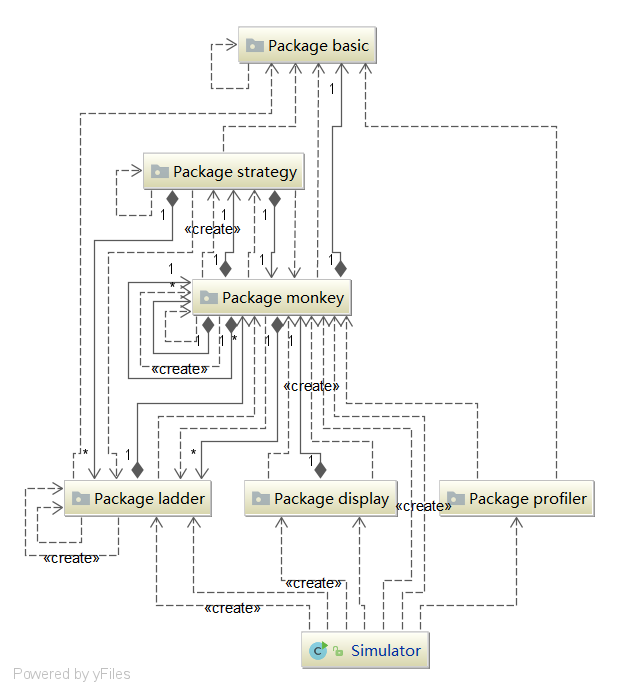
MonkeyPane类图如下：



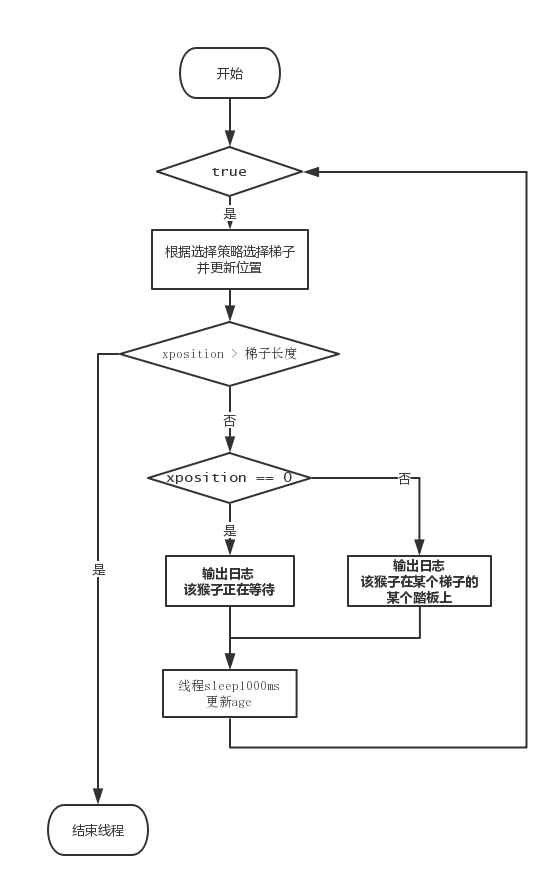
对于MonkeyPane来说，主要的功能就是在上面显示所有的梯子和猴子的过程，所以设计的时候需要保证实时可以向里面加入新的猴子并更新猴子的位置。基于这个原因，paintComponent函数是synchronized加锁的。

### 包之间的依赖关系

由于类的数目比较多，所以直接使用类之间的图来表示关系不够清晰，在这里使用包之间的关系来进行显示。



## Monkey线程的run()的执行流程图



此处的xposition表示如果猴子在梯子上，那么xposition为其所在的踏板编号；如果猴子不在梯子上，那么xposition为0.

## 至少两种“梯子选择”策略的设计与实现方案

### 策略1

#### WaitUtilEmptyStrategy的设计

WaitUtilEmptyStrategy，使用这种策略过河的猴子，在选择梯子的时候只会选择没有猴子的梯子。换句话说，如果梯子上面有猴子，那么使用这种策略的猴子会一直在岸边等待，直到出现有某个梯子上面没有猴子，才会选择通过梯子。

易见这种策略，会使得大量的时间在等待，不利于提高吞吐率，在后面的测试中我们也会发现这一点。

#### WaitUtilEmptyStrategy的实现方案

使用这个策略的猴子，直接遍历所有的梯子，当发现有某个梯子上面没有猴子，将该梯子“占领”，并踏上第1个踏板。

### 策略2

#### WaitFirstPedalStrategy的设计

WaitFirstPedalStrategy，使用这种策略过河的猴子，在选择梯子的时候会优先选择空梯子（没有猴子在上面的梯子），但是如果这个梯子上面有猴子而且上面的猴子与选择梯子的猴子的过河方向相同并且没有猴子在第1个踏板上，这个猴子也认为这个梯子可以选择。

对于这种策略，我们不好直接说明白其优劣，但是对于行进速度比较快的猴子，这种选择方式会导致其速度变慢。

#### WaitFirstPedalStrategy的实现方案

遍历所有的梯子，如果梯子是空的或者是梯子上面的猴子与待选择梯子的猴子的方向相同并且空出来第一个踏板，那么该猴子选择这个梯子。

### 策略3

#### ChooseCrossingFastestStrategy的设计

ChooseCrossingFastestStrategy，使用这种策略过河的猴子，优先选择空梯子；在没有梯子空闲的情况下，一定会遍历所有的梯子，并从中寻找到前进方向与自己一致，并且最后一个猴子的行进速度最快的梯子。

#### ChooseCrossingFastestStrategy的实现方案

遍历所有的梯子，如果可以找到空梯子，登上梯子；否则在所有的梯子中寻找在梯子上的最后一个猴子的前进方向与自己一致并且速度最大的一个梯子，登上该梯子。

## “猴子生成器”MonkeyGenerator

对于“猴子生成器”主要的功能就是按照规定的时间间隔和每次产生的猴子个数，产生指定个数的猴子。所以设计的方式就是，能够从config.properties中读取配置信息，并且按照配置文件的要求生成猴子，在生成猴子后就将对应于该猴子的线程开启即可。

## 如何确保threadsafe？

保证Thread-Safe的方式，由于对于每个猴子，都是单独的线程，所以主要的线程安全保证在于Ladder中数据结构的维护。

所以下面主要来说一下如何维护Ladder中的Thread-safe

在Ladder中最重要的数据结构就是BlockingDeque<Monkey>，首先JDK保证了其线程安全性，所以在使用的任何其原子操作都是线程安全的，进而防止Rep exposure，所以在其外面进行了“包装”。

所有针对其的操作均是synchronized保证线程安全的。

|  |
| --- |
|  |

## 系统吞吐率和公平性的度量方案

### 吞吐率

吞吐率的计算方式就是，将第1个猴子出生的时候作为起始时间，在最后一个猴子到达对岸作为终止时间，将所用的总时间T秒，和通过的猴子总数N做比。得到的 作为表征单位时间内通过的猴子数量。

举例说明：

如果第1只猴子出现，此时开始计时。在最后一只猴子（第10只猴子）通过梯子到达河对岸，此时的时间为20s。

那么这个系统的吞吐率就是

### 公平性

公平性就是指，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。此处注意，我们仅仅区分的是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，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。

基于上面的思想，我们引出两个猴子间公平性的计算公式如下：

其中Ya表示A出生的时间，Za表示A到达河对岸的时间。

然后再将以上的公平性进行组合得到总体的公平性：

其中表示的是N中取2的组合数。

## 输出方案设计

### 日志

对于日志的方式，将所有关于Monkey对象过河的全过程均记录在monkey.log文件中，分别记录等待、过河过程以及到达对岸；对于GUI的方式，主要记录的是多个猴子在河上通过梯子的动态过程。

在实验中综合使用了以上两种方式进行输出。

#### 等待的记录

采取以下格式进行记录：

|  |
| --- |
|  |

如上面这条记录，表示的是在2018-06-04 21:34:17的时候程序正在运行第28个线程，在程序开始25007ms后进行的记录，记录的内容是：Monkey90003，即第9批产生的第3只猴子，待在L（左岸）。时刻是第1只猴子出生后的第24秒，在这只猴子出生后的第0秒。

#### 行进的记录

采取以下格式进行记录

|  |
| --- |
|  |

记录的时间是2018-06-04 21:34:17,965，第22个线程，在程序开始运行的第25898ms进行的记录，记录的内容是：

Monkey70003，即第7批产生的第3只猴子，来到了第4个梯子的第8个踏板上，正在从右岸向左岸进发（R2L），在第一只猴子出生后的第25秒，在自己出生后的第7秒。

#### 到达的记录

记录的格式如下：

|  |
| --- |
|  |

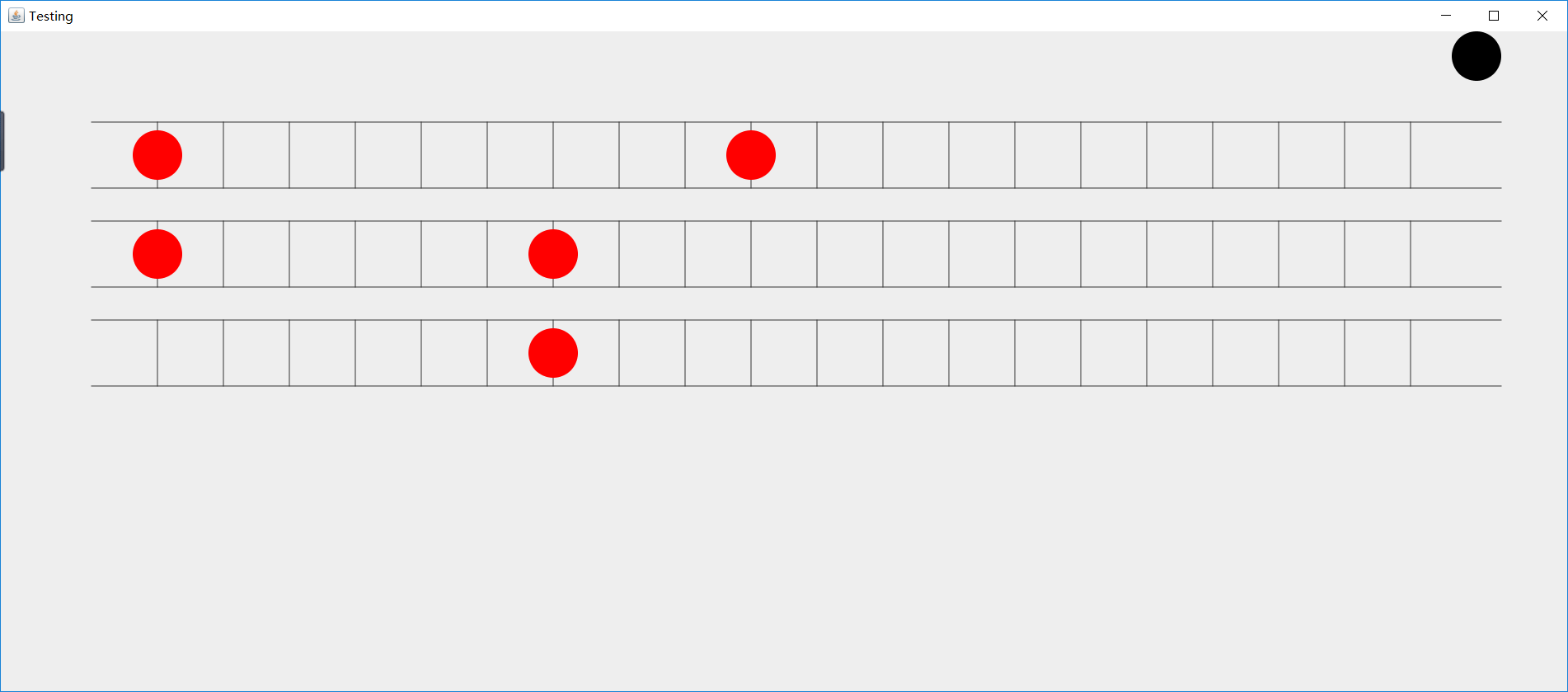
记录的时间是2018-06-04 21:34:20,916，第20个线程，在程序开始运行的第28849ms进行的记录，记录的内容是：

Monkey70001，即第7批产生的第1只猴子，通过第3个梯子到达河的左岸，此时的时间是第1只猴子出生后28秒，自己出生后第10秒。

### GUI

GUI的方式主要是可以形象化的表示每个时刻猴子在梯子上的位置，记录的方式如下，此处使用的测试参数为

h = 20, t = 3, N = 10, k = 3, MV = 5, n = 3



在图中红色的实心圆表示从左岸去右岸的猴子，黑色的实心圆表示从右侧去左侧的猴子。可以看到，在截图的瞬间，从左岸出发的猴子在梯子上面行进，分别到达了第1个踏板等等；从右岸出发的猴子，由于没有梯子可以使用（避免死锁），因此在河的右岸等待。

## 猴子过河模拟器v1

### 参数如何初始化

由于各种参数的面向的类不同，此处使用Apache开源的配置文件读取进行参数的初始化。

在其中使用一个类ConfigReader，当所有的类需要读取默认的参数值时，从Config.properties中直接读取所需要的参数。其中对于所有可能的参数，使用enum进行组织，Params如下表示：

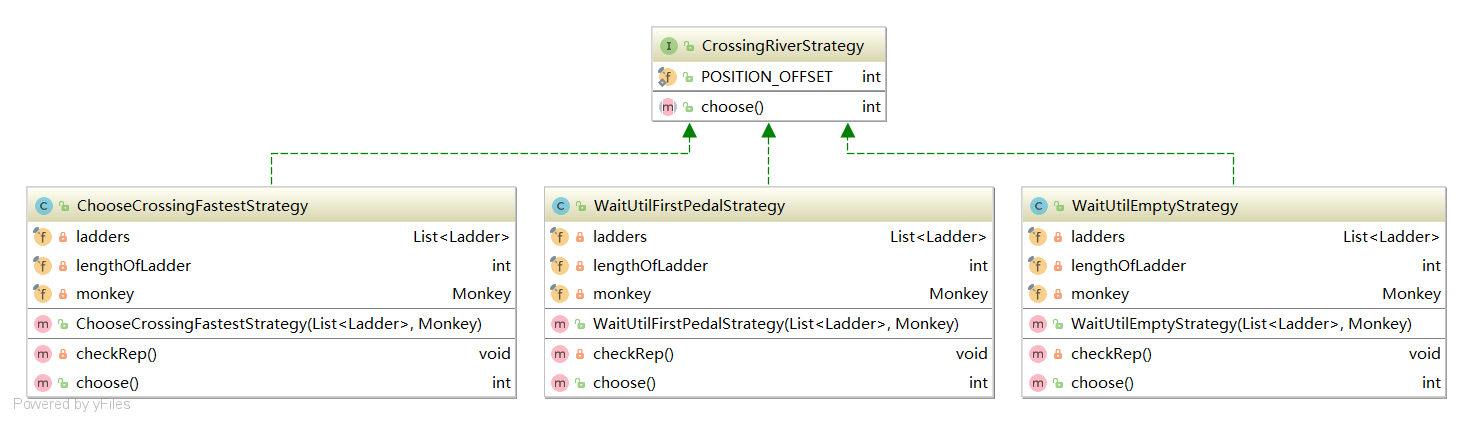
|  |
| --- |
|  |

### 使用Strategy模式为每只猴子随机选择决策策略

在MonkeyGenerator中，使用随机数为Monkey选择决策策略。

|  |
| --- |
|  |

对于Strategy的继承关系，见下面的类图：



## 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。**以下所有的测试结果，都在doc/time.xlsx进行详细的记录。由于对于所有猴子的速度是随机产生的，为了避免随机值对于结果的影响，所以以下所有测试都进行10次取平均值。**

### 对比分析：固定其他参数，选择不同的决策策略

h = 20, t = 3, N = 10, k = 3, MV = 5, n = 3

测试的结果如下：



### 对比分析：变化某个参数，固定其他参数

**1、测试n（梯子数目）的影响**

均使用ChooseCrossingFastestStrategy

相同的参数设置如下

h = 20, t = 3, N = 10, k = 3, MV = 5

测试的结果如下：



**2、测试t（产生猴子的时间间隔）的影响**

均使用ChooseCrossingFastestStrategy

相同的参数设置如下

h = 20, n = 3, N = 10, k = 3, MV = 5

测试的结果如下：



**3、测试k（单次产生猴子的数目）的影响**

均使用ChooseCrossingFastestStrategy

相同的参数设置如下

h = 20, n = 3, N = 10, t = 3, MV = 5



**4、测试N（猴子数目）的影响**

均使用ChooseCrossingFastestStrategy

相同的参数设置如下

h = 20, n = 3, t = 3, k = 3, MV = 5

测试的结果如下：



### 分析：吞吐率是否与各参数/决策策略有相关性？

#### 吞吐率与决策策略的相关性



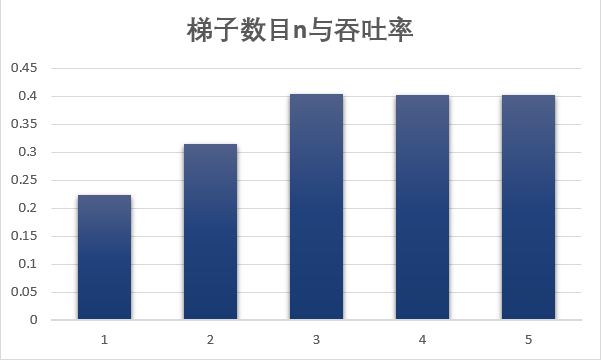
根据[3.9.1节](#_对比分析：固定其他参数，选择不同的决策策略)对于决策策略的测试可以看到，在固定参数的时候，WaitUtilEmpty（等待直到有空梯子为止）的策略吞吐率最低，只有0.30左右。而其余的两个策略相差不大，分别是0.34和0.35左右。虽然已经经过了10次的实验来减少随机速度对于最终结果的影响，但是没有办法消除。

综合来看，**吞吐率与决策策略直接相关，选择WaitUtilEmpty策略的吞吐率较其余两个策略偏低。**

#### 吞吐率与各参数之间的相关性

1、与梯子的数目n

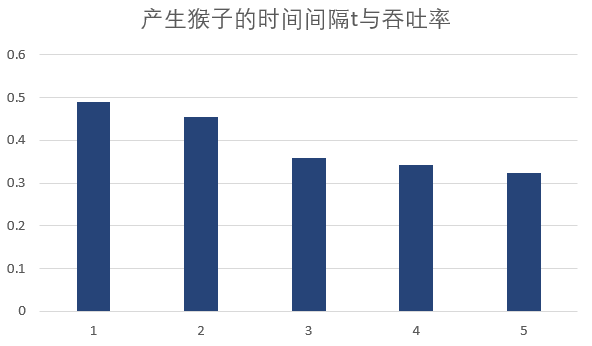
在通过固定其余参数来看与n的相关性，可以看到**随着n的增大，吞吐率在逐渐提升，并最终趋于平稳。**



可以分析一下，在猴子数目较少（N=10）的情况下，并且猴子出生的间隔也比较大的前提下，梯子数目多，因此不会有过多的猴子在同一个梯子上的情况，进而不能够无限制的提高并行的效率。

2、与产生猴子的时间间隔t

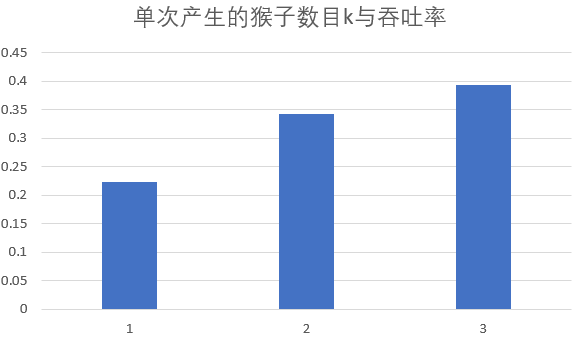
在固定其余参数，来看与t的相关性，可以看到**随着t的增大，吞吐率逐渐下降。**



可以分析一下，在猴子数目比较少（N=10）的情况下，t越大，意味着在下一批猴子出生前，所有的猴子已经通过了梯子，因而没有并行效率提高的体现，进而导致吞吐率的下降。

3、与单次产生的猴子数目k

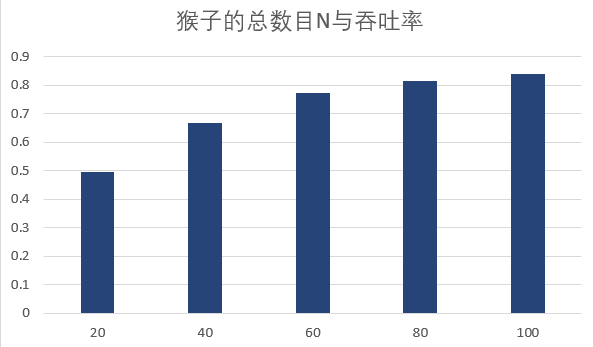
在固定其余参数，来看与k的相关性，可以看到**随着k的增大，吞吐率逐渐提升。**



可以分析一下，k与t恰好是一对相反的影响因素。k越大，意味着我有更多的猴子在同一时刻出生，那么就有可能出现在同一个梯子上有多个猴子一同过河的情况，那么理所应当的可以提高并行的效率。

4、与猴子的总数目N

在固定其余参数的情况下，来看与N的相关性，可以看到**随着N的增大，吞吐率逐渐提升。**



可以想到，在猴子数目越多的情况下，使用策略如果不是等待直到有空梯子的策略，那么在同一时刻会有多个猴子在梯子上，并且随机的速度对于吞吐率的影响要减小，而是与第一次出现并踏上梯子的速度为1猴子的时间相关。

### 压力测试结果与分析

#### 使用多猴子少梯子进行测试

测试使用的参数列表如下：

|  |  |
| --- | --- |
| n | 3 |
| h | 20 |
| t | 3 |
| N | 3000 |
| k | 3 |
| MV | 5 |

最终测试的log文件详见./log/monkey-压力测试1.log

最终的测试结果文件详见./log/record-压力测试1.txt

得到的最后的吞吐率和公平性如下：

Throughput = 0.9940357852882704

Justice = 0.9998959653

#### 使用相差较大的速度进行压力测试

使用下面的参数进行测试

|  |  |
| --- | --- |
| n | 3 |
| h | 20 |
| t | 3 |
| N | 100 |
| k | 3 |
| MV | 5 |

其中，**随机产生的猴子速度只有两种1或者5，**最终得到的完整测试日志文件见./log/monkey-压力测试2.log，最终的完整的测试结果文件见./log/record-压力测试2.txt。

测试得到的吞吐率和公平性见下：

Throughput = 0.8403361344537815

Justice = 1.0000000000

# 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 |
| 2018-5-28 | 13:45-17:00 | 写配置文件 设计类 | 完成 |
| 2018-5-29 |  | 完成monkey的初步设计 | 完成 |
| 2018-5-30 | 2018-6-3 | 配置log4j和写GUI | 延期 |
| 2018-6-3 | 22:00前 | GUI写完 重构代码 | 完成 |
| 2018-6-4 | 23:00前 | 报告 | 延期至次日16点完成 |

# 实验过程中遇到的困难与解决途径

log4j记录日志的功能在输出到文件的时候会将部分日志文本删除，但是如果使用Append=true的方式输出，就不会出现类似的问题。

解决的方式：

由于在Stack Overflow和log4j官方的讨论区都没有给出相应的解决方法，所以只好通过命令行进行输出。然后通过重定向将命令行输出的文本输出到文件中。

# 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）单线程程序的调试是容易复现错误的，而对于多线程来说，很多错误不能复现。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，这种性能的提升更多的来自于过个任务执行时的优势。

（2）使用线程安全的数据结构和synchronized加锁来实现。

优先保证线程安全，然后再考虑性能问题。

（3）遇到了线程不安全的问题，在多个猴子同时访问ladder的时候，将占用的信息没有及时的写回。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。

修改后与预期相同。

（4）是所有实验中感觉最有趣的实验，deadline比较合适。

（5）最后一个实验结束了，感觉做下来6个实验，确实对自己编程方面的能力有不少的提升。